

# Springer Handbook of Speech Processing

DVD ROM

Benesty
Sondhi
Huang
Editors

Springer

Amazon EX1010 Page 1

# Springer Handbook of Speech Processing

Amazon EX1010 Page 2

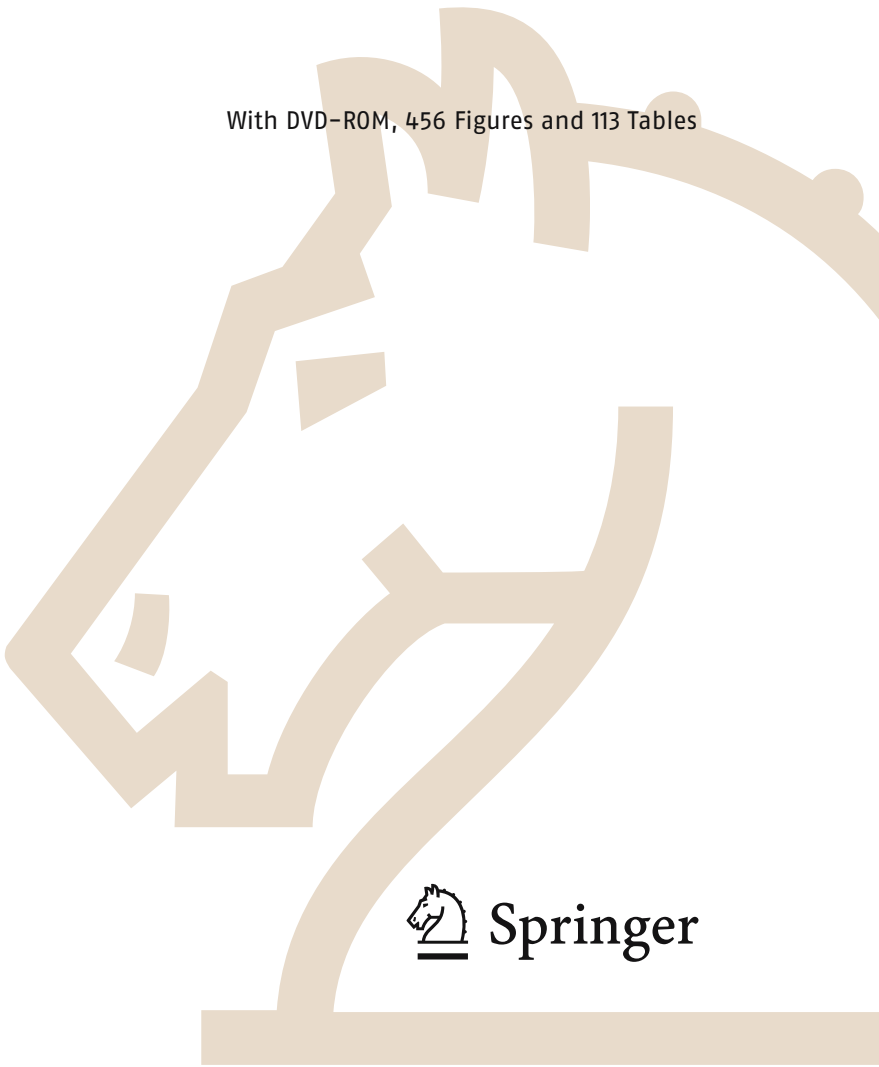
**Springer Handbooks** provide a concise compilation of approved key information on methods of research, general principles, and functional relationships in physical sciences and engineering. The world's leading experts in the fields of physics and engineering will be assigned by one or several renowned editors to write the chapters comprising each volume. The content is selected by these experts from Springer sources (books, journals, online content) and other systematic and approved recent publications of physical and technical information.

The volumes are designed to be useful as readable desk reference books to give a fast and comprehensive overview and easy retrieval of essential reliable key information, including tables, graphs, and bibliographies. References to extensive sources are provided.

# Springer Handbook of Speech Processing

Jacob Benesty, M. Mohan Sondhi, Yiteng Huang (Eds.)

With DVD-ROM, 456 Figures and 113 Tables



Springer

Amazon EX1010 Page 4

Editors:

Jacob Benesty
INRS-EMT, University of Quebec
800 de la Gauchetiere Ouest, Suite 6900
Montreal, Quebec, H5A 1K6, Canada
benesty@emt.inrs.ca

M. Mohan Sondhi
Avayalabs Research
233 Mount Airy Road
Basking Ridge, NJ 07920, USA
mms@research.avayalabs.com

Yiteng Huang
Bell Laboratories, Alcatel-Lucent
600 Mountain Avenue
Murray Hill, NJ 07974, USA
arden_huang@ieee.org

Library of Congress Control Number: 2007931999

ISBN: 978-3-540-49125-5 e-ISBN: 978-3-540-49127-9

This work is subject to copyright. All rights reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September, 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2008

The use of designations, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Product liability: The publisher cannot guarantee the accuracy of any information about dosage and application contained in this book. In every individual case the user must check such information by consulting the relevant literature.

Typesetting and production:
LE-TEX Jelonek, Schmidt&Vöckler GbR, Leipzig
Senior Manager Springer Handbook: Dr. W. Skolaut, Heidelberg
Typography and layout: schreiberVIS, Seeheim
Illustrations: Hippmann GbR, Schwarzenbruck
Cover design: eStudio Calamar Steinen, Barcelona
Cover production: WMXDesign GmbH, Heidelberg
Printing and binding: Stürtz GmbH, Würzburg

Printed on acid free paper

SPIN 11544036 60/3180/YL 5 4 3 2 1 0

# Foreword



**J. L. Flanagan**
**Professor Emeritus**
**Electrical and Computer Engineering**
**Rutgers University**

Over the past three decades digital signal processing has emerged as a recognized discipline. Much of the impetus for this advance stems from research in representation, coding, transmission, storage and reproduction of speech and image information. In particular, interest in voice communication has stimulated central contributions to digital filtering and discrete-time spectral transforms.

This dynamic development was built upon the convergence of three then-evolving technologies: (i) sampled-data theory and representation of information signals (which led directly to digital telecommunication that provides signal quality independent of transmission distance); (ii) electronic binary computation (aided in early implementation by pulse-circuit techniques from radar design); and, (iii) invention of solid-state devices for exquisite control of electronic current (transistors – which now, through microelectronic materials, scale to systems of enormous size and complexity). This timely convergence was soon followed by optical fiber methods for broadband information transport.

These advances impact an important aspect of human activity – information exchange. And, over man's existence, speech has played a principal role in human communication. Now, speech is playing an increasing role in human interaction with complex information systems. Automatic services of great variety exploit the comfort of voice exchange, and, in the corporate sector, sophisticated audio/video teleconferencing is reducing the necessity of expensive, time-consuming business travel. In each instance an overarching target is a user environment that captures some of the naturalness and spatial realism of face-to-face communication. Again, speech is a core element, and new understanding from diverse research sectors can be brought to bear.

Editors-in-Chief Benesty, Sondhi and Huang have organized a timely engineering handbook to answer this need. They have assembled a remarkable compendium of current knowledge in speech processing. And, this accumulated understanding can be focused upon enlarging the human capacity to deal with a world ever increasing in complexity. Benesty, Sondhi and Huang are renowned researchers in their own right, and they have attracted an international cadre of over 80 fellow authors and collaborators who constitute a veritable *Who's Who* of world leaders in speech processing research. The resulting book provides under one cover authoritative treatments that commence with the basic physics and psychophysics of speech and hearing, and range through the related topics of computational tools, coding, synthesis, recognition, and signal enhancement, concluding with discussions on capture and projection of sound in enclosures. The book can be expected to become a valuable resource for researchers, engineers and speech scientists throughout the global community. It should equally serve teachers and students in human communication, especially delimiting knowledge frontiers where graduate thesis research may be appropriate.

Warren, New Jersey
October 2007

Jim Flanagan

Amazon EX1010 Page 6

# Preface

The achievement of this Springer Handbook is the result of a wonderful journey that started in March 2005 at the 30th International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Two of the editors-in-chief (Benesty and Huang) met in one of the long corridors of the Pennsylvania Convention Center in Philadelphia with Dr Dieter Merkle from Springer. Together we had a very nice discussion about the conference and immediately an idea came up for a handbook. After a short discussion we converged without too much hesitation on a handbook of *speech processing*. It was quite surprising to see that, even after 30 years of ICASSP and more than half a century of research in this fundamental area, there was still no major book summarizing the important aspects of speech processing. We thought that the time was ripe for such a large project. Soon after we got home, a third editor-in-chief (Sondhi) joined the efforts.



**Jacob Benesty**



**M. Mohan Sondhi**



**Yiteng Huang**

We had a very clear objective in our minds: to summarize, in a reasonable number of pages, the most important and useful aspects of speech processing. The content was then organized accordingly. This task was not easy since we had to find a good balance between feasible ideas and new trends. As we all know, practical ideas can be viewed as *old stuff* while emerging ideas can be criticized for not having passed the test of time; we hope that we have succeeded in finding a good compromise. For this we relied on many authors who are well established and are recognized as experts in their field, from all over the world, and from academia as well as from industry.

From *simple* consumer products such as cell phones and MP3 players to more-sophisticated projects such as human–machine interfaces and robots that can obey orders, speech technologies are now everywhere. We believe that it is just a matter of time before more applications of the science of speech become impossible to miss in our daily life. So we believe that this Springer Handbook will play a fundamental role in the sustainable progress of speech research and development.

This handbook is targeted at three categories of readers: graduate students of speech processing, professors and researchers in academia and research labs who are active in this field, and engineers in industry who need to understand or implement specific algorithms for their speech-related products. The handbook could also be used as a text for one or more graduate courses on signal processing for speech and various aspects of speech processing and applications.

For the completion of such an ambitious project we have many people to thank. First, we would like to thank the many authors who did a terrific job in delivering very high-quality chapters. Second, we are very grateful to the members of the editorial board who helped us so much in organizing the content and structure of this book, taking part in all phases of this project from conception to completion. Third, we would like to thank all the reviewers, who helped us to improve the quality of the material. Last, but not least, we would like to thank the Springer team for their availability and very professional work. In particular, we appreciated the help of Dieter Merkle, Christoph Baumann, Werner Skolaut, Petra Jantzen, and Claudia Rau.

We hope this Springer Handbook will inspire many great minds to find new research ideas or to implement algorithms in products.

Montreal, Basking Ridge, Murray Hill
October 2007

Jacob Benesty
M. Mohan Sondhi
Yiteng Huang

Amazon EX1010 Page 7

# List of Editors

## Editors-in-Chief

Jacob Benesty, Montreal
M. Mohan Sondhi, Basking Ridge
Yiteng (Arden) Huang, Murray Hill

## Part Editors

### Part A: Production, Perception, and Modeling of Speech

M. M. Sondhi, Basking Ridge

### Part B: Signal Processing for Speech

Y. Huang, Murray Hill; J. Benesty, Montreal

### Part C: Speech Coding

W. B. Kleijn, Stockholm

### Part D: Text-to-Speech Synthesis

S. Narayanan, Los Angeles

### Part E: Speech Recognition

L. Rabiner, Piscataway; B.-H. Juang, Atlanta

### Part F: Speaker Recognition

S. Parthasarathy, Sunnyvale

### Part G: Language Recognition

C.-H. Lee, Atlanta

### Part H: Speech Enhancement

J. Chen, Murray Hill; S. Gannot, Ramat-Gan; J. Benesty, Montreal

### Part I: Multichannel Speech Processing

J. Benesty, Montreal; I. Cohen, Haifa; Y. Huang, Murray Hill

## List of Authors

**Alex Acero**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
e-mail: *alexac@microsoft.com*

**Jont B. Allen**
University of Illinois
ECE
Urbana, IL 61801, USA
e-mail: *JontAllen@ieee.org*

**Jacob Benesty**
University of Quebec
INRS-EMT
800 de la Gauchetiere Ouest
Montreal, Quebec H5A 1K6, Canada
e-mail: *benesty@emt.inrs.ca*

**Frédéric Bimbot**
IRISA (CNRS & INRIA) - METISS
Pièce C 320 - Campus Universitaire de Beaulieu
35042 Rennes, France
e-mail: *bimbot@irisa.fr*

**Thomas Brand**
Carl von Ossietzky Universität Oldenburg
Sektion Medizinphysik
Haus des Hörens, Marie-Curie-Str. 2
26121 Oldenburg, Germany
e-mail: *thomas.brand@uni-oldenburg.de*

**Nick Campbell**
Knowledge Creating Communication Research Centre
Acoustics & Speech Research Project, Spoken Language Communication Group
2-2-2 Hikaridai
619-0288 Keihanna Science City, Japan
e-mail: *nick@nict.go.jp*

**William M. Campbell**
MIT Lincoln Laboratory
Information Systems Technology Group
244 Wood Street
Lexington, MA 02420-9108, USA
e-mail: *wcampbell@ll.mit.edu*

**Rolf Carlson**
Royal Institute of Technology (KTH)
Department of Speech, Music and Hearing
Lindstedtsvägen 24
10044 Stockholm, Sweden
e-mail: *rolf@speech.kth.se*

**Jingdong Chen**
Bell Laboratories
Alcatel-Lucent
600 Mountain Ave
Murray Hill, NJ 07974, USA
e-mail: *jingdong@research.bell-labs.com*

**Juin-Hwey Chen**
Broadcom Corp.
5300 California Avenue
Irvine, CA 92617, USA
e-mail: *rchen@broadcom.com*

**Israel Cohen**
Technion–Israel Institute of Technology
Department of Electrical Engineering
Technion City
Haifa 32000, Israel
e-mail: *icohen@ee.technion.ac.il*

**Jordan Cohen**
SRI International
300 Ravenswood Drive
Menlo Park, CA 94019, USA
e-mail: *jrc@speech.sri.com*

**Corinna Cortes**
Google, Inc.
Google Research
76 9th Avenue, 4th Floor
New York, NY 10011, USA
e-mail: *corinna@google.com*

**Eric J. Diethorn**
Avaya Labs Research
Multimedia Technologies Research Department
233 Mt. Airy Road
Basking Ridge, NJ 07920, USA
e-mail: *ejd@avaya.com*

**Simon Doclo**
Katholieke Universiteit Leuven
Department of Electrical Engineering (ESAT-SCD)
Kasteelpark Arenberg 10 bus 2446
3001 Leuven, Belgium
e-mail: *simon.doclo@esat.kuleuven.be*

**Jasha Droppo**
Microsoft Research
Speech Technology Group
One Microsoft Way
Redmond, WA 98052, USA
e-mail: *jdroppo@microsoft.com*

**Thierry Dutoit**
Faculté Polytechnique de Mons FPMs
TCTS Laboratory
Bvd Dolez, 31
7000 Mons, Belgium
e-mail: *thierry.dutoit@fpms.ac.be*

**Gary W. Elko**
mh acoustics LLC
25A Summit Ave
Summit, NJ 07901, USA
e-mail: *gwe@mhacoustics.com*

**Sadaoki Furui**
Tokyo Institute of Technology Street
Department of Computer Science
2-12-1 Ookayama, Meguro-ku
152-8552 Tokyo, Japan
e-mail: *furui@cs.titech.ac.jp*

**Sharon Gannot**
Bar-Ilan University
School of Electrical Engineering
Ramat-Gan 52900, Israel
e-mail: *gannot@eng.biu.ac.il*

**Mazin E. Gilbert**
AT&T Labs, Inc., Research
180 Park Ave.
Florham Park, NJ 07932, USA
e-mail: *mazin@research.att.com*

**Michael M. Goodwin**
Creative Advanced Technology Center
Audio Research
1500 Green Hills Road
Scotts Valley, CA 95066, USA
e-mail: *mgoodwin@atc.creative.com*

**Volodya Grancharov**
Multimedia Technologies
Ericsson Research, Ericsson AB
Torshamnsgatan 23, Kista, KI/EAB/TVA/A
16480 Stockholm, Sweden
e-mail: *volodya.grancharov@ericsson.com*

**Björn Granström**
Royal Institute of Technology (KTH)
Department for Speech, Music and Hearing
Lindstedsvägen 24
10044 Stockholm, Sweden
e-mail: *bjorn@speech.kth.se*

**Patrick Haffner**
AT&T Labs-Research
IP and Voice Services
200 S Laurel Ave.
Middletown, NJ 07748, USA
e-mail: *haffner@research.att.com*

**Roar Hagen**
Global IP Solutions
Magnus Ladulsgatan 63B
118 27 Stockholm, Sweden
e-mail: *roar.hagen@gipscorp.com*

**Mary P. Harper**
University of Maryland
Center for Advanced Study of Language
7005 52nd Avenue
College Park, MD 20742, USA
e-mail: *mharper@casl.umd.edu*

**Jürgen Herre**
Fraunhofer Institute for Integrated Circuits
(Fraunhofer IIS)
Audio and Multimedia
Am Wolfsmantel 33
91058 Erlangen, Germany
e-mail: *hrr@iis.fraunhofer.de*

Amazon EX1010 Page 10

**Wolfgang J. Hess**
University of Bonn
Institute for Communication Sciences, Dept. of Communication, Language, and Speech
Poppelsdorfer Allee 47
53115 Bonn, Germany
e-mail: *wgh@ifk.uni-bonn.de*

**Kiyoshi Honda**
Université de la Sorbonne Nouvelle-Paris III
Laboratoire de Phonétique et de Phonologie, ATR Cognitive Information Laboratories
UMR-7018-CNRS, 46, rue Barrault
75634 Paris, France
e-mail: *honda@atr.jp*

**Yiteng (Arden) Huang**
Bell Laboratories
Alcatel-Lucent
600 Mountain Avenue
Murray Hill, NJ 07974, USA
e-mail: *arden_huang@ieee.org*

**Matthieu Hébert**
Network ASR Core Technology
Nuance Communications
1500 Université
Montréal, Québec H3A-3S7, Canada
e-mail: *hebert@nuance.com*

**Biing-Hwang Juang**
Georgia Institute of Technology
School of Electrical & Computer Engineering
777 Atlantic Dr. NW
Atlanta, GA 30332-0250, USA
e-mail: *juang@ece.gatech.edu*

**Tatsuya Kawahara**
Kyoto University
Academic Center for Computing and Media Studies
Sakyo-ku
606-8501 Kyoto, Japan
e-mail: *kawahara@i.kyoto-u.ac.jp*

**Ulrik Kjems**
Oticon A/S
9 Kongebakken
2765 Smørum, Denmark
e-mail: *uk@oticon.dk*

**Esther Klabbers**
Oregon Health & Science University
Center for Spoken Language Understanding, OGI School of Science and Engineering
20000 NW Walker Rd
Beaverton, OR 97006, USA
e-mail: *klabbers@cslu.ogi.edu*

**W. Bastiaan Kleijn**
Royal Institute of Technology (KTH)
School of Electrical Engineering, Sound and Image Processing Lab
Osquldas väg 10
10044 Stockholm, Sweden
e-mail: *bastiaan.kleijn@ee.kth.se*

**Birger Kollmeier**
Universität Oldenburg
Medizinische Physik
26111 Oldenburg, Germany
e-mail: *birger.kollmeier@uni-oldenburg.de*

**Ermin Kozica**
Royal Institute of Technology (KTH)
School of Electrical Engineering, Sound and Image Processing Laboratory
Osquldas väg 10
10044 Stockholm, Sweden
e-mail: *ermin.kozica@ee.kth.se*

**Sen M. Kuo**
Northern Illinois University
Department of Electrical Engineering
DeKalb, IL 60115, USA
e-mail: *kuo@ceet.niu.edu*

**Jan Larsen**
Technical University of Denmark
Informatics and Mathematical Modelling
Richard Petersens Plads
2800 Kongens Lyngby, Denmark
e-mail: *jl@imm.dtu.dk*

**Chin-Hui Lee**
Georgia Institute of Technology
School of Electrical and Computer Engineering
777 Atlantic Drive NW
Atlanta, GA 30332-0250, USA
e-mail: *chl@ece.gatech.edu*

Amazon EX1010 Page 11

**Haizhou Li**
Institute for Infocomm Research
Department of Human Language Technology
21 Heng Mui Keng Terrace
Singapore, 119613
e-mail: *hli@i2r.a-star.edu.sg*

**Jan Linden**
Global IP Solutions
301 Brannan Street
San Francisco, CA 94107, USA
e-mail: *jan.linden@gipscorp.com*

**Manfred Lutzky**
Fraunhofer Integrated Circuits (IIS)
Multimedia Realtime Systems
Am Wolfsmantel 33
91058 Erlangen, Germany
e-mail: *manfred.lutzky@iis.fraunhofer.de*

**Bin Ma**
Human Language Technology
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore, 119613
e-mail: *mabin@i2r.a-star.edu.sg*

**Michael Maxwell**
University of Maryland
Center for Advanced Study of Language
Box 25
College Park, MD 20742, USA
e-mail: *mmaxwell@casl.umd.edu*

**Alan V. McCree**
MIT Lincoln Laboratory
Department of Information Systems Technology
244 Wood Street
Lexington, MA 02420-9185, USA
e-mail: *mccree@ll.mit.edu*

**Bernd Meyer**
Carl von Ossietzky Universität Oldenburg
Medical Physics Section, Haus des Hörens
Marie-Curie-Str. 2
26121 Oldenburg, Germany
e-mail: *bernd.meyer@uni-oldenburg.de*

**Jens Meyer**
mh acoustcis
25A Summit Ave.
Summit, NJ 07901, USA
e-mail: *jmm@mhacoustics.com*

**Taniya Mishra**
Oregon Health and Science University
Center for Spoken Language Understanding,
Computer Science and Electrical Engineering, OGI
School of Science and Engineering
20000 NW Walker Road
Beaverton, OR 97006, USA
e-mail: *mishra@cslu.ogi.edu*

**Mehryar Mohri**
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, NY 10012, USA
e-mail: *mohri@cs.nyu.edu*

**Marc Moonen**
Katholieke Universiteit Leuven
Electrical Engineering Department ESAT/SISTA
Arenberg 10
3001 Leuven, Belgium
e-mail: *marc.moonen@esat.kuleuven.be*

**Dennis R. Morgan**
Bell Laboratories, Alcatel-Lucent
700 Mountain Avenue 2D-537
Murray Hill, NJ 07974-0636, USA
e-mail: *drrm@bell-labs.com*

**David Nahamoo**
IBM Thomas J. Watson Research Center
PO BOX 218
Yorktown Heights, NY 10598, USA
e-mail: *nahamoo@us.ibm.com*

**Douglas O'Shaughnessy**
Université du Québec
INRS Énergie, Matériaux et Télécommunications
(INRS-EMT)
800, de la Gauchetiere Ouest
Montréal, Québec H5A 1K6, Canada
e-mail: *dougo@emt.inrs.ca*

Amazon EX1010 Page 12

**Lucas C. Parra**
Steinman Hall, The City College of New York
Department of Biomedical Engineering
140th and Convent Ave
New York, NY 10031, USA
e-mail: *parra@ccny.cuny.edu*

**Sarangarajan Parthasarathy**
Yahoo!, Applied Research
1MC 743, 701 First Avenue
Sunnyvale, CA 94089-0703, USA
e-mail: *parthas@yahoo-inc.com*

**Michael Syskind Pedersen**
Oticon A/S
Kongebakken 9
2765 Smørum, Denmark
e-mail: *msp@oticon.dk*

**Fernando Pereira**
University of Pennsylvania
Department of Computer and Information Science
305 Levine Hall, 3330 Walnut Street
Philadelphia, PA 19104, USA
e-mail: *pereira@cis.upenn.edu*

**Michael Picheny**
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
e-mail: *Picheny@us.ibm.com*

**Rudolf Rabenstein**
University Erlangen-Nuremberg
Electrical Engineering, Electronics, and
Information Technology
Cauerstrasse 7/LMS
91058 Erlangen, Germany
e-mail: *rabe@LNT.de*

**Lawrence Rabiner**
Rutgers University
Department of Electrical and Computer
Engineering
96 Frelinghuysen Road
Piscataway, NJ 08854, USA
e-mail: *lrr@caip.rutgers.edu*

**Douglas A. Reynolds**
Massachusetts Institute of Technology
Lincoln Laboratory, Information Systems
Technology Group
244 Wood Street
Lexington, MA 02420-9108, USA
e-mail: *dar@ll.mit.edu*

**Michael Riley**
Google, Inc., Research
111 Eighth AV
New York, NY 10011, USA
e-mail: *riley@google.com*

**Aaron E. Rosenberg**
Rutgers University
Center for Advanced Information Processing
96 Frelinghuysen Road
Piscataway, NJ 08854-8088, USA
e-mail: *aer@caip.rutgers.edu*

**Salim Roukos**
IBM T. J. Watson Research Center
Multilingual NLP Technologies
Yorktown Heights, NY 10598, USA
e-mail: *roukos@us.ibm.com*

**Jan van Santen**
Oregon Health And Science University
OGI School of Science and Engineering,
Department of Computer Science and Electrical
Engineering
20000 NW Walker Rd
Beaverton, OR 97006-8921, USA
e-mail: *vansanten@cslu.ogi.edu*

**Ronald W. Schafer**
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
e-mail: *ron.schafer@hp.com*

**Juergen Schroeter**
AT&T Labs - Research
Department of Speech Algorithms and Engines
180 Park Ave
Florham Park, NJ 07932, USA
e-mail: *schroeter@att.com*

Amazon EX1010 Page 13

**Stephanie Seneff**
Massachusetts Institute of Technology
Computer Science and Artificial
Intelligence Laboratory
32 Vassar Street
Cambridge, MA 02139, USA
e-mail: *senef@csail.mit.edu*

**Wade Shen**
Massachusetts Institute of Technology
Communication Systems, Information Systems
Technology, Lincoln Laboratory
244 Wood Street
Lexington, MA 02420-9108, USA
e-mail: *swade@ll.mit.edu*

**Elliot Singer**
Massachusetts Institute of Technology
Information Systems Technology Group, Lincoln
Laboratory
244 Wood Street
Lexington, MA 02420-9108, USA
e-mail: *es@ll.mit.edu*

**Jan Skoglund**
Global IP Solutions
301 Brannan Street
San Francisco, CA 94107, USA
e-mail: *jan.skoglund@gipscorp.com*

**M. Mohan Sondhi**
Avayalabs Research
233 Mount Airy Road
Basking Ridge, NJ 07920, USA
e-mail: *mms@research.avayalabs.com*

**Sascha Spors**
Deutsche Telekom AG, Laboratories
Ernst-Reuter-Platz 7
10587 Berlin, Germany
e-mail: *Sascha.Spors@telekom.de*

**Ann Spriet**
ESAT-SCD/SISTA, K.U. Leuven
Department of Electrical Engineering
Kasteelpark Arenberg 10
3001 Leuven, Belgium
e-mail: *ann.spriet@esat.kuleuven.be*

**Richard Sproat**
University of Illinois at Urbana-Champaign
Department of Linguistics
Urbana, IL 61801, USA
e-mail: *rws@uiuc.edu*

**Yannis Stylianou**
Institute of Computer Science
Heraklion, Crete 700 13, Greece
e-mail: *yannis@csd.uoc.gr*

**Jes Thyssen**
Broadcom Corporation
5300 California Avenue
Irvine, CA 92617, USA
e-mail: *jthyssen@broadcom.com*

**Jay Wilpon**
Research AT&T Labs
Voice and IP Services
Florham Park, NJ 07932, USA
e-mail: *jgw@research.att.com*

**Jan Wouters**
ExpORL, Department of Neurosciences, K.U. Leuven
O.& N2, Herestraat 49
3000 Leuven, Belgium
e-mail: *jan.wouters@med.kuleuven.be*

**Arie Yeredor**
Tel-Aviv University
Electrical Engineering - Systems
Tel-Aviv 69978, Israel
e-mail: *arie@eng.tau.ac.il*

**Steve Young**
Cambridge University Engineering Dept
Cambridge, CB21PZ, UK
e-mail: *sjy@eng.cam.ac.uk*

**Victor Zue**
Massachusetts Institute of Technology
CSAI Laboratory
32 Vassar Street
Cambridge, MA 02139, USA
e-mail: *zue@csail.mit.edu*

Amazon EX1010 Page 14

# Contents

**List of Abbreviations** ..... XXXI

1 **Introduction to Speech Processing**
*J. Benesty, M. M. Sondhi, Y. Huang* ..... 1
1.1 A Brief History of Speech Processing ..... 1
1.2 Applications of Speech Processing ..... 2
1.3 Organization of the Handbook ..... 4
**References** ..... 4

**Part A Production, Perception, and Modeling of Speech**

2 **Physiological Processes of Speech Production**
*K. Honda* ..... 7
2.1 Overview of Speech Apparatus ..... 7
2.2 Voice Production Mechanisms ..... 8
2.3 Articulatory Mechanisms ..... 14
2.4 Summary ..... 24
**References** ..... 25

3 **Nonlinear Cochlear Signal Processing and Masking in Speech Perception**
*J. B. Allen* ..... 27
3.1 Basics ..... 27
3.2 The Nonlinear Cochlea ..... 35
3.3 Neural Masking ..... 45
3.4 Discussion and Summary ..... 55
**References** ..... 56

4 **Perception of Speech and Sound**
*B. Kollmeier, T. Brand, B. Meyer* ..... 61
4.1 Basic Psychoacoustic Quantities ..... 62
4.2 Acoustical Information Required for Speech Perception ..... 70
4.3 Speech Feature Perception ..... 74
**References** ..... 81

5 **Speech Quality Assessment**
*V. Grancharov, W. B. Kleijn* ..... 83
5.1 Degradation Factors Affecting Speech Quality ..... 84
5.2 Subjective Tests ..... 85
5.3 Objective Measures ..... 90
5.4 Conclusions ..... 95
**References** ..... 96

## Part B Signal Processing for Speech


6 **Wiener and Adaptive Filters**
*J. Benesty, Y. Huang, J. Chen* ..... 103
6.1 Overview ..... 103
6.2 Signal Models ..... 104
6.3 Derivation of the Wiener Filter ..... 106
6.4 Impulse Response Tail Effect ..... 107
6.5 Condition Number ..... 108
6.6 Adaptive Algorithms ..... 110
6.7 MIMO Wiener Filter ..... 116
6.8 Conclusions ..... 119
**References** ..... 120

7 **Linear Prediction**
*J. Benesty, J. Chen, Y. Huang* ..... 121
7.1 Fundamentals ..... 121
7.2 Forward Linear Prediction ..... 122
7.3 Backward Linear Prediction ..... 123
7.4 Levinson–Durbin Algorithm ..... 124
7.5 Lattice Predictor ..... 126
7.6 Spectral Representation ..... 127
7.7 Linear Interpolation ..... 128
7.8 Line Spectrum Pair Representation ..... 129
7.9 Multichannel Linear Prediction ..... 130
7.10 Conclusions ..... 133
**References** ..... 133

8 **The Kalman Filter**
*S. Gannot, A. Yeredor* ..... 135
8.1 Derivation of the Kalman Filter ..... 136
8.2 Examples: Estimation of Parametric Stochastic Process from Noisy Observations ..... 141
8.3 Extensions of the Kalman Filter ..... 144
8.4 The Application of the Kalman Filter to Speech Processing ..... 149
8.5 Summary ..... 157
**References** ..... 157

9 **Homomorphic Systems and Cepstrum Analysis of Speech**
*R. W. Schafer* ..... 161
9.1 Definitions ..... 161
9.2 *Z*-Transform Analysis ..... 164
9.3 Discrete-Time Model for Speech Production ..... 165
9.4 The Cepstrum of Speech ..... 166
9.5 Relation to LPC ..... 169
9.6 Application to Pitch Detection ..... 171


Amazon EX1010 Page 16

9.7 Applications to Analysis/Synthesis Coding ..... 172
9.8 Applications to Speech Pattern Recognition ..... 176
9.9 Summary ..... 180
**References** ..... 180

10 **Pitch and Voicing Determination of Speech with an Extension Toward Music Signals**
*W. J. Hess* ..... 181
10.1 Pitch in Time-Variant Quasiperiodic Acoustic Signals ..... 182
10.2 Short-Term Analysis PDAs ..... 185
10.3 Selected Time-Domain Methods ..... 192
10.4 A Short Look into Voicing Determination ..... 195
10.5 Evaluation and Postprocessing ..... 197
10.6 Applications in Speech and Music ..... 201
10.7 Some New Challenges and Developments ..... 203
10.8 Concluding Remarks ..... 207
**References** ..... 208

11 **Formant Estimation and Tracking**
*D. O'Shaughnessy* ..... 213
11.1 Historical ..... 213
11.2 Vocal Tract Resonances ..... 215
11.3 Speech Production ..... 216
11.4 Acoustics of the Vocal Tract ..... 218
11.5 Short-Time Speech Analysis ..... 221
11.6 Formant Estimation ..... 223
11.7 Summary ..... 226
**References** ..... 226

12 **The STFT, Sinusoidal Models, and Speech Modification**
*M. M. Goodwin* ..... 229
12.1 The Short-Time Fourier Transform ..... 230
12.2 Sinusoidal Models ..... 242
12.3 Speech Modification ..... 253
**References** ..... 256

13 **Adaptive Blind Multichannel Identification**
*Y. Huang, J. Benesty, J. Chen* ..... 259
13.1 Overview ..... 259
13.2 Signal Model and Problem Formulation ..... 260
13.3 Identifiability and Principle ..... 261
13.4 Constrained Time-Domain Multichannel LMS and Newton Algorithms ..... 262
13.5 Unconstrained Multichannel LMS Algorithm with Optimal Step-Size Control ..... 266
13.6 Frequency-Domain Blind Multichannel Identification Algorithms ..... 268
13.7 Adaptive Multichannel Exponentiated Gradient Algorithm ..... 276

Amazon EX1010 Page 17

13.8 Summary ..... 279
**References** ..... 279

## Part C Speech Coding

### 14 Principles of Speech Coding
*W. B. Kleijn* ..... 283
14.1 The Objective of Speech Coding ..... 283
14.2 Speech Coder Attributes ..... 284
14.3 A Universal Coder for Speech ..... 286
14.4 Coding with Autoregressive Models ..... 293
14.5 Distortion Measures and Coding Architecture ..... 296
14.6 Summary ..... 302
**References** ..... 303

### 15 Voice over IP: Speech Transmission over Packet Networks
*J. Skoglund, E. Kozica, J. Linden, R. Hagen, W. B. Kleijn* ..... 307
15.1 Voice Communication ..... 307
15.2 Properties of the Network ..... 308
15.3 Outline of a VoIP System ..... 313
15.4 Robust Encoding ..... 317
15.5 Packet Loss Concealment ..... 326
15.6 Conclusion ..... 327
**References** ..... 328

### 16 Low-Bit-Rate Speech Coding
*A. V. McCree* ..... 331
16.1 Speech Coding ..... 331
16.2 Fundamentals: Parametric Modeling of Speech Signals ..... 332
16.3 Flexible Parametric Models ..... 337
16.4 Efficient Quantization of Model Parameters ..... 344
16.5 Low-Rate Speech Coding Standards ..... 345
16.6 Summary ..... 347
**References** ..... 347

### 17 Analysis-by-Synthesis Speech Coding
*J.-H. Chen, J. Thyssen* ..... 351
17.1 Overview ..... 352
17.2 Basic Concepts of Analysis-by-Synthesis Coding ..... 353
17.3 Overview of Prominent Analysis-by-Synthesis Speech Coders ..... 357
17.4 Multipulse Linear Predictive Coding (MPLPC) ..... 360
17.5 Regular-Pulse Excitation with Long-Term Prediction (RPE-LTP) ..... 362
17.6 The Original Code Excited Linear Prediction (CELP) Coder ..... 363
17.7 US Federal Standard FS1016 CELP ..... 367
17.8 Vector Sum Excited Linear Prediction (VSELP) ..... 368
17.9 Low-Delay CELP (LD-CELP) ..... 370

Amazon EX1010 Page 18

17.10 Pitch Synchronous Innovation CELP (PSI-CELP) ..... 371
17.11 Algebraic CELP (ACELP) ..... 371
17.12 Conjugate Structure CELP (CS-CELP) and CS-ACELP ..... 377
17.13 Relaxed CELP (RCELP) – Generalized Analysis by Synthesis ..... 378
17.14 eX-CELP ..... 381
17.15 iLBC ..... 382
17.16 TSNFC ..... 383
17.17 Embedded CELP ..... 386
17.18 Summary of Analysis-by-Synthesis Speech Coders ..... 388
17.19 Conclusion ..... 390
**References** ..... 390

18 **Perceptual Audio Coding of Speech Signals**
*J. Herre, M. Lutzky* ..... 393
18.1 History of Audio Coding ..... 393
18.2 Fundamentals of Perceptual Audio Coding ..... 394
18.3 Some Successful Standardized Audio Coders ..... 396
18.4 Perceptual Audio Coding for Real-Time Communication ..... 398
18.5 Hybrid/Crossover Coders ..... 403
18.6 Summary ..... 409
**References** ..... 409

**Part D Text-to-Speech Synthesis**

19 **Basic Principles of Speech Synthesis**
*J. Schroeter* ..... 413
19.1 The Basic Components of a TTS System ..... 413
19.2 Speech Representations and Signal Processing for Concatenative Synthesis ..... 421
19.3 Speech Signal Transformation Principles ..... 423
19.4 Speech Synthesis Evaluation ..... 425
19.5 Conclusions ..... 426
**References** ..... 426

20 **Rule-Based Speech Synthesis**
*R. Carlson, B. Granström* ..... 429
20.1 Background ..... 429
20.2 Terminal Analog ..... 429
20.3 Controlling the Synthesizer ..... 432
20.4 Special Applications of Rule-Based Parametric Synthesis ..... 434
20.5 Concluding Remarks ..... 434
**References** ..... 434

21 **Corpus-Based Speech Synthesis**
*T. Dutoit* ..... 437
21.1 Basics ..... 437

21.2 Concatenative Synthesis with a Fixed Inventory ..... 438
21.3 Unit-Selection-Based Synthesis ..... 447
21.4 Statistical Parametric Synthesis ..... 450
21.5 Conclusion ..... 453
**References** ..... 453

22 **Linguistic Processing for Speech Synthesis**
*R. Sproat* ..... 457
22.1 Why Linguistic Processing is Hard ..... 457
22.2 Fundamentals: Writing Systems and the Graphical Representation of Language ..... 457
22.3 Problems to be Solved and Methods to Solve Them ..... 458
22.4 Architectures for Multilingual Linguistic Processing ..... 465
22.5 Document-Level Processing ..... 465
22.6 Future Prospects ..... 466
**References** ..... 467

23 **Prosodic Processing**
*J. van Santen, T. Mishra, E. Klabbers* ..... 471
23.1 Overview ..... 471
23.2 Historical Overview ..... 475
23.3 Fundamental Challenges ..... 476
23.4 A Survey of Current Approaches ..... 477
23.5 Future Approaches ..... 484
23.6 Conclusions ..... 485
**References** ..... 485

24 **Voice Transformation**
*Y. Stylianou* ..... 489
24.1 Background ..... 489
24.2 Source–Filter Theory and Harmonic Models ..... 490
24.3 Definitions ..... 492
24.4 Source Modifications ..... 494
24.5 Filter Modifications ..... 498
24.6 Conversion Functions ..... 499
24.7 Voice Conversion ..... 500
24.8 Quality Issues in Voice Transformations ..... 501
24.9 Summary ..... 502
**References** ..... 502

25 **Expressive/Affective Speech Synthesis**
*N. Campbell* ..... 505
25.1 Overview ..... 505
25.2 Characteristics of Affective Speech ..... 506
25.3 The Communicative Functionality of Speech ..... 508
25.4 Approaches to Synthesizing Expressive Speech ..... 510
25.5 Modeling Human Speech ..... 512

Amazon EX1010 Page 20

25.6 Conclusion ..... 515
**References** ..... 515

## Part E Speech Recognition

26 **Historical Perspective of the Field of ASR/NLU**
*L. Rabiner, B.-H. Juang* ..... 521
26.1 ASR Methodologies ..... 521
26.2 Important Milestones in Speech Recognition History ..... 523
26.3 Generation 1 – The Early History of Speech Recognition ..... 524
26.4 Generation 2 – The First Working Systems for Speech Recognition ..... 524
26.5 Generation 3 – The Pattern Recognition Approach to Speech Recognition ..... 525
26.6 Generation 4 – The Era of the Statistical Model ..... 530
26.7 Generation 5 – The Future ..... 534
26.8 Summary ..... 534
**References** ..... 535

27 **HMMs and Related Speech Recognition Technologies**
*S. Young* ..... 539
27.1 Basic Framework ..... 539
27.2 Architecture of an HMM-Based Recognizer ..... 540
27.3 HMM-Based Acoustic Modeling ..... 547
27.4 Normalization ..... 550
27.5 Adaptation ..... 551
27.6 Multipass Recognition Architectures ..... 554
27.7 Conclusions ..... 554
**References** ..... 555

28 **Speech Recognition with Weighted Finite-State Transducers**
*M. Mohri, F. Pereira, M. Riley* ..... 559
28.1 Definitions ..... 559
28.2 Overview ..... 560
28.3 Algorithms ..... 567
28.4 Applications to Speech Recognition ..... 574
28.5 Conclusion ..... 582
**References** ..... 582

29 **A Machine Learning Framework for Spoken-Dialog Classification**
*C. Cortes, P. Haffner, M. Mohri* ..... 585
29.1 Motivation ..... 585
29.2 Introduction to Kernel Methods ..... 586
29.3 Rational Kernels ..... 587
29.4 Algorithms ..... 589
29.5 Experiments ..... 591
29.6 Theoretical Results for Rational Kernels ..... 593

Amazon EX1010 Page 21

29.7 Conclusion ..... 594
**References** ..... 595

30 **Towards Superhuman Speech Recognition**
*M. Picheny, D. Nahamoo* ..... 597
30.1 Current Status ..... 597
30.2 A Multidomain Conversational Test Set ..... 598
30.3 Listening Experiments ..... 599
30.4 Recognition Experiments ..... 601
30.5 Speculation ..... 607
**References** ..... 614

31 **Natural Language Understanding**
*S. Roukos* ..... 617
31.1 Overview of NLU Applications ..... 618
31.2 Natural Language Parsing ..... 620
31.3 Practical Implementation ..... 623
31.4 Speech Mining ..... 623
31.5 Conclusion ..... 625
**References** ..... 626

32 **Transcription and Distillation of Spontaneous Speech**
*S. Furui, T. Kawahara* ..... 627
32.1 Background ..... 627
32.2 Overview of Research Activities on Spontaneous Speech ..... 628
32.3 Analysis for Spontaneous Speech Recognition ..... 632
32.4 Approaches to Spontaneous Speech Recognition ..... 635
32.5 Metadata and Structure Extraction of Spontaneous Speech ..... 640
32.6 Speech Summarization ..... 644
32.7 Conclusions ..... 647
**References** ..... 647

33 **Environmental Robustness**
*J. Droppo, A. Acero* ..... 653
33.1 Noise Robust Speech Recognition ..... 653
33.2 Model Retraining and Adaptation ..... 656
33.3 Feature Transformation and Normalization ..... 657
33.4 A Model of the Environment ..... 664
33.5 Structured Model Adaptation ..... 667
33.6 Structured Feature Enhancement ..... 671
33.7 Unifying Model and Feature Techniques ..... 675
33.8 Conclusion ..... 677
**References** ..... 677

34 **The Business of Speech Technologies**
*J. Wilpon, M. E. Gilbert, J. Cohen* ..... 681
34.1 Introduction ..... 682

Amazon EX1010 Page 22

34.2 Network-Based Speech Services ..... 686
34.3 Device-Based Speech Applications ..... 692
34.4 Vision/Predications of Future Services – Fueling the Trends ..... 697
34.5 Conclusion ..... 701
**References** ..... 702

35 **Spoken Dialogue Systems**
*V. Zue, S. Seneff* ..... 705
35.1 Technology Components and System Development ..... 707
35.2 Development Issues ..... 712
35.3 Historical Perspectives ..... 714
35.4 New Directions ..... 715
35.5 Concluding Remarks ..... 718
**References** ..... 718

**Part F Speaker Recognition**

36 **Overview of Speaker Recognition**
*A. E. Rosenberg, F. Bimbot, S. Parthasarathy* ..... 725
36.1 Speaker Recognition ..... 725
36.2 Measuring Speaker Features ..... 729
36.3 Constructing Speaker Models ..... 731
36.4 Adaptation ..... 735
36.5 Decision and Performance ..... 735
36.6 Selected Applications for Automatic Speaker Recognition ..... 737
36.7 Summary ..... 739
**References** ..... 739

37 **Text-Dependent Speaker Recognition**
*M. Hébert* ..... 743
37.1 Brief Overview ..... 743
37.2 Text-Dependent Challenges ..... 747
37.3 Selected Results ..... 750
37.4 Concluding Remarks ..... 760
**References** ..... 760

38 **Text-Independent Speaker Recognition**
*D. A. Reynolds, W. M. Campbell* ..... 763
38.1 Introduction ..... 763
38.2 Likelihood Ratio Detector ..... 764
38.3 Features ..... 766
38.4 Classifiers ..... 767
38.5 Performance Assessment ..... 776
38.6 Summary ..... 778
**References** ..... 779

## Part G Language Recognition


39 **Principles of Spoken Language Recognition**
*C.-H. Lee* ..... 785
39.1 Spoken Language ..... 785
39.2 Language Recognition Principles ..... 786
39.3 Phone Recognition Followed by Language Modeling (PRLM) ..... 788
39.4 Vector-Space Characterization (VSC) ..... 789
39.5 Spoken Language Verification ..... 790
39.6 Discriminative Classifier Design ..... 791
39.7 Summary ..... 793
**References** ..... 793

40 **Spoken Language Characterization**
*M. P. Harper, M. Maxwell* ..... 797
40.1 Language versus Dialect ..... 798
40.2 Spoken Language Collections ..... 800
40.3 Spoken Language Characteristics ..... 800
40.4 Human Language Identification ..... 804
40.5 Text as a Source of Information on Spoken Languages ..... 806
40.6 Summary ..... 807
**References** ..... 807

41 **Automatic Language Recognition Via Spectral and Token Based Approaches**
*D. A. Reynolds, W. M. Campbell, W. Shen, E. Singer* ..... 811
41.1 Automatic Language Recognition ..... 811
41.2 Spectral Based Methods ..... 812
41.3 Token-Based Methods ..... 815
41.4 System Fusion ..... 818
41.5 Performance Assessment ..... 820
41.6 Summary ..... 823
**References** ..... 823

42 **Vector-Based Spoken Language Classification**
*H. Li, B. Ma, C.-H. Lee* ..... 825
42.1 Vector Space Characterization ..... 826
42.2 Unit Selection and Modeling ..... 827
42.3 Front-End: Voice Tokenization and Spoken Document Vectorization 830
42.4 Back-End: Vector-Based Classifier Design ..... 831
42.5 Language Classification Experiments and Discussion ..... 835
42.6 Summary ..... 838
**References** ..... 839


Amazon EX1010 Page 24

## Part H Speech Enhancement


43 **Fundamentals of Noise Reduction**
*J. Chen, J. Benesty, Y. Huang, E. J. Diethorn* ..... 843
43.1 Noise ..... 843
43.2 Signal Model and Problem Formulation ..... 845
43.3 Evaluation of Noise Reduction ..... 846
43.4 Noise Reduction via Filtering Techniques ..... 847
43.5 Noise Reduction via Spectral Restoration ..... 857
43.6 Speech-Model-Based Noise Reduction ..... 863
43.7 Summary ..... 868
**References** ..... 869

44 **Spectral Enhancement Methods**
*I. Cohen, S. Gannot* ..... 873
44.1 Spectral Enhancement ..... 874
44.2 Problem Formulation ..... 875
44.3 Statistical Models ..... 876
44.4 Signal Estimation ..... 879
44.5 Signal Presence Probability Estimation ..... 881
44.6 A Priori SNR Estimation ..... 882
44.7 Noise Spectrum Estimation ..... 888
44.8 Summary of a Spectral Enhancement Algorithm ..... 891
44.9 Selection of Spectral Enhancement Algorithms ..... 896
44.10 Conclusions ..... 898
**References** ..... 899

45 **Adaptive Echo Cancelation for Voice Signals**
*M. M. Sondhi* ..... 903
45.1 Network Echoes ..... 904
45.2 Single-Channel Acoustic Echo Cancelation ..... 915
45.3 Multichannel Acoustic Echo Cancelation ..... 921
45.4 Summary ..... 925
**References** ..... 926

46 **Dereverberation**
*Y. Huang, J. Benesty, J. Chen* ..... 929
46.1 Background and Overview ..... 929
46.2 Signal Model and Problem Formulation ..... 931
46.3 Source Model-Based Speech Dereverberation ..... 932
46.4 Separation of Speech and Reverberation via Homomorphic Transformation ..... 936
46.5 Channel Inversion and Equalization ..... 937
46.6 Summary ..... 941
**References** ..... 942


Amazon EX1010 Page 25

47 **Adaptive Beamforming and Postfiltering**
*S. Gannot, I. Cohen* ..... 945
47.1 Problem Formulation ..... 947
47.2 Adaptive Beamforming ..... 948
47.3 Fixed Beamformer and Blocking Matrix ..... 953
47.4 Identification of the Acoustical Transfer Function ..... 955
47.5 Robustness and Distortion Weighting ..... 960
47.6 Multichannel Postfiltering ..... 962
47.7 Performance Analysis ..... 967
47.8 Experimental Results ..... 972
47.9 Summary ..... 972
47.A Appendix: Derivation of the Expected Noise Reduction for a Coherent Noise Field ..... 973
47.B Appendix: Equivalence Between Maximum SNR and LCMV Beamformers ..... 974
**References** ..... 975

48 **Feedback Control in Hearing Aids**
*A. Spriet, S. Doclo, M. Moonen, J. Wouters* ..... 979
48.1 Problem Statement ..... 980
48.2 Standard Adaptive Feedback Canceller ..... 982
48.3 Feedback Cancellation Based on Prior Knowledge of the Acoustic Feedback Path ..... 986
48.4 Feedback Cancellation Based on Closed-Loop System Identification ..... 990
48.5 Comparison ..... 995
48.6 Conclusions ..... 997
**References** ..... 997

49 **Active Noise Control**
*S. M. Kuo, D. R. Morgan* ..... 1001
49.1 Broadband Feedforward Active Noise Control ..... 1002
49.2 Narrowband Feedforward Active Noise Control ..... 1006
49.3 Feedback Active Noise Control ..... 1010
49.4 Multichannel ANC ..... 1011
49.5 Summary ..... 1015
**References** ..... 1015

## Part I Multichannel Speech Processing

50 **Microphone Arrays**
*G. W. Elko, J. Meyer* ..... 1021
50.1 Microphone Array Beamforming ..... 1021
50.2 Constant-Beamwidth Microphone Array System ..... 1029
50.3 Constrained Optimization of the Directional Gain ..... 1030
50.4 Differential Microphone Arrays ..... 1031
50.5 Eigenbeamforming Arrays ..... 1034

Amazon EX1010 Page 26

50.6 Adaptive Array Systems ..... 1037
50.7 Conclusions ..... 1040
**References** ..... 1040

51 **Time Delay Estimation and Source Localization**
*Y. Huang, J. Benesty, J. Chen* ..... 1043
51.1 Technology Taxonomy ..... 1043
51.2 Time Delay Estimation ..... 1044
51.3 Source Localization ..... 1054
51.4 Summary ..... 1061
**References** ..... 1062

52 **Convolutive Blind Source Separation Methods**
*M. S. Pedersen, J. Larsen, U. Kjems, L. C. Parra* ..... 1065
52.1 The Mixing Model ..... 1066
52.2 The Separation Model ..... 1068
52.3 Identification ..... 1071
52.4 Separation Principle ..... 1071
52.5 Time Versus Frequency Domain ..... 1076
52.6 The Permutation Ambiguity ..... 1078
52.7 Results ..... 1084
52.8 Conclusion ..... 1084
**References** ..... 1084

53 **Sound Field Reproduction**
*R. Rabenstein, S. Spors* ..... 1095
53.1 Sound Field Synthesis ..... 1095
53.2 Mathematical Representation of Sound Fields ..... 1096
53.3 Stereophony ..... 1100
53.4 Vector-Based Amplitude Panning ..... 1103
53.5 Ambisonics ..... 1104
53.6 Wave Field Synthesis ..... 1109
**References** ..... 1113

**Acknowledgements** ..... 1115
**About the Authors** ..... 1117
**Detailed Contents** ..... 1133
**Subject Index** ..... 1161

Amazon EX1010 Page 27

# List of Abbreviations

| | |
|---|---|
| 2TS | two-tone suppression |

## A

| | |
|---|---|
| ACELP | algebraic code excited linear prediction |
| ACF | autocorrelation function |
| ACR | absolute category rating |
| ACS | autocorrelation coefficient sequences |
| ACeS | Asia Cellular Satellite |
| ADC | analog-to-digital converter |
| ADPCM | adaptive differential pulse code modulation |
| AEC | acoustic echo cancelation |
| AFE | advanced front-end |
| AGC | automatic gain control |
| AGN | automatic gain normalization |
| AL | averaged acoustic frame likelihood |
| AMR-WB+ | extended wide-band adaptive multirate coder |
| AMR-WB | wide-band AMR speech coder |
| AMSC-TMI | American Mobile Satellite Corporation Telesat Mobile Incorporated |
| AN | auditory nerve |
| ANC | active noise cancelation |
| ANN | artificial neural networks |
| ANOVA | analysis of variance |
| APA | affine projection algorithm |
| APC | adaptive predictive coding |
| APCO | Association of Public-Safety Communications Officials |
| APP | adjusted test-set perplexity |
| AR | autoregressive |
| ARISE | Automatic Railway Information Systems for Europe |
| ARMA | autoregressive moving-average |
| ARPA | Advanced Research Projects Agency |
| ARQ | automatic repeat request |
| ART | advanced recognition technology |
| ASAT | automatic speech attribute transcription |
| ASM | acoustic segment model |
| ASR | automatic speech recognition |
| ATF | acoustical transfer function |
| ATIS | airline travel information system |
| ATN | augmented transition networks |
| ATR | advanced telecommunications research |
| AW | acoustic word |

## B

| | |
|---|---|
| BBN | Bolt, Beranek and Newman |
| BIC | Bayesian information criterion |
| BILD | binaural intelligibility level difference |
| BM | blocking matrix |
| BN | broadcast news |
| BSD | bark spectral distortion |
| BSS | blind source separation |

## C

| | |
|---|---|
| C | consonants |
| CA | cochlear amplifier |
| CAF | continuous adaptation feedback |
| CART | classification and regression tree |
| CASA | computational auditory scene analysis |
| CAT | cluster adaptive training |
| CCR | comparison category rating |
| CDCN | codeword-dependent cepstral normalization |
| CDF | cumulative distribution function |
| CDMA | code division multiple access |
| CE | categorical estimation |
| CELP | code-excited linear prediction |
| CF | characteristic frequency |
| CF | coherence function |
| CH | call home |
| CHN | cepstral histogram normalization |
| CIS | caller identification system |
| CMLLR | constrained MLLR |
| CMOS | comparison mean opinion score |
| CMR | co-modulation masking release |
| CMS | cepstral mean subtraction |
| CMU | Carnegie Mellon University |
| CMVN | cepstral mean and variance normalization |
| CNG | comfort noise generation |
| COC | context-oriented clustering |
| CR | cross-relation |
| CRF | conditional random fields |
| CRLB | Cramèr–Rao lower bound |
| CS-ACELP | conjugate structure ACELP |
| CS-CELP | conjugate structure CELP |
| CSJ | corpus of spontaneous Japanese |
| CSR | continuous speech recognition |
| CTS | conversational telephone speech |
| CVC | consonant–vowel–consonant |
| CVN | cepstral variance normalization |
| CZT | chirp z-transform |

## D

| | |
|---|---|
| DAC | digital-to-analog |
| DAG | directed acyclic graph |
| DAM | diagnostic acceptability measure |

| | |
|---|---|
| DARPA | Defense Advanced Research Projects Agency |
| DBN | dynamic Bayesian network |
| DCF | detection cost function |
| DCR | degradation category rating |
| DCT | discrete cosine transform |
| DET | detection error tradeoff |
| DF | disfluency |
| DFA | deterministic finite automata |
| DFT | discrete Fourier transform |
| DFW | dynamic frequency warping |
| DM | dialog management |
| DMOS | degradation mean opinion score |
| DP | dynamic programming |
| DPCM | differential PCM |
| DPMC | data-driven parallel model combination |
| DRT | diagnostic rhyme test |
| DSP | digital signal processing |
| DT | discriminative training |
| DTFT | discrete-time Fourier transform |
| DTW | dynamic time warping |
| DoD | Department of Defense |

## E

| | |
|---|---|
| EC | equalization and cancelation |
| ECOC | error-correcting output coding |
| EER | equal error rate |
| EGG | electroglottography |
| EKF | extended Kalman filter |
| ELER | early-to-late energy ratio |
| EM | estimate–maximize |
| EM | expectation maximization |
| EMG | electromyographic |
| EMLLT | extended maximum likelihood linear transform |
| ER AAC-LD | error resilient low-delay advanced audio |
| ERB | equivalent rectangular bandwidth |
| ERL | echo return loss |
| ERLE | echo return loss enhancement |
| EVRC | enhanced variable rate coder |
| eX-CELP | extended CELP |

## F

| | |
|---|---|
| FA | false accept |
| FAP | fast affine projection |
| FB-LPC | forward backward linear predictive coding |
| FBF | fixed beamformer |
| FBS | filter bank summation |
| FC | functional contour |
| FCDT | frame-count-dependent thresholding |
| FEC | frame erasure concealment |
| FFT | fast Fourier transform |
| FIFO | first-in first-out |
| FIR | finite impulse response |
| FM | forward masking |
| FMLLR | maximum-likelihood feature-space regression |
| FR | filler rate |
| FRC | functional residual capacity |
| FRLS | fast recursive least-squares |
| FSM | finite state machine |
| FSN | finite state network |
| FSS | frequency selective switch |
| FST | finite state transducer |
| FT | Fourier transform |
| FTF | fast transversal filter |
| FVQ | fuzzy vector quantization |
| FXLMS | filtered-X LMS |

## G

| | |
|---|---|
| GCC | generalized cross-correlation |
| GCI | glottal closure instant |
| GEVD | generalized eigenvalue decomposition |
| GLDS | generalized linear discriminant sequence |
| GLR | generalized likelihood ratio |
| GMM | Gaussian mixture model |
| GPD | generalized probabilistic descent |
| GSC | generalized sidelobe canceller |
| GSM | Groupe Spéciale Mobile |
| GSV | GMM supervector |

## H

| | |
|---|---|
| HLDA | heteroscedastic LDA |
| HLT | human language technologies |
| HMIHY | How May I Help You |
| HMM | hidden Markov models |
| HMP | hidden Markov processes |
| HNM | harmonic-plus-noise model |
| HOS | higher-order statistics |
| HPF | high-pass filter |
| HRTF | head-related transfer function |
| HSD | honestly significant difference |
| HSR | human speech recognition |
| HTK | hidden Markov model toolkit |

## I

| | |
|---|---|
| IAI | International Association for Identification |
| ICA | independent component analysis |
| IDA | Institute for Defense Analyses |
| IDFT | inverse DFT |
| IDTFT | inverse discrete-time Fourier transform |
| IETF | Internet Engineering Task Force |
| IFSS | inverse frequency selective switch |
| IHC | inner hair cells |
| II | information index |

| | |
|---|---|
| IIR | infinite impulse response |
| iLBC | internet low-bit-rate codec |
| IMCRA | improved minima-controlled recursive averaging |
| IMDCT | inverse MDCT |
| IMM | interacting multiple model |
| IO | input–output |
| IP | internet protocol |
| IP | interruption (disfluent) point |
| IPA | International Phonetic Alphabet |
| IPNLMS | improved PNLMS |
| IQMF | QMF synthesis filterbank |
| IR | information retrieval |
| IS | Itakura–Saito |
| ISU | information state update |
| ITU | International Telecommunication Union |
| IVR | interactive voice response |

## J

| | |
|---|---|
| JADE | joint approximate diagonalization of eigenmatrices |
| JND | just-noticeable difference |

## K

| | |
|---|---|
| KL | Kullbach–Leibler |
| KLT | Karhunen–Loève transform |

## L

| | |
|---|---|
| LAN | local-area network |
| LAR | log-area-ratio |
| LCMV | linearly constrained minimum-variance |
| LD-CELP | low-delay CELP |
| LDA | linear discriminant analysis |
| LDC | Linguistic Data Consortium |
| LDF | linear discriminant function |
| LID | language identification |
| LL | log-likelihood |
| LLAMA | learning library for large-margin classification |
| LLR | (log) likelihood ratio |
| LMFB | log mel-frequency filterbank |
| LMR | linear multivariate regression |
| LMS | least mean square |
| LNRE | large number of rare events |
| LP | linear prediction |
| LPC | linear prediction coefficients |
| LPC | linear predictive coding |
| LPCC | linear predictive cepstral coefficient |
| LRE | language recognition evaluation |
| LRT | likelihood-ratio test |
| LSA | latent semantic analysis |
| LSA | log-spectral amplitude |
| LSF | line spectral frequency |
| LSI | latent semantic indexing |
| LSR | low sampling rates |
| LTI | linear time invariant |
| LTIC | linear time-invariant causal |
| LTP | long term prediction |
| LVCSR | large vocabulary continuous speech recognition |

## M

| | |
|---|---|
| M-step | maximization stage |
| MA | moving average |
| MAP | maximum a posteriori |
| MBE | multiband excited |
| MBR | minimum Bayes-risk |
| MBROLA | multiband resynthesis overlap-add |
| MC | multicategory |
| MCE | minimum classification error |
| MCN | multichannel Newton |
| MDC | multiple description coding |
| MDCT | modified discrete cosine transform |
| MDF | multidelay filter |
| MDP | Markov decision process |
| MELP | mixed excitation linear prediction |
| MFCC | mel-filter cepstral coefficient |
| MFoM | maximal figure-of-merit |
| MIMO | multiple-input multiple-output |
| MIPS | million instructions per second |
| ML | maximum-likelihood |
| MLLR | maximum-likelihood linear regression |
| MLP | multilayer perceptron |
| MMI | maximum mutual information |
| MMSE-LSA | MMSE of the log-spectral amplitude |
| MMSE-SA | MMSE of the spectral amplitude |
| MMSE | minimum mean-square error |
| MNB | measuring normalizing blocks |
| MNRU | modulated noise reference unit |
| MOPS | million operations per second |
| MOS | mean opinion score |
| MP | matching pursuit |
| MPE | minimum phone error |
| MPEG | Moving Pictures Expert Ggroup |
| MPI | minimal pairs intelligibility |
| MPLPC | multipulse linear predictive coding |
| MRI | magnetic resonance imaging |
| MRT | modified rhyme test |
| MS | minimum statistics |
| MSA | modern standard Arabic |
| MSD | minimum significant difference |
| MSE | mean-square error |
| MSG | maximum stable gain |
| MSNR | maximum signal-to-noise ratio |
| MSVQ | multistage VQ |
| MUI | multimodal user interface |
| MUSHRA | multi stimulus test with hidden reference and anchor |

| | |
|---|---|
| MVE | minimum verification error |
| MVIMP | my voice is my password |
| MW | maximum wins |
| MuSIC | multiple signal classification |

## N

| | |
|---|---|
| NAB | North American Business News |
| NASA | National Aeronautics and Space Administration |
| NATO | North Atlantic Treaty Organization |
| NFA | nondeterministic finite automata |
| NFC | noise feedback coding |
| NIST | National Institute of Standards and Technology |
| NLG | natural language generation |
| NLMS | normalized least-mean-square |
| NLP | natural language processing |
| NLU | natural language understanding |
| NN | neural network |
| NSA | National Security Agency |
| NTT | Nippon Telephone & Telegraph |
| NUU | nonuniform units |

## O

| | |
|---|---|
| O | object |
| OAE | otoacoustic emissions |
| OHC | outer-hair cells |
| OLA | overlap-and-add |
| OOV | out-of-vocabulary |
| OQ | open quotient |
| OR | out-of-vocabulary rate |
| OSI | open systems interconnection reference |

## P

| | |
|---|---|
| P-PRLM | parallel PRLM |
| PARCOR | partial correlation coefficients |
| PBFD | partitioned-block frequency-domain |
| PCA | principal component analysis |
| PCBV | phonetic-class-based verification |
| PCFG | probabilistic context-free grammar |
| PCM | pulse-code modulation |
| PDA | pitch determination algorithms |
| PDC | personal digital cellular |
| PDF | probability density function |
| PDP | parallel distributed processing |
| PDS | positive-definite symmetric |
| PEAQ | perceptual quality assessment for digital audio |
| PESQ | perceptual evaluation of speech quality |
| PGG | photoglottography |
| PICOLA | pointer interval controlled overlap and add |
| PIV | particle image velocity |
| PLC | packet loss concealment |
| PLP | perceptual linear prediction |
| PMC | parallel model combination |
| PNLMS | proportionate NLMS |
| POS | part-of-speech |
| PP | word perplexity |
| PPRLM | parallel PRLM |
| PR | phone recognizer |
| PRA | partial-rank algorithm |
| PRLM | phoneme recognition followed by language modeling |
| PSD | power spectral density |
| PSI-CELP | pitch synchronous innovation CELP |
| PSI | pitch synchronous innovation |
| PSQM | perceptual speech quality measure |
| PSRELP | pitch-synchronous residual excited linear prediction |
| PSTN | public switched telephone network |
| PVT | parallel voice tokenization |

## Q

| | |
|---|---|
| QA | question answering |
| QMF | quadrature mirror filter |
| QP | quadratic program |
| QoS | quality-of-service |

## R

| | |
|---|---|
| RD | rate-distortion |
| RASTA | relative spectra |
| RATZ | multivariate Gaussian-based cepstral normalization |
| RCELP | relaxed CELP |
| REW | rapidly evolving waveform |
| RF | radio frequency |
| RIM | repair interval model |
| RIR | room impulse response |
| RL | reticular lamina |
| RLS | recursive least-squares |
| RM | resource management |
| RMS | root mean square |
| RMSE | root-mean-square error |
| ROC | receiver operating characteristic |
| RPA | raw phone accuracy |
| RPD | point of the reparandum |
| RPE-LTP | regular-pulse excitation with long-term prediction |
| RR | reprompt rates |
| RS | Reed–Solomon |
| RSVP | resource reservation protocol |
| RT | rich transcription |

| | |
|---|---|
| RTM | resonant tectorial membrane |
| RTP | real-time transport protocol |

## S

| | |
|---|---|
| S2S | Speech-to-speech |
| S | subject |
| SAT | speaker-adaptive trained |
| SB | switchboard |
| SCTE | Society of Cable Telecommunications Engineers |
| SD-CFG | stochastic dependency context-free grammar |
| SD | spectral distortion |
| SDC | shifted delta cepstral |
| SDR-GSC | speech distortion regularized generalized sidelobe canceller |
| SDW-MWF | speech distortion-weighted multichannel Wiener filter |
| SEW | slowly evolving waveform |
| SGML | standard generalized markup language |
| SI | speech intelligibility |
| SII | speech intelligibility index |
| SIMO | single-input multiple-output |
| SISO | single-input single-output |
| SL | sensation level |
| SLM | statistical language model |
| SLS | spoken language system |
| SM | sinusoidal models |
| SMS | speaker model synthesis |
| SMT | statistical machine translation |
| SMV | selectable mode vocoder |
| SNR | signal-to-noise ratio |
| SOLA | synchronized overlap add |
| SOS | second-order statistics |
| SPAM | subspace-constrained precision and means |
| SPIN | saturated Poisson internal noise |
| SPINE | speech in noisy environment |
| SPL | sound pressure level |
| SPLICE | stereo piecewise linear compensation for environment |
| SQ | speed quotient |
| SR | speaking rate |
| SRT | speech reception threshold |
| SSML | speech synthesis markup language |
| STC | sinusoidal transform coder |
| STFT | short-time Fourier transform |
| SU | sentence unit |
| SUI | speech user interface |
| SUNDIAL | speech understanding and dialog |
| SUR | Speech Understanding Research |
| SVD | singular value decomposition |
| SVM | support vector machines |
| SWB | switchboard |
| SegSNR | segmental SNR |

## T

| | |
|---|---|
| T–F | time–frequency |
| TBRR | transient beam-to-reference ratio |
| TC | text categorization |
| TCP | transmission control protocol |
| TCX | transform coded excitation |
| TD-PSOLA | time-domain pitch-synchronous overlap-add |
| TD | time domain |
| TDAC | time-domain aliasing cancelation |
| TDBWE | time-domain bandwidth extension |
| TDMA | time-division multiple-access |
| TDOA | time difference of arrival |
| TDT | topic detection and tracking |
| TF-GSC | transfer-function generalized sidelobe canceller |
| TFIDF | term frequency inverse document frequency |
| TFLLR | term frequency log-likelihood ratio |
| TFLOG | term frequency logarithmic |
| TI | transinformation index |
| TIA | Telecommunications Industry Association |
| TITO | two-input-two-output |
| TM | tectorial membrane |
| TM | tympanic membrane |
| TMJ | temporomandibular joint |
| TNS | temporal noise shaping |
| TPC | transform predictive coder |
| TSNFC | two-stage noise feedback coding |
| TTS | text-to-speech |
| ToBI | tone and break indices |

## U

| | |
|---|---|
| UBM | universal background model |
| UCD | unit concatenative distortion |
| UDP | user datagram protocol |
| UE | user experience |
| UKF | unscented Kalman filter |
| ULD | ultra-low delay |
| USD | unit segmental distortion |
| USM | upward spread of masking |
| UT | unscented transform |
| UVT | universal voice tokenization |

## V

| | |
|---|---|
| V | verb |
| V | vowels |
| VAD | voice activity detector |
| VBAP | vector based amplitude panning |
| VCV | vowel–consonant–vowel |
| VLSI | very large-scale integration |

| | |
|---|---|
| VMR-WB | variable-rate multimode wide-band |
| VOT | voice onset time |
| VQ | vector quantization |
| VRCP | voice recognition call processing |
| VRS | variable rate smoothing |
| VSC | vector space characterization |
| VSELP | vector sum excited linear prediction |
| VT | voice tokenization |
| VTLN | vocal-tract-length normalization |
| VTR | vocal tract resonance |
| VTS | vector Taylor-series |
| VoIP | voice over IP |

## W

| | |
|---|---|
| WDRC | wide dynamic-range multiband compression |
| WER | word error rate |
| WFSA | weighted finite-state acceptors |
| WFST | weighted finite-state transducer |
| WGN | white Gaussian noise |
| WI | waveform interpolation |
| WLAN | wireless LAN |
| WLS | weighted least-squares |
| WMOPS | weighted MOPS |
| WSJ | Wall Street Journal |
| WSOLA | waveform similarity OLA |
| WiFi | wireless fidelity |

## X

| | |
|---|---|
| XML | extensible mark-up languages |
| XOR | exclusive-or |

## Z

| | |
|---|---|
| ZC | zero crossing |
| ZIR | zero-input response |
| ZSR | zero-state response |

# 1. Introduction to Speech Processing

J. Benesty, M. M. Sondhi, Y. Huang

In this brief introduction we outline some major highlights in the history of speech processing. We briefly describe some of the important applications of speech processing. Finally, we introduce the reader to the various parts of this handbook.

1.1 A Brief History of Speech Processing ..... 1
1.2 Applications of Speech Processing ..... 2
1.3 Organization of the Handbook ..... 4
References ..... 4

## 1.1 A Brief History of Speech Processing

Human beings have long been motivated to create machines that can talk. Early attempts at understanding speech production consisted of building mechanical models to mimic the human vocal apparatus. Two such examples date back to the 13th century, when the German philosopher Albertus Magnus and the English scientist Roger Bacon are reputed to have constructed metal talking heads. However, no documentation of these devices is known to exist. The first documented attempts at making speaking machines came some five hundred years later. In 1769 Kratzenstein constructed resonant cavities which, when he excited them by a vibrating reed, produced the sounds of the five vowels a, e, i, o, and u. Around the same time, and independently of this work, Wolfgang von Kempelen constructed a mechanical speech synthesizer that could generate recognizable consonants, vowels, and some connected utterances. His book on his research, published in 1791, may be regarded as marking the beginnings of speech processing. Some 40 years later, Charles Wheatstone constructed a machine based essentially on von Kempelen's specifications [1.1–3].

Interest in mechanical analogs of the human vocal apparatus continued well into the 20th century. Mimics of the type of von Kempelen's machine were constructed by several people besides Wheatstone, e.g., Joseph Faber, Richard Paget, R. R. Riesz, et al.

It is known that as a young man Alexander Graham Bell had the opportunity to see Wheatstone's implementation. He too made a speaking machine of that general nature. However, it was his other invention – the telephone – that provided a major impetus to modern speech processing. Nobody could have guessed at that time the impact the telephone would have, not only on the way people communicate with each other but also on research in speech processing as a science in its own right. The availability of the speech waveform as an electrical signal shifted interest from mechanical to electrical machines for synthesizing and processing speech.

Some attempts were made in the 1920s and 1930s to synthesize speech electrically. However it is Homer Dudley's work in the 1930s that ushered in the modern era of speech processing. His most important contribution was the clear understanding of the *carrier* nature of speech [1.4]. He developed the analogy between speech signals and modulated-carrier radio signals that are used, for instance, for the transmission or broadcast of audio signals. In the case of the radio broadcast, the message to be transmitted is the audio signal which has frequencies in the range of 0–20 kHz. Analogously, the message to be transmitted in the case of speech is carried mainly by the time-varying shape of the vocal tract, which in turn is a representation of the *thoughts* the speaker wishes to convey to the listener. The movements of the vocal tract are at syllabic rates, i. e., at frequencies between 0 and 20 Hz. In each case – electromagnetic and acoustic – the message is in a frequency range unsuitable for transmission. The solution in each case is to imprint the message on a carrier. In the electromagnetic case the carrier is usually a high-frequency sinusoidal wave. In the acoustic case the carrier can be one of several signals. It is the quasi periodic signal provided by the vocal cords for voiced speech, and a noise-like signal provided by turbulence at a constriction for fricative and aspirated sounds. Or it can be a combination of these for voiced fricative sounds. Indeed, the selection of the carrier as well as the changes in intensity and fundamental frequency of the

vocal cords may be conveniently regarded as additional parts of the message.

Being an electrical engineer himself, Dudley proceeded to exploit this insight to construct an *electrical* speech synthesizer which dispensed with all the mechanical devices of von Kempelen's machine. Electrical circuits were used to generate the carriers. And the message (i. e., the characteristics of the vocal tract) was imprinted on the carrier by passing it through a time-varying filter whose frequency response was adjusted to simulate the transfer characteristics of the vocal tract.

With the collaboration of Riesz and Watkins, Dudley implemented two highly acclaimed devices based on this principle – the Voder and the Vocoder. The Voder was the first versatile talking machine able to produce arbitrary sentences. It was a system in which an operator manipulated a keyboard to control the sound source and the filter bank. This system was displayed with great success at the New York World Fair of 1939. It could produce speech of much better quality than had been possible with the mechanical devices, but remained essentially a curiosity. The Vocoder, on the other hand had a much more serious purpose. It was the first attempt at *compressing* speech. Dudley estimated that since the message in a speech signal is carried by the slowly time-varying filters, it should be possible to send adequate information for the receiver to be able to reconstruct a telephone speech signal using a bandwidth of only about 150 Hz, which is about 1/20 the bandwidth required to send the speech signal. Since bandwidth was very expensive in those days, this possibility was extremely attractive from a commercial point of view.

We have devoted so much space here to Dudley's work because his ideas were the basis of practically all the work on speech signal processing that followed. The description of speech in terms of a carrier (or excitation function) and its modulation (or the time-varying spectral envelope) is still – 70 years later – the basic representation. The parameters used to quantify these components, of course, have evolved in various ways. Besides the channel Vocoder (the modern name for Dudley's Vocoder) many other types of Vocoders have been invented, e.g., formant Vocoder, voice-excited Vocoder. Besides speech compression, Dudley's description was also considered for other applications such as secure voice systems, and the sound spectrograph and its use for communication with the deaf.

Unfortunately, the quality achieved by analog implementations of Vocoders never reached a level acceptable for commercial telephony. Nevertheless they found useful applications for military purposes where poor speech quality was tolerated. The Vocoder representation was also the basis of a speech secrecy system that found extensive use during World War II.

Another example of an analog implementation of Dudley's representation is the sound spectrograph. This is a device that displays the distribution of energy in a speech signal as a function of frequency, and the evolution of this distribution in time. This tool has been extremely useful for investigating properties of speech signals. A real time version of the spectrograph was intended for use as a device for communication with the deaf. That, however, was not very successful. A few people were able to identify about 300 words after 100 hours of training. However, it turned out to be too difficult a task to be practical.

During more than three decades following Dudley's pioneering work, a great amount of research was done on various aspects and properties of speech – properties of the speech production mechanisms, the auditory system, psychophysics, etc. However, except for the three applications mentioned above, little progress was made in speech signal processing and its applications. Exploitation of this research for practical applications had to wait for the general availability of digital hardware starting in the 1970s. Since then much progress has been made in speech coding for efficient transmission, speech synthesis, speech and speaker recognition, and hearing aids [1.5–7]. In the next section we discuss some of these developments.

Today, the area of speech processing is very vast and rich as can be seen from the contents of this Handbook. While we have made great progress since the invention of the telephone, research in the area of speech processing is still very active, and many challenging problems remain unsolved.

## 1.2 Applications of Speech Processing

As mentioned above, one of the earliest goals of speech processing was that of coding speech for efficient transmission. This was taken to be synonymous with reduction of the bandwidth required for transmitting speech. Several advances were needed before the modern success in speech coding was achieved. First, the

notions of information theory introduced during the late 1940s and 1950s brought the realization that the proper goal was the reduction of information rate rather than bandwidth. Second, hardware became available to utilize the sampling theorem to convert a continuous band-limited signal to a sequence of discrete samples. And quantization of the samples allowed digitization of a band-limited speech signal, thus making it usable for digital processing. Finally, the description of a speech signal in terms of linear prediction coefficients (LPC) provided a very convenient representation [1.8–11]. (The theory of predictive coding was in fact developed in 1955. However, its application to speech signals was not made until the late 1970s.)

A telephone speech signal, limited in frequency from 0 to 3.4 kHz, requires 64 kbps (kilobits per second) to be transmitted without further loss of quality. With modern speech compression techniques, the bit rate can be reduced to 13 kbps with little further degradation. For commercial telephony a remaining challenge is to reduce the required bit rate further but without sacrificing quality. Today, the rate can be lowered down to 2.4 kbps while maintaining very high intelligibility, but with a significant loss in quality. Some attempts have been made to reduce the bit rate down to 300 bps, e.g., for radio communication with a submarine. However the quality and intelligibility at these low bit rates are very poor.

Another highly successful application of speech processing is automatic speech recognition (ASR). Early attempts at ASR consisted of making deterministic models of whole words in a small vocabulary (say 100 words) and recognizing a given speech utterance as the word whose model comes closest to it. The introduction of hidden Markov models (HMMs) in the early 1980s provided a much more powerful tool for speech recognition [1.12–14]. Today many products have been developed that successfully utilize ASR for communication between humans and machines. And the recognition can be done for continuous speech using a large vocabulary, and in a speaker-independent manner. Performance of these devices, however, deteriorates in the presence of reverberation and even low levels of ambient noise. Robustness to noise, reverberation, and characteristics of the transducer, is still an unsolved problem.

The goal of ASR is to recognize speech accurately regardless of who the speaker is. The complementary problem is that of recognizing a speaker from his/her voice, regardless of what words he/she is speaking. At present this problem appears to be solvable only if the speaker is one of a small set of $N$ known speakers. A variant of the problem is speaker *verification*, in which the aim is to automatically verify the claimed identity of a speaker. While speaker recognition requires the selection of one out of $N$ possible outcomes, speaker verification requires just a yes/no answer. This problem can be solved with a high degree of accuracy for much larger populations. Speaker verification has application wherever access to data or facilities has to be controlled. Forensics is another area of application. The problem of reduced performance in the presence of noise, as mentioned above for ASR, applies also to speaker recognition and speaker verification.

A third application of speech processing is that of synthesizing speech corresponding to a given text. When used together with ASR, speech synthesis allows a complete two-way spoken interaction between humans and machines. Speech synthesis is also a way to communicate for persons unable to speak. Its use for this purpose by the famous physicist Stephen Hawking is well known.

Early attempts at speech synthesis consisted of deriving the time-varying spectrum for the sequence of phonemes of a given text sentence. From this the corresponding time variation of the vocal tract was estimated, and the speech was synthesized by exciting the time-varying vocal tract with periodic or noise-like excitation as appropriate. The quality of the synthesis was significantly improved by concatenating pre-stored units (i. e., short segments such as diphones, triphones) after modifying them to fit the context. Today the highest-quality speech is synthesized by the unit *selection* method in which the units are selected from a large amount of stored speech and concatenated with little or no modification.

Finally we might mention the application of speech processing to aids for the handicapped. Hearing aid technology has made considerable progress in the last two decades. Part of this progress is due to a slow but steady improvement in our knowledge of the human hearing mechanism. A large part is due to the availability of high-speed digital hardware. At present performance of hearing aids is still poor under noisy and reverberant conditions.

A potentially useful application of speech processing to aid the handicapped is to display the shape of one's vocal tract as one speaks. By trying to match one's vocal tract shape to a displayed shape, a deaf person can learn correct pronunciation. Some attempts to implement this idea have been made, but have still been only in the realm of research.

Another useful application is a reading aid for the blind. The idea is to have a device to scan printed text from a book, and synthesize speech from the

scanned text. Coupled with a device to change speaking rate, this forms a useful aid for the blind. Several products offering this application are available on the market.

Many other application examples are described in the various parts of this handbook. We invite the reader to browse this volume on speech processing to find topics relevant to his/her specific interests.

## 1.3 Organization of the Handbook

This handbook on speech processing is a comprehensive source of knowledge in speech technology and its applications. It is organized as follows. This volume is divided into nine parts. For each part we invited at least one associate editor (AE) to handle it. All the AEs are very well-known researchers in their respective area of research. Part A (AE: M. M. Sondhi) contains four chapters on production, perception, and modeling of speech signals. Part B (AEs: Y. Huang and J. Benesty) concerns signal processing tools for speech, in eight chapters. Part C (AE: B. Kleijn) covers five chapters on speech coding. In part D (AE: S. Narayanan), the areas of text-to-speech synthesis are presented in seven chapters. Part E (AEs: L. Rabiner and B.-H. Juang), with 10 chapters, is a comprehensive overview on speech recognition. Part F (AE: S. Parthasarathy) contains three chapters on speaker recognition. Part G (AE: C.-H. Lee) is about language identification and contains four chapters. In part H (AEs: J. Chen, S. Gannot, and J. Benesty), various aspects of speech enhancement are developed in seven chapters. Finally the last section, part I (AEs: J. Benesty, I. Cohen, and Y. Huang), presents the important aspects of multichannel speech processing in four chapters.

### References


1.1 H. Dudley, T.H. Tarnoczy: The speaking machine of Wolfgang von Kempelen, J. Acoust. Soc. Am. **22**, 151–166 (1950)

1.2 G. Fant: *Acoustic Theory of Speech Production* (Mouton, 's-Gravenhage 1960)

1.3 J.L. Flanagan: *Speech Analysis, Synthesis and Perception* (Springer, New York 1972)

1.4 H. Dudley: The carrier nature of speech, Bell Syst. Tech. J. **19**(4), 495–515 (1940)

1.5 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice Hall, Englewood Cliffs 1978)

1.6 S. Furui, M.M. Sondhi (Eds.): *Advances in Speech Signal Processing* (Marcel Dekker, New York 1992)

1.7 B. Gold, N. Morgan: *Speech and Audio Signal Processing* (Wiley, New York 2000)

1.8 P. Elias: Predictive coding I, IRE Trans. Inform. Theory **1**(1), 16–24 (1955)

1.9 P. Elias: Predictive coding II, IRE Trans. Inform. Theory **1**(1), 24–33 (1955)

1.10 B.S. Atal, M.R. Schroeder: Adaptive predictive coding of speech, Bell Syst. Tech. J. **49**(8), 1973–1986 (1970)

1.11 B.S. Atal: The history of linear prediction, IEEE Signal Proc. Mag. **23**(2), 154–161 (2006)

1.12 L.R. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE **77**(2), 257–286 (1989)

1.13 L.R. Rabiner, B.-H. Juang: *Fundamentals of Speech Recognition* (Prentice-Hall, Englewood Cliff 1993)

1.14 F. Jelinek: *Statistical Methods for Speech Recognition* (MIT, Boston 1998)

# Part A Production

## Part A Production, Perception, and Modeling of Speech

Ed. by M. M. Sondhi


**2 Physiological Processes of Speech Production**
K. Honda, Paris, France

**3 Nonlinear Cochlear Signal Processing and Masking in Speech Perception**
J. B. Allen, Urbana, USA

**4 Perception of Speech and Sound**
B. Kollmeier, Oldenburg, Germany
T. Brand, Oldenburg, Germany
B. Meyer, Oldenburg, Germany

**5 Speech Quality Assessment**
V. Grancharov, Stockholm, Sweden
W. B. Kleijn, Stockholm, Sweden


Amazon EX1010 Page 38

# 2. Physiological Processes of Speech Production

K. Honda

Speech sound is a wave of air that originates from complex actions of the human body, supported by three functional units: generation of air pressure, regulation of vibration, and control of resonators. The lung air pressure for speech results from functions of the respiratory system during a prolonged phase of expiration after a short inhalation. Vibrations of air for voiced sounds are introduced by the vocal folds in the larynx; they are controlled by a set of laryngeal muscles and airflow from the lungs. The oscillation of the vocal folds converts the expiratory air into intermittent airflow pulses that result in a buzzing sound. The narrow constrictions of the airway along the tract above the larynx also generate transient source sounds; their pressure gives rise to an airstream with turbulence or burst sounds. The resonators are formed in the upper respiratory tract by the pharyngeal, oral, and nasal cavities. These cavities act as resonance chambers to transform the laryngeal buzz or turbulence sounds into the sounds with special linguistic functions. The main articulators are the tongue, lower jaw, lips, and velum. They generate patterned movements to alter the resonance characteristics of the supra-laryngeal airway. In this chapter, contemporary views on phonatory and articulatory mechanisms are summarized to illustrate the physiological processes of speech production, with brief notes on their observation techniques.

2.1 **Overview of Speech Apparatus** ..... 7
2.2 **Voice Production Mechanisms** ..... 8
2.2.1 Regulation of Respiration ..... 8
2.2.2 Structure of the Larynx ..... 9
2.2.3 Vocal Fold and its Oscillation ..... 10
2.2.4 Regulation of Fundamental Frequency ($F_0$) ..... 12
2.2.5 Methods for Measuring Voice Production ..... 13
2.3 **Articulatory Mechanisms** ..... 14
2.3.1 Articulatory Organs ..... 14
2.3.2 Vocal Tract and Nasal Cavity ..... 18
2.3.3 Aspects of Articulation in Relation to Voicing ..... 19
2.3.4 Articulators' Mobility and Coarticulation ..... 22
2.3.5 Instruments for Observing Articulatory Dynamics 23
2.4 **Summary** ..... 24
**References** ..... 25

## 2.1 Overview of Speech Apparatus

The speech production apparatus is a part of the motor system for respiration and alimentation. The form of the system can be characterized, when compared with those of other primates, by several unique features, such as small red lips, flat face, compact teeth, short oral cavity with a round tongue, and long pharynx with a low larynx position. The functions of the system are also uniquely advanced by the developed brain with the language areas, direct neural connections from the cortex to motor nuclei, and dense neural supply to each muscle. Independent control over phonation and articulation is a human-specific ability. These morphological and neural changes along human evolution reorganized the original functions of each component into an integrated motor system for speech communication.

The speech apparatus is divided into the organs of phonation (voice production) and articulation (settings of the speech organs). The phonatory organs (lungs and larynx) create voice source sounds by setting the driving air pressure in the lungs and parameters for vocal fold vibration at the larynx. The two organs together adjust the pitch, loudness, and quality of the voice, and further generate prosodic patterns of speech. The articulatory organs give resonances or modulations to the voice source and generate additional sounds for some consonants. They consist of the lower jaw, tongue,

Amazon EX1010 Page 39

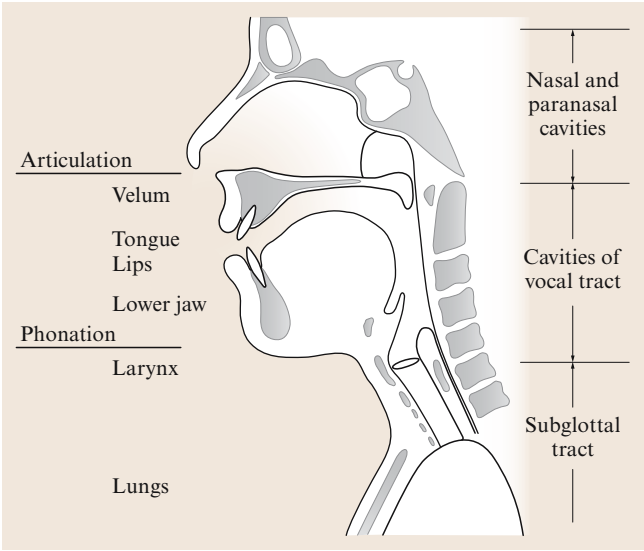



**Fig. 2.1** Sketch of a speech production system. Physiological processes of speech production are realized by combined sequential actions of the speech organs for phonation and articulation. These activities result in sound propagation phenomena at the three levels: subglottal cavities, cavities of the vocal tract, and nasal and paranasal cavities

lips, and the velum. The larynx also takes a part in the articulation of voiced/voiceless distinctions. The tongue and lower lip attach to the lower jaw, while the velum is loosely combined with other articulators. The constrictor muscles of the pharynx and larynx also participate in articulation as well as in voice quality control. The phonatory and articulatory systems influence each other mutually, while changing the vocal tract shape for producing vowels and consonants. Figure 2.1 shows a schematic drawing of the speech production system.

## 2.2 Voice Production Mechanisms

Generation of voice source requires adequate configuration of the airflow from the lungs and vocal fold parameters for oscillation. The sources for voiced sounds are the airflow pulses generated at the larynx, while those for some consonants (i. e., stops and fricatives) are airflow noises made at a narrow constriction in the vocal tract. The expiratory and inspiratory muscles together regulate relatively constant pressure during speech. The laryngeal muscles adjust the onset/offset, amplitude, and frequency of vocal fold vibration.

### 2.2.1 Regulation of Respiration

The respiratory system is divided into two segments: the conduction airways for ventilation between the atmosphere and the lungs, and the respiratory tissue of the lungs for gas exchange. Ventilation (i. e., expiration and inhalation) is carried out by movements of the thorax, diaphragm, and abdomen. These movements involve actions of respiratory muscles and elastic recoil forces of the system. During quiet breathing, the lungs expand to inhale air by the actions of inspiratory muscles (diaphragm, external intercostal, etc.), and expel air by the elastic recoil force of the lung tissue, diaphragm, and cavities of the thorax and abdomen. In effort expiration, the expiratory muscles (internal intercostals, abdominal muscles, etc.) come into action. The inspiratory and expiratory muscles work alternately, making the thorax expand and contract during deep breathing.

During speech production, the respiratory pattern changes to a longer expiratory phase with a shorter inspiratory phase during quiet breathing. Figure 2.2 shows a conventional view of the respiratory pattern during

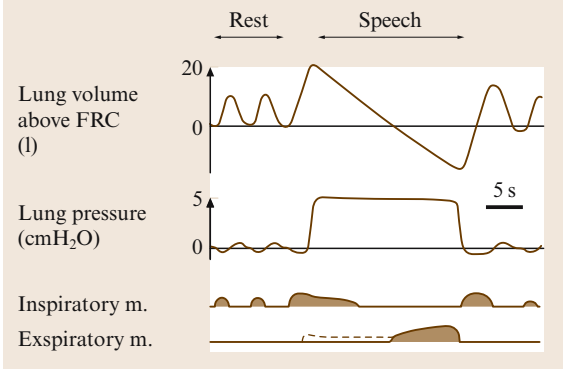



**Fig. 2.2** Respiratory pattern during speech. Top two curves show the changes in the volume and pressure in the lungs. The bottom two curves show schematic activity patterns of the inspiratory and expiratory muscles (after [2.1]). The dashed line for the expiratory muscles indicates their predicted activity for expiration

Amazon EX1010 Page 40

speech [2.1]. The thorax is expanded by inspiration prior to initiation of speech, and then compressed by elastic recoil force by the tissues of the respiratory system to the level of the functional residual capacity (FRC). The lung pressure during speech is kept nearly constant except for the tendency of utterance initial rise and final lowering. In natural speech, stress and emphasis add local pressure increases. The constant lung pressure is due to the actions of the inspiratory muscles to prevent excessive airflow and maintain the long expiratory phase. As speech continues, the lung volume decreases gradually below the level of FRC, and the lung pressure is then maintained by the actions of the expiratory muscles that actively expel air from the lung. It has been argued whether the initiation of speech involves only the elastic recoil forces of the thorax to generate expiratory airflow. Indeed, a few studies have suggested that not only the thoracic system but also the abdominal system assists the regulation of expiration during speech [2.2, 3], as shown by the dashed line in Fig. 2.2. Thus, the contemporary view of speech respiration emphasizes that expiration of air during speech is not a passive process but a controlled one with co-activation of the inspiratory and expiratory muscles.

### 2.2.2 Structure of the Larynx

The larynx is a small cervical organ located at the top of the trachea making a junction to the pharyngeal cavity: it primarily functions to prevent foreign material from entering the lungs. The larynx contains several rigid structures such as the cricoid, thyroid, arytenoid, epiglottic, and other smaller cartilages. Figure 2.3a shows the arrangement of the major cartilages and the hyoid bone. The cricoid cartilage is ring-shaped and supports the lumen of the laryngeal cavity. It offers two bilateral articulations to the thyroid and arytenoid cartilages at the cricothyroid and cricoarytenoid joints, respectively. The thyroid cartilage is a shield-like structure that offers attachments to the vocal folds and the vestibular folds. The arytenoid cartilages are bilateral tetrahedral cartilages that change in location and orientation between phonation and respiration. The whole larynx is mechanically suspended from the hyoid bone by muscles and ligaments.

The gap between the free edges of the vocal folds is called the *glottis*. The space is divided into two portions by the vocal processes of the arytenoid cartilages: the membranous portion in front (essential for vibration) and cartilaginous portion in back (essential for respiration). The glottis changes its form in various ways during speech: it narrows by adduction and widens by abduction of the vocal folds. Figure 2.3b shows that this movement is carried out by the actions of the intrinsic laryngeal muscles that attach to the arytenoid cartilages. These muscles are functionally divided into the adductor and abductor muscles. The adductor muscles include the thyroarytenoid muscles, lateral cricoarytenoid, and arytenoid muscles, and the abductor muscle is the posterior cricoarytenoid muscle. The glottis also changes in length according to the length of the vocal folds, which takes place mainly at the membranous portion. The length of the glottis shows a large developmental sexual variation. The membranous length on average is 10 mm in adult females and 16 mm in adult males, while the cartilaginous length is about 3 mm for both [2.4].

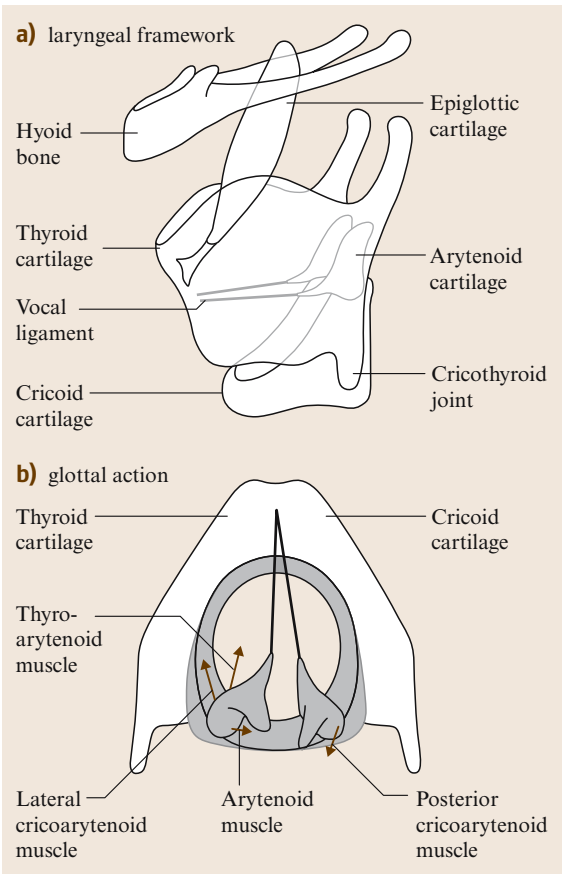



**Fig. 2.3a,b** Laryngeal framework and internal structures. **(a)** Oblique view of the laryngeal framework, which includes the hyoid bone and four major cartilages. **(b)** Adduction (*left*) and abduction (*right*) of the glottis and the effects of the intrinsic laryngeal muscles

### 2.2.3 Vocal Fold and its Oscillation

The larynx includes several structures such as the subglottic dome, vocal folds, ventricles, vestibular folds, epiglottis, and aryepiglottic folds, as shown in Fig. 2.4a. The vocal folds run anteroposteriorly from the vocal processes of the arytenoid cartilages to the internal surface of the thyroid cartilage. The vocal fold tissue consists of the thyroarytenoid muscle, vocal ligament, lamina propria, and mucous membrane. They form a special layer structure that yields to aerodynamic forces to oscillate, which is often described as the *body-cover* structure [2.5].

During voiced speech sounds, the vocal folds are set into vibration by pressurized air passing through the membranous portion of the narrowed glottis. The glottal airflow thus generated induces wave-like motion of the vocal fold membrane, which appears to propagate from the bottom to the top of the vocal fold edges. When this oscillatory motion builds up, the vocal fold membranes on either side come into contact with each other, resulting in repetitive closing and opening of the glottis. Figure 2.4b shows that vocal fold vibration repeats four phases within a cycle: the closed phase, opening phase, open phase, and closing phase. The conditions that determine vocal fold vibration are the stiffness and mass of the vocal folds, the width of the glottis, and the pressure difference across the glottis.

The aerodynamic parameters that regulate vocal fold vibration are the transglottal pressure difference and glottal airflow. The former coincides with the measure of subglottal pressure during mid and low vowels, which is about 5–10 cm $H_2O$ in comfortable loudness and pitch (1 cm $H_2O$ = 0.98 hPa). The latter also coincides with the average measure of oral

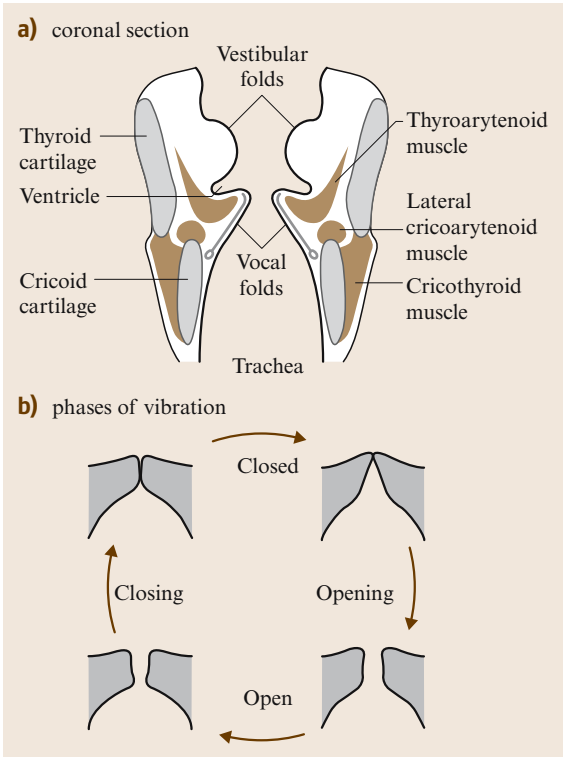



**Fig. 2.4a,b** Vocal folds and their vibration pattern. **(a)** Coronal section of the larynx, showing the tissues of the vocal and vestibular (false) folds. The cavity of the larynx includes supraglottic and subglottic regions. **(b)** Vocal-fold vibration pattern and glottal shapes in open phases. As the vocal-fold edge deforms in a glottal cycle, the glottis follows four phases: closed, opening, open and closing

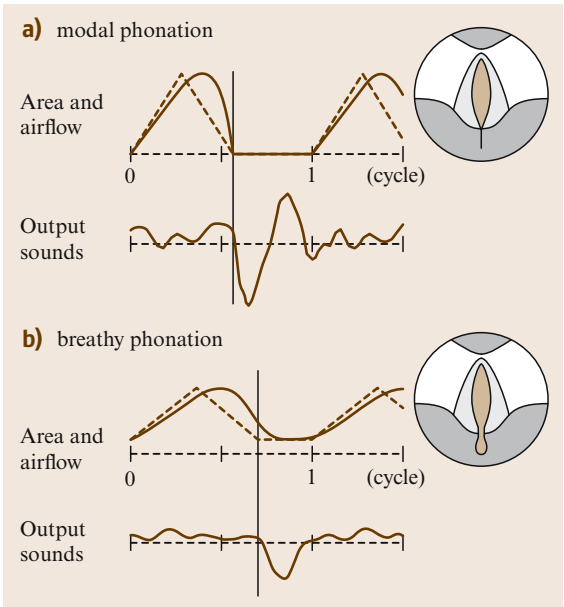



**Fig. 2.5a,b** Changes in glottal area and airflow in relation to output sounds during 1.5 glottal cycles from glottal opening, with glottal shapes at peak opening (in the circles). **(a)** In modal phonation with complete glottal closure in the closed phase, glottal closure causes abrupt shut-off of glottal airflow and strong excitation of the air in the vocal tract during the closed phase. **(b)** In breathy phonation, the glottal closure is incomplete, and the airflow wave includes a DC component, which results in weak excitation of the tract

Amazon EX1010 Page 42

airflow during vowel production, which is roughly 0.1–0.2 l/s. These values show a large individual variation: the pressure range is 4.2–9.6 cm $H_2O$ in males and 4.4–7.6 cm $H_2O$ in females, while the airflow rate ranges between 0.1–0.3 l/s in males and 0.09–0.2 l/s in females [2.6].

Figure 2.5 shows schematically the relationship between the glottal cycle and volumic airflow change in normal and breathy phonation. The airflow varies within each glottal cycle, reflecting the cyclic variation of the glottal area and subglottal pressure. The glottal area curve roughly shows a triangular pattern, while the airflow curve shows a skew of the peak to the right due to the inertia of the air mass within the glottis [2.7]. The closure of the glottis causes a discontinuous decrease of the glottal airflow to zero, which contributes the main source of vocal tract excitation, as shown in Fig. 2.5a. When the glottal closure is more abrupt, the output sounds are more intense with richer harmonic components [2.8]. When the glottal closure is incomplete in soft and breathy voices or the cartilaginous portion of the glottis is open to show the *glottal chink*, the airflow includes a direct-current (DC) component and exhibits a gradual decrease of airflow, which results in a more sinusoidal waveform and a lower intensity of the output sounds, as shown in Fig. 2.5b.

Laryngeal control of the oscillatory patterns of the vocal folds is one of the major factors in voice quality control. In sharp voice, the open phase of the glottal cycle becomes shorter, while in soft voice, the open phase becomes longer. The ratio of the open phase within a glottal cycle is called the *open quotient* (OQ), and the ratio of the closing slope to the opening slope in the glottal cycle is called the *speed quotient* (SQ). These two parameters determine the slope of the spectral envelope. When the open phase is longer (high OQ) with a longer closing phase (low SQ), the glottal airflow becomes more sinusoidal, with weak harmonic components. Contrarily, when the open phase is shorter (low OQ), glottal airflow builds up to pulsating waves with rich harmonics. In modal voice, all the vocal fold layers are involved in vibration, and the membranous glottis is completely closed during the closed phase of each cycle. In falsetto, only the edges of the vocal folds vibrate, glottal closure becomes incomplete, and harmonic components reduce remarkably.

The oscillation of the vocal folds during natural speech is quasiperiodic, and cycle-to-cycle variation are observed in speech waveforms as two types of measures: *jitter* (frequency perturbation) and *shimmer* (amplitude perturbation). These irregularities appear to arise from combinations of biomechanical (vocal fold asymmetry), neurogenic (involuntary activities of laryngeal muscles), and aerodynamic (fluctuations of airflow and subglottal pressure) factors. In sustained phonation of normal voice, the jitter is about 1% in frequency, and the shimmer is about 6% in amplitude.

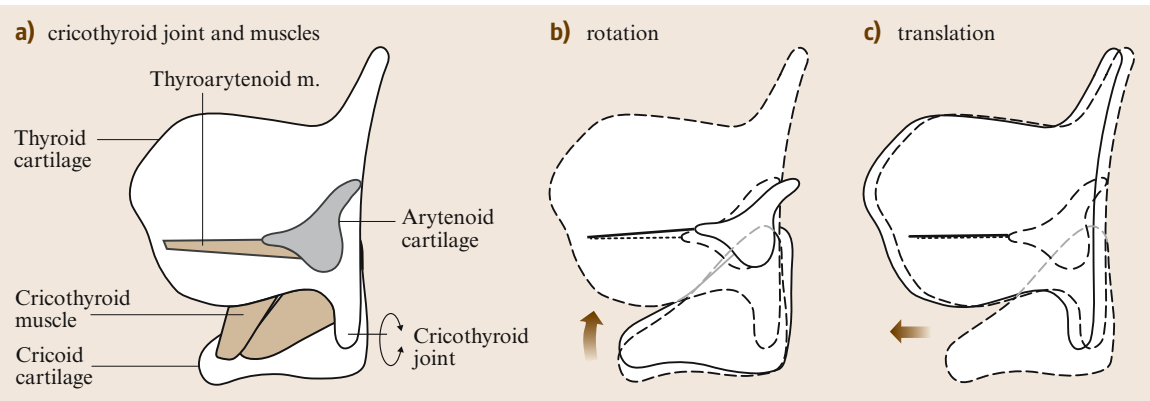



**Fig. 2.6a–c** Cricothyroid joint and $F_0$ regulation mechanism. **(a)** The cricothyroid joint is locally controlled by the thyroarytenoid and two parts of the cricothyroid muscles: Pars recta (anterior) and pars obliqua (posterior). As $F_0$ rises, the thyroid cartilage advances and cricoid cartilage rotates to the direction to stretch the vocal folds, which leads to the increases in the stiffness of vocal fold tissue and in the natural resonance frequency of the vocal folds. **(b)** Rotation of the cricothyroid joint is caused mainly by the action of the pars recta to raise the cricoid arch. **(c)** Translation of the joint is produced mainly by the pars obliqua

### 2.2.4 Regulation of Fundamental Frequency ($F_0$)

The fundamental frequency ($F_0$) of voice is the lowest harmonic component in voiced sounds, which conforms to the natural frequency of vocal fold vibration. $F_0$ changes depending on two factors: regulation of the length of the vocal folds and adjustment of aerodynamic factors that satisfy the conditions necessary for vocal fold vibration. In high $F_0$, the vocal folds become thinner and longer; while in low $F_0$, the vocal folds become shorter and thicker. As the vocal folds are stretched by separating their two attachments (the anterior commissure and vocal processes), the mass per unit length of the vocal fold tissue is reduced while the stiffness of the tissue layer involved in vibration increases. Thus, the mass is smaller and the stiffness is greater for higher $F_0$ than lower $F_0$, and it follows that the characteristic frequency of vibrating tissue increases for higher $F_0$. The length of the vocal folds is adjusted by relative movement of the cricoid and thyroid cartilages. Its natural length is a determinant factor of individual difference in $F_0$. The possible range of $F_0$ in adult speakers is about 80–400 Hz in males, and about 120–800 Hz in females.

The thyroid and cricoid cartilages are articulated at the cricothyroid joint. Any external forces applied to this joint cause rotation and translation (sliding) of the joint, which alters the length of the vocal folds. It is well known that the two joint actions are brought about by the contraction of the cricothyroid muscle to approximate the two cartilages at their front edges. Figure 2.6 shows two possible actions of the cricothyroid muscle on the joint: rotation by the pars recta and translation of the pars obliqua [2.9]. Questions still remain as to whether each part of the cricothyroid conducts pure actions of rotation or translation, and as to which part is more responsible for determining $F_0$.

The extrinsic laryngeal muscles can also apply external forces to this joint as a supplementary mechanism for regulating $F_0$ [2.10]. The most well known among the activities of the extrinsic muscles in this regulation is the transient action of the sternohyoid muscle observed as $F_0$ falls. Since this muscle pulls down the hyoid bone to lower the entire larynx, larynx lowering has long been thought to play a certain role in $F_0$ lowering. Figure 2.7 shows a possible mechanism of $F_0$ lowering by vertical larynx movement revealed by magnetic resonance imaging (MRI). As the cricoid cartilage descends along the anterior surface of the cervical spine, the cartilage rotates in a direction that shortens the vocal folds because the cervical spine shows anterior convexity at the level of the cricoid cartilage [2.11].

Aerodynamic conditions are an additional factor that alters $F_0$, as seen in the local rises of the subglottal pressure during speech at stress or emphasis. The increase of the subglottal air pressure results in a larger airflow rate and a wider opening of the glottis, which causes greater deformation of the vocal folds with larger average tissue stiffness. The rate of $F_0$ increase due to the subglottal pressure is reported to be about 2–5 Hz/cm$H_2O$ when the chest cavity is compressed externally, and is observed to be 5–15 Hz/cm$H_2O$, when

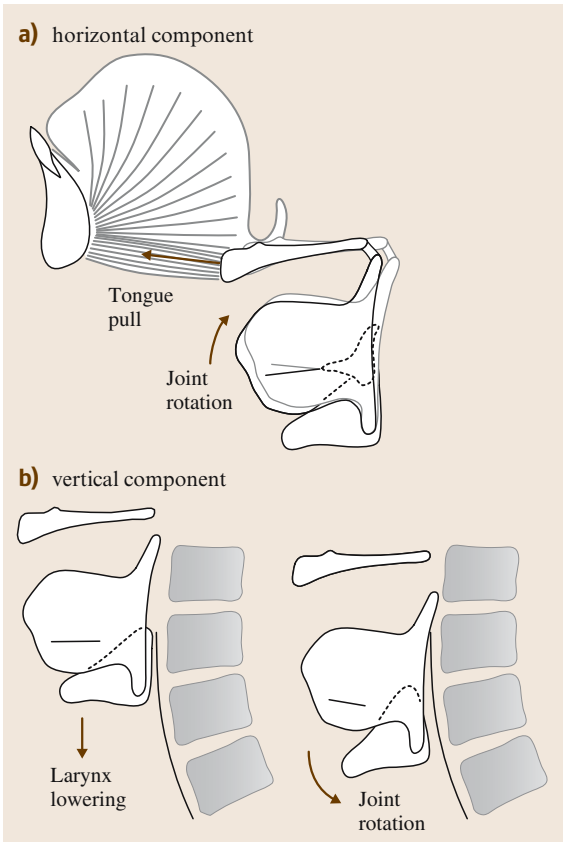



**Fig. 2.7a,b** Extrinsic control of $F_0$. Actions of the cricothyroid joint are determined not only by the cricothyroid muscle but also by other laryngeal muscles. Any external forces applied to the joint can activate the actions of the joint. **(a)** In $F_0$ raising, advancement of the hyoid bone possibly apply a force to rotate the thyroid cartilage. **(b)** In $F_0$ lowering, the cricoid cartilage rotates as its posterior plate descends along the anterior convexity of the cervical spine

Amazon EX1010 Page 44

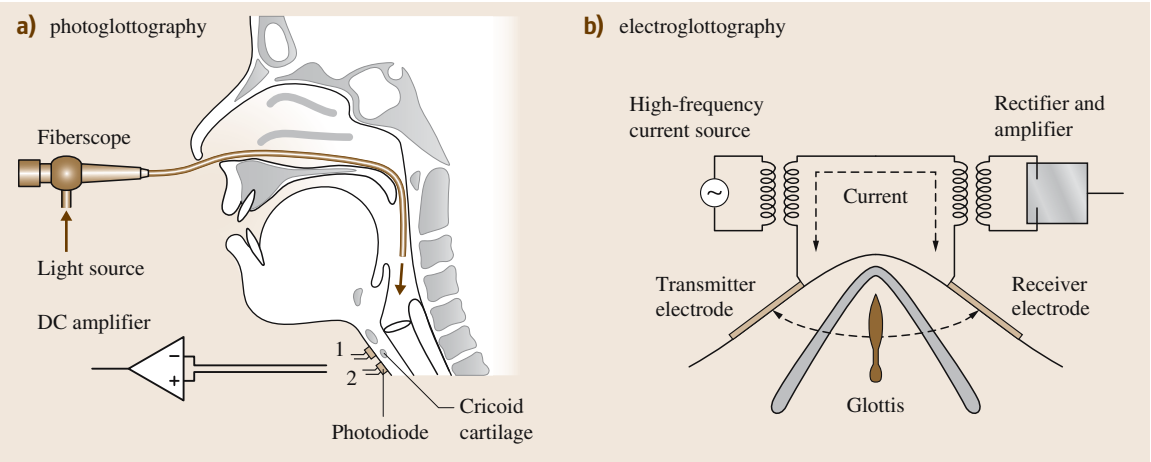



**Fig. 2.8a,b** Glottographic methods. **(a)** PGG with fiberscopy uses a photodetector attached near the cricothyroid cartilage in two locations: one attachment for measuring vibrations, and two attachment for glottal gestures. **(b)** EGG uses a pair of electrodes on the skin above the thyroid lamina to form a induction circuit to record electrical currents passed through the vocal-fold edges

it is measured between the beginning and end of speech utterances.

## 2.2.5 Methods for Measuring Voice Production

Speech production mechanisms arise from the functions of the internal organs of the human body that are mostly invisible. Therefore, better understanding of speech production processes relies on the development of observation techniques. The lung functions in speech can be assessed by the tools for aerodynamic measurements, while examination of the larynx functions during speech requires special techniques for imaging and signal recording.

### Monitoring Respiratory Functions

Respiratory functions during speech are examined by recording aerodynamic measurements of lung volume, airflow, and pressure. Changes in lung volume are monitored with several types of plethysmography (e.g., whole-body, induction, and magnetic). The airflow from the mouth is measured with pneumotachography using a mask with pressure probes (differential-pressure anemometry) or thermal probes (hot-wire anemometry). Measurements of the subglottal pressure require a tracheal puncture of a needle with a pressure sensor or a thin catheter-type pressure transducer inserted from the nostril to the trachea via the cartilaginous part of the glottis.

### Laryngeal Endoscopy

Imaging of the vocal folds during speech has been conducted with a combination of an endoscope and video camera. A solid-type endoscope is capable of observing vocal fold vibration with stroboscopic or real-time digital imaging techniques during sustained phonation. The flexible endoscope is beneficial for video recording of glottal movements during speech with a fiber optic bundle inserted into the pharynx through the nostril via the velopharyngeal port. Recently, an electronic type of flexible endoscope with a built-in image sensor has become available.

### Glottography

Glottography is a technique to monitor vocal fold vibration as a waveform. Figure 2.8 shows two types of glottographic techniques. Photoglottography (PGG) detects light intensity modulated by the glottis using an optical sensor. The sensor is placed on the neck and a flexible endoscope is used as a light source. The signal from the sensor corresponds to the glottal aperture size, reflecting vocal fold vibration and glottal adduction–abduction movement. Electroglottography (EGG) records the contact of the left and right vocal fold edges during vibration. High-frequency current is applied to a pair of surface electrodes placed on the skin above the thyroid lamina, which detect a varying induction current that corresponds to the change in vocal fold contact area.

Amazon EX1010 Page 45

## 2.3 Articulatory Mechanisms

Speech articulation is the most complex motor activity in humans, producing concatenations of phonemes into syllables and syllables into words using movements of the speech organs. These articulatory processes are conducted within a phrase of a single expiratory phase with continuous changes of vocal fold vibration, which is one of the human-specific characteristics of sound production mechanisms.

### 2.3.1 Articulatory Organs

Articulatory organs are composed of the rigid organ of the lower jaw and soft-tissue organs of the tongue, lips, and velum, as illustrated in Fig. 2.9. These organs together alter the resonance of the vocal tract in various ways and generate sound sources for consonants in the vocal tract. The tongue is the most important articulatory organ, and changes the gross configuration of the vocal tract. Deformation of the whole tongue determines vowel quality and produces palatal, velar, and pharyngeal consonants. Movements of the tongue apex and blade contribute to the differentiation of dental and alveolar consonants and the realization of retroflex consonants. The lips deform the open end of the vocal tract by various types of gestures, assisting the production of vowels and labial consonants. Actions of these soft-tissue organs are essentially based on contractions of the muscles within these organs, and their mechanism is often compared with the *muscular hydrostat*. Since the tongue and lips have attachments to the lower jaw, they are interlocked with the jaw to open the mouth. The velum controls opening and closing of the velopharyngeal port, and allows distinction between nasal and oral sounds. Additionally, the constrictor muscles of the pharynx adjust the lateral width of the pharyngeal cavity, and their actions also assist articulation for vowels and back consonants.

#### Upper Jaw

The upper jaw, or the maxilla with the upper teeth, is the structure fixed to the skull, forming the palatal dome on the arch of the alveolar process with the teeth. It forms a fixed wall of the vocal tract and does not belong to the articulatory organs: yet it is a critical structure for speech articulation because it provides the frame of reference for many articulatory gestures. The structures of the upper jaw offer the location for contact or approximation by many parts of the tongue such as the apex, blade, and dorsum. The phonetics literature describes the place of articulation as classified according to the locations of lingual approximation along the upper jaw for dental, alveolar, and palatal consonants. The hard palate is covered by the thick mucoperiosteum, which has several transverse lines of mucosal folds called the *palatine rugae*.

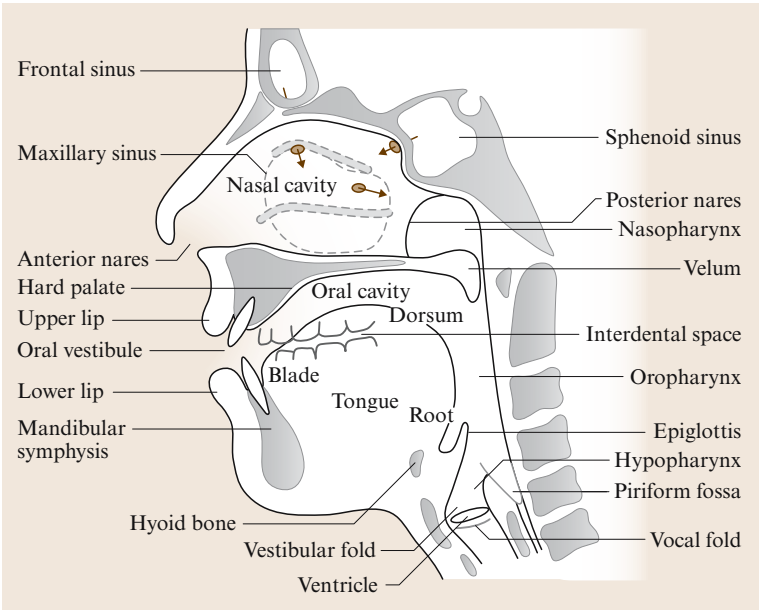



**Fig. 2.9** Illustration of the articulatory system with names of articulators and cavities

### Lower Jaw

The lower jaw, or the mandible with the lower teeth, is the largest rigid motor organ among the speech production apparatus. Its volume is about 100 $cm^3$. As well as playing the major role in opening and closing the mouth, it provides attachments for many speech muscles and supports the tongue, lips, and hyoid bone.

Figure 2.10 shows the action of the jaw and the muscles used in speech articulation. The mandible articulates with the temporal bone at the temporomandibular joint (TMJ) and brings about jaw opening–closing actions by rotation and translation. The muscles that control jaw movements are generally called the masticatory muscles. The jaw opening muscles are the digastric and lateral pterygoid muscles. The strap muscles, such as the geniohyoid and sternohyoid, also assist jaw opening. The jaw closing muscles include the masseter, temporalis, and medial pterygoid muscles. While the larger muscles play major roles in biting and chewing, comparatively small muscles are used for speech articulation. The medial pterygoid is mainly used for jaw closing in articulation, and the elastic recoil force of the connective tissues surrounding the mandible is another factor for closing the jaw from its open position.

### Tongue

The tongue is an organ of complex musculature [2.12]. It consists of a round body occupying its main mass and a short blade with an apex. Its volume is approximately 100 $cm^3$, including the muscles in the tongue floor. The tongue body moves in the oral cavity by variously deforming its voluminous mass, while the tongue blade alters its shape and changes the angle of the tongue apex. Deformation of the tongue tissue is caused by contractions of the extrinsic and intrinsic tongue muscles, which are illustrated schematically in Fig. 2.11.

The extrinsic tongue muscles are those that arise outside of the tongue and end within the tongue tissue. This group includes four muscles, the genioglossus, hyoglossus, styloglossus, and palatoglossus muscles, although the former three muscles are thought to be involved in the articulation of the tongue. The palatoglossus muscle participates in the lowering of the velum as discussed later.

The genioglossus is the largest and strongest muscle in the tongue. It begins from the posterior aspect of the mandibular symphysis and runs along the midline of the tongue. Morphologically, it belongs to the triangular muscle, and its contraction effects differ across portions of the muscle. Therefore, the genioglossus is divided functionally into the anterior, middle, and posterior bundles. The anterior and middle bundles run midsagittally, and their contraction makes the midline groove of the tongue for the production of front vowels. The anterior bundle often makes a shallow notch on the tongue surface called the *lingual fossa* and assists elevation of the tongue apex. The middle bundle runs obliquely, and advances the tongue body for front vowels. The posterior bundle of the genioglossus runs midsagittally and

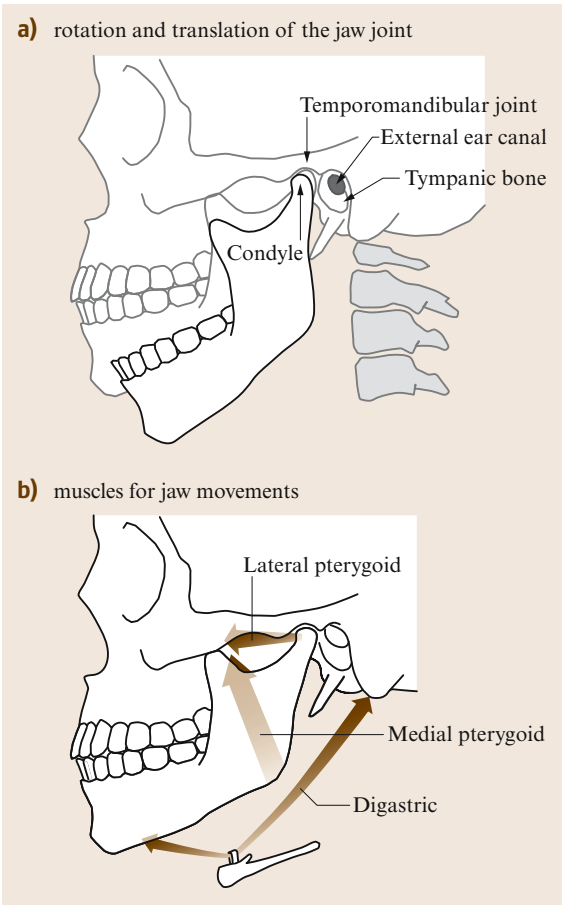



**Fig. 2.10a,b** Actions of the temporomandibular joint and muscles for jaw opening and closing. **(a)** The lower jaw opens by rotation and translation of the mandible at the temporomandibular joint. Jaw translation is needed for wide opening of the jaw because jaw rotation is limited by the narrow space between the condyle and tympanic bone. **(b)** Jaw opening in speech depends on the actions of the digastric and medial pterygoid muscles with support of the strap muscles. Jaw closing is carried out by the contraction of the lateral pterygoid muscle and elastic recoil forces of the tissues surrounding the jaw

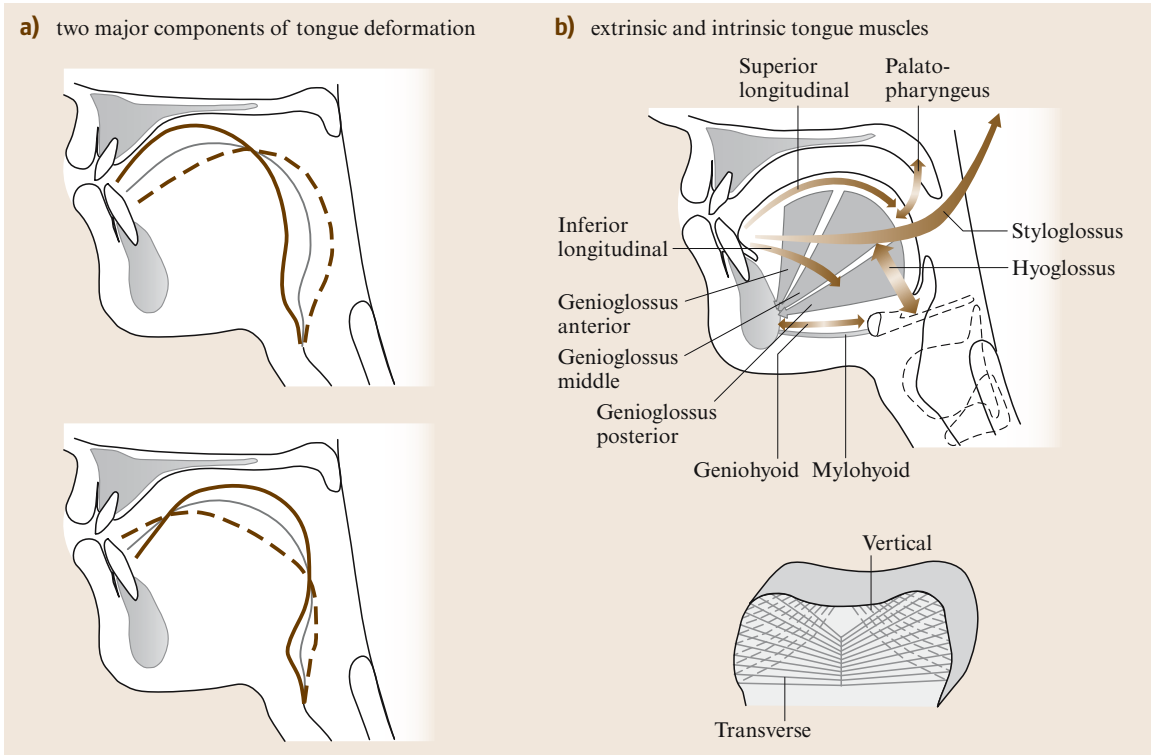



**Fig. 2.11a,b** Actions of the tongue and its musculature. **(a)** Major components of tongue deformation are high-front vs.low-back (*top*) and high back versus low front (*bottom*) motions, (after [2.14]). **(b)** Lateral view (*top*) shows the extrinsic and intrinsic muscles of the tongue with two tongue floor muscles. Coronal section (*bottom*) shows additional intrinsic muscles

also spreads laterally, reaching a wide area of the tongue root. This bundle draws the tongue root forward and elevates the upper surface of the tongue for high vowels and anterior types of oral consonants. The hyoglossus is a bilateral thin-sheet muscle, which arises from the hyoid bone, runs upward along the sides of the tongue, and ends in the tongue tissue, intermingling with the styloglossus. Its contraction lowers the tongue dorsum and pushes the tongue root backward for the production of low vowels. The styloglossus is a bilateral long muscle originating from the styloid process on the skull base, running obliquely to enter the back sides of the tongue. Within the tongue, it runs forward to reach the apex of the tongue, while branching downward to the hyoid bone and medially toward the midline. Although the extra-lingual bundle of the styloglossus runs obliquely, it pulls the tongue body straight back at the insertion point because the bundle is surrounded by fatty and muscular tissues. The shortening of the intra-lingual bundle draws the tongue apex backward and causes an upward bunching of the tongue body [2.13]. Each of the extrinsic tongue muscles has two functions: drawing of the relevant attachment point toward the origin, and deforming the tongue tissue in the orthogonal orientation. The resulting deformation of the tongue can be explained by two antagonistic pairs of extrinsic muscles: posterior genioglossus versus styloglossus, and anterior genioglossus versus hyoglossus. The muscle arrangement appears to be suitable for tongue body movements in the vertical and horizontal dimensions.

The intrinsic tongue muscle is a group of muscles that have both their origin and termination within the tongue tissue. They include four bilateral muscles: the superior longitudinal, inferior longitudinal, transverse, and vertical muscles. The superior and inferior longitudinal muscles operate on the tongue blade to produce vertical and horizontal movements of the tongue tip. The transverse and vertical muscles together compress the tongue tissue medially to change the cross-sectional shape of the tongue.

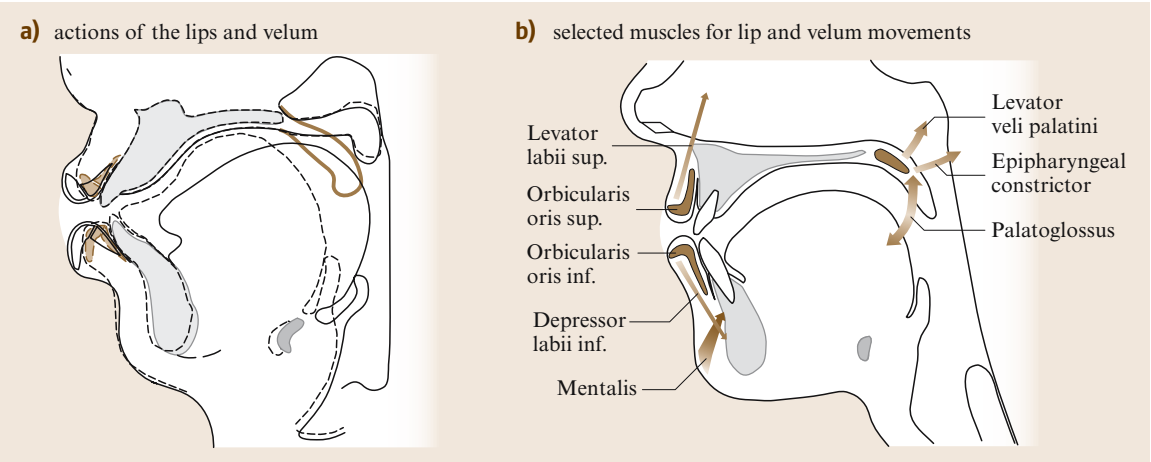



**Fig. 2.12a,b** Actions of the lips and velum, and their muscles. **(a)** Trace of MRI data in the production of /i/ and /u/ with lip protrusion show that two parts of the orbicularis oris, marginal (*front*) and peripheral (*back*) bundles demonstrate their geometrical changes within the vermillion tissue. The shapes of the velum also vary greatly between the rest position (thick gray line) and vowel articulation. **(b)** Five labial muscles are shown selectively from among many facial muscles. The velum shape is determined by the elevator, constrictor, and depressor (palatopharyngeus)

There are two muscles that support the tongue floor: the geniohyoid and mylohyoid muscles. The geniohyoid runs from the genial process of the mandibular symphysis to the body of the hyoid bone. This muscle has two functions: opening the jaw for open vowels and advancing the hyoid bone to help raise $F_0$. The mylohyoid is a sheet-like muscle beneath the tongue body that stretches between the mandible and the hyoid bone to support the entire tongue floor. This muscle supports the tongue floor to assist articulation of high front vowels and oral consonants.

### Lips and Velum

The lips are a pair of soft-tissue organs consisting of many muscles. Their functions resemble those of the tongue because they partly adhere to the mandible and partly run within the soft tissue of the lips. The vermillion, or the part of red skin, is the unique feature of the human lips, which transmits phonetic signals visually. The deformation of the lips in speech can be divided into three components. The first is opening/closing of the lip aperture, which is augmented by jaw movement. The second is rounding/spreading of the lip tissue, produced by the changes in their left–right dimension. The third is protrusion/retraction of the lip gesture, generated by three-dimensional deformation of the entire lip tissue.

The muscles that cause deformation of the lips are numerous. Figure 2.12 shows only a few representative muscles of the lips. The orbicularis oris is the muscle that surrounds the lips, consisting of two portions; the marginal and peripheral bundles. Contraction of the marginal bundles near the vermillion borders is thought to produce lip rounding without protrusion. Contraction of the peripheral bundles that run in the region around the marginal bundles compresses the lip tissue circumferentially to advance the vermillion in lip protrusion [2.15]. The mentalis arises from the mental part of the mandible to the lip surface, and its contraction elevates the lower lip by pulling the skin at the mental region. The levator labii superior elevates the upper lip, and the depressor labii inferior depresses the lower lip relative to the jaw. The superior and inferior angli oris muscles move the lip corners up and down, respectively, which makes facial expressions rather than speech articulation.

The exact mechanism of lip protrusion is still in question. Tissue bunching by muscle shortening as a general rule for the organs of muscle does not fully apply to the phenomenon of lip protrusion. This is because, as the vermilion thickens in lip protrusion, it does not compress on the teeth; its dental surface often detaches from the teeth (Figure 2.12a). A certain three-dimensional stress distribution within the entire labial tissue must be considered to account for the causal factors of lip protrusion.

The velum, or the soft palate, works as a valve behind the hard palate to control the velopharyngeal port, as shown in Fig. 2.12a. Elevation of the velum is carried

out during the production of oral sounds, while lowering takes place during the production of nasal sounds. The action of the velum to close the velopharyngeal port is not a pure hinge motion but is accompanied by the deformation of the velum tissue with narrowing of the nasopharyngeal wall. In velopharyngeal closure, the levator veli palatine contracts to elevate the velum, and the superior pharyngeal constrictor muscle produces concentric narrowing of the port. In velopharyngeal opening, the palatoglossus muscle assists active lowering of the velum.

### 2.3.2 Vocal Tract and Nasal Cavity

The vocal tract is an acoustic space where source sounds for speech propagate. Vowels and consonants rely on strengthening or weakening of the spectral components of the source sound by resonance of the air column in the vocal tract. In the broad definition, the vocal tract includes all the air spaces where acoustic pressure variation takes place in speech production. In this sense, the vocal tract divides into three regions: the subglottal tract, the tract from the glottis to the lips, and the nasal cavities.

The subglottal tract is the lower respiratory tract below the glottis down to the lungs via the trachea and bronchial tubes. The length of the trachea from the glottis to the carina is 10–15 cm in adults, including the length of the subglottic laryngeal cavity (about 2 cm). Vocal source sounds propagate from the glottis to the trachea, causing the subglottal resonance in speech spectra. The resonance frequencies of the subglottal airway are estimated to be 640, 1400, and 2100 Hz [2.16]. The second subglottal resonance is often observed below the second formant of high vowels.

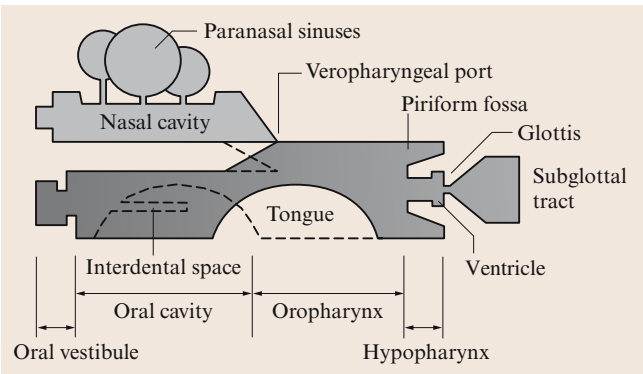



**Fig. 2.13** Acoustic design of the vocal tract. Passages from the subglottal tract to two output ends at the lips and nares are shown with the effects of tongue and velar movements. The resonance of the vocal tract above the supraglottic laryngeal cavity determines major the vowel formants ($F_1$, $F_2$, and $F_3$). The resonance of the subglottal tract and interdental space interacts with the vowel formants, while the hypopharyngeal cavities and other small cavities cause local resonances and antiresonances in the higher-frequency region

The vocal tract, according to the conventional definition, is the passage of vocal sounds from the glottis to the lips, where source sounds propagate and give rise to the major resonances. The representative values for the length of the main vocal tract from the glottis to the lips are 15 cm in adult females and 17.5 cm in adult males. According to the measurement data based on the younger population, vocal tract lengths are 14 cm in females and 16.5 cm in males [2.17, 18], which are shorter than the above values. Considering the elongation of the vocal tract during a course of life, the above representative values appear reasonable. While the oral cavity length is maintained by the rigid structures of the skull and jaw, the pharyngeal cavity length increases due to larynx lowering before and after puberty. Thus, elongation of the pharyngeal cavity is the major factor in the developmental variation in vocal tract length.

The vocal tract anatomically divides into four segments: the hypopharyngeal cavities, the mesopharynx, the oral cavity, and the oral vestibule (lip tube). The hypopharyngeal part of the vocal tract consists of the supraglottic laryngeal cavity (2 cm long) and the bilateral conical cavities of the piriform fossa (2 cm long). The mesopharynx extends from the aryepiglottic fold to the anterior palatal arch. The oral cavity is the segment from the anterior palatal arch to the incisors. The oral vestibule extends from the incisors to the lip opening. The latter shows an anterior convexity, which often makes it difficult to measure the exact location of lip opening.

The vocal tract is not a simple uniaxial tube but has a complex three-dimensional construction. The immobile wall of the vocal tract includes the dental arch and the palatal dome. The posterior pharyngeal wall is almost rigid, but it allows subtle changes in convexity and orientation. The soft walls include the entire tongue surface, the velum with the uvula, the lateral pharyngeal wall, and the lip tube. The shape of the vocal tract varies individually due to a few factors. First, the lateral width of the upper and lower jaws relative to the pharyngeal cavity width affects tongue articulation and results in a large individual variation of vocal tract shape observed midsagittally. Second, the mobility of the jaw

Amazon EX1010 Page 50

depending on the location of the mandibular symphysis relative to the skull can vary the openness of vowels. Third, the size of the tongue relative to the oral and pharyngeal cavities varies individually; the larger the tongue size, the smaller the articulatory space for vowels.

Figure 2.13 shows a schematic drawing of the vocal tract and nasal cavity. The vocal tract has nearly constant branches such as the piriform fossa (entrance to the esophagus) and the vallecula (between the tongue root and epiglottis). The vocal tract also has controlled branches to the nasal cavity at the velopharyngeal port and to the *interdental space* (the space bounded by the upper and lower teeth and the lateral cheek wall). The latter forms a pair of side-branches when the tongue is in a higher position as in /i/ or /e/, while it is unified with the oral cavity when the tongue is in a lower position as in /a/.

The nasal cavity is an accessory channel to the main vocal tract. Its horizontal dimension from the anterior nares to the posterior wall of the epipharynx is approximately 10–11 cm. The nasal cavity can be divided into the single-tube segment (the velopharyngeal region and epipharynx) and the dual-tube segment (the nasal cavity proper and nasal vestibule). Each of the bilateral channels of the nasal cavity proper has a complex shape of walls with the three turbinates with thick mucous membrane, which makes a narrower cross section compared with the epipharyngeal area [2.19]. The nasal cavity has its own side-branches of the paranasal sinuses; the maxillary, sphenoid, ethomoid, and frontal sinuses.

The nasal cavity builds nasal resonance to accomplish phonetic features of nasal sounds and nasalized vowels. The paranasal sinuses also contribute to acoustic characteristics of the nasal sounds. The nasal murmur results from these characteristics: a Helmholtz resonance of the entire nasopharyngeal tract from the glottis to the anterior nares and regional Helmholtz resonances caused by the paranasal sinuses, together characterized by a resonance peak at 200–300 Hz and spectral flattening up to 2 kHz [2.20, 21]. The nasal resonance could takes place even in oral vowels with a complete closure of the velopharyngeal port: the soft tissue of the velum transmits the pressure variation in the oral cavity to the nasal cavity, which would enhances sound radiation for close vowels and voiced stops.

### 2.3.3 Aspects of Articulation in Relation to Voicing

Here we consider a few phonetic evidences that can be considered as joint products of articulation and phonation. Vowel production is the typical example for this topic, in view of its interaction with the larynx. Regulation of voice quality, which has been thought to be a laryngeal phenomenon, is largely affected by the lower part of the vocal tract. The voiced versus voiceless distinction is a pertinent issue of phonetics that involves both phonatory and articulatory mechanisms.

#### Production of Vowels

The production of vowels is the result of the joint action of phonatory and articulatory mechanisms. In this process, the larynx functions as a source generator, and the vocal tract plays the role of an acoustic filter to modulate the source sounds and radiate from the lip opening, as described by the *source-filter* theory [2.22, 23]. The quality of oral vowels is determined by a few peak frequencies of vocal tract resonance (formants). In vowel production, the vocal tract forms a *closed tube* with the closed end at the glottis and the open end at the lip opening. Multiple reflections of sound wave between the two ends of the vocal tract give rise to vowel formants ($F_1$, $F_2$, $F_3$). The source-filter theory has been supported by many studies as the fundamental concept explaining the acoustic process of speech production, which is further discussed in the next section.

Vowel articulation is the setup for the articulatory organs to determine vocal tract shape for each vowel. When the jaw is in a high position and the tongue is in a high front position, the vocal tract assumes the shape for /i/. Contrarily, when the jaw is in a low position and the tongue is in a low back position, the vocal tract takes the shape for /a/. The articulatory organ that greatly influences vocal tract shape for vowels is the tongue. When the vocal tract is modeled as a tube with two segments (front and back cavities), the movements of the tongue body between its low back and high front positions creates contrasting diverging and converging shapes of the main vocal tract. Jaw movement enhances these changes in the front cavity volume, while pharyngeal constriction assists in the back cavity volume. When the vocal tract is modeled as a tube with three segments, the movements of the tongue body between its high back and low front positions determine the constriction or widening of the vocal tract in its middle portion. The velum also contributes to the articulation of open vowels by decreasing the area of the vocal tract at the velum or making a narrow branch to the nasal cavity. The lip tube is another factor for vowel articulation that determines the vocal tract area near the open end.

Although muscular control for vowel articulation is complex, a simplified view can be drawn based

Amazon EX1010 Page 51

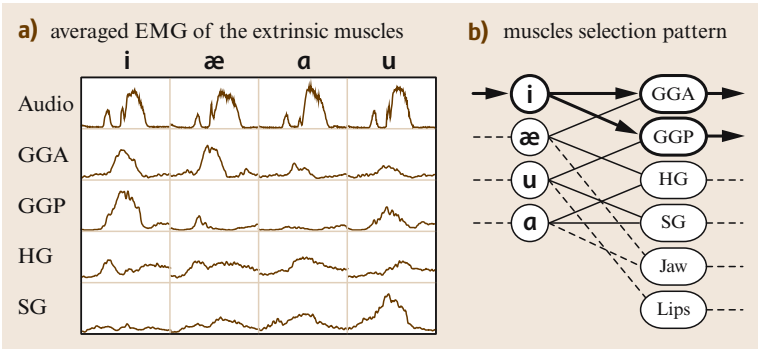



**Fig. 2.14a,b** Tongue EMG data during VCV utterances and muscle selection pattern in vowel articulation. **(a)** Averaged EMG data for four English corner vowels are shown for the major muscles of the tongue: the anterior genioglossus (GGA), posterior genioglossus (GGP), hyoglossus (HG), and styloglossus (SG). **(b)** The systematic variation observed in the muscle–vowel matrix suggests a muscle selection pattern

on electromyographic (EMG) data obtained from the tongue muscles [2.24]. Figure 2.14a shows a systematic pattern of muscle activities for CVC (consonant-vowel-consonant) utterances with /p/ and four English corner vowels. The anterior and posterior genioglossus are active for front vowels, while the styloglossus and hyoglossus are active for back vowels. These muscles also show a variation depending on vowel height. These observations are shown schematically in Fig. 2.14b: the basic control pattern for vowel articulation is the selection of two muscles among the four extrinsic muscles of the tongue [2.25].

As the tongue or jaw moves for vowel articulation, they apply forces to the surrounding organs and cause secondary effects on vowel sounds. For example, articulation of high vowels such as /i/ and /u/ is mainly produced by contraction of the posterior genioglossus, which is accompanied by forward movement of the hyoid bone. This action applies a force to rotate the thyroid cartilage in a direction that stretches the vocal folds. In evidence, higher vowels tend to have a higher $F_0$, known as the *intrinsic vowel* $F_0$ [2.26,27]. When the jaw opens to produce open vowels, jaw rotation compresses the tissue behind the mandibular symphysis, which applies a force to rotate the thyroid cartilage in the opposite direction, thereby shortening the vocal folds. Thus, the jaw opening has the secondary effect of lowering the intrinsic $F_0$ for lower vowels.

### Supra-Laryngeal Control of Voice Quality

The laryngeal mechanisms controlling voice quality were described in an earlier section. In this section, the supra-laryngeal factors are discussed. Recent studies have shown evidence that the resonances of the hypopharyngeal cavities determine the spectral envelope in the higher frequencies above 2.5 kHz by causing an extra resonance and antiresonances [2.28–31]. The hypopharyngeal cavities include a pair of vocal-tract

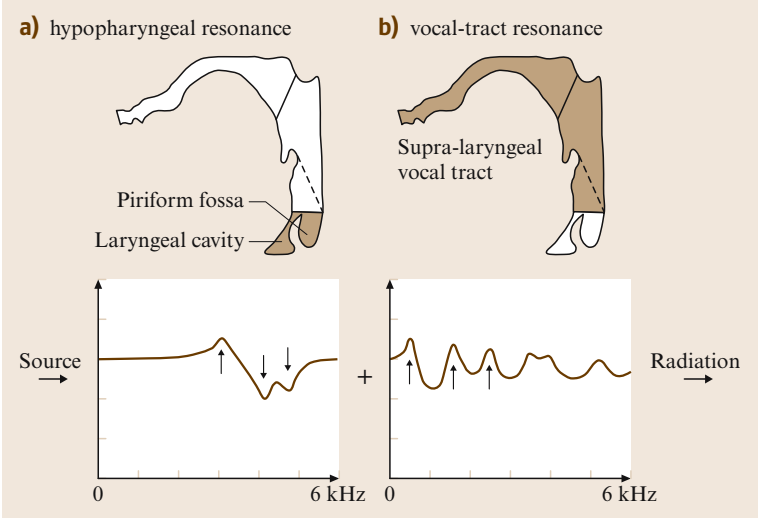



**Fig. 2.15a,b** Vocal-tract resonance with hypopharyngeal cavity coupling in vowel production. **(a)** The supraglottal laryngeal cavity contributes a resonance peak at 3–3.5 kHz, and the bilateral cavities of the piriform fossa cause antiresonances at 4–5 kHz. **(b)** The main vocal tract above the laryngeal cavity determines the major vowel formants

side-branches formed by the piriform fossa. Each fossa maintains a relatively constant cavity during speech, which is collapsed only in deep inhalation by the wide abduction of the arytenoid cartilage. The piriform fossa causes one or two obvious antiresonances in the higher frequencies above 4 kHz [2.29] and affects the surrounding formants. The laryngeal cavity above the vocal folds also contributes to shaping the higher frequencies [2.28, 32]. The supraglottic laryngeal cavity, from the ventricles to the aryepiglottic folds via the ventricular folds, forms a type of Helmholtz resonator and gives rise to a resonance at higher frequencies of 3–3.5 kHz. This resonance can be counted as the fourth formant (F4) but it is actually an *extra formant* to the resonance of the vocal tract above the laryngeal cavity [2.30]. When the glottis opens in the open phase of vocal fold vibration, the supraglottic laryngeal cavity no longer constitutes a typical Helmholtz resonator, and demonstrates a strong damping of the resonance, which is observed as the disappearance of the affiliated extra formant. Therefore, the laryngeal cavity resonance shows a cyclic nature during vocal fold vibration, and it is possibly absent in breathy phonation or pathological conditions with insufficient glottal closure [2.31]. Figure 2.15 shows an acoustic model of the vocal tract to illustrate this coupling of the hypopharyngeal cavities.

The hypopharyngeal cavities are not an entirely fixed structure but vary due to physiological efforts to control $F_0$ and voice quality. A typical case of the hypopharyngeal adjustment of voice quality is found in the *singing formant* [2.28]. When high notes are produced by opera singers, the entire larynx is pulled forward due to the advanced position of the tongue, which widens the piriform fossa to deepens the fossa's antiresonances, resulting in a decrease of the frequency of the adjacent lower formant ($F_5$). When the supraglottic laryngeal cavity is constricted, its resonance ($F_4$) comes down towards the lower formant ($F_3$). Consequently, the third to fifth formants come closer to each other and generate a high resonance peak observed near 3 kHz.

### Regulation of Voiced and Voiceless Sounds

Voiced and voiceless sounds are often attributed to the glottal state with and without vocal fold vibration, while their phonetic characteristics result from phonatory and articulatory controls over the speech production system. In voiced vowels, the vocal tract forms a closed tube with no significant constrictions except for the narrow laryngeal cavity. On the other hand, in whispered vowels, the membranous glottis is closed, and the supraglottic laryngeal cavity forms an extremely narrow channel continued from the open cartilaginous glottis, with a moderate constriction of the lower pharynx. Devoiced vowels exhibit a wide open glottis and a reduction of tongue articulation. Phonetic distinctions of voiced and voiceless consonants further involve fine temporal control over the larynx

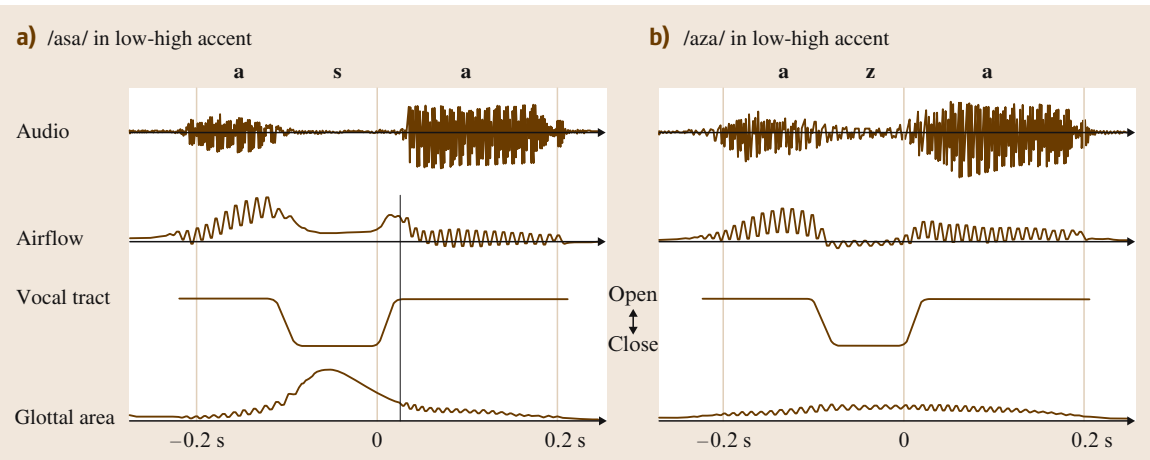



**Fig. 2.16a,b** Laryngeal articulatory patterns in producing VCV utterances with voiceless and voiced fricatives as in /asa/ and /aza/. From the top to bottom, speech signals, oral airflow, schematic patterns of vocal tract constriction, and glottal area variations are shown schematically. This figure is based on the author's recent experiment with anemometry with an open-type airflow transducer and photoglotto-graphy with an external lighting technique, conducted by *Dr. Shinji Maeda* (ENST) and the author

Amazon EX1010 Page 53

and supra-laryngeal articulators in language-specific ways.

In the production of voiced consonants, vocal fold vibration typically continues during the voiced segments. In voiced stops and fricatives, the closure or narrowing of the vocal tract results in decrease in glottal airflow and transglottal pressure difference. The glottal airflow during the stop closure is maintained during the closure due to the increases in vocal tract volume: the expansion of the oral cavity (jaw lowering and cheek wall expansion) and the expansion of the pharyngeal cavity (lateral wall expansion and larynx lowering). During the closure period, air pressure variations are radiated not only from the vocal tract wall but also from the anterior nares due to transvelar propagation of the intra-oral sound pressure into the nasal cavities.

In the production of voiceless consonants, vocal fold vibration is suppressed due to a rapid reduction of the transglottal pressure difference and abduction of the vocal folds. During stop closures, the intra-oral pressure builds up to reach the subglottal pressure, which enhances the rapid airflow after the release of the closure. Then, vocal fold vibration restarts with a delay to the release, which is observed as a long voice onset time (VOT) for voiceless stops. The process of suppressing vocal fold vibration is not merely a passive aerodynamic process on the vocal folds, but is assisted by a physiological process to control vocal fold stiffness. The cricothyroid muscle has been observed to increase its activity in producing voiceless consonants. This activity results in a high–falling $F_0$ pattern during the following vowel, contributing a phonetic attribute to voiceless consonants [2.33]. In glottal stops, vocal fold vibration stops due to forced adduction of the vocal folds with an effort closure of the supraglottic laryngeal cavity.

Figure 2.16 illustrates the time course of the processes during vowel-consonant-vowel (VCV) utterances with a voiceless fricative in comparison to the case with a voiced fricative. The voiceless segment initiates with glottal abduction and alveolar constriction, and vocal fold vibration gradually fades out during the phase of glottal opening. After reaching the maximum glottal abduction, the glottis enters the adduction phase, followed by the release of the alveolar constriction. Then, the glottis becomes narrower and vocal fold vibration restarts. There is the time lag between the release of the constriction and full adduction of the glottis, which results in the peak flow seen in Fig. 2.16a, presumably accompanied by aspiration sound at the glottis.

### 2.3.4 Articulators' Mobility and Coarticulation

The mobility of speech articulators varies across organs and contributes certain phonetic characteristics to speech sounds. Rapid movements are essential to a sequence from one distinctive feature to another, as observed in the syllable /sa/ from a narrow constriction to the vocalic opening, while gradual movements are found to produce nasals and certain labial sounds. These variations are principally due to the nature of articulators with respect to their mobility. The articulatory mechanism involves a complex system that is built up by organs with different motor characteristics. Their variation in temporal mobility may be explained by a few biological factors. The first is the phylogenetic origin of the organs: the tongue muscles share their origin with the fast motor systems such as the eyeball or finger, while other muscles such as in the lips or velum originate from the slow motor system similar to the musculature of the alimentary tract. The second is the innervation density to each muscle: the faster organs are innervated by thicker nerve bundles, and vise versa, which derives from an adaptation of the biological system to required functions. In fact, the human hypoglossal nerve that supplies the tongue muscles is much thicker than that of other members of the primate family. The third is the composition of muscle fiber types in the musculature, which varies from organ to organ. The muscles in the larynx have a high concentration of the ultrafast fibers (type 2B), while the muscle to elevate the velum predominantly contains the slow fibers (type 1). In accordance with these biological views, the rate of the articulators movement indexed by the maximum number of syllables per second follows the order of the tongue apex, body, and lips: the tongue moves at a maximum rate of 8.2 syllables per second at the apex, and 7.1 syllables per second with the back of the tongue, while the lips and facial structures move at a maximum rate of 2.5–3 syllables per second [2.34]. More recent measurements indicate that the lips are slower than the tongue apex but faster than the tongue dorsum. The velocities during speech tasks reach 166 mm/sec for the lower lip, 196 mm/sec for the tongue tip, and 129 mm/sec for the tongue dorsum [2.35]. The discrepancy between these two reports regarding the mobility of the lips may be explained by the types of movements measured: opening–closure movement by the jaw–lower lip complex is faster than the movement of the lips themselves, such as protrusion and spreading.

It is often noted that speech is characterized by asynchrony among articulatory movements, and the degree

of asynchrony varies with the feature to be realized. Each articulator does not necessarily strictly keep pace with other articulators in a syllable sequence. The physiological basis of this asynchrony may be explained by the mobility of the articulatory organs and motor precision required for the target of articulation. The slower articulators such as the lips and velum tend to exhibit marked coarticulation in production of labial and nasal sounds. In stop–vowel–nasal sequences (such as /tan/), the velopharyngeal port is tightly closed at the stop onset and the velum begins to lower before the nasal consonant. Thus, the vowel before the nasal consonant is partly nasalized. When the vowel /u/ is preceded by /s/, the lips start to protrude during the consonant prior to the rounded vowel.

The articulators' mobility also contributes some variability to speech movements. The faster articulators such as parts of the tongue show various patterns from target undershooting to overshooting. In articulation of close–open–close vowel sequences such as /iai/, tongue movements naturally show undershooting for the open vowel. In contrast, when the alveolar voiceless stop /t/ is placed in the open vowel context as in /ata/, the tongue blade sometimes shows an extreme overshoot with a wide contact on the hard palate because such articulatory variations do not significantly affect the output sounds. On the contrary, in alveolar and postalveolar fricatives such as /s/ and /sh/, tongue movements also show a dependence on articulatory precision because the position of the tongue blade must be controlled precisely to realize the narrow passage for generating frication sounds. The lateral /l/ is similar to the stops with respect to the palatal contact, while the rhotic /r/ with no contact to the palate can show a greater extent of articulatory variations from retroflex to bunched types depending on the preceding sounds.

## 2.3.5 Instruments for Observing Articulatory Dynamics

X-ray and palatography have been used as common tools for articulatory observation. Custom instruments are also developed to monitor articulatory movements, such as the X-ray microbeam system and magnetic sensor system. The various types of newer medical imaging techniques are being used to visualize the movements of articulatory system using sonography and nuclear magnetic resonance. These instruments are generally large scale, although relatively compact instruments are becoming available (e.g., magnetic probing system or portable ultrasound scanner).

### Palatography

The palatograph is a compact device to record temporal changes in the contact pattern of the tongue on the palate. There are traditional static and modern dynamic types. The dynamic type is called electropalatography, or dynamic palatography, which employs an individually customized palatal plate to be placed on the upper jaw. As shown in Fig. 2.17a, this system employs a palatal plate with many surface electrodes to monitor electrical contacts on the tongue's surface.

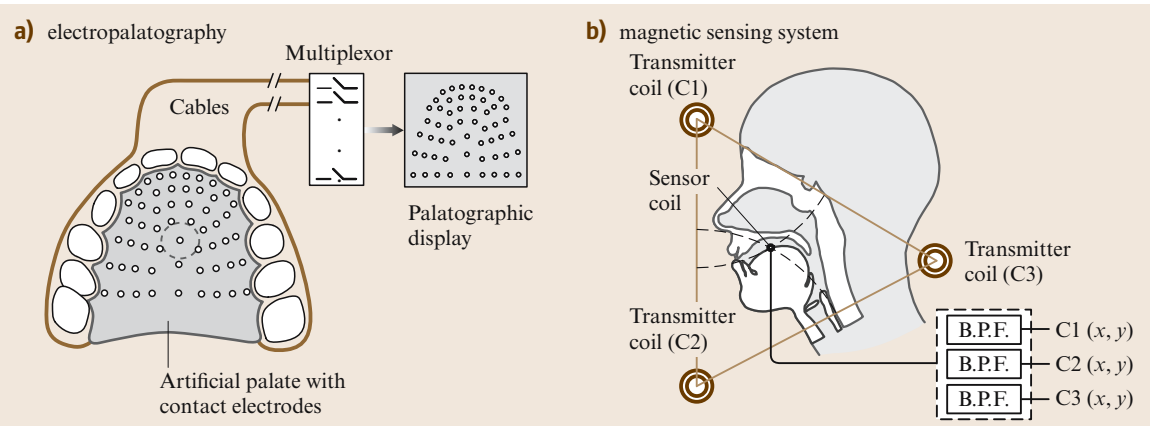



**Fig. 2.17a,b** Electropalatography and magnetic sensing system. **(a)** Electropalatography displays tongue–palate contact patterns by detecting weak electrical current caused by the contact between the electrodes on the artificial palate and the tongue tissue. **(b)** Magnetic sensing system is based on detection of alternate magnetic fields with different frequencies using miniature sensor coils

Amazon EX1010 Page 55

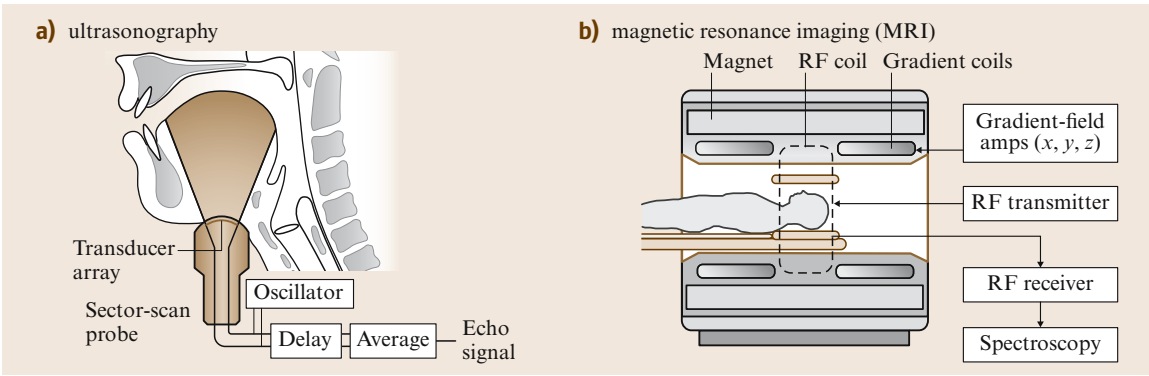



**Fig. 2.18a,b** Medical imaging techniques. **(a)** Ultrasound scanner uses an array of transmitters and receivers to detect echo signals from regions where the ultrasound signals reflect strongly such as at the tissue-air boundaries on the tongue surface. **(b)** Magnetic resonance imaging (MRI) generates strong static magnetic field, controlled gradient fields in the three directions, and radio-frequency (RF) pulses. Hydrogen atoms respond to the RF pulses to generate echo signals, which are detected with a receiver coil for spectral analysis

### Marker Tracking System

A few custom devices have been developed to record movements of markers attached on the articulatory organs. X-ray microbeam and magnetic sensing systems belong to this category. Both can measure 10 markers simultaneously. The X-ray microbeam system uses a computer-controlled narrow beam of high-energy X-rays to track small metal pellets attached on the articulatory organs. This system allows automatic accurate measurements of pellets with a minimum X-ray dosage.

The magnetic sensing system (magnetometer, or magnetic articulograph) is designed to perform the same function as the microbeam system without X-rays. The system uses a set of transmitter coils that generate alternate magnetic fields and miniature sensor coils attached to the articulatory organs, as shown in Fig. 2.17b. The positions of the receiver coils are computed from the filtered signals from the coils.

### Medical Imaging Techniques

X-ray cinematography and X-ray video fluorography have been used for re-cording articulatory movements in two-dimensional projection images. The X-ray images show clear outlines of rigid structures, while they provide less-obvious boundaries for soft tissue. The outline of the tongue is enhanced by the application of liquid contrast media on the surface. Metal markers are often used to track the movements of flesh points on the soft-tissue articulators.

Ultrasonography is a diagnostic technique to obtain cross-sectional images of soft-tissues in real time. Ultrasound scanners consist of a sound probe (phased-array piezo transducer and receiver) and image processor, as illustrated in Fig. 2.18a. The probe is attached to the skin below the tongue to image the tongue surface in the sagittal or coronal plane.

Magnetic resonance imaging (MRI), shown in Fig. 2.18b, is a developing medical technique that excels at soft-tissue imaging of the living body. Its principle relies on excitation and relaxation of the hydrogen nuclei in water in a strong homogeneous magnetic field in response to radio-frequency (RF) pulses applied with variable gradient magnetic fields that determine the slice position. MRI is essentially a method for recording static images, while motion imaging setups with stroboscopic or real-time techniques have been applied to the visualization of articulatory movements or vocal tract deformation three-dimensionally [2.36].

## 2.4 Summary

This chapter described the structures of the human speech organs and physiological mechanisms for producing speech sounds. Physiological processes during speech are multidimensional in nature as described in this chapter. Discoveries of their component mechanisms have been dependent on technical developments

for visualizing the human body and analyses of biological signals, and this is still true today. For example, the hypopharyngeal cavities have long been known to exist, but their acoustic role was underestimated until recent MRI observations. The topics in this chapter were chosen with the author's hope to provide a guideline for the sophistication of speech technologies by reflecting the real and detailed processes of human speech production. Expectations from these lines of studies include speech analysis by recovering control parameters of articulatory models from speech sounds, speech synthesis with full handling of voice quality and individual vocal characteristics, and true speech recognition through biologic, acoustic, and phonetic characterizations of input sounds.

## References


2.1 M.H. Draper, P. Ladefoged, D. Whittenridge: Respiratory muscles in speech, J. Speech Hearing Res. **2**, 16–27 (1959)

2.2 T.J. Hixon, M. Goldman, J. Mead: Kinematics of the chest wall during speech production: volume displacements of the rib cage, abdomen, and lung, J. Speech Hearing Res. **16**, 78–115 (1973)

2.3 G. Weismer: Speech production. In: *Handbook of Speech-Language Pathology and Audiology*, ed. by N.J. Lass, L.V. McReynolds, D.E. Yoder (Decker, Toronto 1988) pp. 215–252

2.4 J. Kahane: A morphological study of the human prepubertal and pubertal larynx, Am. J. Anat. **151**, 11–20 (1979)

2.5 M. Hirano, Y. Kakita: Cover-body theory of vocal cord vibration. In: *Speech Science*, ed. by R. Daniloff (College Hill, San Diego 1985) pp. 1–46

2.6 E.B. Holmberg: Glottal airflow and transglottal air pressure measurements for male and female speakers in soft, normal, and loud voice, J. Acoust. Soc. Am. **84**, 511–529 (1988)

2.7 M.R. Rothenberg: Acoustic interaction between the glottal source and the vocal tract. In: *Vocal Fold Physiology*, ed. by K.N. Stevens, M. Hirano (Univ. Tokyo Press, Tokyo 1981) pp. 305–328

2.8 G. Fant, J. Liljencrants, Q. Lin: A four-parameter model of glottal flow, Speech Transmission Laboratory - Quarterly Progress and Status Report (STL-QPSR) **4**, 1–13 (1985)

2.9 B.R. Fink, R.J. Demarest: *Laryngeal Biomechanics* (Harvard Univ. Press, Cambridge 1978)

2.10 J.E. Atkinson: Correlation analysis of the physiological features controlling fundamental frequency, J. Acoust. Soc. Am. **63**, 211–222 (1978)

2.11 K. Honda, H. Hirai, S. Masaki, Y. Shimada: Role of vertical larynx movement and cervical lordosis in F0 control, Language Speech **42**, 401–411 (1999)

2.12 H. Takemoto: Morphological analysis of the human tongue musculature for three-dimensional modeling, J. Speech Lang. Hearing Res. **44**, 95–107 (2001)

2.13 S. Takano, K. Honda: An MRI analysis of the extrinsic tongue muscles during vowel production, Speech Commun. **49**, 49–58 (2007)

2.14 S. Maeda: Compensatory articulation during speech: evidence from the analysis and synthesis of vocal-tract shapes using an articulatory model. In: *Speech Production and Speech Modeling*, ed. by W.J. Hardcastle, A. Marchal (Kluwer Academic, Dartrecht 1990) pp. 131–149

2.15 K. Honda, T. Kurita, Y. Kakita, S. Maeda: Physiology of the lips and modeling of lip gestures, J. Phonetics **23**, 243–254 (1995)

2.16 K. Ishizaka, M. Matsudaira, T. Kaneko: Input acoustic-impedance measurement of the subglottal system, J. Acoust. Soc. Am. **60**, 190–197 (1976)

2.17 U.G. Goldstein: An articulatory model for the vocal tracts of growing children. Ph.D. Thesis (Massachusetts Institute of Technology, Cambridge 1980)

2.18 W.T. Fitch, J. Giedd: Morphology and development of the human vocal tract: A study using magnetic resonance imaging, J. Acoust. Soc. Am. **106**, 1511–1522 (1999)

2.19 J. Dang, K. Honda, H. Suzuki: Morphological and acoustic analysis of the nasal and paranasal cavities, J. Acoust. Soc. Am. **96**, 2088–2100 (1994)

2.20 O. Fujimura, J. Lindqvist: Sweep-tone measurements of the vocal tract characteristics, J. Acoust. Soc. Am. **49**, 541–557 (1971)

2.21 S. Maeda: The role of the sinus cavities in the production of nasal vowels, Proc. IEEE Int. Conf. Acoustics, Speech, Signal Proc. (ICASSP'82), Vol. 2 (1982) pp. 911–914, Paris

2.22 T. Chiba, M. Kajiyama: *The Vowel – Its Nature and Structure* (Tokyo-Kaiseikan, Tokyo 1942)

2.23 G. Fant: *Acoustic Theory of Speech Production* (Mouton, The Hague 1960)

2.24 T. Baer, P. Alfonso, K. Honda: Electromyography of the tongue muscle during vowels in /*pVp/ environment, Ann. Bull. RILP **22**, 7–20 (1988)

2.25 K. Honda: Organization of tongue articulation for vowels, J. Phonetics **24**, 39–52 (1996)

2.26 I. Lehiste, G.E. Peterson: Some basic considerations in the analysis of intonation, J. Acoust. Soc. Am. **33**, 419–425 (1961)

2.27 K. Honda: Relationship between pitch control and vowel articulation. In: *Vocal Fold Physiology*, ed. by D.M. Bless, J.H. Abbs (College-Hill, San Diego 1983) pp. 286–297

2.28 J. Sundberg: Articulatory interpretation of the singing formant, J. Acoust. Soc. Am. **55**, 838–844 (1974)


Amazon EX1010 Page 57

2.29 J. Dang, K. Honda: Acoustic characteristics of the piriform fossa in models and humans, J. Acoust. Soc. Am. **101**, 456–465 (1996)
2.30 H. Takemoto, S. Adachi, T. Kitamura, P. Mokhtari, K. Honda: Acoustic roles of the laryngeal cavity in vocal tract resonance, J. Acoust. Soc. Am. **120**, 2228–2238 (2006)
2.31 T. Kitamura, H. Takemoto, S. Adachi, P. Mokhtari, K. Honda: Cyclicity of laryngeal cavity resonance due to vocal fold vibration, J. Acoust. Soc. Am. **120**, 2239–2249 (2006)
2.32 I.R. Titze, B.H. Story: Acoustic interactions of the voice source with the lower vocal tract, J. Acoust. Soc. Am. **101**, 2234–2243 (1997)
2.33 A. Lofqvist, N.S. McGarr, K. Honda: Laryngeal muscles and articulatory control, J. Acoust. Soc. Am. **76**, 951–954 (1984)
2.34 R.G. Daniloff: Normal articulation processes. In: *Normal Aspect of Speech, Hearing, and Language*, ed. by F.D. Minifie, T.J. Hixon, F. Williams (Prentice-Hall, Englewood Cliffs 1983) pp. 169–209
2.35 D.P. Kuehn, K.L. Moll: A cineradiographic study of VC and CV articulatory velocities, J. Phonetics **4**, 303–320 (1976)
2.36 K. Honda, H. Takemoto, T. Kitamura, S. Fujita, S. Takano: Exploring human speech production mechanisms by MRI, IEICE Info. Syst. **E87-D**, 1050–1058 (2004)

Amazon EX1010 Page 58

# 3. Nonlinear Cochlear Signal Processing and Masking in Speech Perception

J. B. Allen

There are many classes of masking, but two major classes are easily defined: *neural masking* and *dynamic masking*. Neural masking characterizes the internal noise associated with the neural representation of the auditory signal, a form of loudness noise. Dynamic masking is strictly cochlear, and is associated with cochlear outer-hair-cell processing. This form is responsible for dynamic nonlinear cochlear gain changes associated with sensorineural hearing loss, the upward spread of masking, two-tone suppression and forward masking. The impact of these various forms of masking are critical to our understanding of speech and music processing. In this review, the details of what we know about nonlinear cochlear and basilar membrane signal processing is reviewed, and the implications of neural masking is modeled, with a comprehensive historical review of the masking literature. This review is appropriate for a series of graduate lectures on nonlinear cochlear speech and music processing, from an auditory point of view.

3.1 **Basics** ..... 27
3.1.1 Function of the Inner Ear ..... 28
3.1.2 History of Cochlear Modeling ..... 31

3.2 **The Nonlinear Cochlea** ..... 35
3.2.1 Cochlear Modeling ..... 35
3.2.2 Outer-Hair-Cell Transduction ..... 41
3.2.3 Micromechanics ..... 42

3.3 **Neural Masking** ..... 45
3.3.1 Basic Definitions ..... 47
3.3.2 Empirical Models ..... 51
3.3.3 Models of the JND ..... 51
3.3.4 A Direct Estimate of the Loudness JND ..... 52
3.3.5 Determination of the Loudness SNR 54
3.3.6 Weber–Fraction Formula ..... 54

3.4 **Discussion and Summary** ..... 55
3.4.1 Model Validation ..... 55
3.4.2 The Noise Model ..... 55

**References** ..... 56



## 3.1 Basics

*Auditory masking* is critical to our understanding of speech and music processing. There are many classes of masking, but two major classes are easily defined. These two types of masking and their relation to nonlinear (NL) speech processing and coding are the focus of this chapter.

The *first* class of masking, denoted *neural masking*, is due to internal *neural noise*, characterized in terms of the intensity *just noticeable difference*, denoted $\Delta I(I, f, T)$ (abbreviated $\mathrm{JND_I}$) and defined as the *just discriminable change in intensity*. The $\mathrm{JND_I}$ is a function of intensity $I$, frequency $f$ and stimulus type $T$ (e.g., noise, tones, speech, music, etc.). As an *internal noise*, the $\mathrm{JND_I}$ may be modeled in terms of a loudness (i. e., perceptual intensity) noise density along the length of the cochlea ($0 \le X \le L$), described in terms of a *partial loudness JND* ($\Delta\mathcal{L}(X, T)$, a.k.a. $\mathrm{JND}_{\mathcal{L}}$). The cochlea or inner ear is the organ that converts signals from acoustical to neural signals. The loudness JND is a function of the *partial loudness* $\mathcal{L}(X)$, defined as the loudness contribution coming from each cochlear *critical band*, or more generally, along some *tonotopic central auditory representation*. The critical band is a measure of cochlear bandwidth at a given cochlear *place* $X$. The loudness JND plays a major role in speech and music coding since coding quantization noise may be masked by this internal quantization (i. e., *loudness noise*).

The *second* masking class, denoted here as *dynamic masking*, comes from the NL mechanical action of cochlear *outer-hair-cell* (OHC) signal processing. It can have two forms, simultaneous and nonsimultaneous, also known as *forward masking*, or *post-masking*. Dynamic-masking (i. e., nonlinear OHC signal processing) is well known (i. e., there is a historical literature

Amazon EX1010 Page 59

on this topic) to be intimately related to questions of cochlear frequency selectivity, sensitivity, dynamic range compression and *loudness recruitment* (the loss of loudness dynamic range). Dynamic masking includes the *upward spread of masking* (USM) effect, or in neural processing parlance, *two-tone suppression* (2TS). It may be underappreciated that NL OHC processing (i. e., dynamic masking) is largely responsible for *forward masking* (FM, or post-stimulus masking), which shows large effects over long time scales. For example OHC effects (FM/USM/2TS) can be as large as 50 dB, with an FM *latency* (return to base line) of up to 200 ms. *Forward masking* (FM) and NL OHC *signal onset enhancement* are important to the detection and identification of perceptual features of a speech signal. Some research has concluded that forward masking is not related to OHC processing [3.1, 2], so the topic remains controversial. Understanding and modeling NL OHC processing is key to many speech processing applications. As a result, a vibrant research effort driven by the National Institute of Health on OHC biophysics has ensued.

This OHC research effort is paying off at the highest level. Three key examples are notable. *First* is the development of wide dynamic-range multiband compression (WDRC) hearing aids. In the last 10–15 years WDRC signal processing (first proposed in 1937 by researchers at Bell Labs [3.3]), revolutionized the hearing-aid industry. With the introduction of compression signal processing, hearing aids now address the recruitment problem, thereby providing speech audibility over a much larger dynamic range, at least in quiet. The problems of the impaired ear given speech in noise is poorly understood today, but this problem is likely related to the effects of NL OHC processing. This powerful circuit (WDRC) is not the only reason hearing aids of today are better. Improved electronics and transducers have made significant strides as well. In the last few years the digital barrier has finally been broken, with digital signal processing hearing aids now becoming common.

A *second* example is the development of otoacoustic emissions (OAE) as a hearing diagnostic tool. Pioneered by David Kemp and Duck Kim, and then developed by many others, this tool allows for cochlear evaluation of neonates. The identification of cochlear hearing loss in the first month has dramatically improves the lives of these children (and their parents). While it is tragic to be born deaf, it is much more tragic for the deafness to go unrecognized until the child is three years old, when they fail to learn to talk. If you cannot hear you do not learn to talk. With proper and early cochlear implant intervention, these kids can lead nearly normal-hearing lives and even talk on the phone. However they cannot understand speech in noise. It is at least possible that this loss is due to the lack of NL OHC processing.

A *third* example of the application of NL OHC processing to speech processing is still an underdeveloped application area. The key open problem here is: *How does the auditory system, including the NL cochlea, followed by the auditory cortex, processes human speech*? There are many aspects of this problem including speech coding, speech recognition in noise, hearing aids and language learning and reading disorders in children. If we can solve the *robust phone decoding problem*, we will fundamentally change the effectiveness of human-machine interactions. For example, the ultimate hearing aid is the hearing aid with built in robust speech feature detection and phone recognition. While we have no idea when this will come to be, and it is undoubtedly many years off, when it happens there will be a technology revolution that will change human communications.

In this chapter several topics will be reviewed. First is the history of cochlear models including extensions that have taken place in recent years. These models include both macromechanics and micromechanics of the tectorial membrane and hair cells. This leads to comparisons of the basilar membrane, hair cell, and neural frequency tuning. Hearing loss, loudness recruitment, as well as other key topics of modern hearing health care, are discussed. The role of NL mechanics and dynamic range are reviewed to help the reader understand the importance of modern wideband dynamic range compression hearing aids as well as the overall impact of NL OHC processing.

Any reader desiring further knowledge about cochlear anatomy and function or a basic description of hearing, they may consult *Pickles* [3.4], *Dallos* [3.5], *Yost* [3.6].

### 3.1.1 Function of the Inner Ear

The goal of cochlear modeling is to refine our understanding of how auditory signals are processed. The two main roles of the cochlea are to separate the input acoustic signal into overlapping frequency bands, and to compress the large acoustic intensity range into the much smaller mechanical and electrical dynamic range of the inner hair cell. This is a basic question of information processing by the ear. The eye plays a similar role as a peripheral organ. It breaks the light image into rod- and cone-sized pixels, as it compresses the dynamic range of the visual signal. Based on the intensity JND, the corresponding visual dynamic range is about nine to

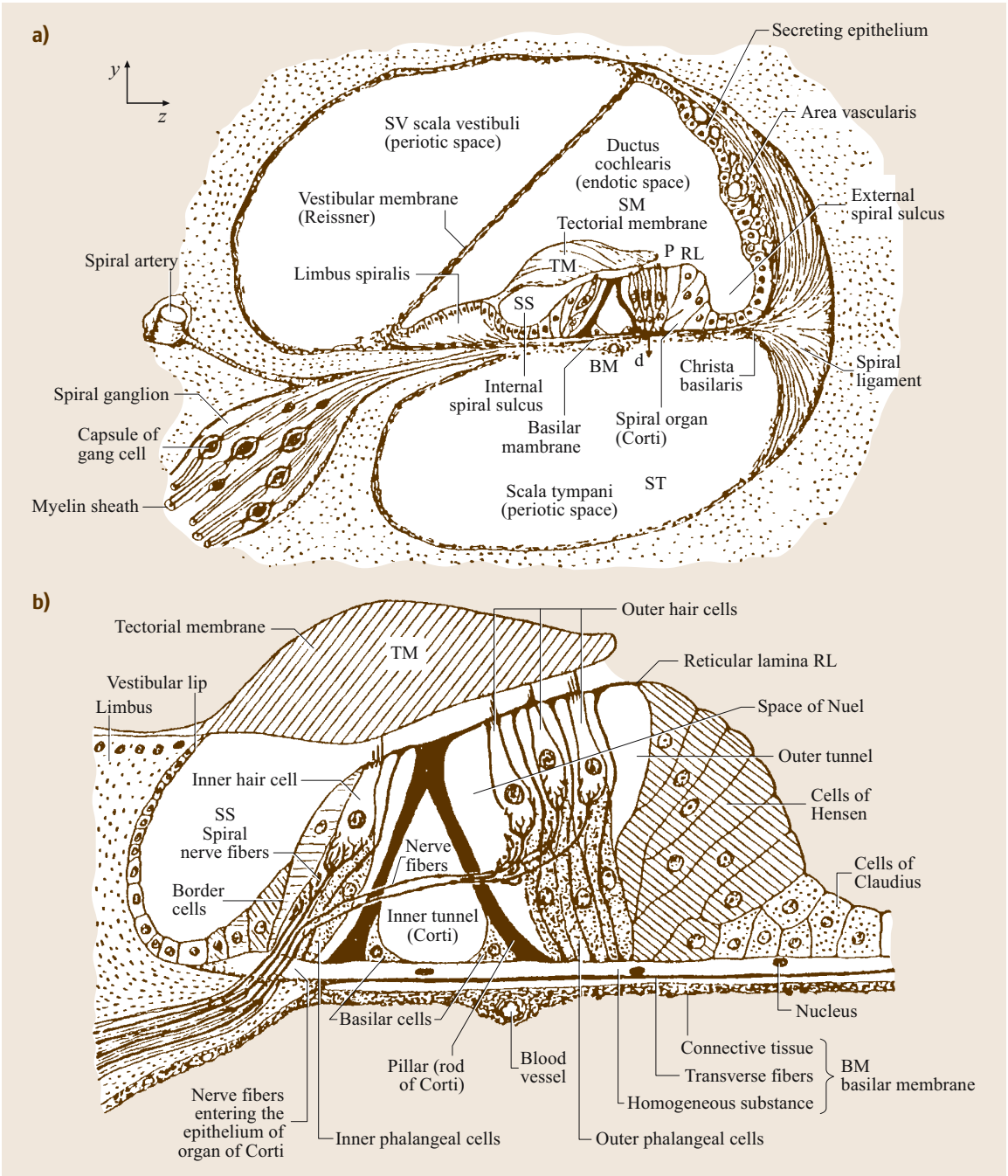



**Fig. 3.1a,b** On the left we see all the major structures of the cochlea **(a)**. The three chambers are filled with fluid. Reissner's membrane is an electrical barrier and is not believed to play a mechanical role. The right panel **(b)** shows the inner and outer hair cells, pillar cells and other supporting structures, the basilar membrane (BM), and the tectorial membrane (TM)

Amazon EX1010 Page 61

ten orders of magnitude of intensity [3.7, 8], while the ear has about 11 to 12. The stimulus has a relatively high information rate. Neurons are low-bandwidth channels. The eye and the ear must cope with this problem by reducing the stimulus to a large number of low bandwidth signals. It is then the job of the cortex to piece these pixel signals back together, to reconstruct the world as we see and hear it.

The acoustic information coding starts in the cochlea (Fig. 3.1a) which is composed of three major chambers formed by Reissner's membrane and the basilar membrane (BM). Mechanically speaking, there are only two chambers, as Reissner's membrane is only for electrical isolation of the scala media (SM) [3.4, 5]. Figure 3.1b shows a blown-up view of the organ of Corti where the inner hair cells (IHC) and outer hair cells (OHC) sit between the BM and the tectorial membrane (TM). As the BM moves up and down, the TM shears against the reticular lamina (RL), causing the cilia of the inner and outer hair cells to bend. The afferent auditory nerve fibers that are connected to the inner hair cells carry the signal information into the auditory system. Many fewer efferent fibers bring signals from the auditory system to the base of the outer hair cells. The exact purpose of these efferent fibers remains unknown.

### Inner Hair Cells

In very general terms, the role of the cochlea is to convert sound at the eardrum into neural pulse patterns along approximately 30 000 neurons of the human auditory (VIIIth) nerve. After being filtered by the cochlea, a low-level pure tone has a narrow spread of excitation which excites the cilia of about 40 contiguous inner hair cells [3.5, 9, 10]. The IHC excitation signal has a narrow bandwidth and a center frequency that depends on the inner hair cell's location along the basilar membrane. Each hair cell is about 10 μm in diameter while the human basilar membrane is about 35 mm in length (35 000 μm). Thus the neurons of the auditory nerve encode the responses of about 3500 inner hair cells which form a single row of cells along the length of the BM. Each inner-hair-cell voltage is a low-pass-filtered representation of the detected inner-hair-cell cilia displacement [3.11]. Each hair cell is connected to many neurons, having a wide range of spontaneous firing rates and thresholds [3.12]. In the cat, for example, approximately 15–20 neurons encode each of these narrow band inner hair cells with a neural timing code. It is commonly accepted that all mammalian cochleae are similar in function except the frequency range of operation differs between species (e.g., human $\approx 0.1-20$ kHz and cat $\approx 0.3-50$ kHz). It is widely believed that the neuron information channel between the hair cell and the *cochlear nucleus* is a combination of the mean firing rate and the relative timing between neural pulses (spikes). The mean firing rate is reflected in the loudness coding, while the relative timing carries more subtle cues, including for example pitch information such as speech voicing distinctions.

### Outer Hair Cells

As shown in Fig. 3.1b there are typically three (occasionally four) outer hair cells (OHCs) for each inner hair cell (IHCs), leading to approximately 12 000 OHCs in the human cochlea. Outer hair cells are used for intensity dynamic-range control. This is a form of NL signal processing, not dissimilar to Dolby sound processing. This form of processing was inspired by cochlear function, and was in use long before it was patented by Dolby, in movie sound systems developed by Bell Labs in the 1930s and 1940s. Telephone speech is similarly compressed [3.13] via $\mu$-law coding. It is well known (as was first proposed by *Lorente de Nó* [3.14] and *Steinberg* [3.3]) that noise damage of *nerve cells* (i. e., OHCs) leads to a reduction of dynamic range, a disorder clinically named *loudness recruitment*. The word *recruitment*, which describes the abnormal growth of loudness in the impaired ear, is a seriously misleading term, since nothing is being recruited [3.15].

We may describe cochlear processing two ways: first in terms of the signal representation at various points in the system; and second, in terms of models which are our most succinct means of conveying the conclusions of years of detailed and difficult experimental work on cochlear function. The body of experimental knowledge has been very efficiently represented (to the extent that it is understood) in the form of these mathematical models. When no model exists (e.g., because we do not understand the function), a more basic description via the experimental data is necessary. Several good books and review papers that make excellent supplemental reading are available [3.4, 8, 16, 17].

For pedagogical purposes this chapter has been divided into four parts. Besides this introduction, we include sections on the NL cochlea, neural masking, and finally a brief discussion. Section 3.2 discusses dynamic masking due to NL aspects of the cochlear outer hair cells. This includes the practical aspects, and theory, of the upward spread of masking (USM) and two-tone suppression. Section 3.3 discusses neural masking, the JND, loudness recruitment, the loudness signal-to-noise ratio

(SNR), and the Weber fraction. Section 3.4 provides a brief summary.

### 3.1.2 History of Cochlear Modeling

Typically the cochlea is treated as an uncoiled long thin box, as shown in Fig. 3.2a. This represents the starting point for the macromechanical models.

#### Macromechanics

In his book *On the Sensations of Tone Helmholtz* [3.18] likened the cochlea to a bank of highly tuned resonators selective to different frequencies, much like a piano or a harp [3.19, p. 22–58], with each string representing a different place $X$ on the basilar membrane. This model as proposed was quite limited since it leaves out key features, the most important of which is the cochlear fluid coupling between the mechanical resonators. But given the early publication date, the great master of physics and psychophysics Helmholtz shows deep insight and his studies provided many very important contributions.

The next major contribution by *Wegel* and *Lane* [3.20] stands in a class of its own even today, as a double-barreled paper having both deep psychophysical and modeling insight. Fletcher published much of the Wegel and Lane data one year earlier [3.21]. It is not clear to me why Wegel and Lane are always quoted for these results rather than Fletcher. In Fletcher's 1930 modeling paper, he mentioned that he was the subject in the Wegel and Lane study. It seems to me that Fletcher deserves some of the credit. The paper was the first to quantitatively describe the details of how a high level low frequency tone affects the audibility of a second low-level higher-frequency tone (i. e., the *upward spread of masking*). It was also the first publication to propose a *modern* model of the cochlea, as shown in Fig. 3.2b. If Wegel and Lane had been able to solve the model equations implied by their circuit (of course they had no computer to do this), they would have predicted cochlear traveling waves. It was their mistake, in my opinion, to make this a single paper. The modeling portion of their paper has been totally overshadowed by their experimental results. Transmission line theory had been widely exploited by Campbell, the first mathematical research at AT&T research (ca. 1898) with the invention of the wave filter [3.22, 23], which had been used for speech articulation studies [3.24–26], and Fletcher and Wegel were fully utilizing Campbell's important discoveries.

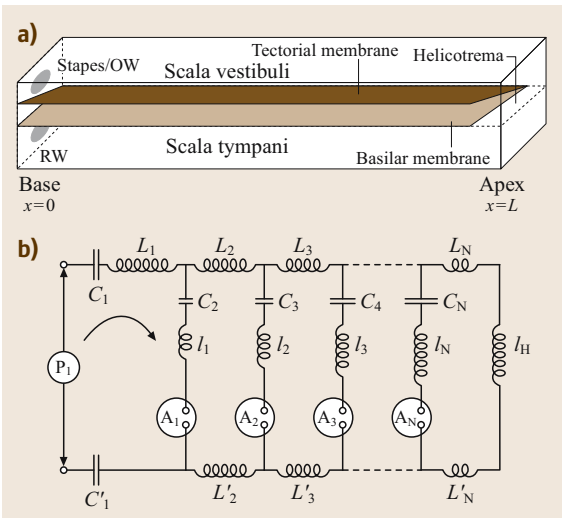



**Fig. 3.2a,b** On the left **(a)** see the basic 2-D box model of the cochlea. The *Base* ($x = 0$) is the high-frequency end of the cochlea while the *Apex* ($x = L$) carries the low frequencies. On the right **(b)** the 1924 Wegel and Lane electrical equivalent circuit. The model is built from a cascade of electrical sections

It was the experimental observations of G. von Békésy starting in 1928 on human cadaver cochleae which unveiled the physical nature of the basilar membrane traveling wave. What von Békésy found (consistent with the 1924 Wegel and Lane model) was that the cochlea is analogous to a *dispersive* transmission line where the different frequency components which make up the input signal travel at different speeds along the basilar membrane, thereby isolating each frequency component at a different place $X$ along the basilar membrane. He properly identified this dispersive wave a *traveling wave,* just as Wegel and Lane had predicted in their 1924 model of the cochlea.

Over the intervening years these experiments have been greatly improved, but von Békésy's fundamental observation of the traveling wave still stands. His original experimental results, however, are *not* characteristic of the responses seen in more-recent experiments, in many important ways. These differences are believed to be due to the fact that Békésy's cochleae were dead, and because of the high sound levels his experiments required. He observed the traveling wave using stroboscopic light, in dead human cochleae, at sound levels well above 140 dB – SPL.

Today we find that for a pure tone input the traveling wave has a more sharply defined location on the basilar membrane than that observed by von Békésy. In fact, according to measurements made over the last 20 years, the response of the basilar membrane to a pure tone

Amazon EX1010 Page 63

can change in amplitude by more than five orders of magnitude per millimeter of distance along the basilar membrane (e.g., 300 dB/oct is equivalent to 100 dB/mm in the cat cochlea).

### The One-Dimensional Model of the Cochlea

To describe this response it is helpful to call upon the macromechanical *transmission line models* of *Wegel* [3.20] (Fig. 3.2b) and *Fletcher* [3.27], first quantitatively analyzed by *Zwislocki* [3.28, 29], *Ranke* [3.30], *Peterson* and *Bogert* [3.31], *Fletcher* [3.32, 33]. This popular transmission line model is now denoted the *one-dimensional* (1-D), or *long-wave* model.

*Zwislocki* [3.28] was first to quantitatively analyze Wegel and Lane's macromechanical cochlear model, explaining Békésy's traveling wave observations. The stapes input pressure $P_1$ is at the left, with the input velocity $V_1$, as shown by the arrow, corresponding to the stapes velocity. This model represents the mass of the fluids of the cochlea as electrical inductors and the BM stiffness as capacitors. Electrical circuit networks are useful when describing mechanical systems. This is possible because of an electrical to mechanical analog that relates the two systems of equations. Electrical circuit elements comprise a de facto standard for describing such equations. It is possible to write down the equations that describe the system from the circuit of Fig. 3.2b, by those trained in the art. Engineers and scientists frequently find it easier to *read* and think in terms of these pictorial circuit diagrams, than to interpret the corresponding equations.

*BM Impedance.* During the following discussion it is necessary to introduce the concept of a *one-port* (two-wire) impedance. *Ohm's law* defines the impedance as

$$\text{Impedance} = \frac{\text{effort}}{\text{flow}} \,. \tag{3.1}$$

In an electrical system the impedance is the ratio of a voltage (effort) over a current (flow). In a mechanical system it is the force (effort) over the velocity (flow).

For *linear time-invariant causal* (LTIC) systems (i. e., an impedance), *phasor* notation is very useful, where the tone is represented as the real part (Re) of the complex exponential

$$\mathrm{e}^{\mathrm{i}2\pi ft+\mathrm{i}\phi} \equiv \cos(2\pi ft+\phi)+\mathrm{i}\sin(2\pi ft+\phi) \,. \tag{3.2}$$

The symbol $\equiv$ denotes *equivalence*. It means that the quantity to the left of $\equiv$ is defined by the quantity on the right. More specifically, impedance is typically defined in the frequency domain using *Laplace transform* notation, in terms of a damped tone

$$A\,\mathrm{e}^{\sigma t}\cos(2\pi ft+\phi) \equiv A\,\mathrm{Re}\,\mathrm{e}^{st+\mathrm{i}\phi} \tag{3.3}$$

excitation, characterized by the tone's amplitude $A$, phase $\phi$ and *complex Laplace frequency* $s \equiv \sigma + \mathrm{i}2\pi f$. When a function such as $Z(s)$ is shown as a function of the complex frequency $s$, this means that its inverse Laplace transform $z(t) \leftrightarrow Z(s)$ must be *causal*. In the time domain, the voltage may be found from the current via a convolution with $z(t)$. Three classic examples of such impedances are presented next.

*Example 3.1:* The impedance of the tympanic membrane (TM, or eardrum) is defined in terms of a pure tone pressure in the ear canal divided by the resulting TM volume velocity (the velocity times the area of TM motion) [3.34, 35]. The pressure (effort) and volume velocity (flow) referred to here are conventionally described using complex numbers, to account for the phase relationship between the two.

*Example 3.2:* The impedance of a spring is given by the ratio of the force $F(f)$ to velocity $V(f) = sX(f)$ with displacement $X$

$$Z(s) \equiv \frac{F}{V} = \frac{K}{s} = \frac{1}{sC} \,, \tag{3.4}$$

where the spring constant $K$ is the stiffness, $C$ the compliance, and $s$ is the complex radian frequency. The stiffness is represented electrically as a capacitor (as parallel lines in Fig. 3.2b). Having $s = \sigma + \mathrm{i}2\pi f$ in the denominator indicates that the impedance of a spring has a phase of $-\pi/2$ (e.g., $-90°$). Such a phase means that when the velocity is $\cos(2\pi ft)$, the force is $\sin(2\pi ft)$. This follows from Hooke's law

$$F = KX = \frac{K}{s}sX = \frac{K}{s}V \,. \tag{3.5}$$

*Example 3.3:* From *Newton's law* $F = Ma$ where $F$ is the force, $M$ is the mass, and acceleration $a(s) = sV(s)$ (i. e., the acceleration in the time domain is $\mathrm{d}v(t)/\mathrm{d}t$). The electrical element corresponding to a mass is an *inductor,* indicated in Fig. 3.2b by a coil. Thus for a mass $Z(s) = sM$.

From these relations the magnitude of the impedance of a spring decreases as $1/f$, while the impedance magnitude of a mass is proportional to $f$. The stiffness with

Amazon EX1010 Page 64

its −90° phase is called a *lagging* phase, while the mass with its +90° phase is called a *leading* phase.

Different points along the basilar membrane are represented by the cascaded sections of the lumped transmission line model of Fig. 3.2b. The position $X$ along the model is called the *place* variable and corresponds to the longitudinal position along the cochlea. The series (horizontal) inductors (coils) denoted by $L_k$ represent the fluid mass (inertia) along the length of the cochlea, while the shunt elements represent the mechanical (acoustical) impedance of the corresponding partition (organ of Corti) impedance, defined as the pressure drop across the partition divided by its volume velocity per unit length

$$Z_{\mathrm{p}}(s, X) = \frac{K_{\mathrm{p}}(X)}{s} + R_{\mathrm{p}}(X) + sM_{\mathrm{p}} \,, \tag{3.6}$$

where $K(X)$ is the partition stiffness, and $R_{\mathrm{p}}$ is the partition resistance. Each inductor going to ground ($l_i$ in Fig. 3.2b) represents the partition plus fluid mass per unit length $M_{\mathrm{p}}$ of the section. Note that $sM$, $R_{\mathrm{p}}$ and $K/s$ are impedances, but the mass $M$ and stiffness $K$ are not. The partition stiffness decreases exponentially along the length of the cochlea, while the mass is frequently approximated as being independent of place.

As shown in Fig. 3.3a, for a given input frequency the BM impedance magnitude has a local minimum at the shunt resonant frequency, where the membrane that can move in a relatively unrestricted manner. The shunt *resonance* has special significance because at this resonance frequency $F_{\mathrm{cf}}(X)$ the inductor and the capacitor reactance cancel each other, creating an acoustic *hole*, where the only impedance element that contributes to the flow resistance is $R_{\mathrm{p}}$. Solving for $F_{\mathrm{cf}}(X)$

$$\frac{K_{\mathrm{p}}(X)}{2\pi \mathrm{i} F_{\mathrm{cf}}} + 2\pi \mathrm{i} F_{\mathrm{cf}} M_{\mathrm{p}} = 0 \tag{3.7}$$

defines the *cochlear map function*, which is a key concept in cochlear modeling:

$$F_{\mathrm{cf}}(X) \equiv \frac{1}{2\pi} \sqrt{\frac{K_{\mathrm{p}}(X)}{M_{\mathrm{p}}}} \,. \tag{3.8}$$

The inverse of this function specifies the location of the *hole* $X_{\mathrm{cf}}(f)$ as shown in Fig. 3.3a. In the example of Fig. 3.3a two frequencies are show, at 1 and 8 kHz, with corresponding resonant points shown by $X_{\mathrm{cf}}(1)$ and $X_{\mathrm{cf}}(8)$.

Basal to $X_{\mathrm{cf}}(f)$ in Fig. 3.3a, the basilar membrane is increasingly stiff, and apically (to the right of the resonant point), the impedance is mass dominated. The above description is dependent on the input frequency $f$ since the location of the hole is frequency dependent. In this apical region the impedance has little influence since almost no fluid flows past the low-impedance hole. This description is key to our understanding of why the various frequency components of a signal are splayed out along the basilar membrane.

If one puts a pulse of current in at the stapes, the highest frequencies that make up the pulse would be shunted close to the stapes since at high frequencies the hole is near the stapes, while the lower frequencies would continue down the line. As the low-pass pulse travels down the basilar membrane, the higher frequencies are progressively removed, until almost nothing is left when the pulse reaches the end of the model (the helicotrema end, the apex of the cochlea).

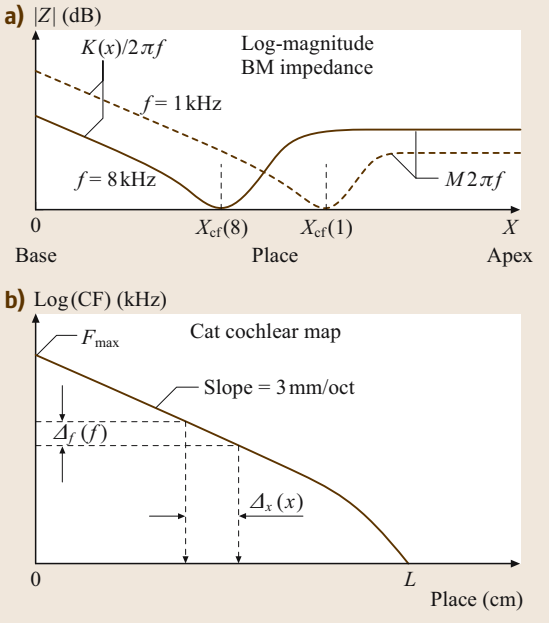



**Fig. 3.3** **(a)** Plot of the log-magnitude of the impedance as a function of place for two different frequencies of 1 and 8 kHz showing the impedance; the region labeled $K(X)$ is the region dominated by the stiffness and has impedance $K(X)/s$. The region labeled $M$ is dominated by the mass and has impedance $sM$. The characteristic places for 1 and 8 kHz are shown as $X_{\mathrm{cf}}$. **(b)** Cochlear map of the cat following Liberman and Dodds. The resonance frequency depends on place according to the *cochlear map function* **(b)**. A *critical bandwidth* $\Delta_f(f)$ and a *critical spread* $\Delta_x(X)$ area related through the cochlear map

Amazon EX1010 Page 65

When a single tone is played, the response in the base increases in proportion to the BM compliance (inversely with the stiffness) until there is is a local maximum just before the traveling wave reaches the resonant hole, at which point the response plummets, since the fluid flow is shorted by the hole. For a fixed stimulus frequency $f$ there is a maximum along the place axis called the *characteristic place*, denoted by $X_{\mathrm{cf}}^{(p)}(f)$. Likewise at a given place $X$ as a function of frequency there is a local maximum called the *characteristic frequency*, denoted by $F_{\mathrm{cf}}^{(p)}(X)$. The relation between the peak in place as a function of frequency or of the peak in frequency as a function of place is also called the *cochlear map*. There is serious confusion with conventional terminology here. The resonant frequency of the BM impedance mathematically defines $F_{\mathrm{cf}}$ and specifies the frequency on the base of the high-frequency steep portion of the tuning slope, *not* the peak. However the peak is used as the visual cue, *not* the base of the high-frequency slope. These two definitions differ by a small factor (that is ignored) that depends directly on the high-frequency slope of the response. Over most of the frequency range this slope is huge, resulting in a very small factor, justifying its being ignored. However at very low frequencies the slope is shallow and the factor can then be large. The *droop* in the cochlear map seen in Fig. 3.3b at the apex ($x = L$) may be a result of these conflicting definitions. The cochlear map function $F_{\mathrm{cf}}(X)$ plays a key role in cochlear mechanics, has a long history, and is known by many names [3.27, 36–40], the most common today being *Greenwood's function*. In the speech literature it is called the *Mel scale*.

The spread of the response around the peak for a fixed frequency is denoted the *critical spread* $\Delta_x(f)$, while the frequency spread at a given place is called the *critical band* denoted $\Delta_f(X)$. As early as 1933 it was clear that the critical band must exist, as extensively discussed by *Fletcher* and *Munson* [3.41]. At any point along the BM the critical band is proportional to the *critical ratio* $\kappa(X)$, defined as the ratio of pure tone detection intensity at threshold in a background of white noise, to the spectral level of the noise [3.42], namely

$$\Delta_f(X) \propto \kappa(X) \,. \tag{3.9}$$

In the next section we shall show how the the relations between these various quantities are related via the cochlear map.

*Derivation of the Cochlear Map Function.* The derivation of the cochlear map is based on *counting* critical bands as shown by *Fletcher* [3.10] and popularized by *Greenwood* [3.43]. The *number of critical bands* $N_{\mathrm{cb}}$ may be found by integrating the critical band density over both frequency and place, and equating these two integrals, resulting in the cochlear map $F_{\mathrm{cf}}(X)$:

$$N_{\mathrm{cb}} \equiv \int_0^{X_{\mathrm{cf}}} \frac{\mathrm{d}X}{\Delta_x(X)} = \int_0^{F_{\mathrm{cf}}} \frac{\mathrm{d}f}{\Delta_f(f)} \,. \tag{3.10}$$

There are approximately 20 pure-tone frequency JNDs per critical band [3.37], [3.42, p. 171], and *Fletcher* showed that the *critical ratio* expressed in dB $\kappa_{\mathrm{dB}}(X)$ is of the form $aX + b$, where $a$ and $b$ are constants [3.10]. As verified by *Greenwood* [3.43, p. 1350, (1)] the critical bandwidth in Hz is therefore

$$\Delta_f(X) \propto 10^{\kappa_{\mathrm{dB}}(X)/10} \,. \tag{3.11}$$

The critical spread $\Delta_x(X)$ is the effective width of the energy spread on the basilar membrane for a pure tone. Based on a suggestion by Fletcher, *Allen* showed that for the cat, $\Delta_x(X)$ corresponds to about 2.75 times the basilar membrane width with $W_{\mathrm{bm}}(X) \propto \mathrm{e}^X$ [3.10]. It is reasonable to assume that the same relation would hold in the human case.

The direct observation of the cochlear map in the cat was made by *Liberman* [3.44] and *Liberman* and *Dodds* [3.45], and they showed the following empirical formula fit the data

$$F_{\mathrm{cf}}(X) = 456(10^{2.1(1-X/L)} - 0.8) \,, \tag{3.12}$$

where the length of the cat cochlea is $L = 21$ mm, and $X$ is measured from the stapes [3.44]. The same formula may be used for the human cochlea if $L = 35$ mm is used, the 456 is replaced by 165.4, and 0.8 by 0.88. Based on (3.12), and as defined in Fig. 3.3b, the *slope* of the cochlear map is 3 mm/oct for the cat and 5 mm/oct for the human, as may be determined from the formula $L \log_{10}(2)/2.1$ with $L = 21$ or 35 for cat and human, respectively.

For a discussion of work after 1960 on the critical band see *Allen* [3.10] and *Hartmann* [3.17].

Amazon EX1010 Page 66

## 3.2 The Nonlinear Cochlea

### 3.2.1 Cochlear Modeling

In cochlear modeling there are two fundamental intertwined complex problems, *cochlear frequency selectivity* and *cochlear/OHC nonlinearity*. Wegel and Lane's 1924 transmission line wave theory was a most important development, since it was published 26 years prior to the experimental results of von Békésy, and it was based on a simple set of physical principles, conservation of fluid mass, and a spatially variable basilar membrane stiffness. It gives insight into both the NL cochlea, as well has two-dimensional (2-D) model frequency-selective wave-transmission effects (mass loading of the BM).

Over a 15 year period starting in 1971, there was a paradigm shift. Three discoveries rocked the field:

1. nonlinear compressive basilar membrane and inner-hair-cell measures of neural-like cochlear frequency selectivity [3.47, 48],
2. otoacoustic (ear canal) nonlinear emissions [3.49], and
3. motile outer hair cells [3.50].

Today we know that these observations are related, and all involve outer hair cells. A theory (e.g., a computational model) is needed to tie these results together. Many groups are presently working out such theories.

On the modeling side during the same period (the 1970's) all the variants of Wegel and Lane 1-D linear theory were becoming dated because:

1. numerical model results became available, which showed that 2- and three-dimensional (3-D) models were more frequency selective than the 1-D model,
2. experimental basilar membrane observations showed that the basilar membrane motion had a nonlinear compressive response growth, and
3. improved experimental basilar membrane observations became available which showed increased nonlinear cochlear frequency selectivity.

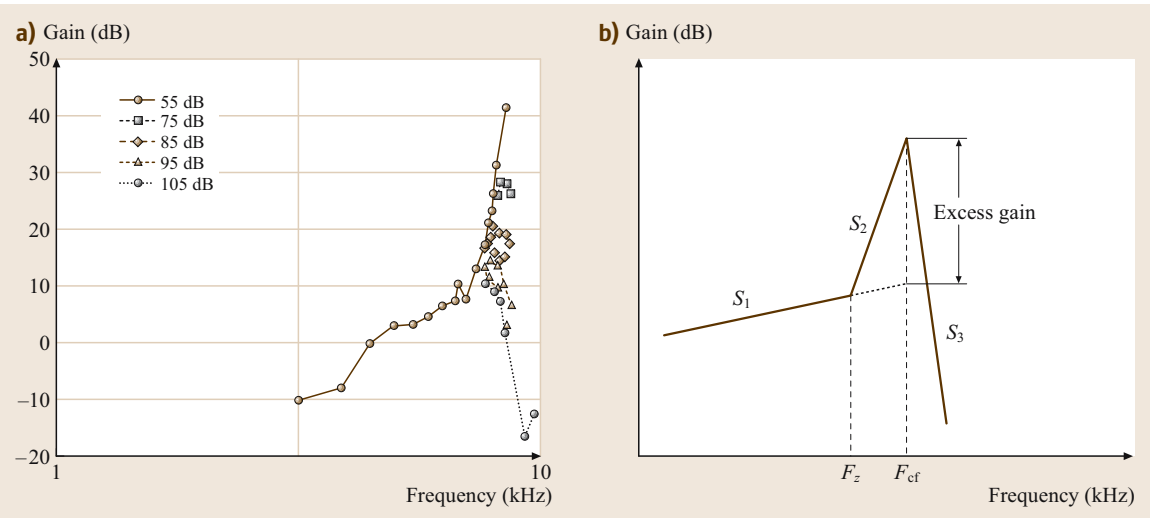



**Fig. 3.4a,b** There are six numbers that characterize every curve, three slopes ($S_1$, $S_2$, $S_3$), in dB/oct, two frequencies ($F_z$, $F_{cf}$), and the *excess gain* characterizes the amount of gain at $F_{cf}$ relative to the gain defined by $S_1$. The excess gain depends on the input level for the case of a nonlinear response like the cochlea. Rhode found up to $\approx 35$ dB of excess gain at 7.4 kHz and 55 dB − SPL, relative to the gain at 105 dB − SPL. From of the 55 dB − SPL curve of **(a)** (the most sensitive case), and his Table I, $S_1 = 9$, $S_2 = 86$, and $S_3 = -288$ (dB/oct), $F_z = 5$ kHz, $F_{cf} = 7.4$ kHz, and an *excess gain* of 27 dB. Rhode reported $S_1 = 6$ dB/oct, but 9 seems to be a better fit to the data, so 9 dB/oct is the value we have used for our comparisons. **(a)** Response of the basilar membrane for his most sensitive animal. The graduations along the abscissa are at 0.1, 1.0 and 10.0 kHz (after [3.46, Fig. 9a]) **(b)** Basic definition of the 6 parameters for characterizing a tuning curve: slopes $S_1$, $S_2$, $S_3$, frequencies $F_z$ and $F_{cf}$, and the excess gain

Because these models and measures are still under development today [the problem has not yet (ca. 2007) been solved], it is necessary to describe the data rather than the models. Data that drives these nonlinear cochlear measures include:

- The upward spread of masking (USM), first described quantitatively by Wegel and Lane in 1924,
- Distortion components generated by the cochlea and described by *Wegel* and *Lane* [3.20], *Goldstein* and *Kiang* [3.52], *Smoorenburg* [3.53], *Kemp* [3.54], *Kim* et al. [3.55], *Fahey* and *Allen* [3.56] and many others,
- Normal loudness growth and recruitment in the impaired ear [3.3, 41],
- The frequency dependent neural two-tone suppression observed by *Sachs* and *Kiang* [3.57], *Arthur* et al. [3.58], *Kiang* and *Moxon* [3.59], *Abbas* and *Sachs* [3.60], *Fahey* and *Allen* [3.56], *Pang* and *Guinan* [3.61], and others,
- The frequency-dependent basilar membrane response-level compression first described by *Rhode* [3.46, 47],
- The frequency-dependent inner-hair-cell receptor potential level compression, first described by *Sellick* and *Russell* [3.48], *Russell* and *Sellick* [3.62].
- Forward masking data that shows a linear return to baseline after up to 0.2 s [3.63]. There may be compelling evidence that OHCs are the source of forward masking.

We shall discuss each of these, but two related measures are the most important for understanding these NL masking effects, the upward spread of masking (USM) and two-tone suppression (2TS).

*Basilar Membrane Nonlinearity.* The most basic early and informative of these nonlinear effects was the NL basilar membrane measurements made by *Rhode* [3.46, 47], as shown in Fig. 3.4a, showing that the basilar membrane displacement to be a highly NL function of level. For every four dB of pressure level increase on the input, the output displacement (or velocity) only changed one dB. This compressive nonlinearity depends on frequency, and only occurs near the most sensitive region (e.g., the tip of the tuning curve). For other frequencies the system was either linear, namely, one dB of input change gave one dB of output change for frequencies away from the best frequency, or very close to linear. This NL effect was highly dependent on the health of the animal, and would decrease or would not be present at all, when the animal was not in its physiologically pristine state.

An important and useful measure of cochlear linear and nonlinear response first proposed by *Rhode* [3.46, Fig. 8], is shown in Fig. 3.4b which describes cochlear tuning curves by straight lines on log–log coordinates. Such straight line approximations are called *Bode plots* in the engineering literature. The *slopes* and *break points*, defined as the locations where the straight lines cross, characterize the response.

*Otoacoustic Emissions.* A few years after Rhode's demonstration of cochlear nonlinearity, David Kemp observed otoacoustic emissions (tonal sound emanating from the cochlea and NL *echos* to clicks and tone bursts) [3.49, 54, 64–66]. Kemp's findings were like a jolt to the field, which led to a cottage industry of objective testing of the auditory system, including both cochlear and middle ear tests.

*Motile OHCs.* Subsequently, *Brownell* et al. [3.50] discovered that isolated OHCs change their length when placed in an electric field, thus that the outer hair cell is motile. This then led to the intuitive and widespread proposal that outer hair cells act as voltage-controlled motors that directly drive the basilar membrane on a cycle by cycle basis. It seems quite clear, from a great deal of data, that the OHC onset response time is on the order of one cycle or so of the BM impulse response, because the first peak is linear [3.67]. The release time must be determined by the OHC membrane properties, which is slow relative to the attack. Thus OHC NL processing is the basis for both the frequency asymmetry of simultaneous (upward versus downward spread) and temporal (forward versus backward) masking.

As summarized in Fig. 3.5, OHCs provide feedback to the BM via the OHC receptor potential, which in turn is modulated by both the position of the basilar

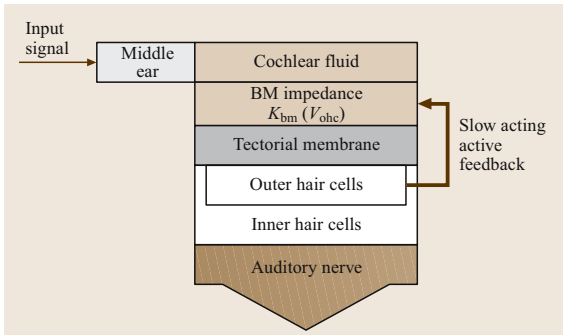



**Fig. 3.5** Block flow diagram of the inner ear (after *Allen* [3.51])

membrane (forming a fast feedback loop), and alternatively by the efferent neurons that are connected to the outer hair cells (forming a slow feedback loop). The details of all this are the topic of a great deal of present research.

OHCs are the one common element that link all the NL data previously observed, and a missing piece of the puzzle that most needs to be understood before any model can hope to succeed in predicting basilar membrane, hair cell, and neural tuning, or NL compression. Understanding the outer hair cell's two-way mechanical transduction is viewed as the key to solving the problem of the cochlea's dynamic range.

Historically the implication that hair cells might play an important role in cochlear mechanics go back at least to 1936 when loudness recruitment was first reported by *Fowler* [3.68] in a comment by *R. Lorente de Nó* [3.14] stating that cochlear hair cells are likely to be involved in loudness recruitment.

The same year *Steinberg* and *Gardner* [3.3] were explicit about the action of recruitment when they concluded:

> *When someone shouts, such a deafened person suffers practically as much discomfort as a normal hearing person would under the same circumstances. Furthermore for such a case, the effective gain in loudness afforded by amplification depends on the amount of variable type loss present. Owing to the expanding action of this type of loss it would be necessary to introduce a corresponding compression in the amplifier in order to produce the same amplification at all levels.*

Therefore as early as 1937 there was a clear sense that cochlear hair cells were related to dynamic range compression.

More recently, theoretical attempts to explain the difference in tuning between normal and damaged cochleae led to the suggestion that OHCs could influence BM mechanics. In 1983 *Neely* and *Kim* [3.69] concluded:

> *We suggest that the negative damping components in the model may represent the physical action of outer hair cells, functioning in the electrochemical environment of the normal cochlea and serving to boost the sensitivity of the cochlea at low levels of excitation.*

In 1999 yet another (a fourth) important discovery was made, that the outer-hair-cell mechanical stiffness depends on the voltage across its membrane [3.70, 71]. This change in stiffness, coupled with the naturally occurring internal static pressure, may well account for the voltage dependent accompanying length changes (the cell's voltage dependent motility). This view follows from the block diagram feedback model of the organ of Corti shown in Fig. 3.5 where the excitation to the OHC changes the cell voltage $V_{ohc}$, which in turn changes the basilar stiffness [3.51]. This is one of several possible theories that have been put forth.

This experimental period set the stage for explaining the two most dramatic NL measures of cochlear response, the upward spread of masking and its related neural correlate, two-tone suppression, and may well turn out to be the explanation of the nonlinear forward-masking effect as well [3.63].

### Simultaneous Dynamic-Masking

The psychophysically measured *upward spread of masking* (USM) and the neurally measured *two-tone suppression* (2TS) are closely related dynamic-masking phenomena. Historically these two measures have been treated independently in the literature. As will be shown, it is now clear that they are alternative objective measures of the same OHC compressive nonlinearity. Both involve the dynamic suppression of a basal (high-frequency) probe due to the simultaneous presentation of an apical (low-frequency) suppressor. These two views (USM versus 2TS) nicely complement each other, providing a symbiotic view of cochlear nonlinearity.

*Upward Spread of Masking (USM).* In a classic paper, *Mayer* [3.72] was the first to describe the asymmetric nature of masking [3.63,73]. Mayer made his qualitative observations with the use of clocks, organ pipes and tuning forks, and found that that the spread of masking is a strong function of the probe-to-masker frequency ratio ($f_p/f_m$) [3.63].

In 1923, Fletcher published the first quantitative results of tonal masking. In 1924, Wegel and Lane extended Fletcher's experiments (Fletcher was the subject [3.27, p. 325]) using a wider range of tones. Wegel and Lane then discuss the results in terms of their 1-D model described above. As shown in Fig. 3.6a, Wegel and Lane's experiments involved presenting listeners with a masker tone at frequency $f_m = 400\,\text{Hz}$ and intensity $I_m$ (the abscissa), along with a probe tone at frequency $f_p$ (the parameter used in the figure). At each masker intensity and probe frequency, the threshold probe intensity $I_p^*(I_m)$ is determined, and displayed relative to its threshold *sensation level* (SL) (the ordinate

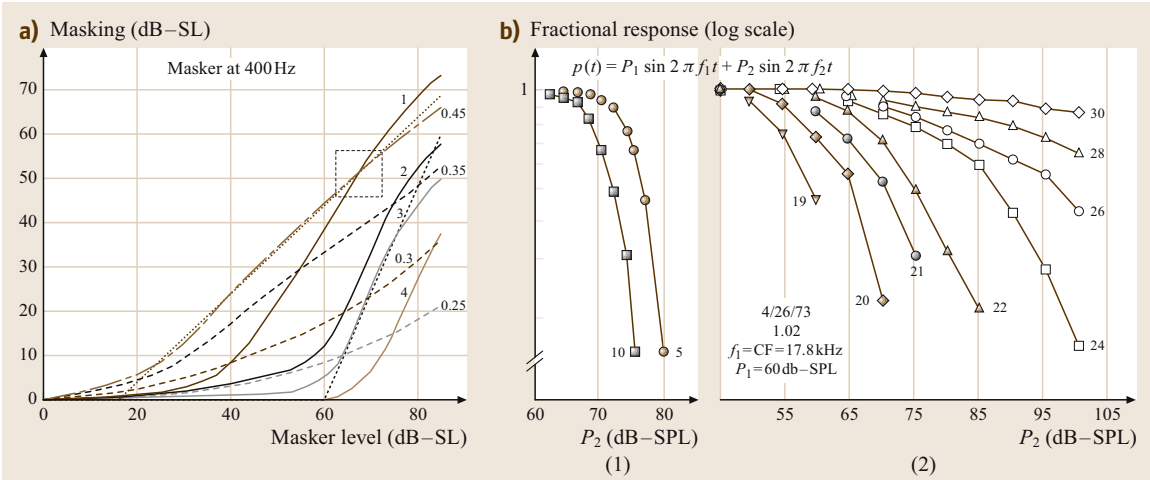



**Fig. 3.6a,b** On the left **(a)** we see the psychoacoustic measure of 2TS, called the upward spread of masking. On the right **(b)** are related measures taken in the auditory nerve by a procedure called two-tone suppression (2TS). Low- and high-side masking or suppression have very different thresholds and slopes. These suppression slopes and thresholds are very similar between 2TS and the USM. **(a)** Upward spread of masking as characterized by Wegel and Lane in 1924. The *solid lines* correspond to the probe being higher than the 400 Hz masker, while the *dashed lines* correspond to the 400 Hz probe lower than the masker. On the *left* we see upward spread of masking functions from Wegel and Lane for a 400 Hz low-frequency masker. The abscissa is the masker intensity $I_m$ in dB − SL while the ordinate is the threshold probe intensity $I_p^*(I_m)$ in dB − SL. The frequency of the probe $f_p$, expressed in kHz, is the parameter indicated on each curve. The *dashed box* shows that the masking due to a 1 kHz tone becomes more than that at 450 Hz, for a 400 Hz probe. This is the first observation of *excitation pattern migration* with input intensity. **(b)** Two-tone suppression (2TS) input–output (IO) functions from *Abbas* and *Sachs* [3.60, Fig. 8]. On the *left* (1) is low-side suppression and on the *right* (2) we see high-side suppression. In 2TS the suppressor plays the role of the masker and the probe the role of the maskee. Note that the threshold of suppression for low-side suppressor (masker) is close to 70 dB − SPL, which is similar to human low-side suppressors, the case of the Wegel and Lane USM (1) (60–70 dB − SPL). The onset of suppression for high-side suppressors is close to the neuron's CF threshold of 50 dB, as elaborated further in Fig. 3.7a

is the probe level at threshold [dB − SL]). The asterisk indicates a threshold measure.

In Fig. 3.6a $f_m = 400$ Hz, $I_m$ is the abscissa, $f_p$ is the parameter on each curve, in kHz, and the threshold probe intensity $I_p^*(I_m)$ is the ordinate. The dotted line superimposed on the 3 kHz curve $(I_m/10^{60/10})^{2.4}$ represents the suppression threshold at 60 dB − SL which has a slope of 2.4 dB/dB. The dotted line superimposed on the 0.45 kHz curve has a slope of 1 and a threshold of 16 dB − SL.

Three regions are clearly evident: the *downward spread* of masking ($f_p < f_m$, dashed curves), *critical band* masking ($f_p \approx f_m$, dashed curve marked 0.45), and the *upward spread* of masking ($f_p > f_m$, solid curves) [3.74].

Critical band masking has a slope close to 1 dB/dB (the superimposed dotted line has a slope of 1). Four years later *Riesz* [3.75] shows critical band masking obeys the *near miss to Weber's law*, as described in Sect. 3.3.2. The downward spread of masking (the dashed lines in Fig. 3.6a) has a low threshold intensity and a variable slope that is less than one dB/dB, and approaches 1 at high masker intensities. The upward spread of masking (USM), shown by the solid curves, has a threshold near 50 dB re sensation level (e.g., 65 dB − SPL), and a growth just less than 2.5 dB/dB. The dotted line superimposed on the $f_p = 3$ kHz curve has a slope of 2.4 dB/dB and a threshold of 60 dB − SL.

The dashed box shows that the upward spread of masking of a probe at 1 kHz can be greater than the masking within a critical band (i. e., $f_p = 450$ Hz $> f_m = 400$ Hz). As the masker frequency is increased, this *crossover effect* occurs in a small frequency region (i. e., 1/2 octave) above the masker frequency. The crossover is a result of a well-documented NL *response migration*, of the excitation pattern with

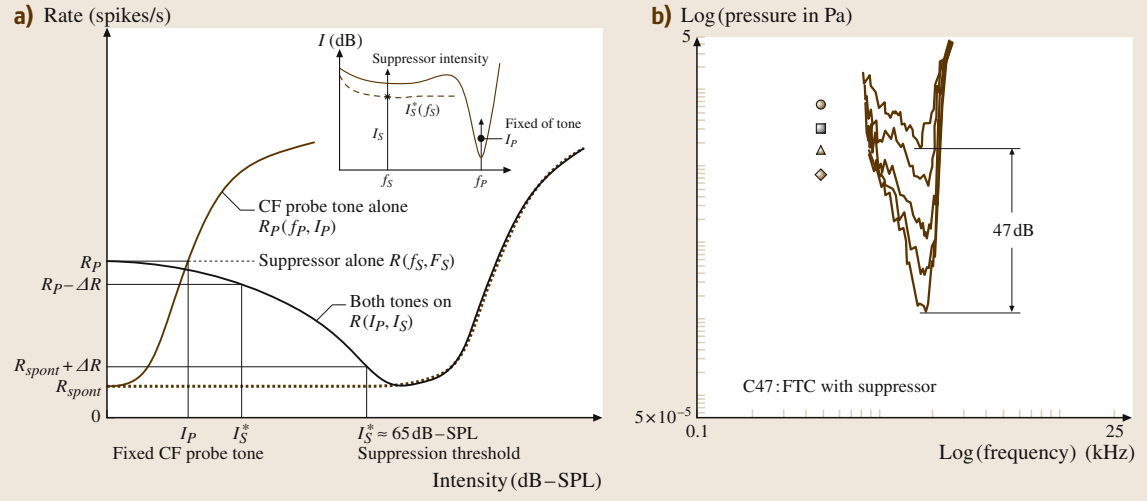



**Fig. 3.7** **(a)** Definitions of 2TS low-side masking procedure (see (3.13) and (3.14)). **(b)** Example of 2TS (low-side masking) in the cat auditory nerve (AN). A cat neural tuning curve taken with various *low-side* suppressors present (suppressor below the best frequency), as indicated by the symbols. The tuning curve with the lowest threshold is for no suppressor. When the suppressor changes by 20 dB, the $F_{cf}$ threshold changes by 36 dB. Thus for a 2 kHz neuron, the slope is 36/20, or 1.8. These numbers are similar to those measure by *Delgutte* [3.80]. One Pa = 94 dB – SPL

stimulus intensity, described in a wonderful paper by *McFadden* [3.76]. Response migration was also observed by *Munson* and *Gardner* in a classic paper on forward masking [3.77]. This important migration effect is beyond the scope of the present discussion, but is reviewed in [3.74, 78, 79] discussed in the caption of Fig. 3.10.

The upward spread of masking is important because it is easily measured psychophysically in normal hearing people, is robust, well documented, and nicely characterizes normal outer-hair-cell nonlinearities. The psychophysically measured USM has correlates in basilar membrane and hair cell signals, and is known as two-tone suppression (2TS) in the auditory nerve literature, as discussed in the caption of Fig. 3.6b.

*Two-Tone Suppression.* The neural correlate of the psychophysically measured USM is called *two-tone suppression* (2TS). As shown in the insert of Fig. 3.7a, first a neural tuning curve is measured. A pure tone probe at intensity $I_p(f_p)$, and frequency $f_p$, is placed a few dB (e.g., 6 to 10) above threshold at the characteristic (best) frequency of the neuron $F_{cf}$ (i. e., $f_p = F_{cf}$). In 2TS a suppressor tone plays the role of the masker. There are two possible *thresholds*. The intensity of the suppressor tone $I_s(f_s)$ at frequency $f_s$ is increased until either

1. the rate response to either the probe alone $R(I_p, I_s = 0)$ decreases by a small increment $\Delta_R$, or
2. drops to the small increment $\Delta_R$, just above the undriven spontaneous rate $R(0, 0)$.

These two criteria are defined in Fig. 3.6b and may be written

$$R_p\left(I_p, I_s^*\right) \equiv R(I_p, 0) - \Delta_R \tag{3.13}$$

and

$$R_{spont}\left(I_p, I_s^*\right) \equiv R(0, 0) + \Delta_R\,; \tag{3.14}$$

$\Delta_R$ indicates a fixed small but statistically significant constant change in the rate (e.g., $\Delta_R = 20$ spikes/s is a typical value). The threshold suppressor intensity is defined as $I_s^*(f_s)$, and as before the $*$ indicates the threshold suppressor intensity. The two threshold definitions (3.13) and (3.14) are very different, and both are useful. The difference in intensity between the two thresholds is quite large, and the more common measure used by *Abbas* and *Sachs* [3.60] is (3.13). The second measure (3.14) is consistent with neural tuning curve suppression, and is therefore the more interesting of the two. It corresponds to suppression of the probe to threshold.

Neural data of *Abbas* and *Sachs* [3.60, Fig. 8] are reproduced in Fig. 3.6b. For this example (see entry in lower-right just below 105), $F_{cf}$ is 17.8 kHz, and the

$f_p = F_{cf}$ probe intensity $20 \log 10(|P_1|)$ is 60 dB. The label on the curves is the frequency $f_1$. The threshold intensity of the associated neural tuning curve is has a low spontaneous rate and a 50–55 dB threshold. The left panel of Fig. 3.6b is for apical suppressors that are lower in frequency than the characteristic frequency (CF) probe ($f_s < f_p$). In this case the threshold is just above 65 dB − SPL. The suppression effect is relatively strong and almost independent of frequency. In this example the threshold of the effect is less than 4 dB apart (the maximum shift of the two curves) at suppressor frequencies $f_s$ of 10 and 5 kHz (a one octave separation).

The right panel shows the case $f_s > f_p$. The suppression threshold is close to the neuron's threshold (e.g., 50 dB − SPL) for probes at 19 kHz, but increases rapidly with frequency. The strength of the suppression is weak in comparison to the case of the left panel ($f_s < f_p$), as indicated by the slopes of the family of curves.

*The Importance of the Criterion.* The data of Fig. 3.6b uses the first suppression threshold definition (3.13) $R_p$ (a small drop from the probe driven rate). In this case the $F_{cf}$ probe is well above its detection threshold at the suppression threshold, since according to definition (3.13), the probe is just detectably reduced, and thus audible. With the second suppression threshold definition (3.14), the suppression threshold corresponds to the detection threshold of the probe. Thus (3.14), *suppression to the spontaneous rate*, is appropriate for Wegel and Lane's masking data where the probe is at its detection threshold $I_p^*(I_m)$. Suppression threshold definition (3.14) was used when taking the 2TS data of Fig. 3.7b, where the suppression threshold was estimated as a function of suppressor frequency.

To be consistent with a detection threshold criterion, such as the detection criterion used by Wegel and Lane in psychophysical masking, (3.14) must be used. To have a tuning curve pass through the $F_{cf}$ probe intensity of a 2TS experiment (i. e., be at threshold levels), it is necessary to use the suppression to rate criterion given by (3.14). This is shown in Fig. 3.7b where a family of tuning curves is taken with different suppressors present. As described by *Fahey* and *Allen* [3.56, Fig. 13], when a probe is placed on a specific tuning curve of Fig. 3.7b, corresponding to one of the suppressor level symbols of Fig. 3.7b, and a suppression threshold is measured, that suppression curve will fall on the corresponding suppression symbol of Fig. 3.7b. There is a symmetry between the tuning curve measured in the presents of a suppressor, and a suppression threshold obtained with a given probe. This symmetry only holds for criterion (3.14), the detection threshold criterion, which is appropriate for Wegel and Lane's data. If one uses (3.13) as in [3.60] they will not see this symmetry as cleary.

*Suppression Threshold.* Using the criterion (3.14), *Fahey* and *Allen* [3.56, Fig. 13] showed that the suppression threshold $I_s^*(I_p)$ in the tails is near 65 dB − SPL (0.04 Pa). This is true for suppressors between 0.6 and 4 kHz. A small amount of data are consistent with the threshold being constant to much higher frequencies, but the Fahey and Allen data are insufficient on that point.

*Suppression Slope.* *Delgutte* has written several insightful papers on masking and suppression [3.80–82]. He estimated how the intensity growth slope (in dB/dB) of 2TS varies with suppressor frequency for several probe frequencies [3.80]. As may be seen in his figure, the suppression growth slope for the case of a low frequency apical suppressor on a high frequency basal neuron (the case of the left panel of Fig. 3.6b), is $\approx 2.4$ dB/dB. This is the same slope as for Wegel and Lane's 400 Hz masker, 3 kHz probe USM data shown in Fig. 3.6a. For suppressor frequencies greater than the probe's ($f_s > f_p$), Delgutte reports a slope that is significantly less than 1 dB/dB. Likewise Wegel and Lane's data has slopes much less than 1 for the downward spread of masking.

One may conclude that USM and 2TS data show systematic and quantitative correlations between the threshold levels and slopes. The significance of these correlations has special importance because

1. they come from very different measurement methods, and
2. Wegel and Lane's USM are from human, while the 2TS data are from cat, yet they show similar responses. This implies that the cat and human cochleae may be quite similar in their NL responses.

The USM and 2TS threshold and growth slope (e.g., 50 dB − SL and 2.4 dB/dB) are important features that must be fully understood and modeled before we can claim to understand cochlear function. While there have been several models of 2TS [3.83–85] as discussed in some detail by *Delgutte* [3.80], none are in quantitative agreement with the data. The two-tone suppression model of *Hall* [3.84] is an interesting contribution to this problem because it qualitatively explores many of the key issues. Finally forward-masking data also show related nonlinear properties that we speculate may turn out to be related to NL OHC function as well [3.78, 86, 87].

Amazon EX1010 Page 72

### 3.2.2 Outer-Hair-Cell Transduction

The purpose of this section is to address two intimately intertwined problems *cochlear frequency selectivity* and *cochlear nonlinearity*. The fundamental question in cochlear research today is: *What is the role of the outer hair cell (OHC) in cochlear mechanics?* The OHC is the source of the NL effect, and the end product is dynamic masking, including the USM, 2TS and forward masking, all of which include dramatic amounts of gain and tuning variation. The issues are the nature of the NL transformations of the BM, OHC cilia motion, and OHC soma motility, at a given location along the basilar membrane.

The prevailing and popular *cochlear-amplifier* view is that the OHC provides *cochlear sensitivity* and *frequency selectivity* [3.5, 88–94]. The alternative view, argued here, is that the OHC compresses the excitation to the inner hair cell, thereby providing dynamic range expansion.

There is an important difference between these two views. The *first* view deemphasizes the role of the OHC in providing dynamic range control (the OHC's role is to improve sensitivity and selectivity), and assumes that the NL effects result from OHC saturation.

The *second* view places the dynamic range problem as the top priority. It assumes that the sole purpose of the OHC nonlinearity is to provide dynamic range compression, and that the OHC plays no role in either sensitivity or selectivity, which are treated as important but independent issues. Of course other views besides these two are possible.

#### The Dynamic-Range Problem

The question of how the large (up to 120 dB) dynamic range of the auditory system is attained has been a long standing problem which remains fundamentally incomplete. For example, *recruitment*, the most common symptom of neurosensory hearing loss, is best characterized as the loss of dynamic range [3.3, 10, 15, 95]. Recruitment results from outer-hair-cell damage [3.96]. To successfully design hearing aids that deal with the problem of recruitment, we need models that improve our understanding of *how* the cochlea achieves its dynamic range.

Based on a simple analysis of the IHC voltage, one may prove that the dynamic range of the IHC must be less than 65 dB [3.97]. In fact it is widely accepted that IHC dynamic range is less than 50 dB.

The IHC's transmembrane voltage is limited at the high end by the cell's open circuit (unloaded) membrane voltage, and at the low end by thermal noise. There are two obvious sources of thermal noise, cilia Brownian motion, and Johnson (shot) noise across the cell membrane (Fig. 3.8).

The obvious question arises: *How can the basic cochlear detectors (the IHCs) have a dynamic range of less than* 50 dB *(a factor of* $0.3\times10^2$*), and yet the auditory system has a dynamic range of up to* 120 dB

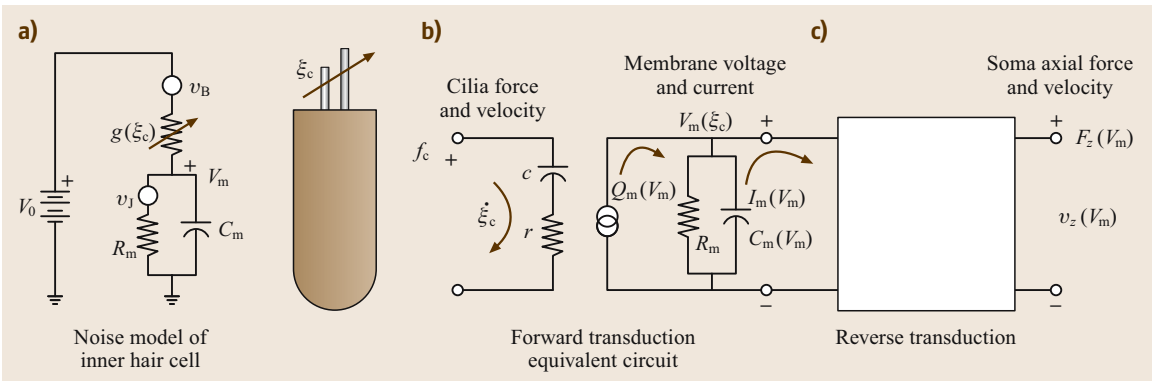



**Fig. 3.8a–c** On the far left **(a)** is the electrical equivalent circuit model of an IHC with thermal noise sources due to the cell leakage resistance Johnson and shot noise $v_J$ and the Brownian motion of the cilia, represented by the voltage noise source $v_B$. The cilia force $f_c$ and velocity $\dot{\xi}_c$ are the stimulus (input) variables to the forward transduction **(b)**, and are loaded by the mechanical impedance of the cilia viscous drag $r$ and compliance $c$. **(c)** For OHCs, when the cilia move, current flows into the cell charging the membrane capacitance, thus changing the membrane voltage $V_m$. This membrane capacitance $C_m(V_m)$ is voltage dependent (i. e., it is NL). The membrane voltage has also been shown to control the cell's soma axial stiffness. It follows that the axial force $F_z(V_m)$ the cell can deliver, and the axial velocity $\mathcal{V}_z(V_m)$ of the cell, must also depend on the membrane voltage. The precise details of how all this works is unknown

*(a factor of* $10^6$*)?* The huge amount of indirect evidence has shown that this increased dynamic range results from mechanical NL signal compression provided by outer hair cells. This dynamic-range compression shows up in auditory psychophysics and in cochlear physiology in many ways.

This thus forms the basic dynamic-range dilemma.

#### Outer-Hair-Cell Motility Model

A most significant finding in 1985 was of OHC *motility*, namely that the OHC changes its length by up to 5% in response to the cell's membrane voltage [3.50, 99, 100]. This less than 5% change in length must account for a 40 dB (100 times) change in cochlear sensitivity. This observation led to a significant increases in research on the OHC cell's motor properties.

In 1999 it was shown that the cell's longitudinal soma stiffness changes by at least a factor of 2 ($> 100\%$), again as a function of cell membrane voltage [3.70, 71]. A displacement of the cilia in the direction of the tallest cilia, which is called a *depolarizing* stimulus, decreases the magnitude of the membrane voltage $|V_m|$, *decreases* the longitudinal soma stiffness, and *decreases* the cell soma length. A hyperpolarizing stimulus increases the stiffness and extends the longitudinal soma length.

Given this much larger relative change in stiffness (a factor of 2) compared to the relative change in length (a factor of 1.05), for a maximum voltage change, it seems possible, or even likely, that the observed length changes (the motility) are simply a result of the voltage dependent stiffness. For example, imagine a spring stretched by applying a constant force (say a weight), and then suppose that the spring's stiffness decreases. It follows from Hooke's law (3.5) that the spring's length will *increase* when the stiffness decreases.

Each cell is stretched by its internal static pressure $\mathcal{P}$ [3.101], and its stiffness is voltage controlled [3.70, 71]. The voltage dependent relative stiffness change is much greater than the relative motility change. Thus we have the necessary conditions for stiffness-induced motility.

### 3.2.3 Micromechanics

Unlike the case of macromechanical models, the physics of every micromechanical model differs significantly. This is in part due to the lack of direct experimental evidence of physical parameters of the cochlea. This is an important and very active area of research (e.g., [3.102]).

To organize our discussion of cochlear micromechanics, we represent each radial cross-section through the cochlear partition (Fig. 3.1b) as a linear two-port network. A general formalization in transmission matrix form of the relation between the basilar membrane *input* pressure $P(x, s)$ and velocity $V(x, s)$ and the OHC *output* cilia bundle shear force $f(x, s)$ and shear velocity

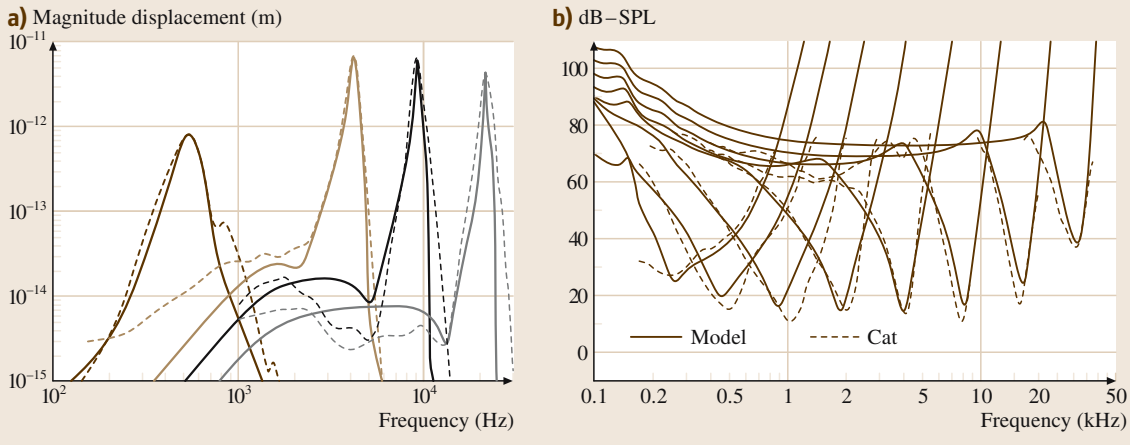



**Fig. 3.9a,b** The tuning curves shown by the dashed lines are the average of single nerve fiber responses from six cats obtained by M. C. Liberman and B. Delgutte. **(a)** Comparison between neural data and the computed model excitation patterns from Allen's passive RTM model (transfer function format). This CA model assumes an IHC cilia bundle displacement of about 50 pm at the neural rate threshold. **(b)** Comparison between neural data computed tuning curves from *Neely*'s active model [3.98]. This CA model assumes an IHC cilia bundle displacement of 300 pm (0.3 nm) at the neural rate threshold

$v(x, s)$

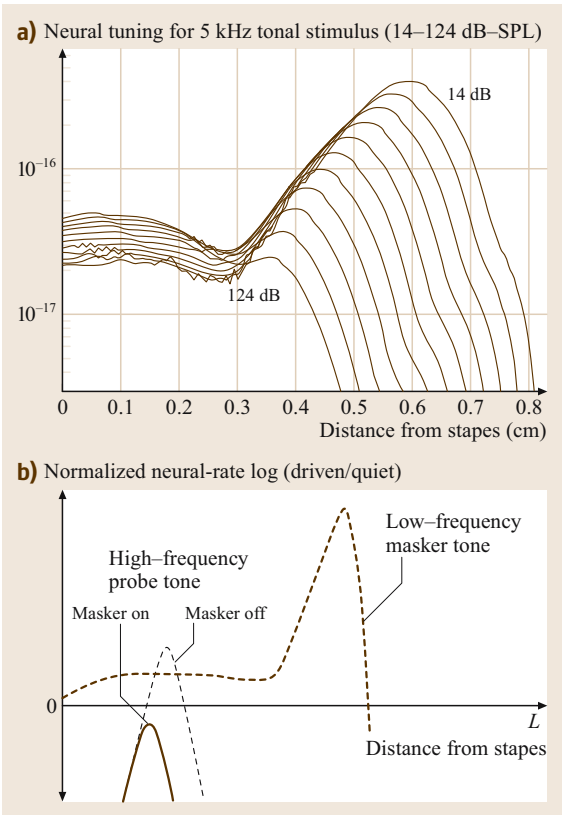
$$\begin{pmatrix} P \\ V \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} f \\ v \end{pmatrix} , \qquad (3.15)$$

where $A$, $B$, $C$, and $D$ are complex functions of place $X$ and radian frequency $s$.

### Passive BM Models

The most successful *passive* model of cochlear tuning is the resonant tectorial membrane (RTM) model [3.9, 104]. The RTM model starts from the assumption that the slope $S_2$ of BM tuning is insufficient to account for the slope $S_2$ of neural tuning, as seen in Fig. 3.4b. This sharpening is accounted for by a reflection in the tectorial membrane, introducing an antiresonance (*spectral zero*) at frequency $F_z$ (Fig. 3.4b), which is about half an octave below the resonant frequency $F_{cf}$ of the basilar membrane. As described by *Allen* and *Neely* [3.9], the detailed $A$, $B$, $C$, $D$ elements of (3.15) are given by *Allen* [3.104], *Allen* and *Neely* [3.9].

As described in *Allen* [3.105], the response ratio of IHC cilia bundle displacement to basilar membrane displacement is defined as $H_{ihc}(x, s)$. The parameters of the RTM model may be chosen such that model results fit the experimental neural threshold tuning curves closely, as shown in Fig. 3.9a.




*The Nonlinear RTM Model.* The resonant tectorial membrane (RTM) model is made NL by control of the BM stiffness via OHC's stiffness is based on Fig. 3.1b. The OHC soma stiffness has been shown to be voltage dependent by *Dallos* et al. [3.106] and dependent on prestin in the membrane wall [3.107]. If an elastic connection is assumed where the TM attaches to the Limbus, and if this elasticity is similar to that of the cilia of the OHC, then the resulting transfer function between the BM and IHC cilia is strongly filtered at low frequencies [3.51,103,108,109]. Such models are actively under consideration [3.102].

It is postulated that the decrease in OHC stiffness accompanying cilia stimulation results in a decrease of the net BM partition stiffness $K_p(x)$ (i. e., increasing compliance) of (3.6). As shown in Fig. 3.3, this decrease in the local BM stiffness would result in the partition excitation pattern shifting basally towards the stapes. Such shifts in the BM response patterns are commonly seen. Another way to view this is shown in Fig. 3.10. This migration of the excitation pattern, combined with the assumption that the TM has a high-pass characteristic, means that the cilia excitation gain at CF is nonlinearly compressed

**Fig. 3.10a,b** In **(a)** results of model calculations by *Sen* and *Allen* [3.103] are shown of a NL BM stiffness model. On the right shows a cartoon of what might happen to the excitation pattern of a low-level probe when a suppressor is turned on given such a nonlinearity. The presence of the suppressor causes the probe to be suppressed and shifted slightly toward the base when the stiffness is decreased with increased level. It may be inferred from Fig. 3.3a that, if the BM stiffness is reduced, the location of the maximum will shift to the base, as is seen in real data. **(a)** Compression in the NL RTM model. Note how the response at the peak is reduced as the BM stiffness changes, causing the peak to shift to the base. As this happens the response in the tail region between $0 \le X \le 0.3$ cm becomes more sensitive, and thus shows an expansive NL response. All of these effects have been seen in real BM data. **(b)** Cartoon showing the effect of a low-side masker on a high-frequency tone as a function of position along the basilar membrane. When the suppressor is turned on, the CF of the high-frequency probe becomes less sensitive and shifts to higher frequencies. We model this effect in the panel on the left as BM stiffness that depends on level (i. e., $K_p(I_s)$)

as the intensity increases. This compression effect is shown in a cartoon format in Fig. 3.10b, while Fig. 3.10a shows the actual calculated model results. Note how the bandwidth $\Delta_f(X)$) remains approximately constant as a function of input intensity.

*Sewell* [3.110] has nicely demonstrated that as the voltage driving the hair cells changes, the neural gain in dB at CF changes proportionally. It is not yet known why the dB gain is proportional to the voltage (1 dB/mv), however this would explain why forward masking decays linearly in dB value with time, after a strong excitation, since the membrane voltage $V_m(t)$ is proportional to $e^{-t/\tau_m}$, due to the OHC membrane's $\tau_m = RC$ time constant. In my view, explaining the proportionality between the neural threshold in dB and the linear membrane voltage, is key.

*Discussion.* Two important advantages of the NL RTM model include its physically based assumptions (described above), and its simplicity. Given these physical assumptions, we show next that the NL RTM model can explain:

1. the basal-ward half-octave traveling-wave migration as a function of increasing intensity [3.76],
2. the upward spread of masking (USM) [3.20, 21], two-tone suppression (2TS) (see Sect. 3.2.1),
3. distortion product generation [3.49,55,56,111–113],
4. normal and recruiting loudness growth, and
5. hypersensitive tails [3.45].

From the steep 2.5 dB/dB slope of the USM and 2TS (Fig. 3.6a) it seems necessary that the low-frequency suppressor is turning down the high-frequency probe even though the growth of the masker at the high frequency's place is linear with masker level, as shown in Fig. 3.10b.

### Active BM Models

One obvious question about active cochlear models is *Are they really necessary?* At least three attempts to answer this question based on detailed comparisons of basilar membrane responses have concluded that the measured responses *cannot* be accounted for by a passive cochlear model [3.93, 114–117].

*The CA Hypothesis.* The most popular active micromechanical theory is called the *cochlear amplifier* (CA) hypothesis. The concept of the *cochlear amplifier*, originated by Gold, Kemp, Kim and Neely, and named by H. Davis, refers to a hypothetical mechanism within the cochlear partition which increases the *sensitivity* of basilar membrane vibrations to low-level sounds and, at the same time, increases the *frequency selectivity* of these vibrations [3.94]. The CA adds mechanical energy to the cochlear partition at acoustic frequencies by drawing upon the electrical and mechanical energy available from the outer hair cells. In response to a tone, the CA adds mechanical energy to the cochlear traveling wave in the region defined by $S_2$ as it approaches the place of maximum response. This energy is reabsorbed at other places along the cochlear partition. The resulting improvement in sensitivity of the ear due to the CA is thought to be 40 dB, or more under certain conditions; however, the details of how this amplification might be accomplished are still unknown [3.118, 119]. A general discussion of this model is presented in *Geisler* [3.90], and in *Allen* and *Fahey* [3.91].

It is presumed that this OHC action amplifies the BM signal energy on a cycle-by-cycle basis, increasing the sensitivity [3.69, 92]. In some of the models it is assumed that this cycle-by-cycle pressure (force) due to the OHCs causes the sharp BM tuning tip. In most of these models, the CA is equivalent to introducing a frequency-dependent negative damping (resistance) into the BM impedance [3.120]. Nonlinear compression is introduced by assuming that the resistance is signal level dependent. This NL resistance model was first described by *Hall* [3.84] for the case of $R > 0$. Thus the CA model is an extension of Hall's model to the case of $R < 0$. In several models NL negative damping is obtained with a nonlinear stiffness and a small delay. The addition of a small delay introduces a negative real part into the impedance. In mathematical physics, NL damping resonators are described by *van der Pol equations*, while NL stiffness resonators are described by *Duffing equations* [3.121].

*Allen* and *Fahey* [3.91] developed a method for directly measuring the cochlear amplifier (CA) gain. All of the studies to date using this method have found no gain. However many researchers continue to believe that the CA has gain. Given that the gain is order 40–50 dB this is difficult to understand. A nice summary of this situation has been recently published in *Shera* and *Guinan* [3.120]. The reasons for the failure to directly measure any CA gain are complex and multifaceted, and many important questions remain open. One possibility that remains open is that the many observed large NL OHC BM effects we see are not due to cycle-by-cycle power amplification of the BM traveling wave.

### Discussion and Summary

*Discussion.* Both active and passive BM models are reasonably successful at simulating the neural thresh-

old response tuning curves. Thus we may need to look elsewhere to contrast the difference between these two approaches, such as 2TS/USM. While the passive RTM model is easily made NL with the introduction of $K_{ohc}(V_m)$, differences between *nonlinear* RTM and CA models have not yet been investigated. The CA and RTM models differ in their interpretation of damaged cochlear responses. In CA models, the loss of sensitivity of the cochlea with damage is interpreted as a loss of CA gain while in passive models, the loss of sensitivity has been interpreted as a 2:1 change in the BM stiffness [3.122].

The discovery of OHC motility demonstrates the existence of a potential source of mechanical energy within the cochlear partition which is suitably positioned to influence vibrations of the basilar membrane. It is still an open question whether this source of energy is sufficient to power a CA at high frequencies.

One possible advantage of the CA is that of improving the signal-to-noise ratio in front of the IHC detector. A weakness of the CA models has been their lack of specificity about the physical realization of the active elements. Until we have a detailed physical representation for the CA, RTM models have the advantage of being simpler and more explicit.

The discovery by He and Dallos that the OHC soma stiffness is voltage dependent is an exciting development for the NL passive RTM model, as it greatly simplifies the implementation of the physical model. The RTM model has been in disfavor because many feel it does not account for basilar membrane tuning. This criticism is largely due to the experimental results of physiologists who have measured the BM–ear canal transfer function, and found the tuning of BM velocity to be similar to neural threshold response data. Much of the experimental BM data, however, are not convincing on this point, with the BM slope $S_2$ (Fig. 3.4b) generally being much smaller than that of neural responses [3.97]. The question of whether an active model is required to simulate measured BM responses is still being debated.

Better estimates of the amplitude of cilia bundle displacement at a given sound pressure level directly address the sensitivity questions. If the estimate of *Russell* of 30 mV/degree is correct [3.123], then the cochlear sensitivity question may be resolved by having very sensitive detectors. Also, better estimates are needed of the ratio of the BM frequency response to the IHC frequency response, both at high and low frequencies. Rhode's approach of using the slopes of Fig. 3.4b rather than traditional ad hoc bandwidth measures, is a useful tool in this regard. The bandwidth 10 dB down relative to the peak has been popular, but arbitrary and thus poor, criterion in cochlear research. A second, somewhat better, bandwidth measure is Fletcher's *equivalent rectangular bandwidth* discussed in *Allen* [3.10].

*Summary.* This section has reviewed what we know about the cochlea. The *Basics* section reviews the nature of modeling and briefly describes the anatomy of the inner ear, and the function of inner and outer hair cells. In Sect. 3.1.2 we reviewed the history of cochlear modeling. The Wegel and Lane paper was a key paper that introduced the first detailed view of masking, and in the same paper introduced the first modern cochlear model Fig. 3.2b. We presented the basic tools of cochlear modeling, *impedance*, and introduced the *transmission matrix* method (two-port analysis). We describe how these models work in intuitive terms, including how the basilar membrane may be treated as having a frequency dependent acoustic hole. The location of the hole, as a function of frequency, is called the cochlear map. This hole keeps fluid from flowing beyond a certain point, producing the cochlear traveling wave.

We reviewed and summarized the NL measures of cochlear response. Since these data are not fully understood, and have not been adequately modeled, this is the most difficult section. However it is worth the effort to understand these extensive data and to appreciate the various relations between them, such as the close parallel between two-tone suppression and the upward spread of masking, and between loudness recruitment and outer hair cell damage.

We review several models of the hair cell, including forward and reverse transduction. Some of this material is recently published, and the view of these models could easily change over the next few years as we better understand reverse transduction.

Finally in Sect. 3.2.3 we reviewed the basics of micromechanics. We have presented the two basic types of models, *passive* and *active* models, with a critical review of each.

## 3.3 Neural Masking

When modeling human psychophysics one must carefully distinguish the external *physical* variables, which we call $\Phi$ variables, from the internal *psychophysical* variables, or $\Psi$ variables. It may be helpful to note that

$\Phi$ and $\Psi$ sound similar to the initial syllable of the words *phy*sical and *psy*chological, respectively [3.124]. Psychophysical modeling seeks a transformation from the $\Phi$ domain to the $\Psi$ domain. The $\Phi$ intensity of a sound is easily quantified by direct measurement. The $\Psi$ intensity is the loudness. The idea that loudness could be quantified was first suggested by *Fechner* [3.125] in 1860, who raised the question of the quantitative transformation between the physical and psychophysical intensity. For a recent review of this problem, and a brief summary of its long history, see *Schlauch* et al. [3.126]. This section is based on an earlier report by *Allen* [3.79], and *Allen* and *Neely* [3.127].

An increment in the intensity of a sound that results in a *just noticeable difference* is called an intensity JND. Fechner suggested quantifying the intensity-loudness growth transformation by counting the number of the *loudness JNDs* between two intensity values. However, after many years of work, the details of the relationship between loudness and the intensity JNDs remain unclear [3.128–130].

The contribution of *Allen* and *Neely* [3.127] and *Allen* [3.79] is that it takes a new view of the problem of the intensity JND and loudness by merging the 1953 Fletcher neural excitation pattern model of loudness [3.10, 131] with auditory signal detection theory [3.132].

It is generally accepted that the intensity JND is the physical correlate of the psychological-domain uncertainty corresponding to the psychological intensity representation of a signal. Along these lines, for long duration pure tones and wide-band noise, we assume that the $\Psi$-domain intensity is the loudness, and that the loudness JND results from loudness *noise* due to its stochastic representation.

To model the intensity JND we must define a *decision variable* associated with loudness and its random fluctuations. We call this loudness random decision variable the *single-trial loudness*. Accordingly we define the loudness and the loudness JND in terms of the first and second moments of the single-trial loudness, that is the mean and variance of the distribution of the single-trial loudness decision variable. We also define the ratio of the mean loudness to the loudness standard deviation as the *loudness signal-to-noise ratio*, $\mathrm{SNR_L}$.

Our ultimate goal in this work is to use signal detection theory to unify masking and the JND, following the 1947 outline of this problem by *Miller* [3.133]. Tonal data follows the *near miss to Weber's law* (thus does not obey Weber's law), while the wide-band noise data does obey Weber's law. We will show that the transformation of the $\Phi$-domain (intensity) JND data (both tone and noise) into the $\Psi$ domain (loudness) unifies these two types of JND data, since $\mathrm{SNR_L}(L)$ is the same for both the tone and noise cases. To help understand these results, we introduce the concept of a near miss to Stevens' law, which we show cancels the near–miss to Weber's law, giving the invariance in $\mathrm{SNR_L}$ for the tone case [3.127]. This work has applications in speech and audio coding.

For the case of tones, we have chosen to illustrate our theoretical work using the classical intensity modulation measurements of *Riesz* [3.75] who measured the intensity JND using small, low-frequency (3 Hz), sinusoidal modulation of tones. *Modern* methods generally use *pulsed* tones which are turned on and off somewhat abruptly, to make them suitable for a two-alternative, forced-choice (2AFC) paradigm. This transient could trigger cochlear forward masking. Riesz's modulation method has a distinct advantage for characterizing the internal signal detection process, because it maintains a nearly steady-state small-signal condition within the auditory system, minimizing any cochlear forward masking component. The interpretation of intensity JNDs is therefore simplified since underlying stochastic processes are stationary.

An outline of this neural masking section is as follows. After some basic definitions in Sect. 3.3.1 and a review of historical models (e.g., Weber and Fechner), in Sect. 3.3.2, we explore issues surrounding the relation between the intensity JND and loudness, for the special cases of tones in quiet and for wide-band noise. First, we look at formulae for counting the number of intensity and loudness JNDs and we use these formulae, together with decision-theoretic principles, to relate loudness to the intensity JND. We then review the loudness-JND theory developed by *Hellman* and *Hellman* [3.134], which provided the inspiration for the present work. Next, we empirically estimate the loudness SNR, defined as the mean loudness over the loudness variance, and proportional to $L/\Delta L$, as a function of both intensity and loudness, using the tonal JND data of *Riesz* [3.75] and the loudness growth function of *Fletcher* and *Munson* [3.41]. We then repeat this calculation for Miller's wide-band noise JND and loudness data. Finally we propose a model of loudness that may be used to compute the JND. This model merges Fletcher's neural excitation pattern model of loudness with signal detection theory.

### 3.3.1 Basic Definitions

We need a flexible yet clear notation that accounts for important time fluctuations and modulations that are present in the signals, such as beats and gated signals. We include a definition of *masked threshold* because we view the intensity JND as a special case of the masked threshold [3.133]. We include a definition of *beats* so that we can discuss their influence on Riesz's method for the measurement of intensity JNDs.

*Intensity.* In the time domain, it is common to define the $\Phi$ *intensity* in terms of the time-integrated squared signal pressure $s(t)$, namely,

$$I_s(t) \equiv \frac{1}{\varrho c T} \int_{t-T}^{t} s^2(t)\,\mathrm{d}t\,, \tag{3.16}$$

where $T$ is the integration time and $\varrho c$ is the specific acoustic impedance of air. The *intensity level* is defined as $I_s/I_{\mathrm{ref}}$, and the *sound pressure level* as $|s|/s_{\mathrm{ref}}$, where the reference intensity is $I_{\mathrm{ref}}$ or $10^{-10}\,\mu\mathrm{W/cm^2}$ and the reference pressure $s_{\mathrm{ref}} = 20\,\mu\mathrm{Pa}$. These two reference levels are equivalent at only one temperature, but both seem to be in use. Equivalence of the pressure and intensity references requires that $\varrho c = 40\,\mathrm{cgs}$ Rayls. At standard atmospheric pressure, this is only true when the temperature is about 39 °C. Such levels are typically expressed in dB units.

*Intensity of Masker plus Probe.* The JND is sometimes called *self-masking*, to reflect the view that it is determined by the internal noise of the auditory system. To model the JND it is useful to define a more-general measure called the *masked threshold*, which is defined in the $\Phi$ domain in terms of a nonnegative pressure scale factor $\alpha$ applied to the probe signal $p(t)$ that is then added to the masking pressure signal $m(t)$. The relative intensity of the probe and masker is varied by changing $\alpha$. Setting $s(t) = m(t) + \alpha p(t)$, we denote the combined intensity as

$$I_{\mathrm{m+p}}(t,\alpha) \equiv \frac{1}{\varrho c T} \int_{t-T}^{t} [m(t) + \alpha p(t)]^2\,\mathrm{d}t\,. \tag{3.17}$$

The unscaled probe signal $p(t)$ is chosen to have the same long-term average intensity as the masker $m(t)$, defined as $I$. Let $I_{\mathrm{m}}(t)$ be the intensity of the masker with no probe ($\alpha = 0$), and $I_{\mathrm{p}}(t,\alpha) = \alpha^2 I$ be the intensity of the scaled probe signal with no masker. Thus

$$I \equiv I_{\mathrm{m+p}}(t,0) = I_{\mathrm{m}}(t) = I_{\mathrm{p}}(t,1)\,.$$

Because of small fluctuations in $I_{\mathrm{m}}$ and $I_{\mathrm{p}}$ due to the finite integration time $T$, this equality cannot be exactly true. We are specifically ignoring these small rapid fluctuations – when these rapid fluctuations are important, our conclusions and model results must be reformulated.

*Beats.* Rapid fluctuations having frequency components outside the bandwidth of the period $T_{\mathrm{second}}$ rectangular integration window are very small and will be ignored ($T$ is assumed to be large). Accordingly we drop the time dependence in terms $I_{\mathrm{m}}$ and $I_{\mathrm{p}}$. The beats between $m(t)$ and $p(t)$ of these signals are within a common critical band. Slowly varying correlations, between the probe and masker having frequency components within the bandwidth of the integration window, may *not* be ignored, as with beats between two tones separated in frequency by a few Hz. Accordingly we keep the time dependence in the term $I_{\mathrm{m+p}}(t,\alpha)$ and other slow–beating time dependent terms. In the $\Phi$ domain these beats are accounted for as a probe–masker correlation function $\rho_{\mathrm{mt}}(t)$ [3.132, p. 213].

*Intensity Increment $\delta I(t,\alpha)$.* Expanding (3.17) and solving for the *intensity increment* $\delta I$ we find

$$\delta I(t,\alpha) \equiv I_{\mathrm{m+p}}(t,\alpha) - I = \left[2\alpha\rho_{\mathrm{mp}}(t) + \alpha^2\right] I\,, \tag{3.18}$$

where

$$\rho_{\mathrm{mp}}(t) = \frac{1}{\varrho c T I} \int_{t-T}^{t} m(t)p(t)\,\mathrm{d}t \tag{3.19}$$

defines a normalized cross-correlation function between the masker and the probe. The correlation function must lie between $-1$ and 1.

*Detection Threshold.* As the probe-to-masker ratio $\alpha$ is increased from zero, the probe can eventually be detected. We specify the probe *detection threshold* as $\alpha_*$, where the asterisk indicates the threshold value of $\alpha$ where a subject can discriminate intensity $I_{\mathrm{m+p}}(t,\alpha_*)$ from intensity $I_{\mathrm{m+p}}(t,0)$ 50% of the time, corrected for chance (i. e., obtain a 75% correct score in a direct comparison of the two signals [3.132, p. 129]). The quantity $\alpha_*(t,I)$ is the probe to masker root-mean-square (RMS) pressure ratio at the detection threshold. It is a function of the masker intensity $I$ and, depending on the experimental setup, time. $\alpha_*$ summarizes the experimental measurements.

*Masked Threshold Intensity.* When $\rho_{mp} = 0$, the *masked threshold intensity* is defined in terms of $\alpha_*$ as

$$I_p^*(I) \equiv I_p(\alpha_*) = \alpha_*^2 I \,,$$

which is the threshold intensity of the probe in the presence of the masker.

The masked threshold intensity is a function of the stimulus modulation parameters. For example, tone maskers and narrow-band noise maskers of equal intensity, and therefore approximately equal loudness, give masked thresholds that are about 20 dB different [3.135]. As a second example, when using the method of beats [3.75], the just–detectable modulation depends on the beat frequency. With *modern* 2AFC methods, the signals are usually gated on and off (100% modulation) [3.136]. According to *Stevens* and *Davis* [3.137, p. 142]

> *A gradual transition, such as the sinusoidal variation used by Riesz, is less easy to detect than an abrupt transition; but, as already suggested, an abrupt transition may involve the production of unwanted transients.*

One must conclude that the *relative masked threshold* [i. e., $\alpha_*(t, I)$] is a function of the modulation conditions, and depends on $\rho_{mp}$, and therefore $T$.

*$\Psi$-Domain Temporal Resolution.* When modeling time-varying psychological decision variables, the relevant integration time $T$ is not the duration defined by the $\Phi$ intensity (3.16), rather the integration time is determined in the $\Psi$ domain. This important $\Psi$-domain model parameter is called *loudness temporal integration* [3.138]. It was first explicitly modeled by *Munson* in 1947 [3.139].

The $\Phi$-domain temporal resolution ($T$) is critical to the definition of the JND in Riesz's experiment because it determines the measured intensity of the beats. The $\Psi$-domain temporal resolution plays a different role. Beats cannot be heard if they are faster than, and therefore *filtered* out by, the $\Psi$ domain response. The $\Psi$-domain temporal resolution also impacts results for gated stimuli, such as in the 2AFC experiment, though its role is poorly understood in this case. To model the JND as measured by Riesz's method of just-detectable beats, one must know the $\Psi$-domain resolution duration to calculate the probe–masker effective correlation $\rho_{mp}(t)$ in the $\Psi$ domain. It may be more practical to estimate the $\Psi$ domain resolution from experiments that estimate the degree of correlation, as determined by the beat modulation detection threshold as a function of the beat frequency $f_b$.

In summary, even though Riesz's modulation detection experiment is technically a masking task, we treat it, following *Riesz* [3.75], *Miller* [3.133], and *Littler* [3.16], as characterizing the intensity JND. It follows that the $\Psi$-domain temporal resolution plays a key role in intensity JND and masking models.

*The Intensity JND $\Delta I$.* The intensity *just-noticeable difference* (JND) is

$$\Delta I(I) \equiv \delta(t, \alpha_*) \,, \tag{3.20}$$

the intensity increment at the masked threshold, for the special case where the probe signal is equal to the masking signal ($p(t) = m(t)$). From (3.18) with $\alpha$ set to threshold $\alpha_*$ and $\rho_{mp}(t) = 1$

$$\Delta I(I) = (2\alpha_* + \alpha_*^2) I \,. \tag{3.21}$$

It is traditional to define the intensity JND to be a function of $I$, rather than a function of $\alpha(I)$, as we have done here. We shall treat both notations as equivalent [i. e., $\Delta I(I)$ or $\Delta I(\alpha)$].

An important alternative definition for the special case of the *pure-tone JND* is to let the masker be a pure tone, and let the probe be a pure tone of a slightly different frequency (e.g., a beat frequency difference of $f_b = 3$ Hz). This was the definition used by *Riesz* [3.75]. Beats are heard at $f_b = 3$ Hz, and assuming the period of 3 Hz is within the passband of the $\Psi$ temporal resolution window, $\rho_{mp}(t) = \sin(2\pi f_b t)$. Thus

$$\Delta I(t, I) = \left[2\alpha_* \sin(2\pi f_b t) + \alpha_*^2\right] I \,. \tag{3.22}$$

If the beat period is less than the $\Psi$ temporal resolution window, the beats are *filtered* out by the auditory brain (the effective $\rho_{mn}$ is small) and we do not hear the beats. In this case $\Delta I(I) = \alpha_*^2 I$. This model needs to be tested [3.139].

*Internal Noise.* It is widely accepted that the pure-tone intensity JND is determined by the *internal noise* of the auditory system [3.140, 141], and that $\Delta I$ is proportional to the standard deviation of the $\Psi$-domain decision variable that is being discriminated in the intensity detection task, reflected back into the $\Phi$ domain. The usual assumption, from signal detection theory, is that $\Delta I = d'\sigma_I$, where $d'$ is defined as the proportionality between the change in intensity and the variance $d' \equiv \Delta I/\sigma_I$. Threshold is typically when $d' = 1$ but can

Amazon EX1010 Page 80

depend on the the experimental design; $\sigma_I$ is the intensity standard deviation of the $\Phi$-domain intensity due to $\Psi$-domain auditory noise [3.15, 17, 127].

*Hearing Threshold.* The *hearing threshold* (or unmasked threshold) *intensity* may be defined as the intensity corresponding to the first (lowest intensity) JND. The hearing threshold is represented as $I_p^*(0)$ to indicate the probe intensity when the masker intensity is small (i. e., $I \to 0$). It is believed that internal noise is responsible for the hearing threshold.

*Loudness L.* The *loudness* $L$ of a sound is the $\Psi$ intensity. The *loudness growth function* $L(I)$ depends on the stimulus conditions. For example $L(I)$ for a tone and for wide-band noise are not the same functions. Likewise the loudness growth function for a 100 ms tone and a 1 s tone differ. When defining a *loudness scale* it is traditional to specify the intensity, frequency, and duration of a tone such that the loudness growth function is one [$L(I_{ref}, f_{ref}, T_{ref}) = 1$ defines a loudness scale]. For the sone scale, the reference signal is a $I_{ref} = 40\,\text{dB} - \text{SPL}$ tone at $f_{ref} = 1\,\text{kHz}$ with duration $T_{ref} = 1\,\text{s}$. For Fletcher's LU scale the reference intensity is the hearing threshold, which means that 1 sone = 975 LU [3.42] for a normal hearing person. Fletcher's LU loudness scale seems a more-natural scale than the sone scale used in the American National Standards Institute (ANSI) and International Organization for Standardization (ISO) standards.

*The Single-Trial Loudness.* A fundamental postulate of psychophysics is that all decision variables (i. e., $\Psi$ variables) are random variables, drawn from some probability space [3.132, Chap. 5]. For early discussions of this point see *Montgomery* [3.142] and p. 144 of *Stevens* and *Davis* [3.137]. To clearly indicate the distinction between random and nonrandom variables, a tilde ($\sim$) is used to indicate a random variable. As a mnemonic, we can think of the $\sim$ as a *wiggle* associated with randomness.

We define the loudness decision variable as the *single-trial loudness* $\widetilde{L}$, which is the sample loudness heard on each stimulus presentation. The loudness $L$ is then the expected value of the single-trial loudness $\widetilde{L}$

$$L(I) \equiv \mathcal{E}\widetilde{L}(I)\,. \tag{3.23}$$

The second moment of the single-trial loudness

$$\sigma_L^2 \equiv \mathcal{E}(\widetilde{L} - L)^2 \tag{3.24}$$

defines the loudness *variance* $\sigma_L^2$ and *standard deviation* $\sigma_L$.

### Derived Definitions

The definitions given above cover the basic variables. However many alternative forms (various normalizations) of these variables are used in the literature. These derived variables were frequently formed with the hope of finding an invariance in the data. This could be viewed as a form of modeling exercise that has largely failed (e.g., the near miss to Weber's law), and the shear number of combinations has led to serious confusions [3.138, p. 152]. Each normalized variable is usually expressed in dB, adding an additional unnecessary layer of confusion to the picture. For example, *masking* is defined as the masked threshold normalized by the unmasked (quiet) threshold, namely

$$M \equiv \frac{I_p^*(I_m)}{I_p^*(0)}\,.$$

It is typically quoted in dB re sensation level (dB − SL). The intensity JND is frequently expressed as a *relative JND* called the *Weber fraction* defined by

$$J(I) \equiv \frac{\Delta I(I)}{I}\,. \tag{3.25}$$

From the signal detection theory premise that $\Delta I = d'\sigma_I$ [3.17], $J$ is just the reciprocal of an effective signal-to-noise ratio defined as

$$\text{SNR}_I(I) \equiv \frac{I}{\sigma_I(I)} \tag{3.26}$$

since

$$J = d'\frac{\sigma_I}{I} = \frac{d'}{\text{SNR}_I}\,. \tag{3.27}$$

One conceptual difficulty with the Weber fraction $J$ is that it is an *effective* signal-to-noise ratio, expressed in the $\Phi$ (physical) domain, but determined by a $\Psi$ (psychophysical) domain mechanism (internal noise), as may be seen from Fig. 3.11.

*Loudness JND $\Delta L$.* Any suprathreshold $\Psi$-domain increments may be quantified by corresponding $\Phi$ domain increments. The *loudness* JND $\Delta L(I)$ is defined as the change in loudness $L(I)$ corresponding to the intensity JND $\Delta I(I)$. While it is not possible to measure $\Delta L$ directly, we assume that we may expand the loudness function in a Taylor series (Fig. 3.11), giving

$$L(I + \Delta I) = L(I) + \Delta I \left.\frac{dL}{dI}\right|_I + \text{HOT}\,,$$

where HOT represents *higher-order terms*, which we shall ignore. If we solve for

$$\Delta L \equiv L(I + \Delta I) - L(I) \tag{3.28}$$



Amazon EX1010 Page 81

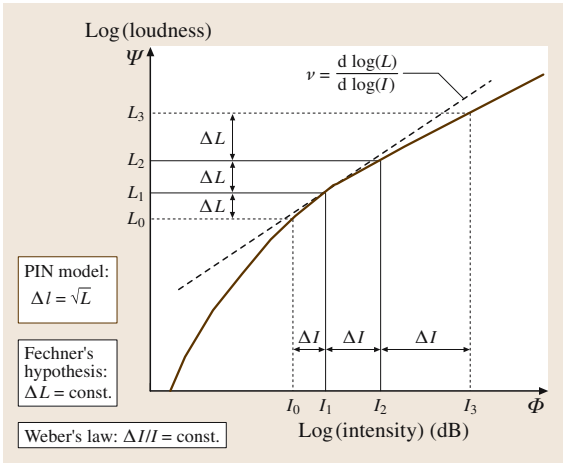



**Fig. 3.11** Summary of all historical ideas about psychophysics and the relations between the $\Phi$ and $\Psi$ variables. Along the abscissa we have the physical variable, intensity, and along the ordinate, the psychological variable loudness. The *curve* represents the loudness, on a log-intensity log-loudness set of scales. A JND in loudness is shown as $\Delta L$ and it depends on loudness, as described by the *Poisson internal noise* (PIN) model shown in the box on the left. Fechner assumed that $\Delta L$ was constant, which we now know to be incorrect. The loudness JND is reflected back into the physical domain as an intensity JND $\Delta I$, which also depends on level. Weber's law, is therefore not true in general (but is approximately true for wide-band noise). Our analysis shows that the loudness SNR and the intensity SNR must be related by the slope of the loudness growth function, as given by (3.32). These relations are verified in Fig. 3.12, as discussed in detail in *Allen* and *Neely* [3.127]

we find

$$\Delta L = \Delta I \left.\frac{\mathrm{d}L}{\mathrm{d}I}\right|_{\mathrm{I}} . \tag{3.29}$$

We call this expression the *small*-JND approximation. The above shows that the loudness JND $\Delta L(I)$ is related to the intensity JND $\Delta I(I)$ by the slope of the loudness function, evaluated at intensity $I$. According to the signal detection model, the standard deviation of the single-trial loudness is proportional to the loudness JND, namely

$$\Delta L = d' \sigma_{\mathrm{L}} . \tag{3.30}$$

A more explicit way of expressing this assumption is

$$\frac{\Delta L}{\Delta I} = \frac{\sigma_{\mathrm{L}}}{\sigma_{\mathrm{I}}} , \tag{3.31}$$

where $d'$ in both the $\Phi$ and $\Psi$ domains is the same and thus cancels.

*Loudness SNR.* In a manner analogous to the $\Phi$-domain $\mathrm{SNR_I}$, we define the $\Psi$-domain loudness SNR as $\mathrm{SNR_L}(L) \equiv L/\sigma_{\mathrm{L}}(L)$. Given (3.30), it follows that

$$\mathrm{SNR_I} = \nu \mathrm{SNR_L} , \tag{3.32}$$

where $\nu$ is the slope of the log-loudness function with respect to log-intensity. If we express the loudness as a power law

$$L(I) = I^{\nu}$$

and let $x = \log(I)$ and $y = \log(L)$, then $y = \nu x$. If the change of $\nu$ with respect to dB – SPL is small, then $\mathrm{d}y/\mathrm{d}x \approx \Delta y/\Delta x \approx \nu$. Since $\mathrm{d}\log(y) = \mathrm{d}y/y$ we get

$$\frac{\Delta L}{L} = \nu \frac{\Delta I}{I} . \tag{3.33}$$

Equation (3.32) is important because

1. it tells us how to relate the SNRs between the $\Phi$ and $\Psi$ domains,
2. every term is dimensionless,
3. the equation is simple, since $\nu \approx 1/3$ is approximately constant above 40 dB – SL (i. e., Stevens' law), and because
4. we are used to seeing and thinking of loudness, intensity, and the SNR, on log scales, and $\nu$ as the slope on log–log scales.

*Counting JNDs.* While the concept of counting JNDs has been frequently discussed in the literature, starting with Fechner, unfortunately the actual counting formula (i. e., the equation) is rarely provided. As a result of a literature search, we found the formula in *Nutting* [3.143], *Fletcher* [3.21], *Wegel* and *Lane* [3.20], *Riesz* [3.75], *Fletcher* [3.144], and *Miller* [3.133].

To derive the JND counting formula, (3.29) is rewritten as

$$\frac{\mathrm{d}I}{\Delta I} = \frac{\mathrm{d}L}{\Delta L} . \tag{3.34}$$

Integrating over an interval gives the total number of intensity JNDs

$$N_{12} \equiv \int_{I_1}^{I_2} \frac{\mathrm{d}I}{\Delta I} = \int_{L_1}^{L_2} \frac{\mathrm{d}L}{\Delta L} , \tag{3.35}$$

where $L_1 = L(I_1)$ and $L_2 = L(I_2)$. Each integral counts the total number of JNDs in a different way between $I_1$ and $I_2$ [3.75,144]. The number of JNDs must be the same regardless of the domain (i. e., the abscissa variable), $\Phi$ or $\Psi$.

Amazon EX1010 Page 82

## 3.3.2 Empirical Models

This section reviews some earlier empirical models of the JND and its relation to loudness relevant to our development.

### Weber's Law

In 1846 it was suggested by Weber that $J(I)$ is independent of $I$. According to (3.21) and (3.25)

$$J(I) = 2\alpha_* + \alpha_*^2 \,.$$

If $J$ is constant, then $\alpha_*$ must be constant, which we denote by $\alpha_*(\not I)$ (we strike out $I$ to indicate that $\alpha_*$ is not a function of intensity). This expectation, which is called Weber's law [3.145], has been successfully applied to many human perceptions. We refer the reader to the helpful and detailed review of these questions by *Viemeister* [3.129], *Johnson* et al. [3.146], and *Moore* [3.147].

Somewhat frustrating is the empirical observation that $J(I)$ is not constant for the most elementary case of a pure tone [3.75, 136]. This observation is referred to as *the near miss to Weber's law* [3.148].

Weber's law does make one simple prediction that is potentially important. From (3.35) along with Weber's law $J_0 \equiv J(\not I)$ we see that the formula for the number of JNDs is

$$N_{12} = \int_{I_1}^{I_2} \frac{\mathrm{d}I}{J_0 I} = \frac{1}{J_0} \ln\left(\frac{I_2}{I_1}\right) . \tag{3.36}$$

It remains unexplained why Weber's law holds as well as it does [3.149, 150, p. 721] (it holds approximately for the case of wide band noise), or even why it holds at all. Given the complex and NL nature of the transformation between the $\Phi$ and $\Psi$ domains, coupled with the belief that the noise source is in the $\Psi$ domain, it seems unreasonable that a law as simple as Weber's law could hold in any general way. A transformation of the JND from the $\Phi$ domain to the $\Psi$ domain greatly clarifies the situation.

### Fechner's Postulate

In 1860 Fechner postulated that the loudness JND $\Delta L(I)$ is a constant [3.125, 130, 151, 152]. We are only considering the auditory case of Fechner's more general theory. We shall indicate such a constancy with respect to $I$ as $\Delta L(\not I)$ (as before, we strike out the $I$ to indicate that $\Delta L$ is *not* a function of intensity). As first reported by *Stevens* [3.153], we shall show that Fechner's postulate is not generally true.

### The Weber–Fechner Law

It is frequently stated [3.152] that Fechner's postulate ($\Delta L(\not I)$) and Weber's law ($J_0 \equiv J(\not I)$) lead to the conclusion that the difference in loudness between any two intensities $I_1$ and $I_2$ is proportional to the logarithm of the ratio of the two intensities, namely

$$\frac{L(I_2) - L(I_1)}{\Delta L} = \frac{1}{J_0} \log\left(\frac{I_2}{I_1}\right) . \tag{3.37}$$

This is easily seen by eliminating $N_{12}$ from (3.36) and by assuming Weber's law and Fechner's hypothesis. This result is called *Fechner's law* (also called the *Weber–Fechner law*). It is not true because of the faulty assumptions, Weber's law and Fechner's postulate.

## 3.3.3 Models of the JND

Starting in 1923, *Fletcher* and *Steinberg* studied loudness coding of pure tones, noise, and speech [3.21, 154–156], and proposed that loudness was related to neural spike count [3.41], and even provided detailed estimates of the relation between the number of spikes and the loudness in sones [3.42, p. 271]. In 1943 *De Vries* first introduced a photon-counting Poisson process model as a theoretical basis for the threshold of vision [3.157]. *Siebert* [3.140] proposed that Poisson point-process noise, resulting from the neural rate code, acts as the internal noise that limits the frequency JND [3.136, 150]. A few years later [3.158], and independently [3.159] *McGill* and *Goldberg* [3.160] proposed that the Poisson internal noise (PIN) model might account for the intensity JND, but they did not find this to produce a reasonable loudness growth function. *Hellman* and *Hellman* [3.134] further refined the argument that Poisson noise may be used to relate the loudness growth to the intensity JND, and they found good agreement between the JND and realistic loudness functions.

Given Poisson noise, the variance is equal to the mean, thus

$$\Delta L(L) \propto \sqrt{L} \,. \tag{3.38}$$

This may also be rewritten as $\sigma_L^2 \propto L$. We would expect this to hold if the assumptions of *McGill* [3.148] (i. e., the PIN model) are valid.

Amazon EX1010 Page 83

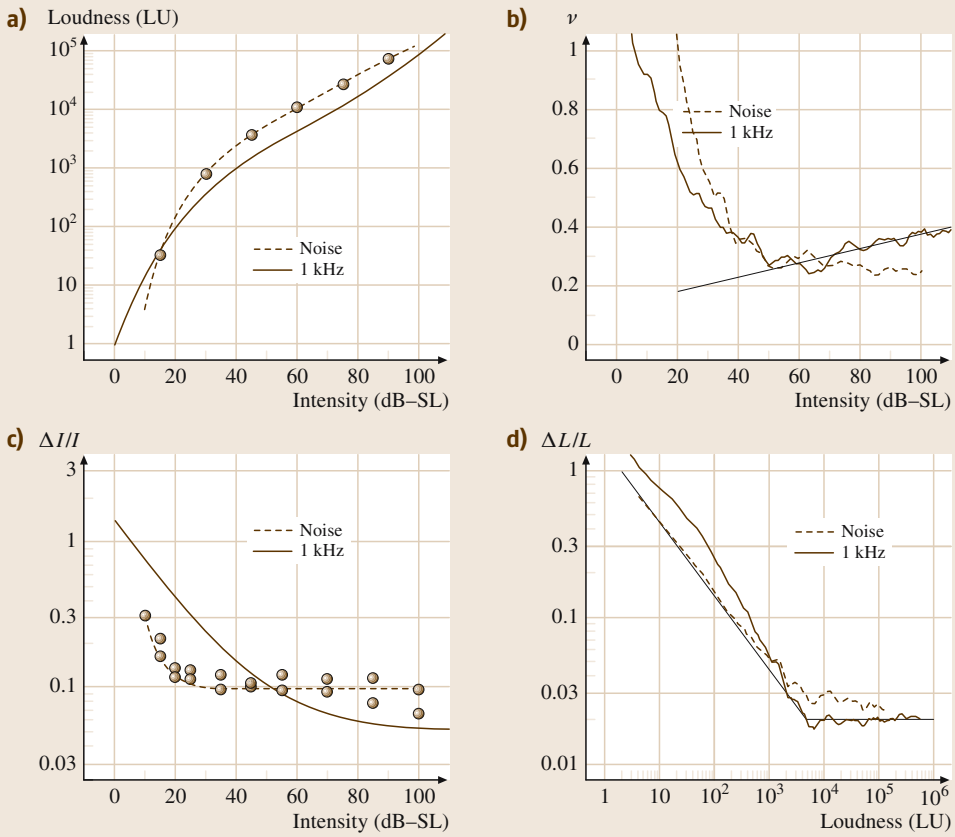



**Fig. 3.12a–d** In 1947 Miller measured the $JND_I$ and the loudness level for two subjects using wide-band modulated noise (0.15–7 kHz) for levels between 3 and 100 dB − SL. The noise (*dashed line*) and pure tone (*solid line*) loudness are shown in **(a)**. The similarity between $\Delta L/L$ derived from the loudness curves for pure tones and for noise provide an almost perfect fit to the SPIN model which results from assuming the noise is neural point-process noise. See the text for a summary of these results. The direct derivation of $\Delta L$ based on pure tone JND and loudness data from *Miller* [3.133], *Riesz* [3.75], *Fletcher* and *Munson* [3.41].

In the following we directly compare the loudness–growth function of Fletcher and Munson to the number of JNDs $N_{12}$ from Riesz [3.75, 127] to estimate $\Delta L/L$.

### 3.3.4 A Direct Estimate of the Loudness JND

Given its importance, it is important to estimate $\Delta L$ directly from its definition (3.28), using Riesz's $\Delta I(I)$ and Fletcher and Munson's 1933 estimate of $L(I)$.

Miller's 1947 famous JND paper includes wide-band-noise loudness-level results. We transformed these JND data to loudness using *Fletcher* and *Munson* [3.41] reference curve (i. e., Fig. 3.12a).

#### Loudness Growth, Recruitment, and the OHC

In 1924 *Fletcher* and *Steinberg* published an important paper on the measurement of the loudness of speech signals [3.155]. In this paper, when describing the growth of loudness, the authors state

> *the use of the above formula involved a summation of the cube root of the energy rather than the energy.*

This cube–root dependence had first been described by *Fletcher* the year before [3.21].

In 1930 *Fletcher* [3.27] postulated that there was a monotonic relationship between central nerve firings rates and loudness. Given a tonal stimulus at the ear

Amazon EX1010 Page 84

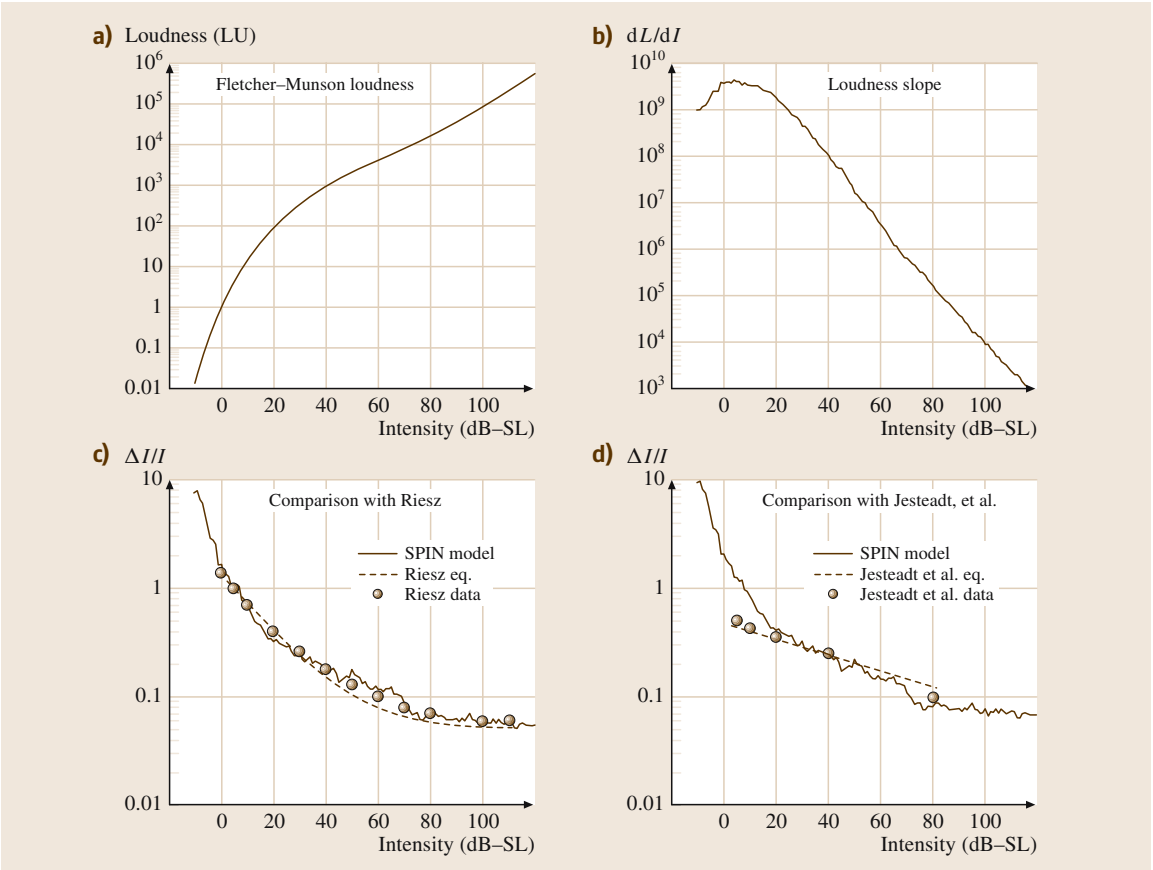



**Fig. 3.13a–d** Test of the SPIN model against the classic results of *Riesz* [3.75], *Jesteadt* et al. [3.136]. Test of the model derived on the *left* based on a comparison between loudness data and intensity JND data at 1 kHz, using the SPIN model

drum, Stevens' law says that the loudness is given by

$$L \equiv L(f, x, I) \propto I^{\nu} , \qquad (3.39)$$

where $(f, x, I)$ are the frequency, place, and intensity of the tone, respectively. The exponent $\nu$ has been experimentally established to be in the range between 1/4 and 1/3 for long duration pure tones at 1 kHz. *Fletcher* and *Munson* [3.41] found $\nu \approx 1/4$ at high intensities and approximately 1 near threshold. Although apparently it has not been adequately documented, $\nu$ seems to be close to 1 for the *recruiting ear* [3.15].

*Recruitment.* What is the source of Fletcher's cube-root loudness growth (i. e., Stevens' law)? Today we know that cochlear outer hair cells are the source of the cube-root loudness growth observed by Fletcher.

From noise trauma experiments on animals and humans, we may conclude that recruitment (abnormal loudness growth) occurs in the cochlea [3.3, 96]. *Steinberg* and *Gardner* described such a loss as a *variable loss* (i. e., sensory neural loss) and partial recruitment as a mixed loss (i. e., having a conductive component) [3.3, 161]. They and Fowler verified the conductive component by estimating the air–bone gap. In a comment to Fowler's original presentation on loudness recruitment in 1937, the famous anatomist *Lorente de Nó* theorized that recruitment is due to hair cell damage [3.14]. *Steinberg* and *Gardner* clearly understood recruitment, as is indicated in the following quote [3.3, p. 20]

> *Owing to the expanding action of this type of loss it would be necessary to introduce a corresponding compression in the amplifier in order to produce the same amplification at all levels.*

This compression/loss model of hearing and hearing loss, along with the loudness models of *Fletcher* and *Munson* [3.41], are basic to an eventual quantitative understanding of NL cochlear signal processing and the cochlea's role in detection, masking and loudness in normal and impaired ears. The work by *Fletcher* [3.162] and *Steinberg* and *Gardner* [3.3], and work on modeling hearing loss and recruitment [3.122] support this view.

In summary, many studies conclude that the cube-root loudness growth starts with the NL compression of basilar membrane motion due to stimulus-dependent voltage changes within the OHC.

### 3.3.5 Determination of the Loudness SNR

In Fig. 3.12 we show a summary of $L(I)$, $\nu(I)$, $J(I)$ and $\Delta L/L = d'/\mathrm{SNR_L}$ for the tone and noise data.

The pure-tone and wide-band noise JND results may be summarized in terms of the loudness $\mathrm{SNR_L}(L)$ data shown in Fig. 3.12d where we show $\Delta L/L = d'/\mathrm{SNR_L}$, as a function of loudness.

For noise below $55\,\mathrm{dB} - \mathrm{SL}$ ($L < 5000\,\mathrm{LU}$) the loudness signal-to-noise ratio $\mathrm{SNR_L} \equiv L/\sigma_\mathrm{L}$ decreases as the square root of the loudness. For a loudness greater than $5000\,\mathrm{LU}$ ($N \approx 5$ sones), $\Delta L/L \approx 0.025$ fn both tones and noise (Fig. 3.12d)

In the lower-right panel (Fig. 3.12d) we provide a functional summary of $\Delta L/L$ for both tones and noise with the light solid line described by

$$\frac{\Delta L(L)}{L} = h\,[\min(L, L_0)]^{-1/2}\ , \tag{3.40}$$

where $h = \sqrt{2}$ and $L_0 = 5000\,\mathrm{LU}$ ($\approx 5$ sone). We call this relation the *saturated Poisson internal noise* (SPIN) model. With these parameter values, (3.40) appears to be a lower bound on the relative loudness $\mathrm{JND_L}$ for both tones and noise. From (3.33) $\Delta L/L = \nu(I)J(I)$. Note how the product of $\nu(I)$ and $J(I)$ is close to a constant for tones above 5000 LU.

In Fig. 3.12b the second top panel shows the exponent $\nu(I)$ for both Fletcher and Munson's and Miller's loudness growth function. In the lower-left panel (Fig. 3.12c) we see $\Delta I/I$ versus $I$ for Miller's subjects, Miller's equation, and Riesz's JND equation.

#### Near miss to Stevens' Law

For tones the intensity exponent $\nu(I)$ varies systematically between 0.3 and 0.4 above $50\,\mathrm{dB} - \mathrm{SL}$, as shown by the solid line in the upper-right panel of Fig. 3.12b. We have highlighted this change in the power law with intensity for a 1 kHz tone in the upper-right panel with a light solid straight line. It is logical to call this effect the *near miss to Stevens' law,* since it cancels the near miss to Weber's law, giving a constant relative loudness JND $\Delta L/L$ for tones.

Figure 3.13a shows the Fletcher–Munson loudness data from Table III in [3.41]. The upper-right panel (Fig. 3.13b) is the slope of the loudness with respect to intensity ($\mathrm{LU\,cm^2/W}$).In the lower–right (Fig. 3.13d) we compare the SPIN model relative JND (3.43) (with $h = 3.0$), and the relative JND computed from the *Jesteadt* et al. [3.136] formula (dashed line) and data from their Table B-I (circles). They measured the JND using pulsed tones for levels between 5 and 80 dB. The Jesteadt et al. data were taken with gated stimuli (100% modulation) and 2AFC methods. It is expected that the experimental method would lead to a different value of $h$ than the valued required for Riesz's data set. The discrepancy between 0 and 20 dB may be due to the 100% modulation for these stimuli. The fit from 20 to $80\,\mathrm{dB} - \mathrm{SL}$ is less than a 5% maximum error, and much less in terms of RMS error. Note the similarity in slope between the model and the data.

### 3.3.6 Weber–Fraction Formula

In this section we derive the relation between the Weber fraction $J(I)$ given the loudness $L(I)$ starting from the *small-JND approximation*

$$\Delta L = \Delta I L'(I)\,, \tag{3.41}$$

where $L'(I) \equiv \mathrm{d}L/\mathrm{d}I$. If we solve this equation for $\Delta I$ and divide by $I$ we find

$$J(I) \equiv \frac{\Delta I}{I} = \frac{\Delta L}{IL'(I)}\ . \tag{3.42}$$

Finally we substitute the SPIN model (3.40)

$$J(I) = \frac{hL(I)}{IL'(I)}\,[\min(L(I), L_0)]^{-1/2}\ . \tag{3.43}$$

This formula is the same as that derived by *Hellman* and *Hellman* [3.134], when $L \le L_0$. In Fig. 3.13c we plot (3.43) labeled *SPIN-model* with $h = 2.4$ and $L_0 = 10\,000\,\mathrm{LU}$. For levels between 0 and $100\,\mathrm{dB} - \mathrm{SL}$, the SPIN model (solid curve) fit to Riesz's data and Riesz's formula is excellent. Over this 100 dB range the curve defined by the loudness function fits as well as the curve defined by Riesz's formula [3.127]. The excellent fit gives us further confidence in the basic assumptions of the model.

## 3.4 Discussion and Summary

Inspired by the Poisson internal noise (PIN)-based theory of *Hellman* and *Hellman* [3.134], we have developed a theoretical framework that can be used to explore the relationship between the pure-tone loudness and the intensity JND. The basic idea is to combine Fletcher's neural excitation response pattern model of loudness with signal detection theory. We defined a random decision variable called the single-trial loudness. The *mean* of this random variable is the loudness, while its *standard deviation* is proportional to the loudness JND. We define the loudness signal-to-noise ratio $\mathrm{SNR_L}$ as the ratio of loudness (the signal) to standard deviation (a measure of the noise).

### 3.4.1 Model Validation

To evaluate the model we have compared the loudness data of *Fletcher* and *Munson* [3.41] with the intensity JND data of *Riesz* [3.75], for tones. A similar comparison was made for noise using loudness and intensity JND data from *Miller* [3.133]. We were able to unify the tone and noise data by two equivalent methods in Fig. 3.12d. Since the loudness SNR is proportional to the ratio of the loudness to the JND $L/\Delta L$, the SNR is also a piecewise power-law function which we call the SPIN model. All the data are in excellent agreement with the SPIN model, providing support for the validity of this theory.

The above discussion has

- drawn out the fundamental nature of the JND,
- shown that the PIN loudness model holds below 5 sone (5000 LU) (the solid line in the lower right panel of Fig. 3.11 below 5000 LU obeys the PIN model, and the data for both tones and wide band noise fall close to this line below 5000 LU) (one sone is 975 LU [3.127, p. 3631], thus 5000 LU = 5.13 LU. From the loudness scale this corresponds to a 1 kHz pure tone at 60 dB − SL),
- shown that above 5 sone the PIN model fails and the loudness SNR remains constant.

### 3.4.2 The Noise Model

#### The SPIN Model

Equation (3.40) summarizes our results on the relative loudness JND for both tones and noise. Using this formula along with (3.32), the JND may be estimated for tones and noise once the loudness has been determined, by measurement, or by model. Fechner's postulate, that the loudness JND is constant, is not supported by our analysis, in agreement with *Stevens* [3.153].

#### The PIN Model

The success of the PIN model is consistent with the idea that the pure-tone loudness code is based on neural discharge rate. This theory should apply between threshold and moderate intensities (e.g., < 60 dB) for *frozen stimuli* where the JND is limited by internal noise.

#### CNS Noise

Above 60 dB − SL we find that the loudness signal-to-noise ratio saturated (Fig. 3.12d) with a constant loudness SNR between 30 and 50 for both the tone and noise conditions, as summarized by Ekman's law [3.163]. We conclude that the Hellman and Hellman theory must be modified to work above 5 sones.

#### Weber's Law

It is significant that, while both $J(I)$ and $\nu(I)$ vary with intensity, the product is constant above 60 dB − SL. Given that $J = d'/\nu\mathrm{SNR_L}$, the saturation in $\mathrm{SNR_L}$ explains Weber's law for wideband signals (since $\nu$ and $\mathrm{SNR_L}$ for that case are constant) as well as the near miss to Weber's law for tones, where $\nu$ is not constant (the near miss to Stevens' law, Fig. 3.12a).

#### Generalization to Other Data

If $\sigma_L(L, I)$ depends on $L$, and is independent of $I$, then the $\mathrm{SNR_L}(L)$ should not depend on the nature of the function $L(I)$ (i. e., it should be true for any $L(I)$). This prediction is supported by our analysis summarized by (3.40). It will be interesting to see how $\mathrm{SNR_L}$ depends on $L$ and $I$ for subjects having a hearing-loss-induced recruitment, and how well this theory explains other data in the literature, such as loudness and JNDs with masking-induced recruitment [3.126].

#### Conditions for Model Validity

To further test the SPIN model, several conditions must be met. First the loudness and the JND must have been measured under the same stimulus conditions. Second, the internal noise must be the dominant factor in determining the JND. This means that the stimuli must be frozen (or have significant duration and bandwidth), and the subjects well trained in the task. As the signal uncertainty begins to dominate the internal noise, as it does in

Amazon EX1010 Page 87

the cases of roving the stimulus, the intensity JND will become independent of the loudness.

As discussed by *Stevens* and *Davis* [3.164, pp. 141-143], JND data are quite sensitive to the modulation conditions. The *Riesz* [3.75] and *Munson* [3.165] data make an interesting comparison because they are taken under steady–state conditions and are long duration tonal signals. Both sets of experimental data (i. e., Riesz and Munson) were taken in the same laboratory within a few years of each other. In 1928 Wegel, Riesz, and Munson were all members of Fletcher's department. *Riesz* [3.75] states that he used the same methods as *Wegel* and *Lane* [3.20], and it is likely that *Munson* [3.165] did as well.

Differences in the signal conditions are the most likely explanation for the differences observed in the intensity JND measurements of Riesz and Jesteadt shown in Fig. 3.13d. One difference between the data of *Riesz* [3.75] and *Jesteadt* et al. [3.136] is that Riesz varied the amplitude of the tones in a sinusoidal manner with a small (i. e., just detectable) modulation index, while Jesteadt et al. alternated between two intervals of different amplitude, requiring that the tones be gated on and off (i. e., a 100% modulation index).

The neural response to transient portions of a stimulus is typically larger than the steady-state response (e.g., neural overshoot) and, therefore, may dominate the perception of stimuli with large, abrupt changes in amplitude. The fact that the intensity JND is sensitive to the time interval between two tones of different amplitude [3.164] is another indication that neural overshoot may play a role.

It would be interesting to check the SPIN model on loudness and JND data taken using gated signals, given the observed sensitivity to the modulation. While these JND data are available [3.136], one would need loudness data taken with identical (or at least similar) modulations. We are not aware of such data.

## References


3.1 E. Relkin, C. Turner: A reexamination of forward masking in the auditory nerve, J. Acoust. Soc. Am. **84**(2), 584–591 (1988)

3.2 M. Hewitt, R. Meddis: An evaluation of eight computer models of mammalian inner hair-cell function, J. Acoust. Soc. Am. **90**(2), 904–917 (1991)

3.3 J. Steinberg, M. Gardner: Dependence of hearing impairment on sound intensity, J. Acoust. Soc. Am. **9**, 11–23 (1937)

3.4 J.O. Pickles: *An Introduction to the Physiology of Hearing* (Academic, London 1982)

3.5 P. Dallos: Cochlear neurobiology. In: *The Cochlea*, ed. by P. Dallos, A. Popper, R. Fay (Springer, New York 1996) pp. 186–257

3.6 W.A. Yost: *Fundamentals of Hearing, An Introduction* (Academic, San Diego, London 2006)

3.7 S. Hecht: Vision II, The nature of the photoreceptor process. In: *Handbook of General Experimental Psychology*, ed. by C. Murchison (Clark Univ. Press, Worcester 1934)

3.8 G. Gescheider: *Psychophysics: The Fundamentals*, 3rd edn. (Lawrence Erlbaum, Mahwah 1997)

3.9 J.B. Allen, S. Neely: Micromechanical models of the cochlea, Phys. Today **45**(7), 40–47 (1992)

3.10 J.B. Allen: Harvey Fletcher's role in the creation of communication acoustics, J. Acoust. Soc. Am. **99**(4), 1825–1839 (1996)

3.11 A. Hudspeth, D. Corey: Sensitivity, polarity, and conductance change in the response of vertebrate hair cells to controlled mechanical stimuli, Proc. Nat. Acad. Sci. **74**(6), 2407–2411 (1977)

3.12 M. Liberman: Single-neuron labeling in the cat auditory nerve, Science **216**, 163–176 (1982)

3.13 J. Steinberg: Stereophonic sound-film system-pre- and post-equalization of compandor systems, J. Acoust. Soc. Am. **13**, 107–114 (1941), B-1327

3.14 R.L. de Nó: The diagnosis of diseases of the neural mechanism of hearing by the aid of sounds well above threshold, T. Am. Otol. Soc. **27**, 219–220 (1937looseness1)

3.15 S.T. Neely, J.B. Allen: Relation between the rate of growth of loudness and the intensity DL. In: *Modeling Sensorineural Hearing Loss*, ed. by W. Jesteadt (Lawrence Erlbaum, Mahwah 1997) pp. 213–222

3.16 T.S. Littler: *The Physics of the Ear* (Pergamon, Oxford 1965)

3.17 W.M. Hartmann: *Signals, Sound, and Sensation* (AIP Press, Woodbury 1997)

3.18 H.L.F. Helmholtz: *On the Sensations of Tone* (Dover, New York 1954), 1863

3.19 H.L.F. Helmholtz: *Helmholtz's popular scientific lectures* (Dover, New York 1962), 1857

3.20 R.L. Wegel, C. Lane: The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear, Phys. Rev. **23**, 266–285 (1924)

3.21 H. Fletcher: Physical measurements of audition and their bearing on the theory of hearing, J. Franklin Inst. **196**(3), 289–326 (1923)

3.22 G. Campbell: On loaded lines in telephonic transmission, Phil. Mag. **5**, 313–331 (1903)

3.23 G. Campbell: Physical theory of the electric wave filter, Bell System Tech. J. **1**(1), 1–32 (1922)
3.24 G.A. Campbell: Telephonic intelligibility, Phil. Mag. **19**(6), 152–159 (1910)
3.25 H. Fletcher: The nature of speech and its interpretation, J. Franklin Inst. **193**(6), 729–747 (1922)
3.26 H. Fletcher, J. Steinberg: Articulation testing methods, J. Acoust. Soc. Am. **1**(2.2), 17–113 (1930), Intelligibility Lists pp. 65–113)
3.27 H. Fletcher: A space-time pattern theory of hearing, J. Acoust. Soc. Am. **1**(1), 311–343 (1930)
3.28 J. Zwislocki: Theorie der Schneckenmechanik, Acta Otolaryngol. **72**, 1–112 (1948)
3.29 J. Zwislocki: Theory of the acoustical action of the cochlea, J. Acoust. Soc. Am. **22**, 779–784 (1950)
3.30 O. Ranke: Theory operation of the cochlea: A contribution to the hydrodynamics of the cochlea, J. Acoust. Soc. Am. **22**, 772–777 (1950)
3.31 L.C. Peterson, B.P. Bogert: A dynamical theory of the cochlea, J. Acoust. Soc. Am. **22**, 369–381 (1950)
3.32 H. Fletcher: On the dynamics of the cochlea, J. Acoust. Soc. Am. **23**, 637–645 (1951)
3.33 H. Fletcher: Acoustics, Phys. Today **4**, 12–18 (1951)
3.34 S. Puria, J.B. Allen: Measurements and model of the cat middle ear: Evidence for tympanic membrane acoustic delay, J. Acoust. Soc. Am. **104**(6), 3463–3481 (1998)
3.35 J.B. Allen, P.S. Jeng, H. Levitt: Evaluating human middle ear function via an acoustic power assessment, J. Rehabil. Res. Dev. **42**(4), 63–78 (2005)
3.36 H. Fletcher, W. Munson: Relation between loudness and masking, J. Acoust. Soc. Am. **9**, 1–10 (1937)
3.37 H. Fletcher: Loudness, masking and their relation to the hearing process and the problem of noise measurement, J. Acoust. Soc. Am. **9**, 275–293 (1938)
3.38 H. Fletcher: Auditory patterns, Rev. Mod. Phys. **12**(1), 47–65 (1940)
3.39 J. Steinberg: Positions of stimulation in the cochlea by pure tones, J. Acoust. Soc. Am. **8**, 176–180 (1937), cochlear map estimate, Monograph B-973
3.40 D.D. Greenwood: Auditory masking and the critical band, J. Acoust. Soc. Am. **33**(4), 484–502 (1961)
3.41 H. Fletcher, W. Munson: Loudness, its definition, measurement, and calculation, J. Acoust. Soc. Am. **5**, 82–108 (1933)
3.42 H. Fletcher: *Speech and Hearing in Communication* (Krieger, Huntington 1953)
3.43 D.D. Greenwood: Critical bandwidth and the frequency coordinates of the basilar membrane, J. Acoust. Soc. Am. **33**, 1344–1356 (1961)
3.44 M. Liberman: The cochlear frequency map for the cat: Labeling auditory-nerve fibers of known characteristic frequency, J. Acoust. Soc. Am. **72**(5), 1441–1449 (1982)
3.45 M. Liberman, L. Dodds: Single neuron labeling and chronic cochlear pathology III: Stereocilia damage and alterations of threshold tuning curves, Hearing Res. **16**, 55–74 (1984)
3.46 W. Rhode: Some observations on cochlear mechanics, J. Acoust. Soc. Am. **64**, 158–176 (1978)
3.47 W. Rhode: Observations of the vibration of the basilar membrane in squirrel monkeys using the Mössbauer technique, J. Acoust. Soc. Am. **49**, 1218–1231 (1971)
3.48 P. Sellick, I. Russell: Intracellular studies of cochlear hair cells: Filling the gap between basilar membrane mechanics and neural excitation. In: *Evoked Electrical Activity in the Auditory Nervous System*, ed. by F. Naunton, C. Fernandez (Academic, New York 1978) pp. 113–140
3.49 D. Kemp: Stimulated acoustic emissions from within the human auditory system, J. Acoust. Soc. Am. **64**, 1386–1391 (1978)
3.50 W. Brownell, C. Bader, D. Bertran, Y. de Rabaupierre: Evoked mechanical responses of isolated cochlear outer hair cells, Science **227**, 194–196 (1985)
3.51 J.B. Allen: OHCs shift the excitation pattern via BM tension. In: *Diversity in Auditory Mechanics*, ed. by E. Lewis, G. Long, R. Lyon, P. Narins, C. Steele, E. Hecht-Poinar (World Scientific, Singapore 1997) pp. 167–175
3.52 J.L. Goldstein, N. Kiang: Neural correlates of the aural combination tone 2f1-f2, Proc. IEEE **56**, 981–992 (1968)
3.53 G. Smoorenburg: Combination tones and their origin, J. Acoust. Soc. Am. **52**(2), 615–632 (1972)
3.54 D. Kemp: Evidence of mechanical nonlinearity and frequency selective wave amplification in the cochlea, Arch. Oto-Rhino-Laryngol. **224**, 37–45 (1979)
3.55 D. Kim, J. Siegel, C. Molnar: Cochlear nonlinear phenomena in two-tone responses, Scand. Audiol. **9**, 63–82 (1979)
3.56 P.F. Fahey, J.B. Allen: Nonlinear phenomena as observed in the ear canal and at the auditory nerve, J. Acoust. Soc. Am. **77**(2), 599–612 (1985)
3.57 M.B. Sachs, Y.S. Kiang: Two-tone inhibition in auditory-nerve fibers, J. Acoust. Soc. Am. **43**, 1120–1128 (1968)
3.58 R. Arthur, R. Pfeiffer, N. Suga: Properties of "two-tone inhibition" in primary auditory neurons, J. Physiol. (London) **212**, 593–609 (1971)
3.59 N.-S. Kiang, E. Moxon: Tails of tuning curves of auditory-nerve fibers, J. Acoust. Soc. Am. **55**, 620–630 (1974)
3.60 P. Abbas, M. Sachs: Two-tone suppression in auditory–nerve fibers: Extension of a stimulus–response relationship, J. Acoust. Soc. Am. **59**(1), 112–122 (1976)
3.61 X. Pang, J. Guinan: Growth rate of simultaneous masking in cat auditory-nerve fibers: Relationship to the growth of basilar-membrane motion and the origin of two-tone suppression, J. Acoust. Soc. Am. **102**(6), 3564–3574 (1997), beautiful data showing the slope of suppression for low frequency suppressors




Amazon EX1010 Page 89

3.62 I. Russell, P. Sellick: Intracellular studies of hair cells in the mammalian cochlea, J. Physiol. **284**, 261–290 (1978)

3.63 H. Duifhuis: Consequences of peripheral frequency selectivity for nonsimultaneous masking, J. Acoust. Soc. Am. **54**(6), 1471–1488 (1973)

3.64 D. Kemp: The evoked cochlear mechanical response and the auditory microstructure – evidence for a new element in cochlear mechanics, Scand. Audiol. **9**, 35–47 (1979)

3.65 D. Kemp: Towards a model for the origin of cochlear echoes, Hearing Res. **2**, 533–548 (1980)

3.66 D. Kemp: Otoacoustic emissions, travelling waves and cochlear mechanisms, J. Acoust. Soc. Am. **22**, 95–104 (1986)

3.67 A. Recio, W. Rhode: Basilar membrane responses to broadband stimuli, J. Acoust. Soc. Am. **108**(5), 2281–2298 (2000)

3.68 E. Fowler: A method for the early detection of otosclerosis, Archiv. Otolaryngol. **24**(6), 731–741 (1936)

3.69 S. Neely, D.O. Kim: An active cochlear model showing sharp tuning and high sensitivity, Hearing Res. **9**, 123–130 (1983)

3.70 D. He, P. Dallos: Somatic stiffness of cochlear outer hair cells is voltage-dependent, Proc. Nat. Acad. Sci. **96**(14), 8223–8228 (1999)

3.71 D. He, P. Dallos: Properties of voltage-dependent somatic stiffness of cochlear outer hair cells, J. Assoc. Res. Otolaryngol. **1**(1), 64–81 (2000)

3.72 A.M. Mayer: Research in acoustics, Phil. Mag. **2**, 500–507 (1876), in Benchmark Papers in Acoustics, Vol. 13, edited by E. D. Schubert

3.73 E. Titchener: *Experimental Psychology, A Manual of Laboratory Practice*, Vol. II (Macmillan, London 1923)

3.74 J.B. Allen: A short history of telephone psychophysics, J. Audio Eng. Soc. Reprint **4636**, 1–37 (1997)

3.75 R.R. Riesz: Differential intensity sensitivity of the ear for pure tones, Phys. Rev. **31**(2), 867–875 (1928)

3.76 D. McFadden: The curious half-octave shift: Evidence of a basalward migration of the traveling-wave envelope with increasing intensity. In: *Applied and Basic Aspects of Noise-Induced Hearing Loss*, ed. by R. Salvi, D. Henderson, R. Hamernik, V. Coletti (Plenum, New York 1986) pp. 295–312

3.77 W.A. Munson, M.B. Gardner: Loudness patterns – a new approach, J. Acoust. Soc. Am. **22**(2), 177–190 (1950)

3.78 B. Strope, A. Alwan: A model of dynamic auditory perception and its application to robust word recognition, IEEE Trans. Acoust. Speech Signal Proc. **5**(5), 451–464 (1997)

3.79 J.B. Allen: Psychoacoustics. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*, Vol. 17, ed. by J. Webster (Wiley, New York 1999) pp. 422–437

3.80 B. Delgutte: Two-tone suppression in auditory-nerve fibres: Dependence on suppressor frequency and level, Hearing Res. **49**, 225–246 (1990)

3.81 B. Delgutte: Physiological mechanisms of psychophysical masking: Observations from auditory-nerve fibers, J. Acoust. Soc. Am. **87**, 791–809 (1990)

3.82 B. Delgutte: Physiological models for basic auditory percepts. In: *Auditory Computation*, ed. by H. Hawkins, T. McMullen (Springer, New York 1995)

3.83 L. Kanis, E. de Boer: Two-tone suppression in a locally active nonlinear model of the cochlea, J. Acoust. Soc. Am. **96**(4), 2156–2165 (1994)

3.84 J. Hall: Two-tone distortion products in a nonlinear model of the basilar membrane, J. Acoust. Soc. Am. **56**, 1818–1828 (1974)

3.85 C.D. Geisler, A.L. Nuttall: Two-tone suppression of basilar membrane vibrations in the base of the guinea pig cochlea using "low-side" suppressors, J. Acoust. Soc. Am. **102**(1), 430–440 (1997)

3.86 M. Régnier, J.B. Allen: *The Importance of Across-Frequency Timing Coincidences in the Perception of Some English Consonants in Noise* (ARO, Denver 2007), Abstract

3.87 M. Régnier, J.B. Allen: *Perceptual Cues of Some CV Sounds Studied in Noise* (AAS, Scottsdale 2007), Abstract

3.88 S. Narayan, A. Temchin, A. Recio, M. Ruggero: Frequency tuning of basilar membrane and auditory nerve fibers in the same cochleae, Science **282**, 1882–1884 (1998), shows BM and neural differ by 3.8 dB/oct

3.89 E. deBoer: Mechanics of the cochlea: Modeling efforts. In: *The Cochlea*, ed. by P. Dallos, A. Popper, R. Fay (Springer, New York 1996) pp. 258–317

3.90 D.C. Geisler: *From Sound to Synapse: Physiology of the Mammalian Ear* (Oxford Univ. Press, Oxford 1998)

3.91 J.B. Allen, P.F. Fahey: Using acoustic distortion products to measure the cochlear amplifier gain on the basilar membrane, J. Acoust. Soc. Am. **92**(1), 178–188 (1992)

3.92 S. Neely, D.O. Kim: A model for active elements in cochlear biomechanics, J. Acoust. Soc. Am. **79**, 1472–1480 (1986)

3.93 E. deBoer, A. Nuttall: The mechanical waveform of the basilar membrane, III Intensity effects, J. Acoust. Soc. Am. **107**(3), 1496–1507 (2000)

3.94 D. Kim, S. Neely, C. Molnar, J. Matthews: An active cochlear model with negative damping in the cochlear partition: Comparison with Rhode's ante- and post-mortem results. In: *Psychological, Physiological and Behavioral Studies in Hearing*, ed. by G. van den Brink, F. Bilsen (Delft Univ. Press, Delft 1980) pp. 7–14

3.95 J.B. Allen: DeRecruitment by multiband compression in hearing aids. In: *Psychoacoustics, Speech, and Hearing aids*, ed. by B. Kollmeier (World Scientific, Singapore 1996) pp. 141–152

3.96 W.F. Carver: Loudness balance procedures. In: *Handbook of Clinical Audiology*, 2nd edn., ed. by J. Katz (Williams Wilkins, Baltimore 1978) pp. 164–178, Chap. 15


Amazon EX1010 Page 90

3.97 J.B. Allen: Nonlinear cochlear signal processing. In: *Physiology of the Ear*, 2nd edn., ed. by A. Jahn, J. Santos-Sacchi (Singular, San Diego 2001) pp. 393–442, Chap. 19

3.98 S. Neely: A model of cochlear mechanics with outer hair cell motility, J. Acoust. Soc. Am. **94**, 137–146 (1992)

3.99 J. Ashmore: A fast motile response in guinea-pig outer hair cells: the molecular basis of the cochlear amplifier, J. Physiol. (London) **388**, 323–347 (1987)

3.100 J. Santos-Sacchi: Reversible inhibition of voltage–dependent outer hair cell motility and capacitance, J. Neurosci. **11**(10), 3096–3110 (1991)

3.101 K. Iwasa, R. Chadwick: Elasticity and active force generation of cochlear outer hair cells, J. Acoust. Soc. Am. **92**(6), 3169–3173 (1992)

3.102 I. Russell, P. Legan, V. Lukashkina, A. Lukashkin, R. Goodyear, G. Richardson: Sharpened cochlear tuning in a mouse with a genetically modified tectorial membrane, Nat. Neurosci. **10**, 215–223 (2007)

3.103 D. Sen, J.B. Allen: Functionality of cochlear micromechanics – as elucidated by the upward spread of masking and two tone suppression, Acoustics Australia **34**(1), 43–51 (2006)

3.104 J.B. Allen: Cochlear micromechanics: A physical model of transduction, J. Acoust. Soc. Am. **68**(6), 1660–1670 (1980)

3.105 J.B. Allen: Cochlear micromechanics – A mechanism for transforming mechanical to neural tuning within the cochlea, J. Acoust. Soc. Am. **62**, 930–939 (1977)

3.106 P. Dallos, D.Z. He, X. Lin, I. Sziklai, S. Mehta, B.N. Evans: Acetylcholine, outer hair cell electromotility, and the cochlear amplifier, J. Neurosci. **17**(6), 2212–2226 (1997)

3.107 P. Dallos: Prestin and the electromechanical responses of outer hair cells, ARO-2002 **25**, 189 (2002)

3.108 J.B. Allen: Derecruitment by multiband compression in hearing aids. In: *The Efferent Auditory System*, ed. by C. Berlin (Singular, San Diego 1999) pp. 73–86, Chap. 4 (includes a CDROM video talk by J. B. Allen in MP3 format)

3.109 J.B. Allen, D. Sen: Is tectorial membrane filtering required to explain two tone suppression and the upward spread of masking?. In: *Recent Developments in Auditory Mechanics*, ed. by H. Wada, T. Takasaka, K. Kieda, K. Ohyama, T. Koike (World Scientific, Singapore 1999) pp. 137–143

3.110 W. Sewell: The effects of furosemide on the endocochlear potential and auditory-nerve fiber tuning curves in cats, Hearing Res. **14**, 305–314 (1984)

3.111 J.B. Allen, P.F. Fahey: Nonlinear behavior at threshold determined in the auditory canal on the auditory nerve. In: *Hearing – Physiological Bases and Psychophysics*, ed. by R. Klinke, R. Hartmann (Springer, Berlin, Heidelberg 1983) pp. 128–134

3.112 J.B. Allen, B.L. Lonsbury-Martin: Otoacoustic emissions, J. Acoust. Soc. Am. **93**(1), 568–569 (1993)

3.113 P.F. Fahey, J.B. Allen: Measurement of distortion product phase in the ear canal of cat, J. Acoust. Soc. Am. **102**(5), 2880–2891 (1997)

3.114 R. Diependaal, E. de Boer, M. Viergever: Cochlear power flux as an indicator of mechanical activity, J. Acoust. Soc. Am. **82**, 917–926 (1987)

3.115 G. Zweig: Finding the impedance of the organ of Corti, J. Acoust. Soc. Am. **89**, 1229–1254 (1991)

3.116 E. deBoer, A. Nuttall: The "inverse problem" solved for a three-dimensional model of the cochlear, III Brushing-up the solution method, J. Acoust. Soc. Am. **105**(6), 3410–3420 (1999)

3.117 E. deBoer, A. Nuttall: The mechanical waveform of the basilar membrane, II From data to models – and back, J. Acoust. Soc. Am. **107**(3), 1487–1496 (2000)

3.118 G. Zweig: Finding the impedance of the organ of Corti, J. Acoust. Soc. Am. **89**(3), 1276–1298 (1991)

3.119 E. deBoer, A. Nuttall: The "inverse problem" solved for a three-dimensional model of the cochlea. II Application to experimental data sets, J. Acoust. Soc. Am. **98**(2), 904–910 (1995)

3.120 C. Shera, J. Guinan: Cochlear traveling-wave amplification, suppression, and beamforming probed using noninvasive caliration of intracochlear distortion sources, J. Acoust. Soc. Am. **121**(2), 1003–1016 (2007)

3.121 L.A. Pipes: *Applied Mathematics for Engineers and Physicists* (McGraw-Hill, New York 1958)

3.122 J.B. Allen: Modeling the noise damaged cochlea. In: *The Mechanics and Biophysics of Hearing*, ed. by P. Dallos, C.D. Geisler, J.W. Matthews, M.A. Ruggero, C.R. Steele (Springer, New York 1991) pp. 324–332

3.123 I.J. Russell, G.P. Richardson, A.R. Cody: Mechanosensitivity of mammalian auditory hair cells in vitro, Nature **321**(29), 517–519 (1986)

3.124 E.G. Boring: *History of Psychophysics* (Appleton–Century, New York 1929)

3.125 G. Fechner: Translation of "Elemente der Psychophysik". In: *Elements of Psychophysics*, Vol. I, ed. by H. Adler (Holt Rinehart Winston, New York 1966)

3.126 R. Schlauch, S. Harvey, N. Lanthier: Intensity resolution and loudness in broadband noise, J. Acoust. Soc. Am. **98**(4), 1895–1902 (1995)

3.127 J.B. Allen, S.T. Neely: Modeling the relation between the intensity JND and loudness for pure tones and wide–band noise, J. Acoust. Soc. Am. **102**(6), 3628–3646 (1997)

3.128 J. Zwislocki, H. Jordan: On the relation of intensity JNDs to loudness and neural noise, J. Acoust. Soc. Am. **79**, 772–780 (1986)

3.129 N.F. Viemeister: Psychophysical aspects of auditory intensity coding. In: *Auditory Function*, ed. by G. Edelman, W. Gall, W. Cowan (Wiley, New York 1988) pp. 213–241, Chap. 7

3.130 C. Plack, R. Carlyon: Loudness perception and intensity coding. In: *Hearing, Handbook of Perception and Cognition*, ed. by B. Moore (Academic, San Diego 1995) pp. 123–160, Chap. 4




Amazon EX1010 Page 91

3.131 J.B. Allen: Harvey Fletcher 1884–1981. In: *The ASA edition of Speech, Hearing in Communication*, ed. by J.B. Allen (Acoust. Soc. Am., Woodbury 1995) pp. 1–34

3.132 D.M. Green, J.A. Swets: *Signal Detection Theory and Psychophysics* (Wiley, New York 1966)

3.133 G.A. Miller: Sensitivity to changes in the intensity of white noise and its relation to masking and loudness, J. Acoust. Soc. Am. **19**, 609–619 (1947)

3.134 W. Hellman, R. Hellman: Intensity discrimination as the driving force for loudness, Application to pure tones in quiet, J. Acoust. Soc. Am. **87**(3), 1255–1271 (1990)

3.135 J. Egan, H. Hake: On the masking pattern of a simple auditory stimulus, J. Acoust. Soc. Am. **22**, 622–630 (1950)

3.136 W. Jesteadt, C. Wier, D. Green: Intensity discrimination as a function of frequency and sensation level, J. Acoust. Soc. Am. **61**(1), 169–177 (1977)

3.137 S. Stevens, H. Davis: *Hearing, Its Psychology and Physiology* (Acoust. Soc. Am., Woodbury 1983)

3.138 W.A. Yost: *Fundamentals of Hearing, An Introduction* (Academic, San Diego, London 1994)

3.139 W. Munson: The growth of auditory sensation, J. Acoust. Soc. Am. **19**, 584–591 (1947)

3.140 W. Siebert: Some implications of the stochastic behavior of primary auditory neurons, Kybernetik **2**, 205–215 (1965)

3.141 D. Raab, I. Goldberg: Auditory intensity discrimination with bursts of reproducible noise, J. Acoust. Soc. Am. **57**(2), 437–447 (1975)

3.142 H.C. Montgomery: Influence of experimental technique on the measurement of differential intensty sensitivity of the ear, J. Acoust. Soc. Am. **7**, 39–43 (1935)

3.143 P.G. Nutting: The complete form of Fechner's law, Bull. Bureau Standards **3**(1), 59–64 (1907)

3.144 H. Fletcher: *Speech and Hearing* (Van Nostrand, New York 1929)

3.145 E.H. Weber: Der Tastsinn und das Gemainfül. In: *Handwörterbuch der Physiologie*, Vol. 3, ed. by R. Wagner (Vieweg, Braunschweig 1988) pp. 481–588, Chap. 7

3.146 J. Johnson, C. Turner, J. Zwislocki, R. Margolis: Just noticeable differences for intensity and their relation to loudness, J. Acoust. Soc. Am. **93**(2), 983–991 (1993)

3.147 B.C.J. Moore: *An Introduction to the Psychology of Hearing*, 2nd edn. (Academic, London, New York 1982)

3.148 W.J. McGill, J.P. Goldberg: A study of the near–miss involving Weber's law and pure tone intensity discrimination, Percept. Psychophys. **4**, 105–109 (1968)

3.149 D. Green: Audition: Psychophysics and perception. In: *Stevens' Handbook of Experimental Psychology*, ed. by R. Atkinson, R. Herrnstein, G. Lindzey, R. Luce (Wiley, New York 1988) pp. 327–376, Chap. 6

3.150 D. Green: Application of detection theory in psychophysics, Proc. IEEE **58**(5), 713–723 (1970)

3.151 S. Stevens: Mathematics, measurement, and psychophysics. In: *Handbook of Experimental Psychology*, ed. by S. Stevens (Wiley, New York 1951) pp. 1–49, Chap. 1

3.152 R. Luce: *Sound and Hearing* (Lawrence Erlbaum, Hilldale 1993)

3.153 S. Stevens: To honor Fechner and repeal his law, Science **133**(3446), 80–86 (1961)

3.154 H. Fletcher: Physical measurements of audition and their bearing on the theory of hearing, Bell System Tech. J. **ii**(4), 145–180 (1923)

3.155 H. Fletcher, J. Steinberg: The dependence of the loudness of a complex sound upon the energy in the various frequency regions of the sound, Phys. Rev. **24**(3), 306–317 (1924)

3.156 J. Steinberg: The loudness of a sound and its physical stimulus, Phys. Rev. **26**, 507 (1925)

3.157 H. De Vries: The quantum character of light and its bearing upon the threshold of vision, the differential sensitivity and the acuity of the eye, Physica **10**, 553–564 (1943)

3.158 W. Siebert: Stimulus transformations in the peripheral auditory system. In: *Recognizing Patterns*, ed. by P. Kolers, M. Eden (MIT Press, Cambridge 1968) pp. 104–133, Chap. 4

3.159 W. Siebert: *Personal communication* (1989)

3.160 W.J. McGill, J.P. Goldberg: Pure–tone intensity discrimination as energy detection, J. Acoust. Soc. Am. **44**, 576–581 (1968)

3.161 J.C. Steinberg, M.B. Gardner: On the auditory significance of the term hearing loss, J. Acoust. Soc. Am. **11**, 270–277 (1940)

3.162 H. Fletcher: A method of calculating hearing loss for speech from an audiogram, J. Acoust. Soc. Am. **22**, 1–5 (1950)

3.163 G. Ekman: Weber's law and related functions, Psychology **47**, 343–352 (1959)

3.164 S. Stevens, H. Davis: *Hearing, Its Psychology and Physiology* (Acoust. Soc. Am., Woodbury 1938)

3.165 W.A. Munson: An experimental determination of the equivalent loudness of pure tones, J. Acoust. Soc. Am. **4**, 7 (1932), Abstract

# 4. Perception of Speech and Sound

B. Kollmeier, T. Brand, B. Meyer

The transformation of acoustical signals into auditory sensations can be characterized by psychophysical quantities such as loudness, tonality, or perceived pitch. The resolution limits of the auditory system produce spectral and temporal masking phenomena and impose constraints on the perception of amplitude modulations. Binaural hearing (i. e., utilizing the acoustical difference across both ears) employs interaural time and intensity differences to produce localization and binaural unmasking phenomena such as the binaural intelligibility level difference, i. e., the speech reception threshold difference between listening to speech in noise monaurally versus listening with both ears.

The acoustical information available to the listener for perceiving speech even under adverse conditions can be characterized using the articulation index, the speech transmission index, and the speech intelligibility index. They can objectively predict speech reception thresholds as a function of spectral content, signal-to-noise ratio, and preservation of amplitude modulations in the speech waveform that enter the listener's ear. The articulatory or phonetic information available to and received by the listener can be characterized by speech feature sets. Transinformation analysis allows one to detect the relative transmission error connected with each of these speech features. The comparison across man and machine in speech recognition allows one to test hypotheses and models of human speech perception. Conversely, automatic speech recognition may be improved by introducing human signal-processing principles into machine processing algorithms.

4.1 **Basic Psychoacoustic Quantities** ..... 62
4.1.1 Mapping of Intensity into Loudness 62
4.1.2 Pitch ..... 64
4.1.3 Temporal Analysis and Modulation Perception ..... 65
4.1.4 Binaural Hearing ..... 67
4.1.5 Binaural Noise Suppression ..... 68
4.2 **Acoustical Information Required for Speech Perception** ..... 70
4.2.1 Speech Intelligibility and Speech Reception Threshold (SRT) ..... 70
4.2.2 Measurement Methods ..... 71
4.2.3 Factors Influencing Speech Intelligibility ..... 72
4.2.4 Prediction Methods ..... 72
4.3 **Speech Feature Perception** ..... 74
4.3.1 Formant Features ..... 75
4.3.2 Phonetic and Distinctive Feature Sets ..... 76
4.3.3 Internal Representation Approach and Higher-Order Temporal-Spectral Features ..... 77
4.3.4 Man–Machine Comparison ..... 80
**References** ..... 81



Acoustically produced speech is a very special sound to our ears and brain. Humans are able to extract the information contained in a spoken message extremely efficiently even if the speech energy is lower than any competing background sound. Hence, humans are able to communicate acoustically even under adverse listening conditions, e.g., in a cafeteria. The process of understanding speech can be subdivided into two stages. First, an auditory pre-processing stage where the speech sound is transformed into its *internal representation* in the brain and special speech features are extracted (such as, e.g., acoustic energy in a certain frequency channel as a function of time, or instantaneous pitch of a speech sound). This process is assumed to be mainly bottom-up with no special preference for speech sounds as compared to other sounds. In other words, the information contained in any of the speech features can be described quite well by the acoustical contents of the input signal to the ears. In a second step, speech pattern recognition takes place under cognitive control where the internally available speech cues are assembled by our brain to convey the underlying message of the speech sound. This

Amazon EX1010 Page 93

process is assumed to be top-down and cognitive controlled, and is dependent on training, familiarity, and attention. In the context of this handbook we primarily consider the first step while assuming that the second step operates in a nearly perfect way in normal human listeners, thus ignoring the vast field of cognitive psychology, neuropsychology of speech, and psycholinguistics. Instead, we consider the psychoacoustics of transforming speech and other sounds into its *internal representation*. We will concentrate on the acoustical prerequisites of speech perception by measuring and modeling the speech information contained in a sound entering the ear, and finally the speech features that are presumably used by our brain to recognize speech.

## 4.1 Basic Psychoacoustic Quantities



The ear converts the temporally and spectrally fluctuating acoustic waveform of incoming speech and sound into a stream of auditory percepts. The most important dimensions of auditory perception are:

- the transformation of sound intensity into subjectively perceived loudness,
- the transformation of major frequency components of the sound into subjectively perceived pitch,
- the transformation of different temporal patterns and rhythms into subjectively perceived fluctuations,
- the transformation of the spectro-temporal contents of acoustic signals into subjectively perceived timbre (which is not independent of the dimensions listed above),
- the transformation of interaural disparities (i. e., differences across both ears) and spectro-temporal contents of acoustical signals into the perceived spatial location and spatial extent of an auditory object.

A basic prerequisite for being able to assign these dimensions to a given sound is the ear's ability to internally separate acoustically superimposed sound sources into different auditory streams or objects.

Psychoacoustics is the scientific discipline that measures and models the relation between physical acoustical quantities (e.g., the intensity of a sinusoidal stimulus specified by sound pressure level, frequency, and duration) and their respective subjective impression (e.g., loudness, pitch, and perceived temporal extent).

### 4.1.1 Mapping of Intensity into Loudness

The absolute threshold in quiet conditions for a continuous sinusoid is highly dependent on the frequency of the pure tone (Fig. 4.1). It shows highest sensitivity in the frequency region around 1 kHz, which also carries most speech information, and increases for low and high frequencies. The normalized threshold in quiet at 1 kHz averaged over a large number of normal hearing subjects is defined as 0 dB sound pressure level (SPL), which corresponds to 20 μPa. As the level of the sinusoid increases, the perceived loudness increases with approximately a doubling of perceived loudness with a level increase by 10 dB. All combinations of sound pressure levels and frequencies of sinusoid that produce the same loudness as a reference 1 kHz sinusoid of a given level (in dB SPL) are denoted as isophones (Fig. 4.1). Hence, the loudness level (in phon) can only be assigned to a sinusoid and not to a multi-frequency mixture of sounds such as speech. This difference is due to the fact that a broadband sound is perceived as being louder than a narrow-band sound at the same

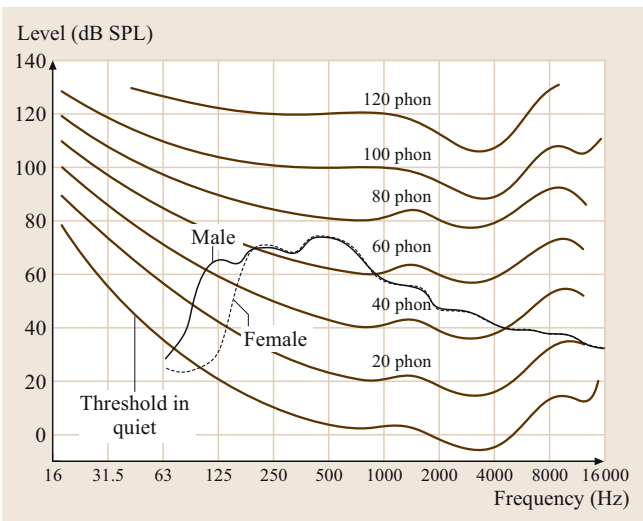



**Fig. 4.1** Auditory field described by the threshold in quiet, the isophones, and the uncomfortable listening level of a continuous sinusoid as a function of tone frequency. Also given is an average speech spectrum for male and female speech plotted as a power density

Amazon EX1010 Page 94

sound pressure level, which puts all of its energy into one *critical band*. In order to express the speech sound pressure level in a way that approximates the human loudness impression, there are several options available:

- Unweighted root-mean-square (RMS) level: the total signal intensity over the audio frequency range is averaged within a certain time window (denoted as *slow*, *fast*, or *impulse*, respectively, for standardized sound level meters, or as the RMS value for a digitized speech signal) and is expressed as dB SPL, i. e., as $10\log(I/I_0)$ where $I$ denotes the signal intensity and $I_0$ denotes the reference signal intensity at auditory threshold. The usage of the dB scale already takes into account the Weber–Fechner law of psychophysics: roughly, a sound intensity difference of 1 dB can be detected as the just notice-

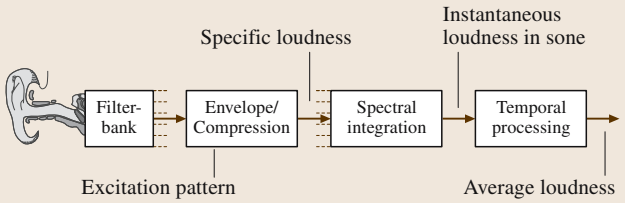



**Fig. 4.2** Block diagram of a loudness model

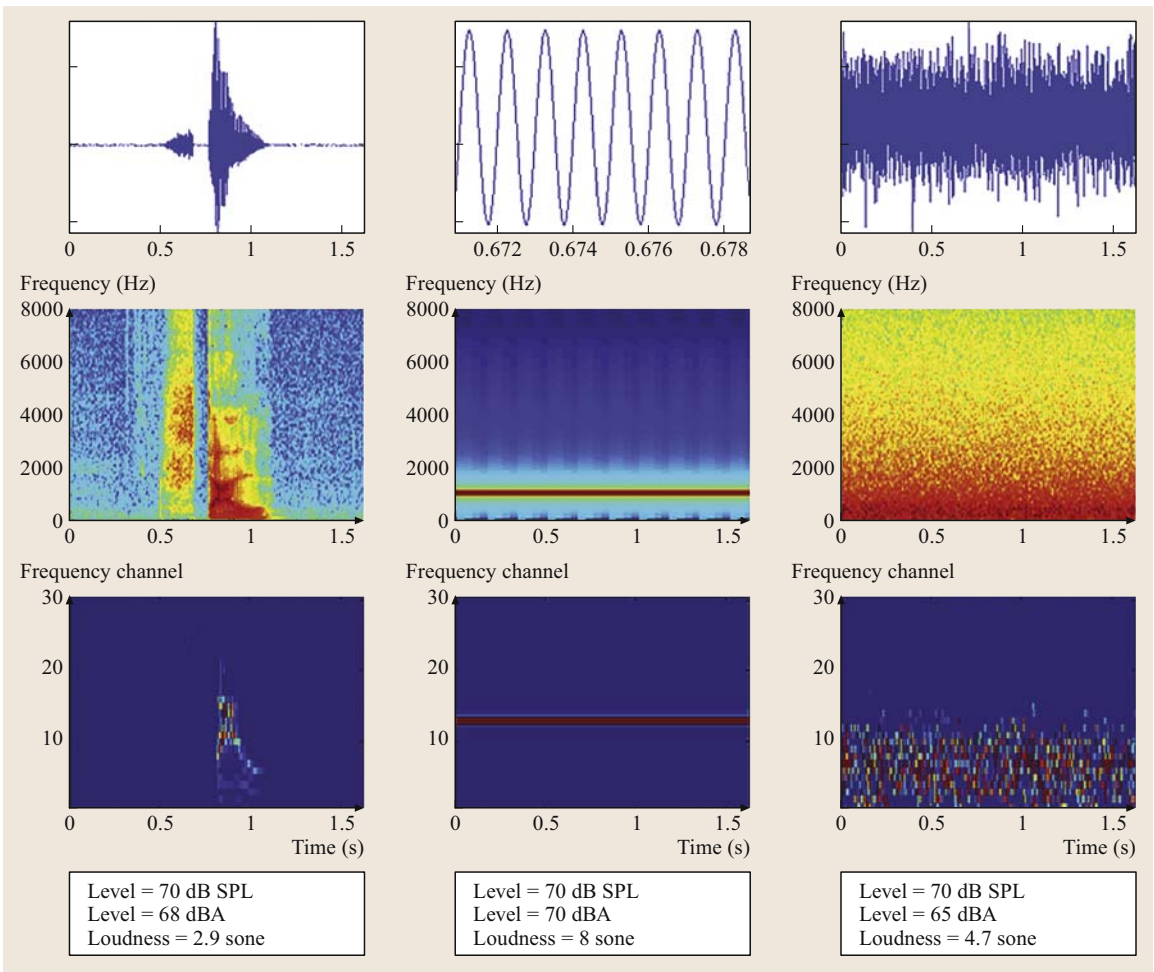



**Fig. 4.3** Example plot of a speech sample, 1 kHz sinusoidal tone and a continuous speech-shaped noise sample represented as a waveform, spectrogram, partial loudness pattern and resulting value in dB SPL, dBA and loudness in sone

able intensity difference irrespective of the reference level.

- A-weighted signal level: in order to account for the higher sensitivity of the ear to mid-frequencies at low levels, a spectral weighting function that approximates the isophones between 20 and 30 phon (denoted as A-weighting) is applied to the spectral components of the sound before they are summed up to give the total sound level. The B- and C-weighting curves are available for higher signal levels. Note that the A-, B-, and C-weighted speech levels do not differ too much from the unweighted speech level because the long-term average spectral shape of speech includes most energy in the frequency range close to 1 kHz where the weighting curves coincide. Since none of these definitions include the psychophysical effect of loudness summation across frequency and the temporal integration performed by our ear, a speech sound at a given sound pressure level does not necessarily produce the same perceived loudness as a 1 kHz signal at this level.
- Loudness in sone. A more exact measure of perceived loudness for sounds that differ in frequency contents is given by a loudness calculation scheme based on the sone scale. For a narrow-band sound (like a sinusoid), the loudness in sone is expressed as

$$N[\text{sone}] = (I/I_0)^{\alpha} , \tag{4.1}$$

where $I_0$ is the reference intensity which is set to 40 dB SPL for a sinusoid at 1 kHz. Since the exponent $\alpha$ amounts to $\approx 0.3$ according to *Stevens* and *Zwicker* [4.1], this yields a compression of sound intensity similar to the nonlinear compression in the human ear. For broadband sounds (such as speech), the same compressive power law as given above has to be applied to each *critical* frequency band (see later) before the partial loudness contributions are summed up across frequencies, which results in the total loudness. The detailed loudness calculation scheme (according to ISO 532b) also accounts for the spread of spectral energy of a narrow-band sound into adjacent frequency bands known from cochlear physiology (Sect. 4.1.2). This *leakage* of spectral energy can also be modeled by a bank of appropriately shaped bandpass filters with a limited upper and lower spectral slope (Fig. 4.2).

To account for the time dependency of loudness perception, a temporal integration process is assumed that sums up all intensity belonging to the same auditory object within a period of approximately 200 ms. This roughly models the effect that sounds with a constant sound level are perceived as being louder if their duration increases up to 200 ms while remaining constant in loudness if the duration increases further. For fluctuating sounds with fluctuating instantaneous loudness estimates, the overall loudness impression is dominated by the respective loudness maxima. This can be well represented by considering the 95 percentile loudness (i. e., the loudness value that is exceeded for only 5% of the time) as the *average* loudness value of a sequence of fluctuating sounds [4.1]. For illustration, Fig. 4.3 displays the relation between waveform, spectrogram, partial loudness pattern and resulting value in dB SPL, dBA and loudness in sone for three different sounds.

### 4.1.2 Pitch

If the frequency of a sinusoid at low frequencies up to 500 Hz is increased, the perceived tone height (or pitch, also denoted as tonality) increases linearly with frequency. At higher frequencies above 1 kHz, however, the perceived pitch increases approximately logarithmically with increasing frequency. The combination of both domains yields the psychophysical mel-scale (Fig. 4.4).

This relation between frequency and subjective frequency perception also represents the mapping of frequencies on the basilar membrane (Sect. 4.2.1 and

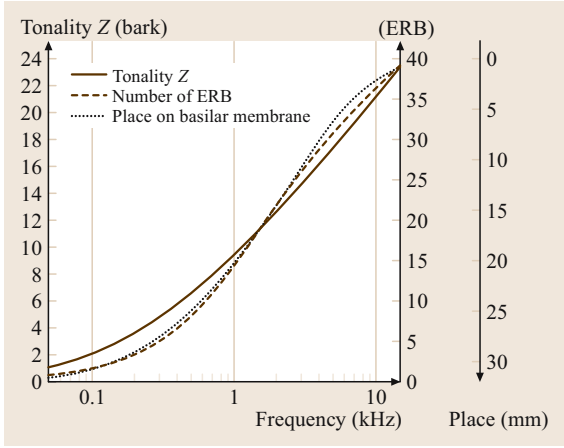



**Fig. 4.4** Tonality in bark and in mel over frequency (one bark equals 100 mel). For comparison the (hypothetical) place of maximum excitation on the basilar membrane and the psychoacoustical frequency scale based on equivalent rectangular bandwidth (ERB) is plotted

Fig. 4.4), where frequencies up to approximately 2 kHz occupy half of the basilar membrane and those between 2 kHz and 20 kHz the remaining half. The slope of this function relates to the just noticeable difference (JND) for frequency. The frequency JND is about 3 Hz for frequencies below 500 Hz and about 0.6% for frequencies above 1000 Hz, which is approximately 3 mel. This value amounts to 1/30 of the frequency-dependent bandwidth of the *critical band* which plays a role both in loudness summation (see above) and in the psychophysical effect of spectral masking. All spectral energy that falls into one *critical band* is summed up and *masks* (or disables) the detection of a sinusoidal tone centered within that critical band as long as its level is below this *masked threshold*. According to *Zwicker* et al. [4.2] the auditory critical bandwidth is expressed in bark after the German physicist *Barkhausen* as a function of frequency $f_0$ (in Hz) as:

$$\begin{aligned} 1\,\text{bark} &= 100\,\text{mel} \\ &\approx 100\,\text{Hz for frequencies below 500 Hz} \\ &\approx 1/5\,f_0 \text{ for frequencies above 500 Hz}\,. \end{aligned} \tag{4.2}$$

The psychophysical frequency scale resulting from integrating the critical bandwidth over frequency is denoted as *bark scale* $Z$ and can be approximated [4.4] by the inverse function of the hyperbolical sinus

$$Z[\text{bark}] = 7 \cdot \text{arcsinh}(f_0/650) \tag{4.3}$$

More-refined measurements of spectral masking performed by *Moore* and *Patterson* [4.5] resulted in the *equivalent rectangular bandwidth* (ERB) as a measure of the psychoacoustical critical bandwidth. It deviates slightly from the bark scale, especially at low frequencies (Fig. 4.4).

It should be noted, however, that the *pitch strength* of a sinusoid decreases steadily as the frequency increases. A much more distinct pitch perception, which is also related to the pitch of musical instruments and the pitch of voiced speech elements, can be perceived for a periodic, broad band sound. For such complex sounds, the term *pitch* should primarily be used to characterize the (perceived) fundamental frequency while *tone height* or *tonality* refers to the psychophysical equivalence of frequency and coincides with pitch only for sinusoids. The perception of pitch results both from temporal cues (i. e., the periodicity information within each critical band) primarily at low frequencies, and from spectral pitch cues, i. e., the regular harmonic structure/line spectrum of a periodic sound primarily dominating at high frequencies. Several theories exist about the perception and relative importance of temporal and spectral pitch cues [4.6]. For the perception of the pitch frequency range of normal speech with fundamental frequencies between approximately 80 Hz and 500 Hz, however, predominantly temporal pitch cues are exploited by our ear. In this range, the pitch JND amounts to approximately 1 Hz, i. e., better resolution occurs for the fundamental frequency of a complex tone than for the audio frequency of a single sinusoid at the fundamental frequency.

### 4.1.3 Temporal Analysis and Modulation Perception

When perceiving complex sounds such as speech that fluctuate in spectral contents across time, the ear can roughly be modeled as a bank of critical-band wide bandpass filters that transform the speech signal into a number of narrow-band time signals at center frequencies that are equally spaced across the bark scale. Hence, the temporal analysis and resolution within each of these frequency channels is of special importance

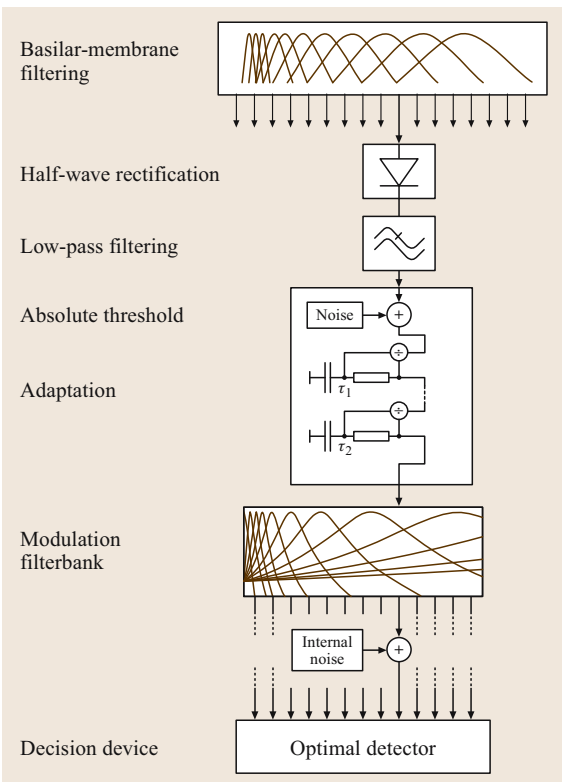



**Fig. 4.5** Model of the *effective* signal processing in the auditory system (after [4.3])



Amazon EX1010 Page 97

for the overall function of our auditory system. For the within-channel analysis, the following phenomena are relevant.

- For center frequencies below ≈ 1000 Hz, the temporal fine structure of the bandpass channel is coded in the auditory nerve and is therefore accessible to the brain. Hence the signal's phase can be exploited during the central processing stages, e.g. for a comparison between different frequency bands to produce a difference in perceived timbre and for a comparison between ears to produce a difference in perceived localization as the phase characteristic is changed. The latter results from extracting the interaural phase difference of a sound signal arriving from a point in space with a certain travel time to either ear (see later).
- For center frequencies above 1 kHz, primarily the envelope of the signal is extracted and analyzed. This makes the ear comparatively phase-deaf above 1 kHz. This envelope extraction is due to the asymmetry between depolarization and hyperpolarization at the synapses between inner hair cells and the auditory nerve as well as due to the temporal integration observed in auditory nerve fibers (Chap. 3). In auditory models this can be modeled to a good approximation by a half-wave rectifier followed by a low-pass filter with a cut-off frequency of ≈ 1 kHz.
- The resulting envelope is subject to a compression and adaptation stage that is required to map the large dynamic range of auditory input signals to the comparatively narrow dynamic range of the nervous system. It is also necessary to set the operation point of the respective further processing stages according to some average value of the current input signal. This compression and adaptation characteristic can either be modeled as a logarithmic compression in combination with the temporal leaky integrator using an *effective* auditory temporal integration window or, alternatively, by a series of nonlinear adaptation loops (Fig. 4.5).

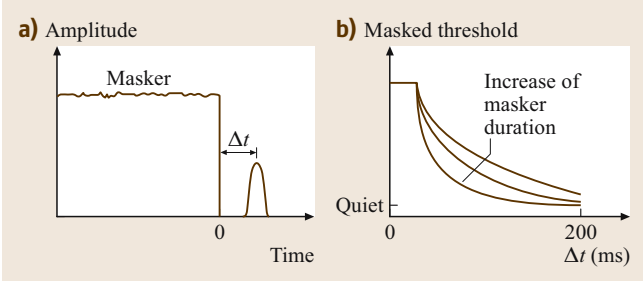



**Fig. 4.6a,b** Schematic plot of a masked threshold **(a)** of a short probe tone in the presence of (or following) a masking noise burst that extends across a variable amount of time **(b)** (simultaneous and forward masking)

Such an adaptation stage produces the *temporal integration* effect already outlined in Sect. 4.1.1, i. e., all temporal energy belonging to the same acoustical object is summed up within a time window with an effective duration of up to 200 ms. In addition, *temporal masking* is due to this integration or adaptation circuit. A short probe signal (of an intensity higher than the threshold intensity in quiet) will become inaudible in the presence of a *masking* signal if the probe signal is presented either before, during, or after the masker. Hence, the masker extends its masking property both back in time (*backward masking*, extending to approximately 5 ms prior to the onset of the masker), simultaneously with the probe signal (*simultaneous masking*, which becomes less efficient if the masker duration is decreased below 200 ms) and subsequent to the masker (*forward masking*, extending up to 200 ms, Fig. 4.6).

Note that forward masking in speech sounds can prevent detection of soft consonant speech components that are preceded by high-energy vocalic parts of speech.

An important further property of temporal analysis in the auditory system is the perception and analysis of the incoming *temporal envelope fluctuations*. While slow amplitude modulations (modulation frequencies below approximately 4 Hz) are primarily perceived as temporal fluctuations, amplitude modulations between approximately 8 Hz and 16 Hz produce a rolling, R-type roughness percept. Modulations between 16 Hz and approximately 80 Hz are perceived as roughness of a sound. Higher modulation frequencies may be perceived as spectral coloration of the input signal without being resolved in the time domain by the auditory system.

The auditory processing of sounds that differ in their composition of modulation frequencies is best described by the modulation spectrum concept which can be modeled by a modulation filter bank (Fig. 4.5). The separation of different modulation frequencies into separate modulation frequency channels (similar to separating the audio frequencies into different center frequency channels in the inner ear) allows the brain to group together sound elements that are generated from the same sound source even if they interfere with sound elements from a different sound source at the same center frequency. Natural objects are usually characterized by a common

Amazon EX1010 Page 98

modulation of the emitted frequency components as a function of time. By grouping those sound components that exhibit the same modulation spectrum across different center frequencies, the brain is able to recombine all the sound components of a certain object that are spread out across different audio frequencies. This property of the auditory system is advantageous in performing a figure-background analysis (such as required for the famous *cocktail-party phenomenon*, i. e., a talker can be understood even in the background of a lively party with several interfering voices).

A way of quantitatively measuring the auditory grouping effect is the so-called co-modulation masking release (CMR) depicted in Fig. 4.7.

A probe tone has to be detected against a narrowband, fluctuating noise at the same frequency (*on-frequency masker*). If the adjacent frequency bands are stimulated with uncorrelated noise samples, the threshold of the tone in the modulated noise is comparatively high. However, if the masking noise in the adjacent bands fluctuates with the same amplitude modulations across time (this is usually done by duplicating the on-frequency masker and shifting its center frequency appropriately), the probe tone becomes better audible and a distinct threshold shift occurs. This is called co-modulation masking release. It is mainly due to within-channel cues, i. e., the modulation minima become more distinct with increasing noise bandwidth and hence allow for a better detection of the continuous probe tone at a certain instant of time. It is also due to some across-channel cues and cognitive processing, i. e., the co-modulated components at different frequencies are grouped to form a single auditory object which is distinct from the probe tone. Since speech is usually characterized by a high degree of co-modulation across different frequencies for a single speaker, the co-modulation masking release helps to detect any irregularity which is not co-modulated with the remainder of the speech signal. Such detectable irregularities may reflect, e.g., any speech pathology, a second, faint acoustical object or even speech processing artefacts. The CMR effect is most prominent for amplitude modulation frequencies between approximately 4 Hz and 50 Hz [4.7], which is a region where most of the modulation spectrum energy of speech is located. Hence this effect is very relevant for speech perception.

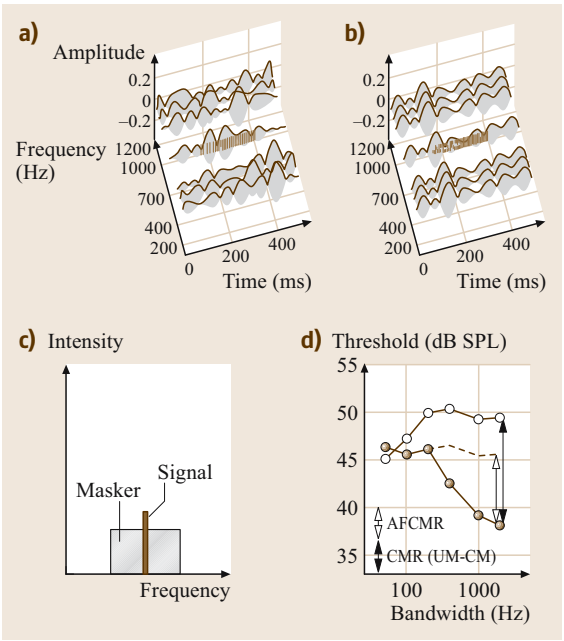



**Fig. 4.7a–d** Schematic plot of co-modulation masking release (after [4.7]). **(a)** Denotes the temporal-spectral difference in the unmodulated condition (flanking bands are modulated in an uncorrelated way) whereas **(b)** shows the pattern for co-modulated sidebands. In the latter case, the detection of a sinusoid at the on-frequency masking band is facilitated. **(c)**, **(d)** Shows typical psychophysical data for a band-widening experiment with unmodulated (*open symbols*) and co-modulated masker (*filled symbols*). A considerable difference in masked threshold for the sinusoidal signal is observed

### 4.1.4 Binaural Hearing

Binaural processing, i. e., the central interaction between signal information entering the right and the left ear contributes significantly to

- suppression of subjectively perceived reverberation in closed rooms,
- localization of sound sources in space,
- suppression of *unwanted* sound sources in real acoustical environments.

To perform these tasks, our brain can utilize

- interaural time (or phase) cues, i. e., the central auditory system extracts the travel time difference between the left and right ear,
- interaural intensity difference, i. e., our brain can utilize the head-shadow effect: sound arriving at the ear pointing towards the sound source in space is not attenuated, while the sound arriving at the opposite ear is attenuated,



Amazon EX1010 Page 99

- spectral changes (coloration) of the sound reaching the inner ear due to interference and scattering effects if the direction of the incoming sound varies (Fig. 4.8).

Normal listeners can localize sound sources with a precision of approximately 1° if sound arrives at the head from the front. This relates to a just noticeable difference (JND) in interaural time difference as small as 10 μs and an interaural level difference JND as low as 1 dB. This remarkable high resolution is due to massive parallel processing at the brain-stem level where the first neural comparison occurs between activation from the right and left ear, respectively.

The binaural performance of our auditory system is extremely challenged in complex acoustical everyday situations characterized by several nonstationary sound sources, reverberation, and a continuous change of the interaural cues due to head movements in space. For perceiving, localizing, and understanding speech in such situations, the following phenomena are relevant:

- Spectral integration of localization cues. Continuous narrow-band signals are hard to localize because their respective interaural time and level difference achieved at the ear level are ambiguous: they can result from any direction within a *cone of confusion*, i. e., a surface that includes all spatial angles centered around the interaural axis that yield the same path difference between right and left ear. For a broadband signal, the comparison across different frequency channels helps to resolve this ambiguity. Also, the onset cues in strongly fluctuating, broadband sounds contain more reliable localization cues than the running cues in the steady-state situation for continuous sounds.
- Precedence effect or the law of the first wavefront. The direct sound (first wavefront) of a sound source hitting the receiver's ears determines the subjective localization percept. Conversely, any subsequent wavefront (that is due to reflections from surrounding structures in a real acoustical environment and hence carries the *wrong* directional information) is not used to create the subjective localization impression. Even though reflections arriving approximately 5–20 ms after the first wavefront are perceivable and their energetic contribution to the total stimulus percept is accessible to the brain, their respective directional information seems to be suppressed. This effect is utilized in some public address loudspeaker systems that deliberately delay the amplified sound in order for the small, unamplified direct sound to reach the listener prior to the amplified sound with the *wrong* directional information.

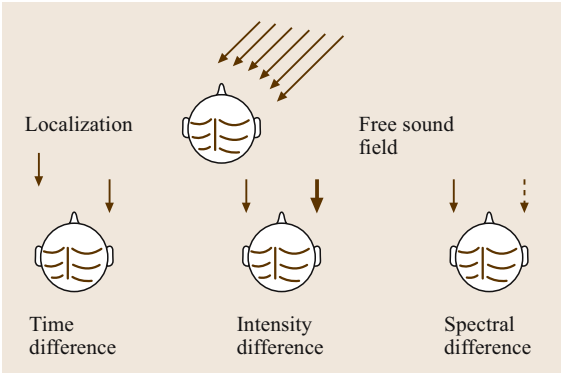



**Fig. 4.8** Schematics of interaural cues due to interference and scattering effects that can be utilized by the auditory system

### 4.1.5 Binaural Noise Suppression

The localization mechanisms described above are not only capable of separating the perceived localization of several simultaneously active acoustical objects. They are also a prerequisite for binaural noise suppression, i. e., an enhancement of the *desired* signal and a suppression of *undesired* parts of the input signals that originate from a different spatial direction. This enhancement is also denoted as binaural release from masking. It can be demonstrated by a tone-in-noise detection experiment where in the reference condition tone and noise are the same at both ears (i. e., exhibit the phase difference 0). The detection threshold can be compared to the threshold using the same noise, but inverting the signal on one side (i. e., a phase difference of $\pi$ for the signal), which yields a higher detectability. This difference in threshold is denoted as binaural masking level difference and amounts up to 20 dB for short probe tones at frequencies below 1000 Hz.

For speech signals, the binaural unmasking can be measured by comparing the speech reception threshold (i. e., the signal-to-noise ratio required to understand 50% of the presented speech material, see later) for different spatial arrangements of target speech sound and interfering noise: in the reference condition, speech and noise are presented directly in front of the subject, while in the test condition speech comes from the front, but the interfering sound source from the side. The gain in speech reception threshold is called the intelligibil-

Amazon EX1010 Page 100

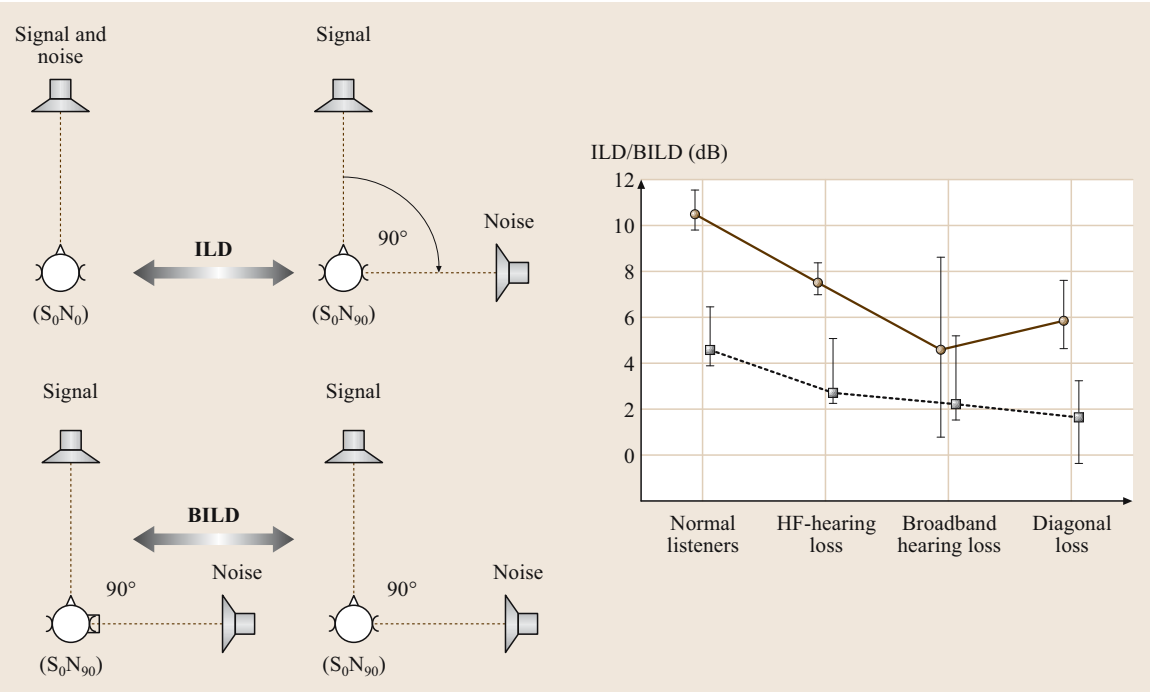



**Fig. 4.9** Intelligibility level difference (ILD, *filled circles*) and binaural intelligibility level difference (BILD, *filled squares*) averaged across a group of normal and impaired listeners that differ in the shape of their respective audiogram (high-frequency hearing loss abbreviated as HF-hearing loss). The difference in speech reception threshold across both situations plotted on the left-hand side is plotted as average value and intersubject standard deviation

ity level difference. (The abbreviation ILD is used for this difference, but is also used for interaural level difference.) Intelligibility level difference is due to a monaural effect (i. e., improved signal-to-noise ratio at the ear opposite to the interfering sound source) and a binaural effect. To separate this latter effect, another threshold in the same spatial situation is used where the *worse* ear is plugged and the speech reception threshold is obtained using only the *better* ear, i. e., the ear with the better signal-to-noise ratio. The difference in speech reception threshold (SRT= between the latter two situations (i. e., the difference due to *adding* the *worse* ear) is a purely binaural effect, called the binaural intelligibility level difference (BILD). Figure 4.9 gives an example of the ILD and BILD at an incidence angle for the interfering noise of 90° in an anechoic condition for different groups of listeners that vary in their hearing loss.

A basic model which describes binaural unmasking phenomena quite well for speech signals in complex acoustical environments is a multichannel equalization and cancelation (EC) model such as that depicted in Fig. 4.10 [4.8].

Within each frequency band, an *equalization and cancelation* mechanism [4.9] is used that first delays

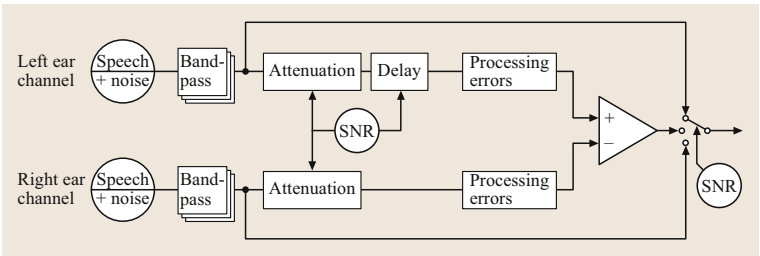



**Fig. 4.10** Sketch of a multichannel binaural noise canceling model describing the binaural release from masking for speech in complex environments (after [4.8])

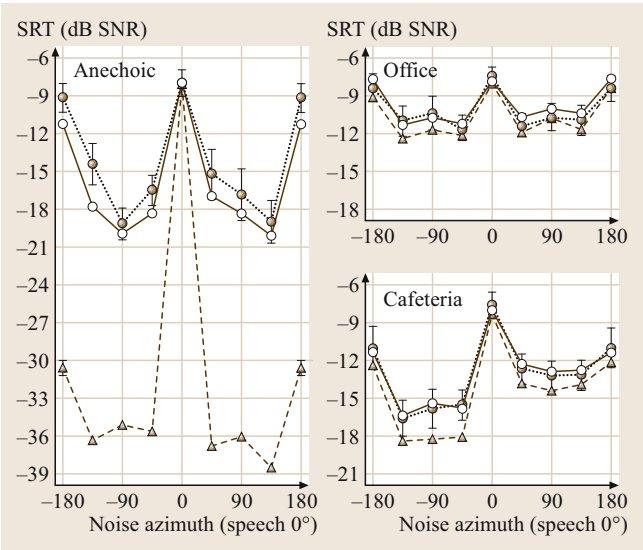



**Fig. 4.11** SRT data (*filled symbols*) and predictions for three different acoustical conditions and normal listeners. The *triangles* denote model predictions without introducing appropriate processing errors, whereas the open symbols denote predictions employing internal processing errors, that have been taken from average values in other psychoacoustical tasks (after [4.8])

and amplifies one or both input channels to yield an approximate match (*equalization*) of the composite input signal within each frequency band. In a second, the *cancelation* stage, the signals from both respective sides of the head are (imperfectly) subtracted from each other. Hence, if the masker (after the *equalization* step) is approximately the same in both ears, the cancelation step will eliminate the masker with the exception of some remaining error signal. Conversely, the *desired* signal, which differs in interaural phase and/or intensity relation from the masker, should stay nearly unchanged, yielding an improvement in signal-to-noise ratio. Using an appropriate numerical optimization strategy to fit the respective equalization parameters across frequency, the model depicted in Fig. 4.11 can predict human performance quite well even under acoustically *difficult* situations, such as, e.g., several interfering talkers within a reverberant environment. Note that this model effectively corresponds to an adaptive spatial beam former, i. e., a frequency-dependent optimum linear combination of the two sensor inputs to both ears that yields a directivity optimized to improve the signal-to-noise ratio for a given target direction and interfering background noise. If the model output is used to predict speech intelligibility with an appropriate (monaural) speech intelligibility prediction method [such as, e.g., the speech intelligibility index (SII), see later], the binaural advantage for speech intelligibility in rooms can be predicted quite well (Fig. 4.11 from [4.8]).

Note that in each frequency band only one EC circuit is employed in the model. This reflects the empirical evidence that the brain is only able to cancel out one direction for each frequency band at each instant of time. Hence, the processing strategy adopted will use appropriate compromises for any given real situation.

## 4.2 Acoustical Information Required for Speech Perception

### 4.2.1 Speech Intelligibility and Speech Reception Threshold (SRT)

Speech intelligibility (SI) is important for various fields of research, engineering, and diagnostics for quantifying very different phenomena such as the quality of recordings, communication and playback devices, the reverberation of auditoria, characteristics of hearing impairment, benefit using hearing aids, or combinations of these topics. The most useful way to define SI is: *speech intelligibility SI is the proportion of speech items (e.g., syllables, words, or sentences) correctly repeated by (a) listener(s) for a given speech intelligibility test*. This operative definition makes SI directly and quantitatively measurable.

The intelligibility function (Fig. 4.12) describes the listener's speech intelligibility SI as a function of speech level $L$ which may either refer to the sound pressure level (measured in dB) of the speech signal or to the speech-to-noise ratio (SNR) (measured in dB), if the test is performed with interfering noise.

In most cases it is possible to fit the logistic function SI ($L$) to the empirical data

$$\mathrm{SI}(L) = \frac{1}{A}\left(1 + \mathrm{SI}_{\max}\frac{A-1}{1+\exp\left(-\frac{L-L_{\mathrm{mid}}}{s}\right)}\right), \quad (4.4)$$

with $L_{\mathrm{mid}}$: speech level of the midpoint of the intelligibility function; $s$: slope parameter, the slope at $L_{\mathrm{mid}}$ is

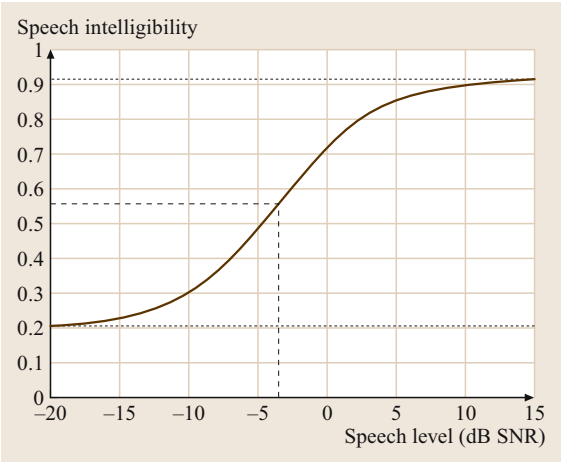



**Fig. 4.12** Typical example of SI function (*solid line*) for word intelligibility test (closed response format with five response alternatives). The *dashed line* denotes $L_{mid}$. The *dotted lines* denote the lower limit $(1/A)$ and the asymptotic maximum $SI_{asymp}$ of the SI function. Parameters: $L_{mid} = -3.5$ dB SNR, $SI_{max} = 0.9 (SI_{asymp} = 0.92)$, $A = 5$, slope $= 0.05$/dB ($s = 3.6$ dB)

given by $\frac{SI_{max}(A-1)}{4As}$; $SI_{max}$: parameter for maximum intelligibility which can be smaller than 1 in some cases (e.g., distorted speech signals or listeners with hearing impairment). The asymptotic maximum of SI is given by $SI_{max} + (1 - SI_{max})/A$. $A$ is the number of response alternatives (e.g., $A = 10$ when the listener should respond in a closed response format for instance using digits between '0' and '9'). In SI tests with *open response format*, like word tests without limiting the number of response alternatives, $A$ is assumed to be infinite, that means

$$SI = SI_{max} \frac{1}{1 + \exp\left(-\frac{L - L_{mid}}{s}\right)} \quad \text{and}$$

$$\text{slope} = \frac{SI_{max}}{4s} \,. \tag{4.5}$$

The primary interest of many applications is the speech reception threshold (SRT) which denotes the speech level (measured in dB), which belongs to a given intelligibility (e.g., SI = 0.5 or 0.7).

The accuracy of SI measurements is given by the binomial distribution. Consequently, the standard error SE(SI) of an SI estimate based on $n$ items (e.g., words) is given by

$$SE(SI) = \sqrt{\frac{SI(1 - SI)}{n}} \,. \tag{4.6}$$

A further increase of this standard error is caused by the fact that SI tests consist of several items (e.g., 50 words) which unavoidably differ in SI. Therefore, SI tests should be constructed in a way that the SI of all items is as homogeneous as possible.

To a first approximation, the standard error of the SRT is equal to $SE(SI_{SRT})$ (the standard error of the SI estimate at the SRT) divided by the slope of the intelligibility function at the SRT. Thus

$$SE(SRT) = \frac{SE(SI_{SRT})}{\text{slope}_{SRT}} \,. \tag{4.7}$$

## 4.2.2 Measurement Methods

### Speech Materials

A speech material (i. e., a set of speech items like words or sentences) is suitable for SI tests when certain requirements are fulfilled: the different speech items have to be homogeneous in SI to yield high measurement accuracy and reproducibility in a limited measuring time, and the distribution of phonemes should be representative of the language being studied. Only speech materials that have been optimized properly by a large number of evaluation measurements can fulfill these requirements.

A large number of SI tests using different materials are available for different languages. An overview of American SI tests can be found in *Penrod* [4.10]. There are different formats, i. e., nonsense syllables, single words, and sentences. Sentences best represent a realistic communication situation. Nonsense syllables and words allow assessing of confusion matrices and analyzing transmission of information. Furthermore, the intelligibility functions of most sentence tests [4.11–16] show slopes between 0.15 and 0.25 per dB, which are considerably steeper than the values obtained with nonsense syllables or single-word tests.

Since the standard deviation of SRT estimates is inversely proportional to the slope of the intelligibility function (see Sect. 4.2.1), these sentence tests are better suited for efficient and reliable SRT measurements than single-word tests.

### Presentation Modes

Signals can be presented either via loudspeakers (free field condition) or via headphones. The free field condition is more natural. Drawbacks are a larger experimental effort and difficulties in calibration. Especially in spatial speech/noise situations (see Sect. 4.2.3), small movements of the listener's head may influence the result of the SI measurement.



Amazon EX1010 Page 103

The advantages of presentation via headphones are: very good reproducibility for each individual listener, smaller experimental effort, and spatial speech/noise conditions can easily be realized using virtual acoustics. Drawbacks are the individual calibration is complicated because headphones may produce different sound pressures in different ears. Measurements with hearing aids are not possible.

Adaptive procedures can be used to concentrate presentation levels near the SRT, which yields highest efficiency for SRT estimates. In sentence tests, each word can be scored independently, which allows one to design adaptive procedures which converge more efficiently than adaptive procedures usually used in psychoacoustics [4.17].



### 4.2.3 Factors Influencing Speech Intelligibility

#### Measuring Method

The various speech materials mentioned above generate different results. Therefore, only standardized speech materials or speech materials with well known reference intelligibility functions should be used.

#### Noise and Room Acoustics

Noise and reverberation reduce SI. Therefore, if SI in silence is to be measured, environmental noise and reverberation have to be minimized (e.g., using sound insulated cabins and damping headphones). On the other hand, SI measurements can be used to investigate the influence of different noises and room acoustics on SI, which is important for the so called *cocktail-party phenomenon* (see later).

#### Cocktail-Party Phenomenon

The human auditory system has very impressive abilities in understanding a target talker even if maskers, i. e., competitive sound sources like different talkers, are present at the same time. An interesting review of research on this so-called *cocktail-party phenomenon* can be found in *Bronkhorst* [4.18]. The SI in these multi-talker conditions is influenced by many masker properties such as sound pressure level, frequency spectrum, amplitude modulations, spatial direction, and the number of maskers. The spatial configuration of target speaker and masker plays a very important role. Binaural hearing (hearing with both ears) produces a very effective release from masking (improvement of the SRT) of up to 12 dB compared to monaural hearing (hearing with one ear) [4.18].

#### Hearing Impairment

An introduction to SI in clinical audiology can be found in *Penrod* [4.10]. Hearing impairment can lead to an increase of the SRT, a decrease of the maximum reachable intelligibility $SI_{asymp}$ and a flatter slope of the intelligibility function. The most difficult situations for hearing impaired listeners are noisy environments with many interfering sound sources (*cocktail-party situation*). Therefore, SI tests in noise are important diagnostic tools for assessing the daily-life consequences of a hearing impairment and the benefit of a hearing aid. SI plays a very important role for the research on and the fitting of hearing aids.

### 4.2.4 Prediction Methods

#### Articulation Index (AI), Speech Intelligibility Index (SII), and Speech Transmission Index (STI)

The most common methods for the prediction of speech intelligibility are the articulation index (AI) [4.19–21] which was renamed the speech intelligibility index (SII) [4.22], and the speech transmission index (STI) [4.23, 24] [Table 4.1]. The strength of these models is the large amount of empirical knowledge they are based on. All of these models assume that speech is coded by several frequency channels that carry independent information. This can be expressed by

$$\mathrm{AI} = \sum_i \mathrm{AI}_i \,, \tag{4.8}$$

with AI denoting the cumulative articulation index of all channels and $\mathrm{AI}_i$ denoting the articulation index of the single channels (including a weighting of the respective channel).

AI and SII are derived from the speech signal by calculating the signal to noise ratio SNR in the different frequency channels:

$$\mathrm{AI} = \sum_i \frac{W_i(\mathrm{SNR}_i + 15)}{30} \,, \tag{4.9}$$

with $W_i$ denoting a frequency channel weighting factor and $\mathrm{SNR}_i$ denoting the signal-to-noise ratio in channel $i$. $W_i$ depends on the speech material used and takes into account that high frequencies are more important for the recognition of consonants than for the recognition of meaningful sentences. The main differences between the different versions of AI and SII are the way they include nonlinearities like distortion, masking, and broadening of frequency bands.

Amazon EX1010 Page 104

The speech transmission index (STI) uses the modulation transfer function instead of the SNR and is especially successful for predicting SI in auditoria and rooms, because it explicitly takes into account the flattening of the information-carrying speech envelopes due to reverberation.

The transformation of AI, SII, or STI to speech intelligibility requires a nonlinear transformation that has to be fitted to empirical data. The transformation depends on the kind of speech material used and is usually steeper at its steepest point for materials with context (e.g., sentences) compared to single words.

### Statistical Methods

The assumption of independent information in different frequency channels does not hold in all situations because synergetic as well as redundant interactions between different channels occur. The speech recognition sensitivity model [4.25, 26] takes these interactions into account using statistical decision theory in order to model the linguistic entropy of speech.

### Functional Method

These methods are based on relatively rough parameterizations of speech (i. e., long-term frequency spectrum and sometimes modulation transfer function). The method (Table 4.1) proposed by *Holube* and *Kollmeier* [4.27], however, is based on physiological and psychoacoustical data and is a combination of a functional model of the human auditory system (Sect. 4.1) and a simple automatic speech recognition system (Sect. 4.3). A drawback of this approach is that there is still a large gap between recognition rates of humans and automatic speech recognition systems (for a review see [4.28], Sect. 4.3).



**Table 4.1** Examples of methods for the prediction of speech intelligibility (SI)

| Method | Signal parameters | Comments |
|---|---|---|
| Articulation index, AI [4.19] | Levels and frequency spectra of speech and noise, kind of speech material | Macroscopic model that describes the influence of the frequency content of speech on intelligibility |
| Articulation index, AI [4.20] | Levels and frequency spectra of speech and noise, kind of speech material | More complex than French and Steinberg version, describes more nonlinear effects, seldom used |
| Articulation index, AI [4.21] | Levels and frequency spectra of speech and noise | Simplified version based on [4.19], not in use anymore |
| Speech intelligibility index, SII [4.22] | Levels and frequency spectra of speech and noise, kind of speech material, hearing loss | Revision of ANSI S3.5-1969, includes spread of masking, standard speech spectra, relative importance of frequency bands |
| Speech transmission index, STI [4.24] | Modulation transfer function | Predicts the change of intelligibility caused by a speech transmission system (e.g., an auditorium) based on the modulation transfer function of the system |
| Speech recognition sensitivity model, SRS [4.25, 26] | Levels and frequency spectra of speech and noise, number of response alternatives | Alternative to SII, handles frequency band interactions and is better suited for unsteady frequency spectra |
| *Holube* and *Kollmeier* [4.27] | Speech and noise signals, hearing loss | Microscopic modeling of signal processing of auditory system combined with simple automatic speech recognition |

## 4.3 Speech Feature Perception

The information-theoretic approach to describing speech perception assumes that human speech recognition is based on the combined, parallel recognition of several acoustical cues that are characteristic for certain speech elements. While a *phoneme* represents the smallest unit of speech information, its acoustic realization (denoted as *phone*) can be quite variable in its acoustical properties. Such a phone is produced in order to deliver a number of acoustical speech cues to the listener who should be able to deduce from it the underlying phoneme. Each speech cue represents one feature value of more- or less-complex speech features like *voicing*, *frication*, or *duration*, that are linked to phonetics and to perception. These speech feature values are decoded by the listener independently of each other and are used for recognizing the underlying speech element (such as, e.g., the represented phoneme). Speech perception can therefore be interpreted as reception of certain values of several speech features in parallel and in discrete time steps.

Each phoneme is characterized by a unique combination of the underlying speech feature values. The articulation of words and sentences produces (in the sense of information theory) a discrete stream of information via a number of simultaneously active channels (Fig. 4.13).

The spoken realization of a given phoneme causes a certain speech feature to assume one out of several different possible values. For example, the speech feature *voicing* can assume the value one (i. e., voiced sound) or the value zero (unvoiced speech sound). Each of these features is transmitted via its own, specific transmission channel to the speech recognition system of the listener. The *channel* consists of the acoustical transmission channel to the listener's ear and the subsequent decoding of the signal in the central auditory system of the receiver (which can be hampered by a hearing impairment or a speech pathology). The listener recognizes the actually assumed values of certain speech features and combines these features to yield the recognized phoneme.

If $p(i)$ gives the probability (or relative frequency) that a specific speech feature assumes the value $i$ and $p'(j)$ gives the probability (or relative frequency, respectively) that the receiver receives the feature value $j$, and $p(i, j)$ gives the joint probability that the value $j$ is recognized if the value $i$ is transmitted, then the so-called transinformation $T$ is defined as

$$T = -\sum_{i=1}^{N}\sum_{j=1}^{N} p(i, j)\log_2\left(\frac{p(i)p'(j)}{p(i, j)}\right) . \qquad (4.10)$$

The transinformation $T$ assumes its maximal value for perfect transmission of the input values to the output values, i. e., if $p(i, j)$ takes the diagonal form or any permutation thereof. $T$ equals 0 if the distribution of received feature values is independent of the distribution of input feature values, i. e., if $p(i, j) = p(i)p'(j)$. The maximum value of $T$ for perfect transmission (i. e., $p(i, j) = p(i) = p'(j)$) equals the amount of information (in bits) included in the distribution of input feature values $H$, i. e.,

$$H = \sum_{i=1}^{N} p(i)H(i) = -\sum_{i=1}^{N} p(i)\log_2[p(i)] . \qquad (4.11)$$

In order to normalize $T$ to give values between 0 and 1, the so-called transinformation index (TI) is often

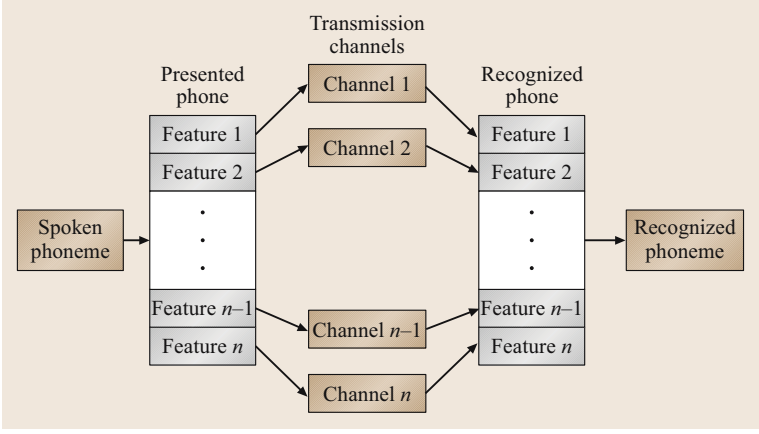



**Fig. 4.13** Schematic representation of speech recognition using speech features. Each speech sound is characterized by a combination of speech features that are modeled to be transmitted independently of each other by specialized (noisy) transmission channels

Amazon EX1010 Page 106

used, i. e.,

$$\mathrm{TI} = T/H \,. \tag{4.12}$$

For speech perception experiments, the distribution $p(i, j)$ can be approximated by a confusion matrix, i. e., a matrix denoting the frequency of a recognized speech element $j$ for all different presented speech elements $i$. This confusion matrix can be condensed to a confusion matrix of each specific speech feature if for each speech element the corresponding value of the respective feature is assigned. For example, a $20 \times 20$ confusion matrix of consonants can be reduced to $2 \times 2$ matrix of the feature *voicing* if for each consonant the feature value *voiced* or *unvoiced* is given. The transinformation analysis of this feature-specific confusion matrix therefore allows extracting the transmission of all respective speech features separately and hence can be used to characterize a certain speech information transmission channel. Note however, that such an analysis requires a sufficiently high number of entries in the confusion matrix to appropriately sample $p(i, j)$, which requires a large data set. Also, it is not easy and straightforward to assign appropriate speech features to all of the presented and recognized speech sounds that will allow an adequate analysis of the acoustical deficiencies in the transmission process. From the multitude of different features and feature sets that have been used in the literature to describe both human speech production and speech perception, only a very limited set of the most prominent features can be discussed here ([4.29], for a more-complete coverage).

### 4.3.1 Formant Features

Vowels are primarily discriminated by their formant structure, i. e., the resonance frequencies of the vocal tract when shaping the vowels. For stationary vowels, the relation between the first and second formant frequencies ($F_1$ and $F_2$), respectively, and the perceived vowel is quite well established (Fig. 4.14 and the introduction).

The *classical* theory of vowel perception has the advantage of being linked to the physical process of speech production (i. e., the formant frequencies are closely linked to the position and elevation of the tongue in the vocal tract). However, modern theories of speech perception no longer assume that vowel identification is based solely on the position of the formant frequencies, for the following reasons.

- For short vowels and for vocalic segments of running speech the perceived vowel often differs from the expected spectral shapes, which are based on long, isolated vowels. This indicates that the mapping between formants and perceived vowels shows different boundaries and different regions of overlap depending on the respective speech context.
- The spectral shape of speech in real-life environments varies considerably due to large spectral variations of the room transfer function, and due to the presence of reverberation and background noise. The pure detection and identification of spectral peaks would yield a much less robust perception of vowels than can actually be observed in human listeners.
- Vowel discrimination and speech understanding is even possible under extreme spectral manipulations, such as, e.g., flat spectrum speech [4.30] and slit-filtered speech (i. e., listening to speech through very few spectral slits, [4.31]).

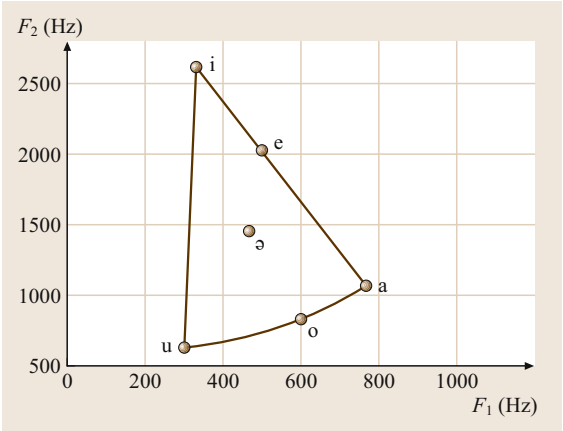



**Fig. 4.14** Schematic plot of the perceived vowels as a function of $F_1$ and $F_2$ in the vowel triangle (sometimes plotted as a quadrangle) for stationary vowels. Note that the vowel boundaries overlap and that for short vowels and for segments of vowels in real speech these boundaries can vary significantly

These findings indicate that speech perception is at least partially based on temporal cues rather than purely on the detection of spectral peaks or formants. Modern speech perception theories therefore assume that our brain monitors the temporal intensity pattern in each frequency band characterized by a *critical band* filter (Sect. 4.1). By comparing these temporal patterns across a few center frequencies that are not too closely spaced, a reliable estimate of the presented vowel is possible. Such principles are both implemented in state-of-the-art perception models (Sect. 4.1) and in preprocessing/feature extraction strategies for automatic speech recognition systems (Sect. 4.2.3 and the chapters in part E).



Amazon EX1010 Page 107

### 4.3.2 Phonetic and Distinctive Feature Sets

The *classical* theory of consonant perception assumes that several phonetically and acoustically defined features are used by the auditory system to decode the underlying, presented consonant. A distinctive feature set combines all binary features that characterize all available consonants in a unique way [4.32]. Both articulatory and (to a lesser degree) acoustical features can be employed to construct such a feature set (see Table 4.2 for an example of a feature set and resulting confusion matrices).

These feature sets where extended and used by *Miller* and *Nicely* [4.33], by *Wang* and *Bilger* [4.34] and subsequently by a large number of researchers to characterize the listeners' ability to discriminate across consonants using, e.g., transinformation analysis (Sect. 4.2.4). Using this approach, the amount of information carried by the specific feature that the receiving side was able to use can be characterized quite well. For example, the confusion matrix listed in Table 4.2 yields a total information transmission index of 0.53 with the features *voicing* assumed to be 0.53 and place 0.46.

However, these phonetic features show only a very weak link to the auditory features actually used by human listeners. From the view point of modern auditory models that assume multichannel temporal energy recording and analysis, most of the phonetic features listed above can be regarded as special prototypes of temporal-spectral patterns that are used by our cognitive system to perform a pattern match between actually presented speech and a stored speech reference database in our brain. Hence, they represent some complex combination of basic auditory perception features that might be defined psychoacoustically or physiologically rather than phonetically.

**Table 4.2** Example for a consonant feature set and the construction of confusion matrices for speech-in-noise recognition data with normal-hearing listeners. *Top right panel*: phonetic feature values for eleven consonants. Voicing is a binary feature (with feature values 0 and 1), while manner and place are ternary features. *Middle*: matrix of confusion for consonants, obtained from human listening tests, where noisy speech was presented at an SNR of -10 dB. Matrix element $(i, j)$ denotes how often the consonant in row $i$ was confused with the consonant in column $j$. *Bottom panels*: confusion matrices for the phonetic features place and voicing, derived from the matrix of confusion for consonants

| | Voicing | Manner | Place |
|---|---|---|---|
| p | 0 | 0 | 0 |
| t | 0 | 0 | 1 |
| k | 0 | 0 | 2 |
| b | 1 | 0 | 0 |
| d | 1 | 0 | 1 |
| g | 1 | 0 | 2 |
| s | 0 | 1 | 1 |
| f | 0 | 1 | 0 |
| v | 1 | 1 | 0 |
| n | 1 | 2 | 2 |
| m | 1 | 2 | 1 |

| | p | t | k | b | d | g | s | f | v | n | m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| p | 379 | 20 | 131 | 45 | 7 | 31 | | 49 | 46 | 4 | 5 |
| t | 3 | 658 | 16 | | 33 | | | 1 | 1 | | |
| k | 42 | 14 | 484 | 10 | 8 | 117 | 1 | 16 | 12 | 6 | |
| b | 58 | 4 | 51 | 260 | 35 | 88 | | 18 | 143 | 16 | 25 |
| d | 5 | 28 | 7 | 21 | 424 | 93 | 1 | 3 | 19 | 49 | 6 |
| g | 11 | 5 | 44 | 43 | 27 | 449 | | 9 | 73 | 18 | 7 |
| s | | 2 | | | | | 702 | 3 | | | |
| f | 23 | | 3 | 4 | | | 88 | 556 | 38 | | |
| v | 19 | 7 | 16 | 78 | 22 | 43 | 7 | 51 | 398 | 9 | 30 |
| n | 1 | 5 | 3 | 13 | 51 | 12 | | 2 | 23 | 364 | 78 |
| m | 7 | 1 | 4 | 43 | 20 | 25 | | 8 | 62 | 95 | 346 |

| | Anterior | Medail | Posterior |
|---|---|---|---|
| Anterior | 2691 | 335 | 392 |
| Medail | 179 | 2317 | 131 |
| Posterior | 223 | 79 | 1094 |

| | Voiced | Unvoiced |
|---|---|---|
| Voiced | 3508 | 375 |
| Unvoiced | 367 | 3191 |

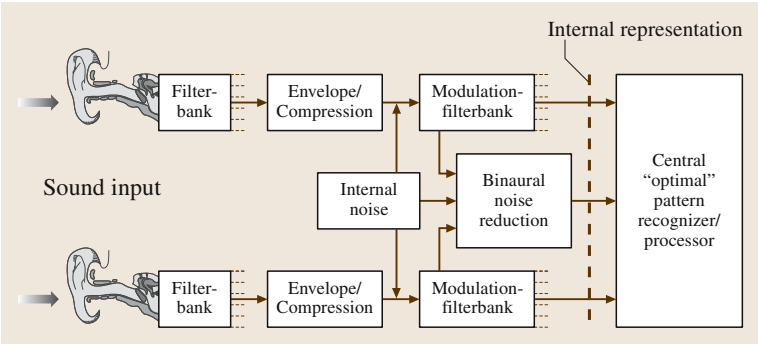



**Fig. 4.15** Schematic diagram of a model of the *effective* auditory processing using a front end to transform the incoming speech signal into and internal representation and a subsequent *back end*, ideal recognition stage which is only limited by the internal noise

### 4.3.3 Internal Representation Approach and Higher-Order Temporal-Spectral Features

The *internal representation* approach of modeling speech reception assumes that the speech signal is transformed by our auditory system with some nonlinear, parallel processing operations into an *internal representation*. This representation is used as the input for a central, cognitive recognition unit which can be assumed to operate as an *ideal observer*, i. e., it performs a pattern match between the incoming internal representation and the multitude of stored internal representations. The accuracy of this recognition process is limited by the *external variability* of the speech items to be recognized, i. e., by their deviation from any of the stored internal templates. It is also limited by the *internal noise* that blurs the received *internal representation* due to neural noise and other physiological and psychological factors. The amount of internal noise can be estimated quite well from psychoacoustical experiments.

Such an internal representation model puts most of the peculiarities and limitations of the speech recognition process into the nonlinear, destructive transformation process from the acoustical speech waveform into its internal representation, assuming that all transformation steps are due to physiological processes that can be characterized completely physiologically or by psychoacoustical means (Fig. 4.15).

Several concepts and models to describe such an internal representation have been developed so far. Some of the basic ideas are as follows.

1. Auditory spectrogram: The basic internal representation assumes that the speech sound is separated into a number of frequency bands (distributed evenly across a psychoacoustically based frequency scale like the bark or ERB scale) and that the compressed frequency-channel-specific intensity is represented

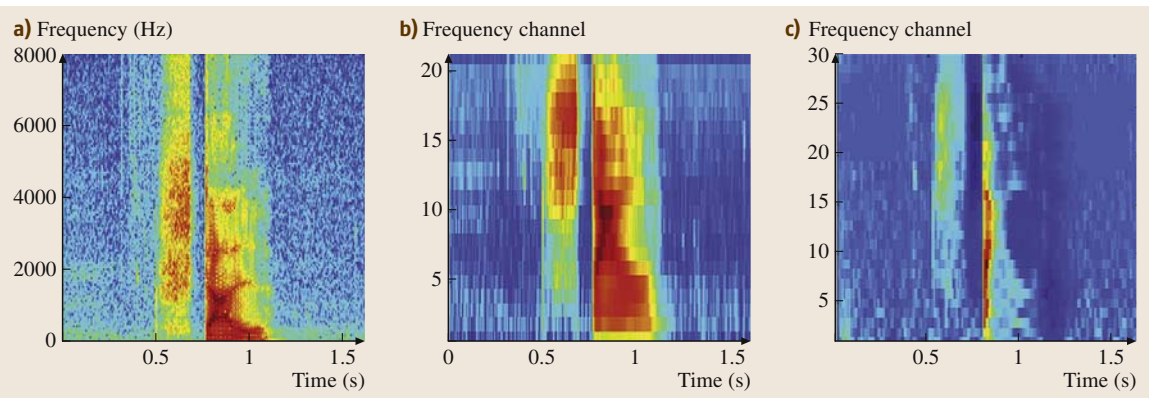



**Fig. 4.16a–c** Auditory spectrogram representation of the German word *Stall*. It can be represented by a spectrogram **(a)**, a bark spectrogram on a log-loudness scale **(b)** or as a contrast-enhanced version using nonlinear feedback loops (after [4.3, 35]) **(c)**

over time. The compression can either be a logarithmic compression or a loudness-derived power law compression that is also required to represent human intensity resolution and loudness mapping. The temporal representation can also include some temporal contrast enhancement and sluggishness in order to represent forward and backward masking and temporal integration (Sect. 4.1). An example of such a representation is given in Fig. 4.16.

Note that speech intelligibility in noise can be modeled quite well with such an approach [4.27]. In addition, such a transformation into the *internal representation* can be implemented as a robust front end for automatic speech recognition (e.g., [4.35]). Finally, it can be used to predict any perceived deviations of the (coded) speech from the original speech [4.36].

2. Modulation spectrogram: one important property of the internal representation is the temporal analysis within each audio frequency band using the modulation filter bank concept. Temporal envelope fluctuations in each audio frequency channel are spectrally analyzed to yield the modulation spectrum in each frequency band, using either a fixed set of modulation filters (modulation filter bank) or a complete spectral analysis (modulation spectrum). This representation yields the so-called amplitude modulation spectrogram for each instant of time, i. e., a two-dimensional representation of modulation frequency across center audio frequencies (Fig. 4.17).

The physiological motivation for this analysis is the finding of amplitude modulation sensitivity in the auditory brain:adjacent cells are tuned to different modulation frequencies. Their arrangement seems to yield a perpendicular representation of modulation frequencies across center frequencies [4.37]. In addition, psychoacoustical findings of modulation sensitivity can best be described by a set of modulation filter banks [4.3]. The advantage of the modulation spectrogram is that the additional dimension of modulation frequency allows separation of acoustical objects that occupy the same center frequency channel, but are modulated at different rates (considering either the syllabic rate at low modulation frequencies or temporal pitch at higher modulation frequencies). Such a more-refined model of internal representation has been used to predict psychoacoustical effects [4.3] and was also used in automatic speech recognition [4.38].

3. Temporal/spectral ripple or Gabor feature approach: A generalization of the modulation frequency feature detectors in the temporal domain outlined above also considers the spectral analysis of ripples in

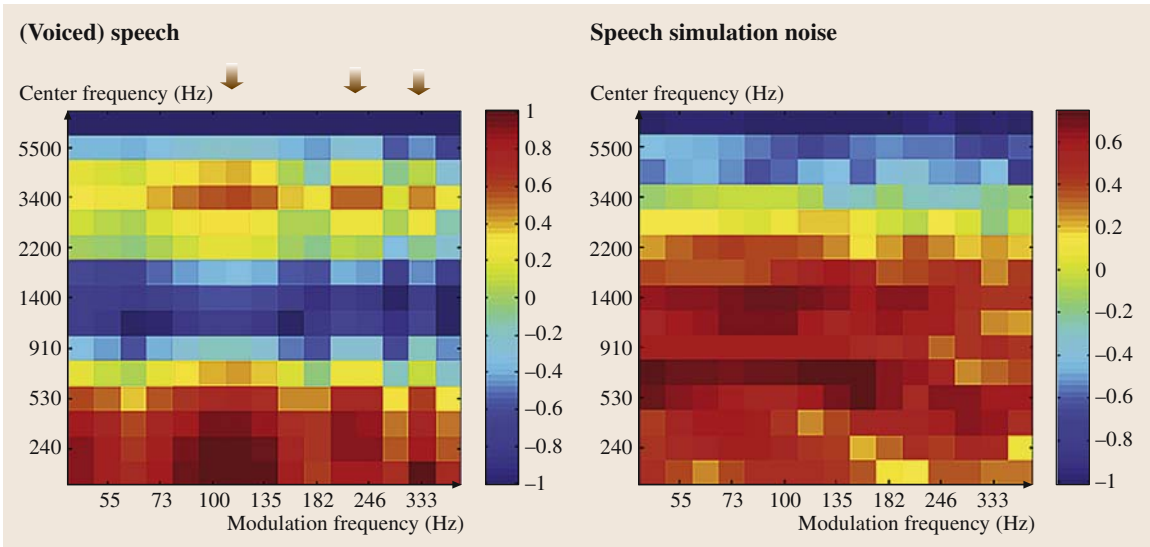



**Fig. 4.17** Amplitude modulation spectrogram of a vowel *I* (fundamental frequency approx. 110 Hz) in comparison to a modulation spectrogram of speech-simulating noise. The modulation spectrum in each audio frequency band is displayed as color (or greyscale, respectively) in the two-dimensional plane given by audio center frequency versus modulation frequency

Amazon EX1010 Page 110

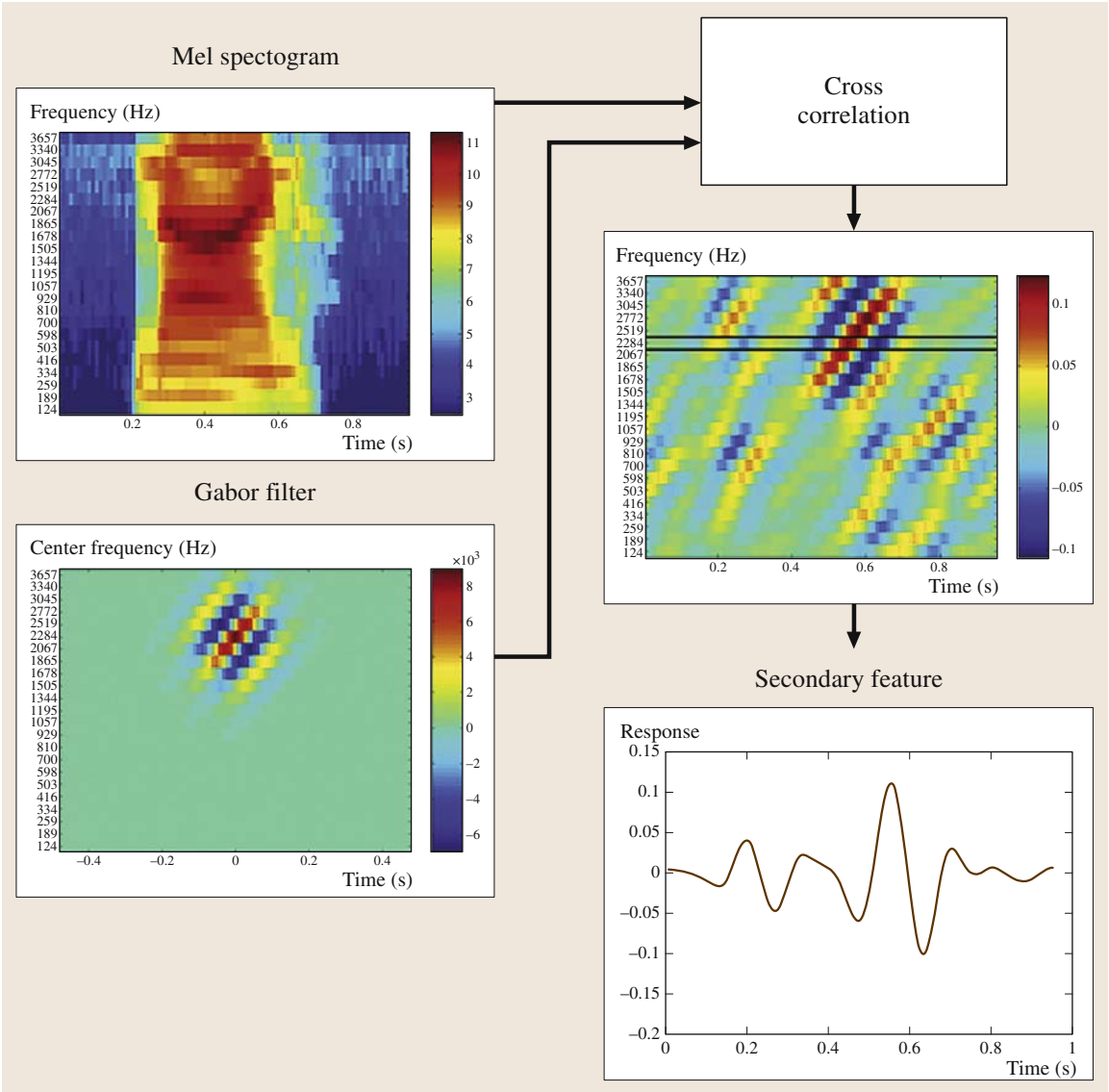



**Fig. 4.18** Sample representation of a Gabor feature that detects a certain speech feature. The two-dimensional Gabor feature (*lower left panel*) that extends both in the time and frequency domain is cross-correlated with the mel spectrogram (*upper-left panel*) to yield the temporal and spatial position of a best match (*middle panel*). In each audio frequency band, the time-dependent output of the cross correlation is used as the input feature to an automatic speech recognizer [4.38]

the frequency domain as well as a ripple frequency analysis for combined temporal and spectral modulations. Such a temporal-spectral ripple analysis is motivated by physiological findings of the auditory receptive fields in ferrets [4.39] as well as psychoacoustical findings by *Kaernbach* [4.40] who demonstrated a sensitivity towards combinations of spectral variations and temporal variations. An elegant way to formalize the sensitivity to joint temporal and spectral energy variations is the Gabor feature concept [4.38] that considers features with a limited spectro-temporal extent tuned to a certain combina-



Amazon EX1010 Page 111

**Table 4.3** Recognition rates (in percent correct) for human speech recognition (HSR) at a signal-to-noise ratio (SNR) of −10 dB, compared to automatic speech recognition (ASR) accuracies at several signal-to-noise ratios. The recognition task for ASR and for human listeners was to classify the middle phoneme in simple nonsense words, which were combinations of either consonant–vowel–consonant or vowel–consonant–vowel. The average rates are broken down into consonant and vowel recognition. At +10 dB SNR, ASR reaches an overall performance that is comparable to HSR at −10 dB SNR. If the same SNR of −10 dB is employed for ASR, error rates are almost 50% higher than for HSR

| Condition | | Average | Consonants | Vowels |
|---|---|---|---|---|
| HSR | −10 dB | 74.5 | 67.7 | 80.5 |
| ASR | | | | |
| | clean | 80.4 | 85.2 | 76.2 |
| | 15 dB | 76.1 | 77.7 | 74.6 |
| | 10 dB | 74.6 | 75.6 | 73.7 |
| | 5 dB | 69.8 | 69.5 | 70.0 |
| | 0 dB | 59.2 | 55.4 | 62.5 |
| | −5 dB | 49.8 | 41.0 | 57.5 |
| | −10 dB | 28.4 | 20.8 | 35.0 |



tion of temporal modulation frequency and spectral ripple frequency (Figure 4.18).

The advantage of such a second-order receptive field (i. e., the sensitivity to a certain combination of a spectral and a temporal cue) is the ability to detect specific spectro-temporal structures, e.g., formant glides or changes of fundamental frequency. It can also be considered as a generalization of the concepts outlined in this section. Even though this approach has successfully been implemented to improve the robustness of automatic speech recognizers [4.38], it has not yet been used to model human speech perception.

### 4.3.4 Man–Machine Comparison

Despite enormous technical advances in recent years, automatic speech recognition (ASR) still suffers from a lack of performance compared to human speech recognition (HSR), which prevents this technology from being widely used. Recognition accuracies of machines drop dramatically in acoustically adverse conditions, i. e., in the presence of additive or convolutive noise, which clearly demonstrates the lack of robustness. For complex tasks such as the recognition of spontaneous speech, ASR error rates are often an order of magnitude higher than those of humans [4.28]. If no high-level grammatical information can be exploited (as in a simple phoneme recognition task), the difference in performance gets smaller, but still remains very noticeable. For example, the HSR consonant recognition rate derived from the confusion matrix in Table 4.2 is 67.7%. The ASR score for the very same task (i. e., the same speech signals at an SNR of −10 dB), obtained with a common recognizer is 20.8%, which corresponds to a relative increase of error rates of 144% (Table 4.3).

This large gap underlines that current state-of-the-art ASR technology is by far not as capable as the human auditory system to recognize speech. As a consequence, the fields of ASR and speech perception modeling in humans may benefit from each other. Since the human auditory system results from a long biological evolution process and seems to be optimally adjusted to perform robust speech recognition, ASR may profit from auditory front ends which are based upon physiological findings and incorporate principles of our hearing system. Ideally, the feature matrix extracted from a speech sound which is used to classify the respective speech element should resemble the internal representation of that speech sound in our brain as closely as possible. Since this internal representation can be approximated by an auditory model, such an auditory model seems to be a good preprocessing stage for a speech recognizer [4.35].

On the other hand, models of the signal processing in the human auditory system can be evaluated using ASR, because–under ideal conditions–human speech perception and its model realization as anthropomorphic ASR system should yield a similar recognition performance and error pattern in well-defined acoustical conditions. Thus, modeling human speech perception can benefit from the computational methods developed in ASR.

Amazon EX1010 Page 112

## References


4.1 H. Fastl, E. Zwicker: *Psychoacoustics: Facts and Models* (Springer, Berlin-Heidelberg 2005)

4.2 E. Zwicker, G. Flottorp, S.S. Stevens: Critical bandwidth in loudness summation, J. Acoust. Soc. Am. **29**, 548 (1957)

4.3 T. Dau, B. Kollmeier, A. Kohlrausch: Modeling auditory processing of amplitude modulation. I. Detection and masking with narrow-band carriers, J. Acoust. Soc. Am. **102**, 2892–2905 (1997)

4.4 M.R. Schroeder: *Computer Speech: Recognition, Compression, Synthesis* (Springer, Berlin-Heidelberg 2005)

4.5 B.C.J. Moore, R.D. Patterson: *Auditory Frequency Selectivity* (Plenum, New York 1986)

4.6 A.J. Houtsma: Pitch perception. In: *Handbook of Perception and Cognition: Hearing*, ed. by B.C.J. Moore (Academic, London 1995) pp. 267–295

4.7 J. Verhey, D. Pressnitzer, I.M. Winter: The psychophysics and physiology of co-modulation masking release, Exp. Brain Res. **153**, 405–417 (2003)

4.8 R. Beutelmann, T. Brand: Prediction of speech intelligibility in spatial noise and reverberation for normal-hearing and hearing-impaired listeners J. Acoust. Soc. Am. **120**(1), 33–42 (2006)

4.9 H.S. Colburn, N.I. Durlach: Models of binaural interaction. In: *Handbook of Perception*, Vol. 4 (Academic, New York 1978) pp. 467–518

4.10 J.P. Penrod: Speech threshold and word recognition/ discrimination testing. In: *Handbook of Clinical Audiology, 4th edn*, ed. by J. Katz (Williams and Wilkins, Baltimore 1994) pp. 147–164

4.11 R. Plomp, A. Mimpen: Improving the reliability of testing the speech-reception threshold for sentences, Audiology **18**, 43–52 (1979)

4.12 B. Hagerman: Sentences for testing speech intelligibility in noise, Scand. Audiol. **11**, 79–87 (1982)

4.13 M. Nilsson, S.D. Soli, J.A. Sullivan: Development of the hearing in noise test for the measurement of speech reception thresholds in quiet and in noise, J. Acoust. Soc. Am. **95**(2), 1085–1099 (1994)

4.14 B. Kollmeier, M. Wesselkamp: Development and evaluation of a German sentence test for objective and subjective speech intelligibility assessment, J. Acoust. Soc. Am. **102**, 2412–2421 (1997)

4.15 K. Wagener, V. Kühnel, B. Kollmeier: Entwicklung und Evaluation eines Satztests für die deutsche Sprache I: Design des Oldenburger Satztests (Development and evaluation of a German sentence test I: Design of the Oldenburg sentence test), Zeitschrift für Audiologie **38**, 4–15 (1999)

4.16 K. Wagener, J.L. Josvassen, R. Ardenkjaer: Design, optimization and evaluation of a Danish sentence test in noise, Int. J. Audiol. **42**(1), 10–17 (2003)

4.17 T. Brand, B. Kollmeier: Efficient adaptive procedures for threshold and concurrent slope estimates for psychophysics and speech intelligibility tests, J. Acoust. Soc. Am. **111**(6), 2801–2810 (2002)

4.18 A. Bronkhorst: The cocktail party phenomenon: a review of research on speech intelligibility in multiple-talker conditions, Acustica **86**, 117–128 (2000)

4.19 N.R. French, J.C. Steinberg: Factors governing the intelligibility of speech sounds, J. Acoust. Soc. Am. **19**, 90–119 (1947)

4.20 H. Fletcher, R.H. Galt: The perception of speech and its relation to telephony, J. Acoust. Soc. Am. **22**, 89–151 (1950)

4.21 ANSI: *Methods for the calculation of the articulation index, ANSI S3.5-1969* (American National Standards Institute, New York 1969)

4.22 ANSI: *Methods for calculation of the speech intelligibility index, ANSI S3.5-1997* (American National Standards Institute, New York 1997)

4.23 T. Houtgast, H.J.M. Steeneken: A review of the MTF concept in room acoustics and its use for estimating speech intelligibility in auditoria, J. Acoust. Soc. Am. **77**, 1069–1077 (1985)

4.24 IEC: Sound system equipment – Part 16: Objective rating of speech intelligibility by speech transmission index. INTERNATIONAL STANDARD 60268-16 Second edition 1998-03 (1998)

4.25 H. Müsch, S. Buus: Using statistical decision theory to predict speech intelligibility. I. Model structure, J. Acoust. Soc. Am. **109**, 2896–2909 (2001)

4.26 H. Müsch, S. Buus: Using statistical decision theory to predict speech intelligibility, II. Measurement and prediction of consonant-discrimination performance J. Acoust. Soc. Am. **109**, 2910–2920 (2001)

4.27 I. Holube, B. Kollmeier: Speech intelligibility prediction in hearing-impaired listeners based on a psychoacoustically motivated perception model, J. Acoust. Soc. Am. **100**, 1703–1716 (1996)

4.28 R. Lippmann: Speech recognition by machines and humans, Speech Commun. **22**, 1–15 (1997)

4.29 S. Greenberg, W.A. Ainsworth, A.N. Popper: Speech Processing in the Auditory System. In: *Handbook of Auditory research*, Vol. 18, ed. by R.R. Fay (Springer, New York 2004)

4.30 M.R. Schroeder, H.W. Strube: Flat spectrum speech, J. Acoust. Soc. Am. **79**, 1580–1583 (1986)

4.31 R.V. Shannon, F.G. Zeng, V. Kamth, J. Wygonsky, M. Ekelid: Speech recognition with primarily temporal cues, Science **270**, 303–304 (1995)

4.32 R. Jakobson, C.G.M. Fant, M. Halle: *Preliminaries to speech analysis: the distinctive features and their correlates* (MIT Press, Cambridge 1963)

4.33 G.A. Miller, P.E. Nicely: An analysis of perceptual confusions among some english consonants, J. Acoust. Soc. Am. **27**, 338–352 (1955)




Amazon EX1010 Page 113

4.34 M.D. Wang, R.C. Bilger: Consonant confusions in noise: a study of perceptual features, J. Acoust. Soc. Am. **54**, 1248–1266 (1973)
4.35 J. Tchorz, B. Kollmeier: A model of auditory perception as front end for automatic speech recognition, J. Acoust. Soc. Am. **106**(4), 2040–2050 (1999)
4.36 M. Hansen, B. Kollmeier: Objective modeling of speech quality with a psychoacoustically validated auditory model, J. Audio Eng. Soc. **48**(5), 395–408 (2000)
4.37 C.E. Schreiner, G. Langner: Periodicity coding in the inferior colliculus of the cat II. Topographical organization, J. Neurophys. **60**, 1823–1840 (1988)
4.38 M. Kleinschmidt: Methods for capturing spectro-temporal modulations in automatic speech recognition, Acustica united with Acta Acustica **88**(3), 416–422 (2002)
4.39 D.A. Depireux, J.Z. Simon, D.J. Klein, S.A. Shamma: Spectro-temporal response field characterization with dynamic ripples in ferret primary auditory cortex, J. Neurophysiol. **85**(3), 1220–1234 (2001)
4.40 C. Kaernbach: The Memory of Noise, Exp. Psychol. **1**(4), 240–248 (2004)

# 5. Speech Quality Assessment

V. Grancharov, W. B. Kleijn

In this chapter, we provide an overview of methods for speech quality assessment. First, we define the term *speech quality* and outline in Sect. 5.1 the main causes of degradation of speech quality. Then, we discuss subjective test methods for quality assessment, with a focus on standardized methods. Section 5.3 is dedicated to objective algorithms for quality assessment. We conclude the chapter with a reference table containing common quality assessment scenarios and the corresponding most suitable methods for quality assessment.

5.1 **Degradation Factors Affecting Speech Quality** ..... 84
5.2 **Subjective Tests** ..... 85
5.2.1 Single Metric (Integral Speech Quality) ..... 85
5.2.2 Multidimensional Metric (Diagnostic Speech-Quality) ..... 87
5.2.3 Assessment of Specific Quality Dimensions ..... 87
5.2.4 Test Implementation ..... 88
5.2.5 Discussion of Subjective Tests ..... 89
5.3 **Objective Measures** ..... 90
5.3.1 Intrusive Listening Quality Measures 90
5.3.2 Non-Intrusive Listening Quality Measures ..... 93
5.3.3 Objective Measures for Assessment of Conversational Quality ..... 94
5.3.4 Discussion of Objective Measures .... 94
5.4 **Conclusions** ..... 95
**References** ..... 96



The rapid deployment of speech processing applications increases the need for speech quality evaluation. The success of any new technology (e.g., network equipment, speech codec, speech synthesis system, etc.) depends largely on end-user opinion of perceived speech quality. Therefore, it is vital for the developers of a new service or speech processing application to assess its speech quality on a regular basis.

In addition to its role for services and speech processing, speech quality evaluation is of critical importance in the areas of clinical hearing diagnostics and psychoacoustical research. Although this chapter addresses speech quality mainly from the viewpoint of telecommunication applications, it is also of general interest for researchers dealing with speech quality assessment methods.

When the speech signal reaches the human auditory system, a speech perception process is initiated. This process results in an *auditory event*, which is internal and can be measured only through a description by the listener (the subject). The subject then establishes a relationship between the perceived and expected auditory event. Thus, the speech quality is a result of a perception and assessment process.

Since the quality of a speech signal does not exist independently of a subject, it is a *subjective* measure. The most straightforward manner to estimate speech quality is to play a speech sample to a group of listeners, who are asked to rate its quality. Since subjective quality assessment is costly and time consuming, computer algorithms are often used to determine an *objective* quality measure that approximates the subjective rating. Section 5.2 provides an overview of subjective tests for speech quality assessment, while Sect. 5.3 is dedicated to objective quality assessment measures.

Speech quality has many perceptual dimensions. Commonly used dimensions are intelligibility, naturalness, loudness, listening effort, etc., while less commonly used dimensions include nasality, graveness, etc. However, the use of a *multidimensional metric* for quality assessment is less common than the use of a *single metric*, mainly as a result of cost and complexity. A single metric, such as the mean opinion score scale, gives an integral (overall) perception of an auditory event and is therefore sufficient to predict the end-user opinion of a speech communication system. However, a single metric does not in general provide sufficient detail for system designers. Multidimensional-metric tests

Amazon EX1010 Page 115

are discussed in Sect. 5.2.2 and single-metric tests are discussed throughout the remainder of Sect. 5.2.

In some applications, it is desirable or historically accepted to measure only specific quality dimensions, such as *intelligibility*, *listening effort*, *naturalness*, and *ability for talker recognition*. The most popular among these measures are covered in Sect. 5.2.3.

The *true* speech quality is often referred to as *conversational quality*. Conversational tests usually involve communication between two people, who are questioned later about the quality aspects of the conversation, see Sect. 5.2.1. However, the most frequently measured quantity is *listening quality*, which is the focus of Sect. 5.2.1. In the listening context, the speech quality is mainly affected by speech distortion due to speech codecs, background noise, and packet loss. One can also distinguish *talking quality*, which is mainly affected by echo associated with delay and sidetone distortion.

The distorted (processed) signal or its parametric representation is always required in an assessment of speech quality. However, based on the availability of the original (unprocessed) signal, two test situations are possible: *reference based* and *not reference based*. This classification is common for both the subjective and objective evaluation of speech quality. The absolute category rating (ACR) procedure, popular in subjective tests, does not require the original signal, while in the degradation category rating (DCR) approach the original signal is needed. In objective speech quality assessment, the historically accepted terms are *intrusive* (with original) and *non-intrusive* (without original). These two test scenarios will be discussed throughout the chapter.



## 5.1 Degradation Factors Affecting Speech Quality

The main underlying causes of degradation of speech quality in modern speech communication systems are delay (latency), packet loss, packet delay variation (jitter), echo, and distortion introduced by the codec. These factors affect psychological parameters such as intelligibility, naturalness, and loudness, which in turn determine the overall speech quality.

In this section, we briefly list the most common impairment factors. We divide them into three classes:

1. factors that lead to listening difficulty
2. factors that lead to talking difficulty
3. factors that lead to conversational difficulty

The reader can find more-detailed information in International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) Rec. G.113 [5.1]. The effect of transmission impairments on users is discussed in ITU-T Rec. P.11 [5.2].

Degradation factors that cause an increase in *listening difficulty* include packet loss, distortion due to speech codecs, speech clipping, and listener echo. *Packet loss* corresponds to the percentage of speech frames that do not reach their final destination. If no protective measures are taken, a packet loss rate of 5% results in significant degradation of the speech quality. Bursts of packet loss also affect speech quality. In systems without error concealment, *speech clipping* occurs at any time when the transmitted signal is lost. Speech clipping may temporarily occur when the connection suffers from packet loss or when voice activity detectors are used. *Listener echo* refers to a transmission condition in which the main speech signal arrives at the listener's end of the connection accompanied by one or more delayed versions (echoes) of the signal. The intelligibility decreases as the *loudness loss* increases. On the other hand, if the loudness loss decreases too much, customer satisfaction decreases because the received speech is too loud.

Degradation factors that cause difficulty while *talking* are talker echo and an incorrectly set sidetone. *Talker echo* occurs when some portion of the talker's speech signal is returned with a delay sufficient (typically more than 30 ms) to make the signal distinguishable from the normal sidetone. The *sidetone* of a telephone set is the transmission of sound from the telephone microphone to the telephone receiver in the same telephone set. Too little sidetone loss causes the returned speech levels to be too loud and thus reduces customer satisfaction. Excessive sidetone loss can make a telephone set sound *dead* as one is talking. In addition, the sidetone path provides another route by which room noise can reach the ear.

*Conversation* difficulties are caused by a third class of degradation factors. *Delay* is defined as the time it takes for the packet to arrive at its destination. Long delays impair a conversation. *Intelligible crosstalk* occurs when the speech signal from one telephone connection is coupled to another telephone connection such that the coupled signal is audible and intelligible to one or both of the participants on the second telephone connection. The *background noise* in the environment of the tele-

phone set may have a substantial effect on the ease of carrying on a conversation.

The study of degradation factors is important in the design of a speech quality assessment test. The set of degradation factors present in a communication system determines the type of test to be performed. If the degradation factors cause only an increase in listening difficulty, it is sufficient to perform relatively inexpensive and simple tests that measure *listening quality*. If the degradation factors cause difficulty while talking, or difficulty while conversing, it is recommended to perform the more-complex *conversational quality* tests.

## 5.2 Subjective Tests

Speech quality is a complex psychoacoustic outcome of the human perception process. As such, it is necessarily subjective, and can be assessed through listening test involving human test subjects that listen to a speech sample and assign a rating to it. In this section, we cover the most commonly used subjective quality tests.

### 5.2.1 Single Metric (Integral Speech Quality)

Users of new speech processing applications are often unaware of the underlying technology. Their main criterion for assessing these applications is based on overall speech quality. Therefore, we start our discussion with *single-metric* subjective tests. In these tests, speech is played to a group of listeners, who are asked to rate the quality of this speech signal based on their *overall perception*.

#### Listening Quality

In an ACR test, a pool of listeners rate a series of audio files using a five-level impairment scale, as shown in Table 5.1. After each sample is heard, the listeners express an opinion, based only on the most recently heard sample. The average of all scores thus obtained for speech produced by a particular system represents its *mean opinion score* (MOS). The ACR listening quality method is standardized in [5.3], and is the most commonly used subjective test procedure in telecommunications. The main reason for the popularity of this test is its simplicity.

A good method for obtaining information on the detectability of a distortion (e.g., echo) as a function of some objective quantity (e.g., listening level) is to use the *detectability opinion scale* (Table 5.2). The decisions on a detectability scale are not equivalent to responses on a continuous scale. It is therefore recommended to use as a method of analysis the probability of response [5.3].

A disadvantage of ACR methods is that for some applications the resolution of their quality scale is not sufficient. In such cases the DCR method is appropriate. DCR methods provide a quality scale of higher resolution, due to comparison of the distorted signal with one or more reference/anchor signals. In a DCR test, the listeners are presented with the unprocessed signal as a reference before they listen to the processed signal. The task for the listener is to rate the perceived degradation by comparing the second stimulus to the first on the scale presented in Table 5.3. The quantity evaluated from the scores is referred to as the *degradation mean opinion score* (DMOS). DCR methods are also standardized in [5.3].

*ABX* is another popular method for speech quality assessment [5.4]. It consists of presenting the listener

**Table 5.1** Grades in the MOS scale. Listeners express their opinion on the quality of the perceived speech signal (no reference presented)

| | |
|---|---|
| Excellent | 5 |
| Good | 4 |
| Fair | 3 |
| Poor | 2 |
| Bad | 1 |

**Table 5.2** Grades in the *detectability opinion* scale. Listeners give their opinion on the detectability of some property of a sound

| | |
|---|---|
| Objectionable | 3 |
| Detectable but not objectionable | 2 |
| Not detectable | 1 |

**Table 5.3** Grades in the DMOS scale. Listeners are asked to describe *degradation* in the second signal in relation to the first signal

| | |
|---|---|
| Inaudible | 5 |
| Audible but not annoying | 4 |
| Slightly annoying | 3 |
| Annoying | 2 |
| Very annoying | 1 |



Amazon EX1010 Page 117

with three samples: A, B, and X. The listener is asked to select which of the samples A and B is identical to X. This method provides a simple assessment of listener reliability, through the calculation of the number of correct answers.

A standardized extension of the ABX method is described in [5.5]. The subject assesses one of three stimuli: A, B, and X. The known reference is always available as stimulus X. The hidden reference and the object are randomly assigned to A and B. The test subject is asked to assess the impairments in A compared to X, and B compared to X, according to a continuous impairment scale. [5.5] defines methods for the subjective assessment of *small impairments* in audio signals (aimed at high-quality audio signals). This method is aimed at so-called expert listeners (who have been trained). The higher the quality reached by the systems to be tested, the more important it is to have expert listeners.

A test procedure more suitable for the subjective assessment of *intermediate*-quality audio signals is the *multistimulus test with hidden reference and anchor* (MUSHRA), standardized in [5.6]. In this test, both a known reference signal and hidden anchors are used. During the test, the users can switch at will between the known reference signal (which is not scored and is known to be the original) and any of the signals under test. They are required to score the stimuli on a continuous quality scale according to a continuous quality scale from 0 to 100, using sliders on an electronic display. Thus, the subjects score the stimuli according to a *continuous quality scale*. This scale is also extensively used for the evaluation of video quality [5.7]. Continuous quality scales provide a continuous rating system to avoid quantization errors, but they are divided into five equal lengths that correspond to the conventional ITU-R fine-point quality scale. Results obtained from continuous quality methods should not be treated as absolute scores but as difference scores between the known reference and the test condition.

In the DCR methods, listeners always rate the amount by which the processed (second) sample is degraded relative to the unprocessed (first) sample. *Comparison category rating* (CCR) procedures can be seen as a refinement of DCR tests. A CCR procedure is a comparison test in which the listeners identify the quality of the second stimulus relative to the first using a two-sided rating scale. This scale is referred to as *comparison mean opinion score* (CMOS) and is presented in Table 5.4. The key idea in the CCR test method is to eliminate the ordering restriction of the DCR test at the expense of doubling the total number of trials. For half of the trials (chosen at random), the unprocessed speech is presented first, followed by the processed speech. For the other half, the order is reversed. An advantage of the CCR method over the DCR method is the possibility to assess speech processing that either degrades or improves the quality of the speech.

**Table 5.4** Grades in the CMOS scale. Listeners grade the perceived quality of a speech signal in relation to a reference speech signal

| | |
|---|---|
| Much better | 3 |
| Better | 2 |
| Slightly better | 1 |
| About the same | 0 |
| Slightly worse | -1 |
| Worse | -2 |
| Much worse | -3 |

For completeness, we mention a method for the general assessment of sound quality that is described in [5.8]. For this method, expert listeners are always preferred over non-expert listeners. Based on the nature and the purpose of the test, each of the scales presented in Tables 5.1, 5.3, and 5.4 can be used.

### Conversational Quality

So far we have discussed test procedures concerned with the *listening quality*. In these tests, the listener rates the signal that is received from the far end. Effects such as echoes at the talker side and transmission delays are ignored. *Conversational* quality refers to how listeners rate their ability to converse during the call (which includes listening quality impairments).

In conversational tests, a pool of subjects are placed in interactive communication scenarios, and asked to complete a task [5.3]. At the end of the conversation, the listeners give an opinion on the connection on five-point category-judgment scale (Table 5.1). The test subjects also give their binary response on a *difficulty scale*. The test subjects answer the question: *Did you or your partner have any difficulty in talking or hearing over the connection* with {Yes = 1, No = 0}. Conversational quality tests are standardized in [5.3], and a thorough study can be found in [5.9].

Conversational tests are exclusively not reference based, yet they are significantly more complex to design and control than conventional listening tests. It is therefore beneficial to perform listening tests where possible. Research on the relations between listening and conversational quality can be found in [5.10].



Amazon EX1010 Page 118

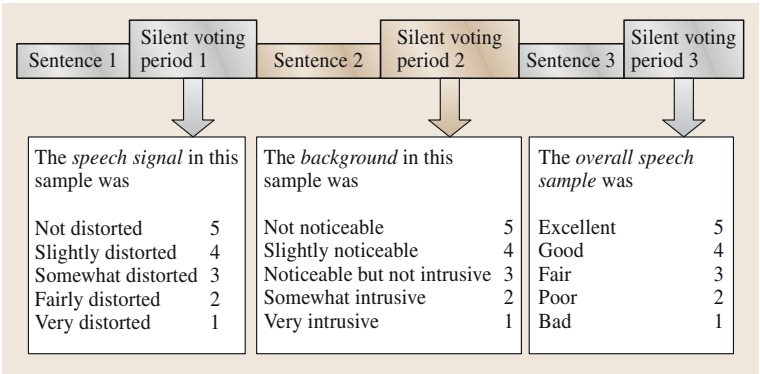



**Fig. 5.1** The ITU-T P.835's scheme for evaluating the subjective quality of noise-suppression algorithms. Each test sample is comprised of three subsamples, where each subsample is followed by a silent voting period

### 5.2.2 Multidimensional Metric (Diagnostic Speech-Quality)

In many cases, the designers of a speech-processing algorithm want to obtain information about speech quality together with diagnostic information. This task requires a *multidimensional metric* subjective test, which provides more insight into system faults and shortcomings than tests providing only an overall speech quality measure.

A procedure that assesses the speech quality on a multidimensional metric is the *diagnostic acceptability measure* (DAM) [5.11], which provides more-systematic feedback and evaluates speech quality on 16 scales. These scales belong to one of three categories: signal quality, background quality, and overall quality. A weighted average of all these scales forms a composite measure that describes the condition under test. The DAM procedure is designed for trained (experienced) listeners and, in contrast to most other measures, the speech material is known to the listeners.

While the DAM procedure is general purpose, it is also possible to design *specialized* multidimensional metrics. One example is the procedure standardized in [5.12], which describes a methodology for evaluating the subjective quality of speech in noise and noise-suppression algorithms. The methodology uses separate rating scales to estimate independently the subjective quality of the speech signal alone, the background noise alone, and overall quality. Each speech sample is comprised of three subsamples, where each subsample is followed by a silent voting period Fig. 5.1. Recent experiments with this procedure are presented in [5.13].

**Table 5.5** Grades in the listening effort scale. Listeners assess the *effort required* to understand the meaning of sentence

| | |
|---|---|
| Complete relaxation possible; no effort required | 5 |
| Attenuation necessary; no appreciable effort required | 4 |
| Moderate effort required | 3 |
| Considerable effort required | 2 |
| No meaning understood with any feasible effort | 1 |

### 5.2.3 Assessment of Specific Quality Dimensions

The *diagnostic rhyme test* (DRT) [5.14] evaluates speech intelligibility. It uses a set of isolated words to test for consonant intelligibility in the initial position. The DRT uses 96 pairs of confusable words (which differ by a single acoustic feature in the initial consonant) spoken in isolation. First, the subject is presented visually with a pair of rhymed words. Then, one word of the pair (selected at random) is presented aurally and the subject is required to indicate which of the two words was played.

The *modified rhyme test* (MRT) [5.15, 16] is an extension to the DRT. It evaluates the intelligibility of both initial and final consonants. A set of six words is played one at a time and the listener marks which word they think they hear.

Intelligibility tests are rarely used to assess the quality of speech coding systems, since while most systems affect naturalness, they do not degrade speech intelligibility significantly. In other applications, such as speech synthesis, intelligibility tests are the major testing tool. In addition to DRT and MRT, other popular tests are the Bellcore test [5.17], and the minimal pairs intelligibility (MPI) test [5.18].

A frequently used opinion scale that focuses on the listener's ability to understand the meaning of the sentence is the *listening effort* scale [5.3] (Table 5.5). Methods using the listening effort scale generally yield results that are better correlated with conversational

opinion scores than methods using the listening quality scale [5.19].

### 5.2.4 Test Implementation

The designer of a subjective test should take care of:

1. the selection of the test material
2. the selection of the test subjects
3. the choice of the test procedure
4. the analysis of the test results

Careful experimental design and planning is needed to ensure that uncontrolled factors, which can cause ambiguity in the test results, are minimized.

Speech material should consist of simple, short, meaningful, and preferably phonetically balanced [5.20], sentences. In the ACR test, the sentences should be made into sets of two without an obvious connection of meaning between the sentences in a set (Fig. 5.2). If the impairments are simulated, the test designer has the responsibility to make the model as close to reality as possible. As an example, for high noise levels people change their talking behavior (the Lombard effect [5.21]), which is often not considered when noise recording is digitally added to the clean-speech database.

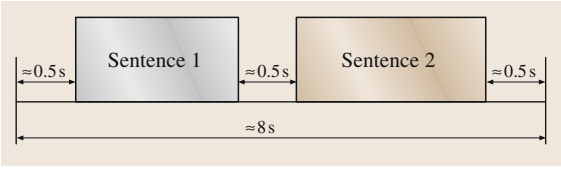



**Fig. 5.2** A typical MOS test uses stimuli that contain two short sentences, separated by a 0.5 s seconds silence gap. The resulting two-utterance stimulus has a duration of about 8–10 s

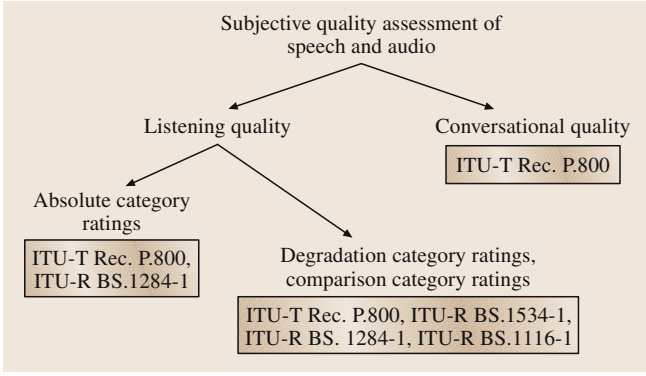



**Fig. 5.3** Classification of subjective quality assessment methods and related ITU standards and recommendations

The most important requirements for the selection of the test subjects are:

1. they must be native speakers of the language in which the tests are being conducted,
2. they do not have known hearing defects, and
3. they have not recently been involved in quality assessment tests of the same or similar speech processing system.

Note that some test procedures require experienced listeners, e.g., [5.5], while others require naïve listeners, e.g., [5.12]. In general, a listening test should use experienced listeners when impairments are small, or when fast test convergency is needed (i. e., small number of listeners). However, listening tests using naïve listeners represent the actual conditions under which the communication system will be used better.

A classification of the most popular subjective tests standardized by the ITU is shown in Fig. 5.3. As discussed earlier, the choice of a particular test procedure is dependent on the type of impairments and the required format (e.g., overall quality, diagnostic information, etc.). Different test procedure may require different listening conditions (headphones or loudspeakers, scoring on a computer or voting on paper, etc.), but these conditions cannot be changed within the test. For most of the test procedures it is recommended that test subjects are given a break after 20–30 min of listening. It is vital that subjects are not overloaded to the point of decreased accuracy of judgment.

Test designers usually include *reference conditions* (well-defined test conditions) in the quality assessment procedure. There are two major reasons to introduce reference conditions in listening tests. The first is to provide a convenient means for comparing subjective test results from different laboratories. The second is that reference conditions provide a spread in quality level, which increases the consistency of human ratings across tests.

Reference conditions, used in telecommunications, typically include a best possible condition (the original signal or a high-quality speech codec), as well as conditions where controlled degradation is introduced. One of the most commonly used types of reference signals is the *modulated noise reference unit* (MNRU), which is standardized in [5.22]. MNRU produces random noise with an amplitude that is proportional to the instantaneous speech amplitude (i. e., multiplicative noise):

$$y_k = s_k\left(1 + 10^{-Q/20} v_k\right) , \tag{5.1}$$



Amazon EX1010 Page 120

where $y_k$ is the output noise-modulated speech, $s_k$ is the original speech signal, $v_k$ is the random noise, $k$ is the sample index, and $Q$ is the ratio of the speech power to the modulated noise power.

The average of the opinion scores should be calculated for each test condition. A variety of statistical procedures is available to compare the results obtained from listening tests.

The $t$-test is a commonly used method to identify a significant difference in means between two conditions. The first step is to calculate the minimum significant difference (MSD) threshold (at the 95% significance level):

$$\mathrm{MSD} = t_{95\%,2(M-1)}\sqrt{\frac{\sigma_\mathrm{A}^2+\sigma_\mathrm{B}^2}{M}}\,, \tag{5.2}$$

where $M$ is the number of observations, $\sigma_\mathrm{A}^2$ is the variance of the scores for condition A, $\sigma_\mathrm{B}^2$ is the variance of the scores for condition B, and $t_{95\%,2(M-1)}$ is a tabulated critical value from Student's $T$ distribution at the 95% significance level with $2(M-1)$ degrees of freedom.

The second step is to compare the absolute difference between the mean values for the two conditions $|\mu_\mathrm{A}-\mu_\mathrm{B}|$ with the previously calculated MSD threshold. If $|\mu_\mathrm{A}-\mu_\mathrm{B}| > \mathrm{MSD}$ one can conclude that the quality of condition A is significantly different from the quality of condition B at the 95% significance level.

If there is a need to compare more than two conditions, or compare conditions created by more than one independent variable, one has to consider the usage of the analysis of variance (ANOVA) or honestly significant difference (HSD) tests [5.23].

### 5.2.5 Discussion of Subjective Tests

Since the auditory event that causes the perception of quality degradation is not available for direct measurement, the test subjects describe their perception on quality scales, which may be of different types. In a *category scale*, the subject assigns a certain class (usually labeled with numbers or symbols) to their perception of signal quality. In an *ordinal scale* the subject arranges test samples into an order (e.g., the loudest sample first). In *interval* and *ratio scales* the differences between classes, or correspondingly the ratio between classes, is quantified.

In ACR procedures, the test subjects rate each presented test item on a discrete ordinal scale, for example labeled as described in Table 5.1. Each of these categories is assigned a number, and the total average of all rating results is often expressed as an MOS. In ACR procedures, the mean values are calculated assuming that each category occupies the same interval on a perceptual continuum, and the statistical processing of the data assumes that this five-point ITU scale is an interval scale.

Some studies [5.24] indicate that the intervals represented by the quality scale labels are not equal. There are also indications that the scale labels cannot be translated adequately into different languages, such that the scale is *equal* in different countries. It is common to apply statistical tests (e.g., analysis of variance) to recorded scores. In the cases where MOS scores are not presumed to represent a linear scale, statistics for ordinal scales may need to be applied [5.25].

In the quality assessment procedures discussed so far, we have assumed that the speech quality does not vary significantly during the speech sample being evaluated. In reality, quality of service may vary even during a single conversation and attention must be paid when investigating time-varying impairments. Time-varying speech quality has been the main focus of several studies, e.g., [5.26–32]. Some important observations from this research are that

1. the long-term perceived speech quality scores are lower than the time average of the corresponding short-term perceived speech quality scores,
2. the perceived long-term quality decreases when the variance of the short-term speech quality increases,
3. listeners detect decreases in speech quality more quickly than increases in speech quality, and
4. long-term scores are more strongly influenced by events near the scoring time than by earlier events.

A methodology for the assessment of time-varying speech quality is standardized in [5.33]. The test subjects are asked to assess the speech quality continuously by moving a slider along a continuous scale so that its position reflects their opinion on quality at that instant (the slider position should be recorded every 500 ms). For each utterance a *mean instantaneous judgment* is obtained by averaging individual instantaneous judgments over the subjects. At the end of the utterance, subjects are asked to rate its overall quality on a five-point ACR scale. For each utterance a *mean overall judgment* is obtained by averaging individual overall judgments over the subjects on the ACR scale. A substantial difference between the continuous score and the overall score indicates a time-varying quality level.



Amazon EX1010 Page 121

## 5.3 Objective Measures

In this Section we discuss *objective* quality measures: computer algorithms designed to estimate quality degradation. Subjective tests are believed to give the *true* speech quality. However, the involvement of human listeners makes them expensive and time consuming. Subjective tests are not suitable to monitor the quality of service (QoS) of a network on a daily basis, but objective measures can be used for this purpose at a very low cost. The main aspects that affect the applicability of objective and subjective measures are summarized in Table 5.6.

Similarly to the subjective tests, some objective quality measures are designed to assess listening quality, while others assess conversational quality. Alternatively, the classification of objective quality measures can be based on the type of input information they require: intrusive quality measures require access to both the original and distorted speech signal, while non-intrusive measures base their estimate only on the distorted signal, as illustrated in Fig. 5.4.

Early work on objective quality assessment focused exclusively on intrusive methods, and non-intrusive methods have received attention only in the last decade. If only the distorted signal is available, sophisticated modeling of the speech and/or distortions is typically needed. In contrast, intrusive measures range from very simple to sophisticated models that consider human perception.

### 5.3.1 Intrusive Listening Quality Measures

#### Simple Time- and Frequency-Domain Measures

The simplest class of intrusive objective quality measures consists of waveform-comparison algorithms, such as those based on the signal-to-noise ratio (SNR) and segmental SNR (SSNR). These two algorithms are easy to implement, have low computational complexity, and can provide indications of perceived speech quality for a specific waveform-preserving speech system [5.34]. Unfortunately, when used to evaluate coding and transmission systems in a more-general context, SNR and SSNR show little correlation to perceived speech quality. These measures are also sensitive to a time shift, and therefore require precise signal alignment, which is not always a trivial problem.

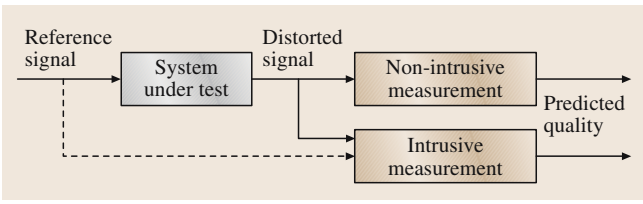



**Fig. 5.4** Intrusive and non-intrusive types of quality assessment. Non-intrusive algorithms do not have access to the reference signal

**Table 5.6** Comparison of subjective and objective methods for quality assessment. The symbol "+" is used to denote that the method is advantageous over the other method, denoted by "−"

| | Subjective measures | Objective measures |
|---|---|---|
| Cost | − | + |
| Reproducibility | − | + |
| Automation | − | + |
| Unforeseen impairments | + | − |

The overall SNR distortion measure between an original $\boldsymbol{s}$ and a distorted $\boldsymbol{y}$ speech vector is calculated as [5.34]:

$$d_{\mathrm{SNR}}(\boldsymbol{s},\boldsymbol{y}) = 10\log_{10}\left(\frac{\boldsymbol{s}^{\mathrm{T}}\boldsymbol{s}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}}\right) , \tag{5.3}$$

where $\boldsymbol{e} = \boldsymbol{s} - \boldsymbol{y}$. The vector dimension is sufficient to contain the entire utterance.

The SSNR is calculated by splitting the two vectors into smaller blocks and calculating an SNR value for each of these blocks. The final SSNR value is obtained by averaging the per-block SNR values, e.g., [5.35]:

$$d_{\mathrm{SSNR}}(\boldsymbol{s},\boldsymbol{y}) = \frac{1}{N}\sum_{n=1}^{N} 10\log_{10}\left(\frac{\boldsymbol{s}_n^{\mathrm{T}}\boldsymbol{s}_n}{\boldsymbol{e}_n^{\mathrm{T}}\boldsymbol{e}_n}\right) , \tag{5.4}$$

where $N$ is the total block number, $n$ is the block index, and the per-block error vector is defined as $\boldsymbol{e}_n = \boldsymbol{s}_n - \boldsymbol{y}_n$. A typical block length is 5 ms. In a perceptual modification of this measure, studied in [5.36], weights that depend on the time-varying spectral envelope of the original speech are applied.

Frequency-domain measures are known to be significantly better correlated with human perception [5.34], but still relatively simple to implement. One of their critical advantages is that they are less sensitive to signal misalignment. Some of the most popular frequency-domain techniques are the *Itakura–Saito* (IS), the cepstral distance (CD), the log-likelihood (LL), and the log-area-ratio (LAR) measures [5.37].

The gain-normalized spectral distortion (SD) is widely accepted as a quality measure of coded speech

Amazon EX1010 Page 122

spectra [5.38]. In its most commonly used form, the SD evaluates the similarity of two autoregressive envelopes on a per-frame basis:

$$d_{\mathrm{SD}}^{(n)}(\boldsymbol{s},\boldsymbol{y}) = \int_{-\pi}^{\pi} \left[10\log_{10}\left(\frac{P_s(\omega,n)}{P_y(\omega,n)}\right)\right]^2 \frac{\mathrm{d}\omega}{2\pi}\,, \quad (5.5)$$

where $P_s(\omega, n)$ and $P_y(\omega, n)$ are the autoregressive spectra of the clean and processed signal. The per-frame distances are generally combined into a global (per-signal) distortion in the form of a root-mean SD:

$$d_{\mathrm{SD}}(\boldsymbol{s},\boldsymbol{y}) = \frac{1}{N}\sqrt{\sum_{n=1}^{N} d_{\mathrm{SD}}^{(n)}(\boldsymbol{s},\boldsymbol{y})}\,, \quad (5.6)$$

where $N$ is the total number of frames. A further enhancement of SD is proposed in [5.39], where the authors apply weighting with a perceived loudness function, which takes in consideration frequency-dependent perception sensitivities. It is noted that perceived distortion of the spectral fine structure is not considered in the SD measures described above.

The distinguishing characteristic of both waveform comparison and frequency-domain techniques is that the basic measure operates on a per-frame basis and that they use simple schemes to combine the estimated per-frame distortions [5.40]. The most commonly employed method for the construction of a global objective distortion measure over a number of $N$ frames can be computed by arithmetically averaging the per-frame computed distance measures. In a more-sophisticated form of this basic measure, unequal contributions to perception from each speech frame can be taken into account. An unequal distribution can be related to frame energy and/or voicing.

The measures discussed in this subsection only have meaning when applied to frames where speech is present. A known problem is that authors do not specify, or use different rules, to select the subset of $N$ speech active frames from the entire frames set. To obtain repeatable results, it is advisable to use speech activity detection based on a speech level meter [5.41].

The intrusive quality measures methods discussed thus far are based on simple and tractable mathematical models. The next topic in our discussion covers a more-sophisticated family of quality measures that is based on knowledge of the human auditory system.

### Psychoacoustically Motivated Measures

Many intrusive quality measures are based on mimicry of the human auditory system. This approach has led to highly accurate objective performance measures that are useful in many contexts where the original signal is present.

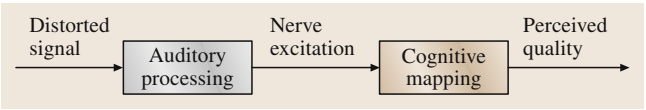



**Fig. 5.5** Human perception of speech quality, involving both hearing and judgment

The process of human assessment of speech quality can be described as consisting of two stages, as illustrated in Fig. 5.5. In the first auditory processing (hearing) stage, the received speech acoustic signal is transformed into an auditory nerve excitation. Essential elements of auditory processing include bark-scale frequency warping and spectral power to subjective loudness conversion. The second stage of the quality assessment process entails cognitive processing in the brain, where compact features (that contain information about the anomalies in the speech signal) are extracted from auditory excitations. These features are combined to form a final impression of the perceived speech signal quality. The cognitive models of speech distortions are less well developed than the auditory model.

Figure 5.6 shows an outline of an ideal mimicry-based speech quality assessment algorithm that incorporates both stages of subjective quality assessment. Both original and distorted signals are first subjected to a perceptual transform that mimics the auditory periphery. Then a process mimicking high-level cognitive process extracts patterns related to the language specifics, context, etc. Finally, the distance between the expected (original) and actual patterns is mapped to a selected speech quality scale. Unfortunately, this scheme is currently not realizable, since the cognitive processing performed by the human brain is largely unknown.

The human auditory periphery is well understood and perceptual transforms are thought to be reasonably accurate. The transforms result in a signal representation that is scaled such that upon thresholding (representing a minimum precision) only perceptually relevant infor-

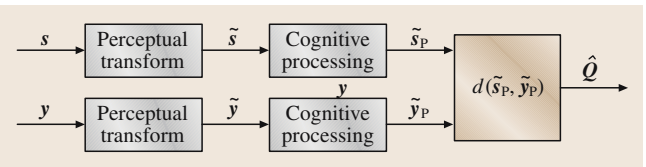



**Fig. 5.6** A speech quality assessment algorithm that mimics the process of human quality assessment

mation is retained. The transforms reflect the fact that the resolution of the ear is not uniform on the Hertz scale and that the relation between perceived loudness and signal intensity is nonlinear. The sensitivity of the ear is a function of frequency and the corresponding absolute hearing thresholds have been characterized. Finally, many studies have demonstrated time- and frequency-domain masking effects [5.42–45]. A study presented in [5.46] argues that nonuniform frequency resolution and nonlinear loudness perception are the most important properties of auditory models.

Historically, researchers have followed two different paths to incorporate knowledge of the auditory periphery into quality assessment schemes. The first is based on the *masking threshold concept*, where the reference signal is used to calculate an estimate of the actual masking threshold operating on a short-term frequency spectrum. The difference between the reference and processed signals is evaluated relative to this masking threshold [5.36, 47]. The second approach aims to compare *internal representations* directly. Based on a model of the auditory periphery, an internal representation of both signals is calculated. This internal representation contains the information that is available to the brain for comparison of signals [5.48]. Note that the second scheme is closer to the desired scheme shown in Fig. 5.6.

It can be shown that an appropriate weighting of the difference for the masking approach is precisely the inverse of the masking threshold. The frequency spectra weighted by the inverse masking function can then be interpreted as a simple internal representation of the human ear, reconciling the two historic approaches.

Internal representations can have different levels of sophistication. The bark spectral distortion (BSD) [5.49] can be seen as one of the first and simplest internal measures. The BSD is the averaged Euclidean distance between the original and distorted speech signals in the Bark domain. A similar measure is the information index (II) [5.50], according to which the auditory system is modeled by dividing the spectrum into 16 critical bands and applying empirical frequency weights and a hearing threshold for each band. The coherence function (CF) [5.51] is a measure of the signal-to-distortion ratio. The objective of the coherence function is to turn off uncorrelated signals and pass correlated signals.

A more-sophisticated internal presentation is used in [5.52], which describes a perceived speech quality assessment algorithm called perceptual speech quality measure (PSQM). Its scope is limited to higher-bitrate speech codecs operating over error-free channels. The transformation from the physical to the psychophysical (internal) domain is performed in three steps – frequency warping, time–frequency smearing, and level compression [5.52].

In ITU-R. BS.1387-1, perceptual quality assessment for digital audio (PEAQ) [5.53, 54], concurrent frames of the original and processed signals are each transformed to a basilar membrane representation (the internal representation), and differences are further analyzed as a function of frequency and time by a cognitive model.

ITU-T Rec. P.862, perceptual evaluation of speech quality (PESQ) [5.55], measures the effects of one-way speech distortion. The effects of loudness loss, delay, sidetone, echo, and other impairments related to two-way interaction are not reflected in PESQ scores. Factors for which PESQ has demonstrated acceptable accuracy are: speech input levels to a codec, transmission channel errors, packet loss and packet loss concealment with CELP codecs, bitrates if a codec has more than one bitrate mode, transcodings, environmental noise at the sending side, the effects of varying delay in listening only tests, short-term time warping of audio signal, and long-term time warping of audio signal.

ITU-T Rec. P.862 was designed to evaluate narrowband (3.4 kHz) speech quality, and cannot deal with wideband (7 kHz) speech quality. A recent research focus has been the development of a wideband extension for PESQ, ITU-T Rec. P.862.2 [5.56].

The above discussion centered on processing that mimics the human auditory periphery. The outcome is a representation that corresponds to that in the auditory nerve. This representation must then be mapped into a quality measure. It is important to note that the lack of knowledge about high-level cognitive brain processing cannot be compensated for by more-sophisticated models of the auditory periphery. Less-audible parts of the signal may be more objectionable if they are of higher importance for the pattern extraction and comparison process performed by the human brain.

Some important cognitive effects that affect the process of quality judgment are:

1. linear distortion is generally less objectionable than nonlinear distortion,
2. speech-correlated distortion is more objectionable than uncorrelated distortion,
3. if the local *information* complexity is high, then distortion is less objectionable, and
4. distortion in some spectral–temporal components, such as formants, is more objectionable, since they carry more information.



Amazon EX1010 Page 124

Even though not completely understood, these effects have been used successfully by researchers to improve perceived quality in speech coding [5.57, 58], and enhancement [5.59] applications. These studies are motivated by the evidence that noise is aurally masked by rapid changes in the speech signal [5.43].

Perhaps the simplest way to develop a relationship between two internal representations corresponding to auditory nerve signals and a target subjective opinion scale is to carry out a weighted summation of their difference along the frequency and time axes. As an example, PESQ performs integration in three steps, first over frequency, then over short time intervals, and finally over the entire speech signal. The authors of [5.60–62] present an error integration scheme that is more consistent with high-level brain processing. *Holier* et al. [5.60] uses an entropy measure of the difference internal representation. The measuring normalizing blocks (MNB) algorithm [5.61, 62] utilizes a relatively simple perceptual transform, but a sophisticated error pooling system. MNB uses a hierarchical structure of integration over a range of time and frequency intervals.

A conceptually simple objective speech quality measure can also be obtained by integrating elements of high-level brain processing. An interesting measure based on distortion in the spectral peaks of speech was proposed in [5.63]. The approach proposed in [5.64] is to measure the phonetic distance between the original and distorted signals (calculated as the weighted difference between spectral slopes over several frequency bands).

A more-general approach to simulate this process is through statistical mapping and data mining [e.g., Gaussian mixture models (GMMs) or neural networks (NNs)], or clustering [5.65]. Another example can be found in [5.66], where the authors recognize the importance of the high-level cognitive process and apply a statistical data-mining approach. In the approach of [5.66], a large pool of candidate features is created and the ones that lead to the most accurate prediction of perceived quality are selected.

### 5.3.2 Non-Intrusive Listening Quality Measures

Intrusive objective speech quality measures can provide a performance measure for a communication system without the need for human listeners. However, intrusive measures require the presence of the original signal, which is not available in some important applications, including QoS monitoring of telecommunication networks. For such applications *non-intrusive* quality assessment must be used.

A wide variety of approaches has been used to obtain non-intrusive quality assessment. These include methods that assess the possibility of the signal being produced by human physiology, methods that compare to the nearest signal from a speech database, and methods that learn the human mapping between signal and quality directly.

Reference [5.67] reports a non-intrusive speech quality assessment that attempts to predict the likelihood that the signal has been generated by the human vocal production system. To achieve this, the speech signal under assessment is first reduced to a set of features. This parameterized data is then used to estimate the perceived quality by means of physiologically based rules.

The measure proposed in [5.68] is based on comparing the output speech to an artificial reference signal that is appropriately selected from an optimally clustered codebook. The perceptual linear prediction (PLP) [5.69] coefficients are used as a parametric representation of the speech signal. A fifth-order all-pole model is performed to suppress speaker-dependent details of the auditory spectrum. The average distance between the unknown test vector and the nearest reference centroids provides an indication of speech degradation.

The authors of [5.70, 71] propose a method that employs intrusive algorithms. However, they avoid the need for the original signal. The method is based on measuring packet degradations at the receive end. The measured degradation is applied to a *typical* speech signal to produce a signal that is similarly degraded. An intrusive algorithm can then be used to map the speech signal and degradation signal to speech quality.

A novel, perceptually motivated, speech quality assessment algorithm based on the temporal envelope representation of speech is presented in [5.72] and [5.73].

A non-intrusive speech quality assessment system, based on a speech spectrogram, is presented in [5.74]. An interesting concept in this approach is that accurate estimation of speech quality is achieved without a perceptual transform of the signal.

The ITU standard for non-intrusive quality assessment is ITU-T P.563 [5.75]. It consists of a combination of a number of approaches. A total of 51 speech features are extracted from the signal. *Key features* are used to determine a dominant distortion class, in each of which a linear combination of features is used to predict a so-called intermediate speech quality. The final speech



Amazon EX1010 Page 125

quality is estimated from the intermediate quality and 11 additional features.

Another approach is to use perceptually motivated spectral envelope representations in combination with a mapping from this representation to the quality measure through a GMM [5.76,77]. The authors of [5.68,78] used a similar approach but with mappings based on hidden Markov models (HMMs) and neural networks.

A recent development [5.79] simplifies this method and combines the GMM mapping with a small set of features selected for optimal performance from a large set of features. As a result, the algorithm has very low complexity, requiring only a small fraction of the computational capability of a mobile phone.

### 5.3.3 Objective Measures for Assessment of Conversational Quality

The emphasis in this chapter, and also in the literature on quality assessment, is on listening quality. However, some applications may require the assessment of conversational quality, i. e., to include impairments such as delay and talker echo.

The E-model is a tool for predicting how an *average user* would rate the voice quality of a phone call with known characterizing transmission parameters (currently 21 input parameters). The E-model is standardized in ITU-T Rec. G.107 [5.80]. The objective of the E-model is to determine a transmission quality rating, i. e., the $R$ factor, with a typical range between 0 and 120. The $R$ factor can be converted to estimated listening and conversational quality MOS scores. The E-model does not compare the original and received signals directly. Instead, it uses the sum of equipment impairment factors, each one quantifying the distortion due to a particular factor. Impairment factors include the type of speech codec, echo, averaged packet delay, packet delay variation, and the fraction of packets dropped. A fundamental assumption is that the impairments on the psychological scale are additive [5.81]. The transmission quality rating is determined as

$$R = R_0 - I_s - I_d - I_{e\text{-}eff} + A \,, \tag{5.7}$$

where $R_0$ represent the basic signal-to-noise ratio (noisiness), including noise sources such as circuit and room noise, $I_s$ is a combination of all impairments that occur simultaneously with the voice signal (loudness), $I_d$ represents the impairments caused by the delay (echo and delay), $I_{e\text{-}eff}$ represents impairments caused by low-bitrate codecs and packet losses (distortion), and $A$ allows for compensation of impairments factors.

The broader scope of conversational quality assessment, as compared to listening quality assessment, is illustrated in Fig. 5.7. For completeness, we note that speech quality can also by discussed solely from the viewpoint of the context of the talker [5.82], with echo and sidetone distortion being the main impairments. There are also studies on the possibility of decomposing conversational quality into listening, talking, and interaction quality, and building a prediction from these components.

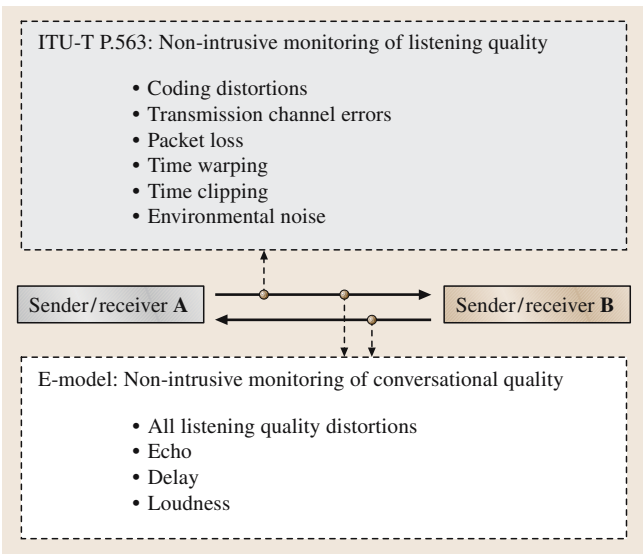



**Fig. 5.7** Non-intrusive monitoring of listening and conversational quality over the network

### 5.3.4 Discussion of Objective Measures

Objective measures of speech quality are now relatively mature, as is reflected in the standards defined by the ITU-T. These standards and a corresponding classification of the methods are presented in Fig. 5.8. This standardization indicates that the performance of these measures is satisfactory for many practical applications. In this subsection we discuss how the performance of objective quality measures can be evaluated, and provide an indication of the performance of the state-of-the-art algorithms.

The development of a computer algorithm that predicts the output of subjective quality test well is not straightforward. Complications come from the fact that speech perception is influenced by grammar rules and semantic context. Modeling of factors such as the speaker's emotional state and expectations of speech quality



Amazon EX1010 Page 126

is even more difficult. It is therefore critical that newly developed objective quality algorithms are properly *calibrated* to the output of subjective quality tests. Objective measures that generate results that closely approximate subjective test results are more useful than objective measures that show larger deviation. To facilitate the reproducibility of the evaluation process, it has to be trained over a large multilanguage database that contains a wide range of distortions, e.g., [5.83].

In the literature, two methods are commonly used to evaluate the performance of quality assessment algorithms. The first is suitable for codec or transmission equipment evaluation. The subjective MOS for the speech files within a particular test condition are first averaged together. The objective MOS are likewise grouped and averaged. Then, the correlation coefficient $R$ and the root-mean-square error (RMSE) $\varepsilon$ between the per-condition averaged subjective and the objective MOS over all the conditions in the database are calculated. Let the measured subjective quality be denoted by $Q$, and the predicted objective quality by $\hat{Q}$, then the RMSE is given by

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{L}(Q_i - \hat{Q}_i)^2}{L}} \tag{5.8}$$

and the correlation coefficient is defined as

$$R = \frac{\sum_{i=1}^{L}(\hat{Q}_i - \mu_{\hat{Q}})(Q_i - \mu_Q)}{\sqrt{\sum_{i=1}^{L}(\hat{Q}_i - \mu_{\hat{Q}})^2}\sqrt{\sum_{i=1}^{L}(Q_i - \mu_Q)^2}}\,, \tag{5.9}$$

where $\mu_Q$ and $\mu_{\hat{Q}}$ are the mean values of the introduced variables and $L$ is the number of conditions in the database.

An alternative method to evaluate the accuracy of objective quality algorithms is to calculate the correlation and RMSE between the objective and subjective MOS for each utterance. This concept is better suited for the assessment of performance in voice quality monitoring applications. The RMSE and correlation coefficient are calculated in a similar fashion as described above,

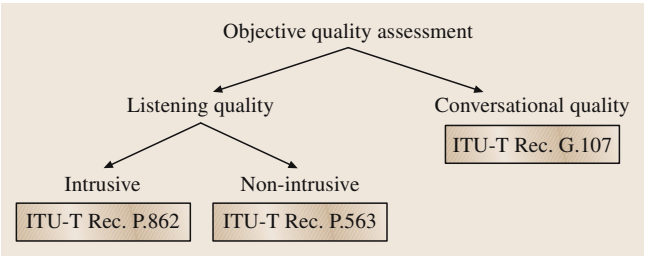



**Fig. 5.8** Classification of objective quality assessment methods and related ITU standards and recommendations

but the summation is over MOS-labeled utterances in the database, and $L$ is the number of these utterances.

One typically overlooked problem in the training and evaluation of objective quality algorithms is that most of the currently available data are for ACR-type listening tests. Thus, intrusive algorithms (e.g., [5.52, 84]) were designed to predict listening quality not based on DCR testing and with a DMOS scale, as is natural, but in terms of MOS values. To the best of our knowledge, there are no studies to assess the impact of this mismatch. This issue is not relevant to non-intrusive algorithms such as ITU-T P.563, [5.75], which are designed to predict the outcome of ACR subjective tests.

Both intrusive and nonintrusive algorithms provide a performance that is useful for practical applications. The intrusive PESQ algorithm (ITU-T Rec. P.862.2) has been observed to have a correlation coefficient $R$ of 0.85–0.95 [5.85, 86]. The nonintrusive ITU-T P.563 standard has a correlation coefficient of 0.8–0.9 [5.79, 86]. Recent work describing new nonintrusive measures reports even higher correlations [5.79].

The discussed measures of listening and conversational quality are designed to predict the speech quality from the simultaneous effect of a large number of distortions. An objective quality assessment measure can also be designed to operate in a particular environment only (e.g., specific speech coding standards in the context of a particular mobile network). These constraints can be used to obtain higher system accuracy and reduce complexity and memory requirements [5.87].

## 5.4 Conclusions

Perhaps the first question that comes to mind after reading an overview on speech quality assessment is: why is it so difficult to assess speech quality and why are so many scales and test procedures used? The explanation is that the auditory event that causes the perception of quality degradation is always

**Table 5.7** A survey of typical quality assessment problems and recommended test methodology

| *Assessment problem* | *Quality assessment method* |
|---|---|
| Subjective assessment of overall conversational quality | Absolute category rating (ACR), Conversation difficulty scale [5.3] |
| Multidimensional subjective speech quality assessment | Diagnostic acceptability measure (DAM) [5.11] |
| Subjective assessment of speech intelligibility | Diagnostic rhyme test (DRT) [5.14], Modified rhyme test (MRT) [5.15], |
| Subjective assessment of general audio quality | ITU-R BS.1284-1 [5.8] |
| Subjective assessment of high audio quality | ITU-R BS.1116-1 [5.5] |
| Subjective assessment of intermediate audio quality | MUSHRA [5.6] |
| Subjective assessment of overall listening quality of speech processing system | Absolute category rating (ACR) [5.3] |
| Subjective evaluation of speech processing algorithms of similar quality (typically for high-quality speech processing algorithms) | Degradation category rating (DCR) [5.3] |
| Subjective evaluation of systems that may increase or decrease the quality of the input speech | Comparison category rating (CCR) [5.3] |
| Subjective evaluation of threshold values of certain quantities | Quantal-response delectability [5.3] |
| Subjective evaluation of noise suppression algorithm | ITU-T Rec. P.835 [5.12], Comparison category rating (CCR) [5.3] |
| Objective assessment of listening speech quality (reference signal not available) | ITU-T Rec. P.563 [5.75] |
| Objective assessment of listening speech quality (reference signal available) | PESQ [5.55] |
| Objective assessment of audio quality (reference signal available) | PEAQ [5.54] |
| Objective assessment of conversational speech quality (network parameters available) | E-model [5.80] |

unknown and can be only projected on a particular scale(s) through the response of the test subject. Thus, we generally deal with one or more projections of speech quality, suitable for the particular application.

It is difficult to give general guidance on which assessment method is adequate for a specific quality assessment problem. Nevertheless, in Table 5.7 we make an attempt to relate some of the typical quality assessment situations with an adequate test methodology.

In line with the topic of this book, we did not cover general audiovisual quality testing in this chapter. Readers interested in this related topic are referred to ITU-T P.920 *Interactive test methods for audiovisual communications* [5.88]. In this recommendation, a five-point quality scale is suggested for assessing video, audio, and overall quality. Trends in speech quality assessment can be obtained from ITU-T study group 12 [5.89], which is the lead study group at the ITU-T on network performance and QoS.

## References


5.1 ITU-T Rec. G.113: *Transmission impairments* (Geneva 2001)

5.2 ITU-T Rec. P.11: *Effects of transmission impairments* (Geneva 1993)

5.3 ITU-T Rec. P.800: *Methods for subjective determination of transmission quality* (Geveva 1996)

5.4 D. Clark: High-resolution subjective testing using a double-blind comparator, J. Audio Eng. Soc. **30**, 330–338 (1982)

5.5 ITU-R Rec. BS.1116-1: *Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems* (Geneva 1997)


Amazon EX1010 Page 128

5.6 ITU-R Rec. BS.1534-1: *Method for the subjective assessment of intermediate quality level of coding systems* (Geneva 2005)
5.7 ITU-R Rec. BT.500-11: *Method for the subjective assessment of the quality of television pictures* (Geneva 2002)
5.8 ITU-R Rec. BS.1284-1: *General methods for the subjective assessment of sound quality* (Geneva 2003)
5.9 S. Möller: *Assessment and Prediction of Speech Quality in Telecommunications* (Kluwer Academic, Boston 2000)
5.10 M. Gueguin, R. Bouquin-Jeannes, G. Faucon, V. Barriac: Towards an objective model of the conversational speech quality, Proc. IEEE ICASSP **1**, 1229–1232 (2006)
5.11 W. Voiers: Diagnostic acceptability measure for speech communication systems, Proc. IEEE ICASSP **2**, 204–207 (1977)
5.12 ITU-T Rec. P.835: *Subjective test methodology for evaluating speech communication systems that include noise suppression algorithm* (Geneva 2003)
5.13 Y. Hu, P. Loizou: Subjective comparison of speech enhancement algorithms, Proc. IEEE ICASSP **1**, 153–156 (2006)
5.14 W. Voiers: Evaluating processed speech using the diagnostic rhyme test, Speech Technol. **1**, 338–352 (1983)
5.15 A. House, C. Williams, M. Hecker, K. Kryter: Articulation testing methods: Consonant differenctiation with a closed response set, J. Acoust. Soc. Am. **37**, 158–166 (1965)
5.16 M. Goldstein: Classification of methods used for assessment of text-to-speech systems according to the demands placed on the listener, Speech Commun. **16**, 225–244 (1995)
5.17 M. Spiegel, M. Altom, M. Macchi, K. Wallace: Comprehensive assessment of the telephone intelligibility of synthesized and natural speech, Speech Commun. **9**, 279–291 (1990)
5.18 J. van Santen: Perceptual experiments for diagnostic testing of text-to-speech systems, Comput. Speech Lang. **7**, 49–100 (1993)
5.19 ITU-T Rec. P.830: *Subjective performance assessment of telephone-band and wideband digital codecs* (Geneva 1996)
5.20 A. Huggins, R. Nickerson: Speech quality evaluation using phonemic-specific sentences, J. Acoust. Soc. Am. **77**, 1896–1906 (1985)
5.21 H. Lane, B. Tranel: The Lombard sign and the role of hearing in speech, J. Acoust. Soc. Am. **47**, 618–624 (1970)
5.22 ITU-T. Rec. P.810: *Modulated noise reference unit* (Geneva 1996)
5.23 J. Tukey: *The Problem of Multiple Comparisons* (Princeton University, Ditton 1953)
5.24 B. Jones, P. McManus: Graphic scaling of qualitative terms, SMPTE J. **95**, 1166–1171 (1986)
5.25 M. Dahlquist, A. Leijon: Paired-comparison rating of sound quality using MAP parameter estimation for data analysis. In: 1st ISCA Tutorial and Research Workshop on Auditory Quality of Systems (2003)
5.26 S. Voran: A basic experiment on time-varying speech quality. In: Proc. 4th Int. Conf.: Measurement of Speech and Audio Quality in Networks (2005)
5.27 M. Hansen, B. Kollmeier: Continuous assessment of the time-varying speech quality, J. Acoust. Soc. Am. **106**, 2888–2899 (1999)
5.28 L. Gros, N. Chateu: Instantaneous and overall judgements for time-varying speech quality: Assessments and relationships, Acta Acust. United Ac. **87**, 367–377 (2001)
5.29 L. Gros, N. Chateu, S. Busson: Effects of context on the subjective assessment of time-varying speech quality: listening/conversation, laboratory/real environment, Acta Acust. United Ac. **90**, 1037–1051 (2004)
5.30 L. Gros: The impact of listening and conversational situations on speech perceived quality for time-varying impairments. P: Int. Conf. Measurement of Speech and Audio Quality in Networks, 17–19 (2002)
5.31 J. Rosenbluth: Testing the quality of connections having time varying impairments, Comitee T1 Standards Contribution ANSI T1A1.7/98-031, (1998)
5.32 P. Gray, R. Massara, M. Hollier: An experimental investigation of the accumulation of perceived error in time-varying speech distortions. In Preprint: Audio Engineering Society 103rd Convention (1997)
5.33 ITU-T Rec. P.880: *Continuous evaluation of time varying speech quality* (Geneva 2004)
5.34 S. Quackenbush, T. Barnwell, M. Clements: *Objective Measures of Speech Quality* (Prentice Hall, Englewood Cliffs 1988)
5.35 S. Jayant, P. Noll: *Digital Coding of Waveforms* (Prentice Hall, Englewood Cliffs 1984)
5.36 M. Schroeder, B. Atal, J. Hall: *Objective Measure of Certain Speech Signal Degradations Based on Masking Properties of Human Auditory Perception* (Academic, New York 1979)
5.37 N. Kitawaki, H. Nagabuchi, K. Itoh: Objective quality evaluation for low-bit-rate speech coding systems, IEEE J. Sel. Area. Commun. **6**, 242–248 (1988)
5.38 W.B. Kleijn, K.K. Paliwal (Eds.): *Speech Coding and Synthesis* (Elsevier Science, Amsterdam 1995)
5.39 R. Viswanathan, J. Makhoul, W. Russel: Towards perceptually consistent measures of spectral distance, Proc. IEEE ICASSP **1**, 485–488 (1976)
5.40 S. Dimolitsas: Objective speech distortion measures and their relevance to speech quality measurements, IEE Proc. **136**, 317–324 (1989)
5.41 ITU-T Rec. P.56: *Objective measurement of active speech level* (Geneva 1993)
5.42 E. Zwicker, H. Fastl: *Psycho-Acoustics: Facts and Models* (Springer, New York 1999)
5.43 B.C.J. Moore: *An Introduction to the Psychology of Hearing* (Academic, London 1989)




Amazon EX1010 Page 129

5.44 T. Dau, D. Püschel, A. Kohlrausch: A quantitative model of the effective signal processing in the auditory system. I. Model structure, J. Acoust. Soc. Am. **99**, 3615–3622 (1996)

5.45 A. Rix, A. Bourret, M. Hollier: Models of human perception, BT Technol. J. **17**, 24–34 (1999)

5.46 S. Voran: A simplified version of the ITU algorithm for objective measurement of speech codec quality, Proc. IEEE ICASSP **1**, 537–540 (1998)

5.47 K. Brandenburg: Evaluation of quality for audio encoding at low bit rates. In preprint: Audio Engineering Society 82nd Convention (1987)

5.48 J. Karjalainen: A new auditory model for the evaluation of sound quality of audio systems, Proc. IEEE ICASSP **10**, 608–611 (1985)

5.49 S. Wang, A. Sekey, A. Gersho: An objective measure for predicting subjective quality of speech coders, IEEE J. Sel. Area. Commun. **10**(5), 819–829 (1992)

5.50 J. Lalou: The information index: An objective measure of speech transmission performance, Ann. Telecommun. **45**, 47–65 (1990)

5.51 R. Bouquin, G. Faucon: Using the coherence function for noise reduction, Proc. IEE **139**(3), 276–282 (1992)

5.52 J. Beerends, J. Stemerdink: A perceptual speech-quality measure based on a psychoacoustic sound representation, J. Audio Eng. Soc. **42**(3), 115–123 (1994)

5.53 C. Colomes, C. Schmidmer, T. Thiede, W. Treurniet: Perceptual quality assessment for digital audio: (PEAQ) – the new ITU standard for objective measurement of the perceived audio quality, In Proc.: AES 17th Int. Conf. 337–351 (1999)

5.54 ITU-R. BS.1387-1: *Method for Objective Measurements of Perceived Audio Quality (PEAQ)* (2001)

5.55 ITU-T Rec. P. 862: *Perceptual evaluation of speech quality (PESQ)* (2001).

5.56 ITU-R Rec. P.862.2: *Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs* (2005)

5.57 H. Knagenhjelm, W.B. Kleijn: Spectral dynamics is more important than spectral distortion, P. IEEE ICASSP **1**, 732–735 (1995)

5.58 F. Norden, T. Eriksson: Time evolution in LPC spectrum coding, IEEE T. Speech Audi. P. **12**, 290–301 (2004)

5.59 T. Quatieri, R. Dunn: Speech enhancement based on auditory spectral change, P. IEEE ICASSP **1**, 257–260 (2002)

5.60 M. Hollier, M. Hawksford, D. Guard: Error activity and error entropy as a measure of psychoacoustic significance in the perceptual domain, IEE P-Vis. Image Sign. **141**(3), 203–208 (1994)

5.61 S. Voran: Objective estimation of perceived speech quality – Part I: Development of the measuring normalizing block technique, IEEE T. Speech Audi. P. **7**(4), 371–382 (1999)

5.62 S. Voran: Objective estimation of perceived speech quality – Part II: Evaluation of the measuring normalizing block technique, IEEE T. Speech Audi. Proc. **7**(4), 383–390 (1999)

5.63 H. Coetzee, T.B. III: An LSP based speech quality measure, Proc. IEEE ICASSP **1**, 596–599 (1989)

5.64 D. Klatt: Prediction of perceived phonetic distance from critical-band spectra: a first step, Proc. IEEE ICASSP **7**, 1278–1281 (1982)

5.65 U. Halka, U. Heute: A new approach to objective quality-measures based on attribute-matching, Speech Commun. **11**, 15–30 (1992)

5.66 W. Zha, W.-Y. Chan: Objective speech quality measurement using statistical data minimg, J. Appl. Signal Process. **9**, 1410–1424 (2005)

5.67 P. Gray, M. Hollier, R. Massara: Non-intrusive speech-quality assessment using vocal-tract models, IEE P-Vis. Image Sign. **147**(6), 493–501 (2000)

5.68 J. Liang, R. Kubichek: Output-based objective speech quality, IEEE 44th Vehicular Technology Conf. **3**(8-10), 1719–1723 (1994)

5.69 H. Hermansky: Perceptual linear prediction (PLP) analysis of speech, J. Acoust. Soc. Am. **87**, 1738–1752 (1990)

5.70 A. Conway: A passive method for monitoring voice-over-IP call quality with ITU-T objective speech quality measurement methods, Proc. IEEE Int. Conf. Commun. **4**, 2583–2586 (2002)

5.71 A. Conway: Output-based method of applying PESQ to measure the perceptual quality of framed speech signals, Proc. IEEE Wireless Commun. Netw. **4**, 2521–2526 (2004)

5.72 D. Kim: ANIQUE: An auditory model for single-ended speech quality estimation, IEEE T. Speech Audi. P. **13**, 821–831 (2005)

5.73 D. Kim, A. Tarraf: Enhanced perceptual model for non-intrusive speech quality assessment, P. IEEE ICASSP **1**, 829–832 (2006)

5.74 O. Au, K. Lam: A novel output-based objective speech quality measure for wireless communication, Signal Process. P, 4th Int. Conf. **1**, 666–669 (1998)

5.75 ITU-T Rec. P.563: *Single ended method for objective speech quality assessment in narrow-band telephony applications* (2004)

5.76 T. Falk, Q. Xu, W.-Y. Chan: Non-intrusive GMM-based speech quality measurement, P. IEEE ICASSP **1**, 125–128 (2005)

5.77 G. Chen, V. Parsa: Bayesian model based non-intrusive speech quality evaluation, P. IEEE ICASSP **1**, 385–388 (2005)

5.78 D. Picovici, A. Mahdi: Output-based objective speech quality measure using self-organizing map, P. IEEE ICASSP **1**, 476–479 (2003)

5.79 V. Grancharov, D. Y. Zhao, J. Lindblom, W. B. Kleijn: Low complexity, non-intrusive speech quality assessment, IEEE Trans. Speech Audio. Process. *14*, 1948-1956 (2006)

5.80 ITU-T Rec. G.107: *The e-model, a computational model for use in transmission planning* (Geneva 2005)




Amazon EX1010 Page 130

5.81 ITU-T Rec. P. Supplement 3: *Models for predicting transmission quality from objective measurements (Withdrawn 1997)* International Telecommunication Union (1993)
5.82 R. Appel, J. Beerends: On the quality of hearing one's own voice, J. Audio Eng. Soc. **50**(4), 237–248 (2002)
5.83 ITU-T Rec. P. Supplement 23: ITU-T coded-speech database, International Telecommunication Union (1998)
5.84 A. Rix, J. Beerends, M. Hollier, A. Hekstra: Perceptual evaluation of speech quality (PESQ) – A new method for speech quality assessment of telephone networks and codecs, P. IEEE ICASSP **2**, 749–752 (2001)
5.85 T. Goldstein, A.W. Rix: Subjective comparison of speech enhancement algorithms, P. IEEE ICASSP **3**, 1064–1067 (2004)
5.86 A.W. Rix: Perceptual speech quality assessment – a review, P. IEEE ICASSP **3**, 1056–1059 (2004)
5.87 M. Werner, T. Junge, P. Vary: Quality control for AMR speech channels in GSM networks, P. IEEE ICASSP **3**, 1076–1079 (2004)
5.88 ITU-T Rec. P.920: Interactive test methods for audiovisual communications, (2000)
5.89 http://www.itu.int/ITU-T/studygroups/com12/index.asp



Amazon EX1010 Page 131

Signal Pr

# Part B

## Part B Signal Processing for Speech

Ed. by Y. Huang, J. Benesty


**6 Wiener and Adaptive Filters**
J. Benesty, Montreal, Canada
Y. Huang, Murray Hill, USA
J. Chen, Murray Hill, USA

**7 Linear Prediction**
J. Benesty, Montreal, Canada
J. Chen, Murray Hill, USA
Y. Huang, Murray Hill, USA

**8 The Kalman Filter**
S. Gannot, Ramat-Gan, Israel
A. Yeredor, Tel-Aviv, Israel

**9 Homomorphic Systems and Cepstrum Analysis of Speech**
R. W. Schafer, Palo Alto, USA

**10 Pitch and Voicing Determination of Speech with an Extension Toward Music Signals**
W. J. Hess, Bonn, Germany

**11 Formant Estimation and Tracking**
D. O'Shaughnessy, Montréal, Canada

**12 The STFT, Sinusoidal Models, and Speech Modification**
M. M. Goodwin, Scotts Valley, USA

**13 Adaptive Blind Multichannel Identification**
Y. Huang, Murray Hill, USA
J. Benesty, Montreal, Canada
J. Chen, Murray Hill, USA

# 6. Wiener and Adaptive Filters

J. Benesty, Y. Huang, J. Chen

The Wiener filter, named after its inventor, has been an extremely useful tool since its invention in the early 1930s. This optimal filter is not only popular in different aspects of speech processing but also in many other applications. This chapter presents the most fundamental results of the Wiener theory with an emphasis on the Wiener–Hopf equations, which are not convenient to solve in practice. An alternative approach to solving these equations directly is the use of an adaptive filter, which is why this work also describes the most classical adaptive algorithms that are able to converge, in a reasonable amount of time, to the optimal Wiener filter.

6.1 **Overview** ..... 103
6.2 **Signal Models** ..... 104
6.2.1 SISO Model ..... 104
6.2.2 SIMO Model ..... 105
6.2.3 MISO Model ..... 105
6.2.4 MIMO Model ..... 106
6.3 **Derivation of the Wiener Filter** ..... 106
6.4 **Impulse Response Tail Effect** ..... 107
6.5 **Condition Number** ..... 108
6.5.1 Decomposition of the Correlation Matrix ..... 108
6.5.2 Condition Number with the Frobenius Norm ..... 108
6.5.3 Fast Computation of the Condition Number ..... 110
6.6 **Adaptive Algorithms** ..... 110
6.6.1 Deterministic Algorithm ..... 110
6.6.2 Stochastic Algorithm ..... 112
6.6.3 Variable-Step-Size NLMS Algorithm 113
6.6.4 Proportionate NLMS Algorithms ..... 114
6.6.5 Sign Algorithms ..... 116
6.7 **MIMO Wiener Filter** ..... 116
6.7.1 Conditioning of the Covariance Matrix ..... 117
6.8 **Conclusions** ..... 119
**References** ..... 120

## 6.1 Overview

In his landmark manuscript on extrapolation, interpolation, and smoothing of stationary time series [6.1], Norbert Wiener was one of the first researchers to treat the filtering problem of estimating a process corrupted by additive noise. The optimum estimate that he derived, required the solution of an integral equation known as the Wiener–Hopf equation [6.2]. Soon after he published his work, Levinson formulated the same problem in discrete time [6.3]. Levinson's contribution has had a great impact on the field. Indeed, thanks to him, Wiener's ideas have become more accessible to many engineers. A very nice overview of linear filtering theory and the history of the different discoveries in this area can be found in [6.4].

In this chapter, we will show that the Wiener theory plays a fundamental role in system identification. For example, in many speech applications, impulse responses between loudspeakers (or speech sources) and microphones need to be identified. Thanks to many (adaptive) algorithms directly derived from the Wiener–Hopf equations, this task is now rather easy.

This chapter is organized as follows. Section 6.2 presents the four basic signal models used in this work. In Sect. 6.3, we derive the optimal Wiener filter for a single-input single-output (SISO) system. Section 6.4 explains what happens if the length of the modeling filter is shorter than the length of the true impulse response (this case always occurs in practice). It is extremely useful in many applications to be able to say how the input signal correlation matrix, which appears in the Wiener–Hopf equations, is conditioned. So we dedicate Sect. 6.5 to a detailed discussion on the condition number of this matrix. In Sect. 6.6, we present a collection of basic adaptive

Amazon EX1010 Page 133

filters. We insist on the classical normalized least-mean-square (NLMS) algorithm. Section 6.7 generalizes the Wiener filter to the multiple-input multiple-output (MIMO) system case. While this generalization is straightforward, the optimal solution does not always exist and identification problems may be possible only in some situations. Finally, we give our conclusions in Sect. 6.8.

## 6.2 Signal Models

In many speech applications, a system with a number of inputs and outputs needs to be identified. In this section, we explain the four basic signal models. This classification is now well accepted and is the basis of many interesting studies in different areas of control and signal processing.

### 6.2.1 SISO Model

The first model we consider is the single-input single-output (SISO) system, as shown in Fig. 6.1a. The output signal is given by

$$x(k) = h * s(k) + b(k) \,, \tag{6.1}$$

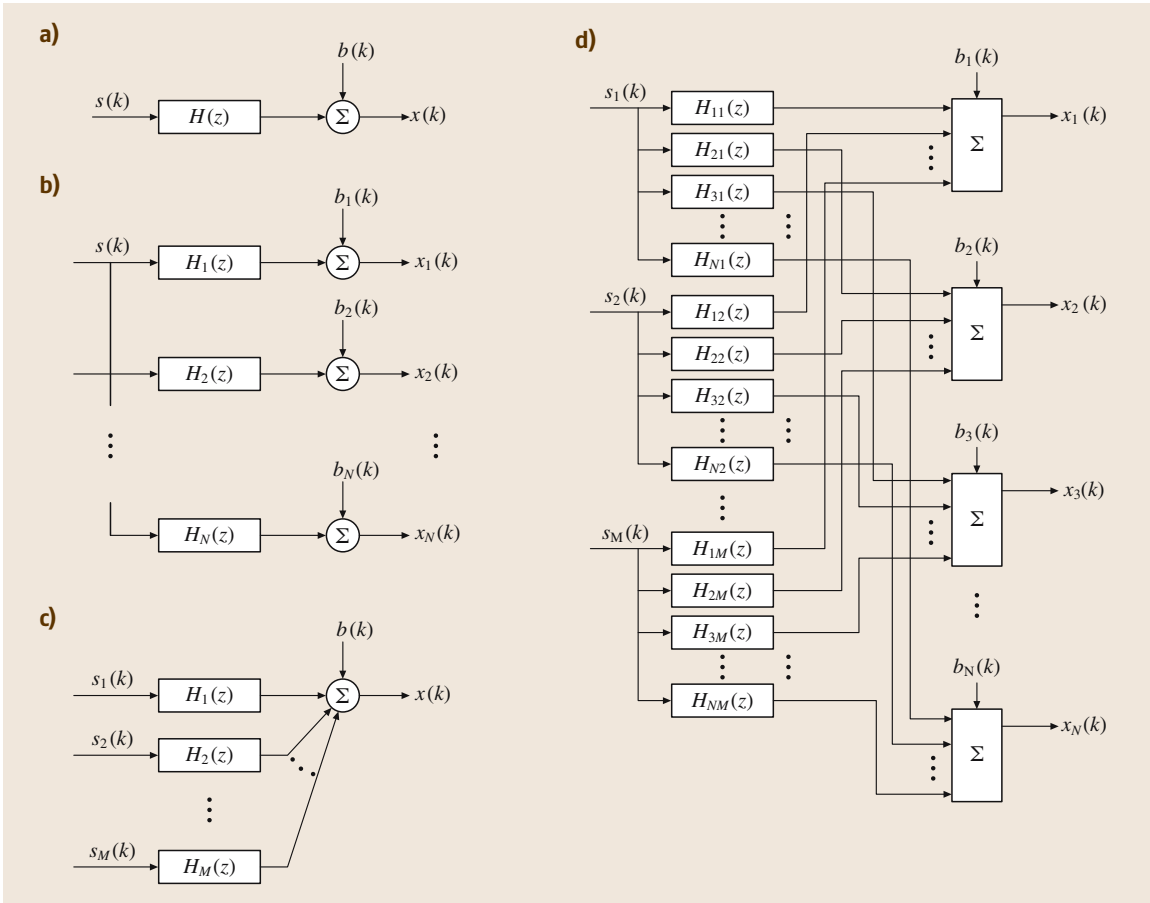



**Fig. 6.1a–d** Illustration of four distinct types of systems. **(a)** A single-input single-output (SISO) system. **(b)** A single-input multiple-output (SIMO) system. **(c)** A multiple-input single-output (MISO) system. **(d)** A multiple-input multiple-output (MIMO) system.

Amazon EX1010 Page 134

where $h$ is the channel impulse response, the symbol $*$ denotes the linear convolution operator, $s(k)$ is the source signal at time $k$, and $b(k)$ is the additive noise at the output. Here we assume that the system is linear and shift-invariant. The channel impulse response is delineated usually with a finite impulse response (FIR) filter rather than an infinite impulse response (IIR) filter. In vector/matrix form, the SISO signal model (6.1) is written as:

$$x(k) = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{s}(k) + b(k)\,, \tag{6.2}$$

where

$$\boldsymbol{h} = [h_0 \;\; h_1 \;\; \cdots \;\; h_{L-1}]^{\mathrm{T}}\,,$$

$$\boldsymbol{s}(k) = [s(k) \;\; s(k-1) \;\; \cdots \;\; s(k-L+1)]^{\mathrm{T}}\,,$$

where $[\cdot]^{\mathrm{T}}$ denotes the transpose of a matrix or a vector, and $L$ is the channel length.

Using the $z$ transform, the SISO signal model (6.2) is described as follows:

$$X(z) = H(z)S(z) + B(z)\,, \tag{6.3}$$

where $X(z)$, $S(z)$, and $B(z)$ are the $z$-transforms of $x(k)$, $s(k)$, and $b(k)$, respectively, and $H(z) = \sum_{l=0}^{L-1} h_l z^{-l}$.

The SISO model is simple and is probably the most widely used and studied model in communication, signal processing, and control theories.

### 6.2.2 SIMO Model

The diagram of a single-input multiple-output (SIMO) system is illustrated in Fig. 6.1b, in which there are $N$ outputs from the same source as input and the $n$-th output is expressed as:

$$x_n(k) = \boldsymbol{h}_n^{\mathrm{T}}\boldsymbol{s}(k) + b_n(k)\,, \quad n = 1, 2, \cdots, N\,, \tag{6.4}$$

where $x_n(k)$, $\boldsymbol{h}_n$, and $b_n(k)$ are defined in a similar way to those in (6.2), and $L$ is the length of the longest channel impulse response in this SIMO system. A more-comprehensive expression of the SIMO model is given by

$$\boldsymbol{x}(k) = \mathbf{H}\boldsymbol{s}(k) + \boldsymbol{b}(k)\,, \tag{6.5}$$

where

$$\boldsymbol{x}(k) = [x_1(k) \;\; x_2(k) \;\; \cdots \;\; x_N(k)]^{\mathrm{T}}\,,$$

$$\mathbf{H} = \begin{pmatrix} h_{1,0} & h_{1,1} & \cdots & h_{1,L-1} \\ h_{2,0} & h_{2,1} & \cdots & h_{2,L-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N,0} & h_{N,1} & \cdots & h_{N,L-1} \end{pmatrix}_{N\times L}\,,$$

$$\boldsymbol{b}(k) = [b_1(k) \;\; b_2(k) \;\; \cdots \;\; b_N(k)]^{\mathrm{T}}\,.$$

The SIMO model (6.5) is described in the $z$-transform domain as:

$$\boldsymbol{X}(z) = \boldsymbol{H}(z)S(z) + \boldsymbol{B}(z)\,, \tag{6.6}$$

where

$$\boldsymbol{X}(z) = [X_1(z) \;\; X_2(z) \;\; \cdots \;\; X_N(z)]^{\mathrm{T}}\,,$$

$$\boldsymbol{H}(z) = [H_1(z) \;\; H_2(z) \;\; \cdots \;\; H_N(z)]^{\mathrm{T}}\,,$$

$$H_n(z) = \sum_{l=0}^{L-1} h_{n,l} z^{-l}, \quad n = 1, 2, \cdots, N\,,$$

$$\boldsymbol{B}(z) = [B_1(z) \;\; B_2(z) \;\; \cdots \;\; B_N(z)]^{\mathrm{T}}\,.$$

### 6.2.3 MISO Model

In the third type of systems as drawn in Fig. 6.1c, we suppose that there are $M$ sources but only one output, whose signal is then expressed as:

$$\begin{aligned} x(k) &= \sum_{m=1}^{M} \boldsymbol{h}_m^{\mathrm{T}}\boldsymbol{s}_m(k) + b(k)\,, \\ &= \mathbf{h}^{\mathrm{T}}\mathbf{s}(k) + b(k)\,, \end{aligned} \tag{6.7}$$

where

$$\mathbf{h} = \left[\boldsymbol{h}_1^{\mathrm{T}} \;\; \boldsymbol{h}_2^{\mathrm{T}} \;\; \cdots \;\; \boldsymbol{h}_M^{\mathrm{T}}\right]^{\mathrm{T}}\,,$$

$$\boldsymbol{h}_m = [h_{m,0} \;\; h_{m,1} \;\; \cdots \;\; h_{m,L-1}]^{\mathrm{T}}\,,$$

$$\mathbf{s}(k) = \left[\boldsymbol{s}_1^{\mathrm{T}}(k) \;\; \boldsymbol{s}_2^{\mathrm{T}}(k) \;\; \cdots \;\; \boldsymbol{s}_M^{\mathrm{T}}(k)\right]^{\mathrm{T}}\,,$$

$$\boldsymbol{s}_m(k) = [s_m(k) \;\; s_m(k-1) \;\; \cdots \;\; s_m(k-L+1)]^{\mathrm{T}}\,.$$

In the $z$-transform domain, the MISO model is given by

$$X(z) = \boldsymbol{H}^{\mathrm{T}}(z)\boldsymbol{S}(z) + B(z)\,, \tag{6.8}$$

where

$$\boldsymbol{H}(z) = [H_1(z) \;\; H_2(z) \;\; \cdots \;\; H_M(z)]^{\mathrm{T}}\,,$$

$$H_m(z) = \sum_{l=0}^{L-1} h_{m,l} z^{-l}, \quad m = 1, 2, \cdots, M\,,$$

$$\boldsymbol{S}(z) = [S_1(z) \;\; S_2(z) \;\; \cdots \;\; S_M(z)]^{\mathrm{T}}\,.$$

Note that $\boldsymbol{H}(z)$ defined here is slightly different from that in (6.6). We do not deliberately distinguish them since their dimension can be easily deduced from the context if slight attention is paid.

### 6.2.4 MIMO Model

Figure 6.1d depicts a multiple-input multiple-output (MIMO) system. A MIMO system with $M$ inputs and $N$ outputs is referred to as an $M \times N$ system. At time $k$, we have

$$\boldsymbol{x}(k) = \mathbf{H}\boldsymbol{s}(k) + \boldsymbol{b}(k)\,, \tag{6.9}$$

where

$$\boldsymbol{x}(k) = [x_1(k)\ \ x_2(k)\ \ \cdots\ \ x_N(k)]^{\mathrm{T}}\,,$$

$$\mathbf{H} = (\boldsymbol{H}_1\ \ \boldsymbol{H}_2\ \ \cdots\ \ \boldsymbol{H}_M)\,,$$

$$\mathbf{H}_m = \begin{pmatrix} h_{1m,0} & h_{1m,1} & \cdots & h_{1m,L-1} \\ h_{2m,0} & h_{2m,1} & \cdots & h_{2m,L-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{Nm,0} & h_{Nm,1} & \cdots & h_{Nm,L-1} \end{pmatrix}_{N\times L}\,,$$

$$m = 1, 2, \cdots, M,$$

$$\boldsymbol{b}(k) = [b_1(k)\ \ b_2(k)\ \ \cdots\ \ b_N(k)]^{\mathrm{T}}\,,$$

where $h_{nm}$ $(n = 1, 2, \cdots, N,\ m = 1, 2, \cdots, M)$ is the impulse response of the channel from input $m$ to output $n$, and $\mathbf{s}(k)$ is defined similarly to that in (6.7). Again, we have the model presented in the $z$-transform domain as

$$\boldsymbol{X}(z) = \mathbf{H}(z)\boldsymbol{S}(z) + \boldsymbol{B}(z)\,, \tag{6.10}$$

where

$$\mathbf{H}(z) = \begin{pmatrix} H_{11}(z) & H_{12}(z) & \cdots & H_{1M}(z) \\ H_{21}(z) & H_{22}(z) & \cdots & H_{2M}(z) \\ \vdots & \vdots & \ddots & \vdots \\ H_{N1}(z) & H_{N2}(z) & \cdots & H_{NM}(z) \end{pmatrix}\,,$$

$$H_{nm}(z) = \sum_{l=0}^{L-1} h_{nm,l} z^{-l}\,, \quad n = 1, 2, \cdots, N\,,$$

$$m = 1, 2, \cdots, M\,.$$

Clearly the MIMO system is the most general model and the other three systems can be treated as special examples of a MIMO system.

## 6.3 Derivation of the Wiener Filter

In this section, we are interested in the SISO system represented by (6.2). We assume that $x(k)$ and the random noise signal $b(k)$ (independent of $s(k)$) are zero-mean and stationary.

With the Wiener theory, it is possible to identify the impulse response $\boldsymbol{h}$, given $s(k)$ and $x(k)$. Define the error signal,

$$\begin{aligned} e(k) &= x(k) - \hat{x}(k) \\ &= x(k) - \hat{\boldsymbol{h}}_{\mathrm{f}}^{\mathrm{T}} \boldsymbol{s}_{\mathrm{f}}(k)\,, \end{aligned} \tag{6.11}$$

where

$$\hat{\boldsymbol{h}}_{\mathrm{f}} = [\hat{h}_0\ \ \hat{h}_1\ \ \cdots\ \ \hat{h}_{L_{\mathrm{f}}-1}]^{\mathrm{T}}$$

is an estimate of $\boldsymbol{h}$ of length $L_{\mathrm{f}} \le L$ and

$$\boldsymbol{s}_{\mathrm{f}}(k) = [s(k)\ \ s(k-1)\ \ \cdots\ \ s(k-L_{\mathrm{f}}+1)]^{\mathrm{T}}\,.$$

To find the optimal filter, we need to minimize a cost function which is always built around the error signal (6.11). The usual choice for this criterion is the mean-square error (MSE) [6.5],

$$J(\hat{\boldsymbol{h}}_{\mathrm{f}}) = E\{e^2(k)\}\,, \tag{6.12}$$

where $E\{\cdot\}$ denotes mathematical expectation.

The optimal Wiener filter, $\hat{\boldsymbol{h}}_{\mathrm{f,o}}$, is the one that cancels the gradient of $J(\hat{\boldsymbol{h}}_{\mathrm{f}})$, i. e.,

$$\frac{\partial J(\hat{\boldsymbol{h}}_{\mathrm{f}})}{\partial \hat{\boldsymbol{h}}_{\mathrm{f}}} = \mathbf{0}_{L_{\mathrm{f}}\times 1}\,. \tag{6.13}$$

We have:

$$\begin{aligned} \frac{\partial J(\hat{\boldsymbol{h}}_{\mathrm{f}})}{\partial \hat{\boldsymbol{h}}_{\mathrm{f}}} &= 2E\left\{e(k)\frac{\partial e(k)}{\partial \hat{\boldsymbol{h}}_{\mathrm{f}}}\right\} \\ &= -2E\{e(k)\boldsymbol{s}_{\mathrm{f}}(k)\}\,. \end{aligned} \tag{6.14}$$

Therefore, at the optimum, we have:

$$E\{e_{\mathrm{o}}(k)\boldsymbol{s}_{\mathrm{f}}(k)\} = \mathbf{0}_{L_{\mathrm{f}}\times 1}\,, \tag{6.15}$$

where

$$e_{\mathrm{o}}(k) = x(k) - \hat{\boldsymbol{h}}_{\mathrm{f,o}}^{\mathrm{T}} \boldsymbol{s}_{\mathrm{f}}(k) \tag{6.16}$$

is the error signal for which $J(\hat{\boldsymbol{h}}_{\mathrm{f}})$ is minimized (i. e., the optimal filter). Expression (6.15) is called the *principle of orthogonality*.

The optimal estimate of $x(k)$ is:

$$\hat{x}_{\mathrm{o}}(k) = \hat{\boldsymbol{h}}_{\mathrm{f,o}}^{\mathrm{T}} \boldsymbol{s}_{\mathrm{f}}(k)\,. \tag{6.17}$$

It is then easy to check, with the help of the principle of orthogonality, that we also have:

$$E\left\{e_{\mathrm{o}}(k)\hat{x}_{\mathrm{o}}(k)\right\} = 0\,. \tag{6.18}$$

Amazon EX1010 Page 136

The previous expression is called the *corollary to the principle of orthogonality*.

If we substitute (6.16) into (6.15), we find the *Wiener–Hopf equations*,

$$\mathbf{R}_\mathrm{f}\hat{\boldsymbol{h}}_{\mathrm{f,o}} = \boldsymbol{p}_\mathrm{f}\,, \tag{6.19}$$

where

$$\mathbf{R}_\mathrm{f} = E\left\{\boldsymbol{s}_\mathrm{f}(k)\boldsymbol{s}_\mathrm{f}^\mathrm{T}(k)\right\}$$

is the correlation matrix of the signal $s(k)$ and

$$\boldsymbol{p}_\mathrm{f} = E\{\boldsymbol{s}_\mathrm{f}(k)x(k)\}$$

is the cross-correlation vector between $\boldsymbol{s}_\mathrm{f}(k)$ and $x(k)$.

The correlation matrix is symmetric and positive semidefinite. It is also Toeplitz, i. e., a matrix which has constant values along diagonals,

$$\mathbf{R}_\mathrm{f} = \begin{pmatrix} r(0) & r(1) & \cdots & r(L_\mathrm{f}-1) \\ r(1) & r(0) & \cdots & r(L_\mathrm{f}-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(L_\mathrm{f}-1) & r(L_\mathrm{f}-2) & \cdots & r(0) \end{pmatrix},$$

with $r(l) = E\{s(k)s(k-l)\}$, $l = 0, 1, \cdots, L_\mathrm{f}-1$. In the SISO system case, this matrix is usually positive definite even for quasistationary signals like speech; however, it can be very ill-conditioned.

Assuming that $\mathbf{R}_\mathrm{f}$ is nonsingular, the optimal Wiener filter is:

$$\hat{\boldsymbol{h}}_{\mathrm{f,o}} = \mathbf{R}_\mathrm{f}^{-1}\boldsymbol{p}_\mathrm{f}\,. \tag{6.20}$$

The MSE can be rewritten as:

$$J(\hat{\boldsymbol{h}}_\mathrm{f}) = \sigma_x^2 - 2\boldsymbol{p}_\mathrm{f}^\mathrm{T}\hat{\boldsymbol{h}}_\mathrm{f} + \hat{\boldsymbol{h}}_\mathrm{f}^\mathrm{T}\mathbf{R}_\mathrm{f}\hat{\boldsymbol{h}}_\mathrm{f}\,, \tag{6.21}$$

where $\sigma_x^2 = E\{x^2(k)\}$ is the variance of the input signal $x(k)$. The criterion $J(\hat{\boldsymbol{h}}_\mathrm{f})$ is a quadratic function of the filter coefficient vector $\hat{\boldsymbol{h}}_\mathrm{f}$ and has a single minimum point. This point combines the optimal Wiener filter, as shown above, and a value called the minimum MSE (MMSE), which is obtained by substituting (6.20) into (6.21):

$$\begin{aligned} J_\mathrm{min} &= J(\hat{\boldsymbol{h}}_{\mathrm{f,o}}) \\ &= \sigma_x^2 - \boldsymbol{p}_\mathrm{f}^\mathrm{T}\mathbf{R}_\mathrm{f}^{-1}\boldsymbol{p}_\mathrm{f} \\ &= \sigma_x^2 - \sigma_{\hat{x}_\mathrm{o}}^2\,, \end{aligned} \tag{6.22}$$

where $\sigma_{\hat{x}_\mathrm{o}}^2 = E\{\hat{x}_\mathrm{o}^2(k)\}$ is the variance of the optimal filter output signal $\hat{x}_\mathrm{o}(k)$. This MMSE can be rewritten as:

$$J_\mathrm{min} = \sigma_b^2 + \boldsymbol{h}^\mathrm{T}\mathbf{R}\boldsymbol{h} - \hat{\boldsymbol{h}}_{\mathrm{f,o}}^\mathrm{T}\mathbf{R}_\mathrm{f}\hat{\boldsymbol{h}}_{\mathrm{f,o}}\,, \tag{6.23}$$

where $\sigma_b^2 = E\{b^2(k)\}$ is the variance of the noise and $\mathbf{R} = E\{\boldsymbol{s}(k)\boldsymbol{s}^\mathrm{T}(k)\}$. The value $J_\mathrm{min}$ is bounded,

$$\sigma_b^2 \le J_\mathrm{min} \le \sigma_b^2 + \boldsymbol{h}^\mathrm{T}\mathbf{R}\boldsymbol{h}\,, \quad \forall L_\mathrm{f}\,. \tag{6.24}$$

We can easily check that for $L_\mathrm{f} = L$, $J_\mathrm{min} = \sigma_b^2$, and as $L_\mathrm{f}$ decreases compared to $L$, $J_\mathrm{min}$ gets closer to its maximum value $\sigma_b^2 + \boldsymbol{h}^\mathrm{T}\mathbf{R}\boldsymbol{h}$.

We define the normalized MMSE as:

$$\tilde{J}_\mathrm{min} = \frac{J_\mathrm{min}}{\sigma_x^2} = 1 - \frac{\sigma_{\hat{x}_\mathrm{o}}^2}{\sigma_x^2}\,. \tag{6.25}$$

According to (6.24), the normalized MMSE always satisfies,

$$\frac{\sigma_b^2}{\sigma_x^2} \le \tilde{J}_\mathrm{min} \le 1\,. \tag{6.26}$$

## 6.4 Impulse Response Tail Effect

In many scenarios, the impulse response that we try to estimate is either very long or its length is not known so that the length ($L_\mathrm{f}$) of any FIR modeling filter $\hat{\boldsymbol{h}}_\mathrm{f}$ will usually be shorter than the length ($L$) of the actual impulse response. Let us split this impulse response into two parts:

$$\mathbf{h} = \begin{pmatrix} \boldsymbol{h}_\mathrm{f} \\ \boldsymbol{h}_\mathrm{t} \end{pmatrix}$$

where $\boldsymbol{h}_\mathrm{f}$ is a vector of size $L_\mathrm{f}$ and $\boldsymbol{h}_\mathrm{t}$ is the *tail* of the impulse response that is not modeled by $\hat{\boldsymbol{h}}_\mathrm{f}$. Equation (6.2), which represents the SISO system, is now:

$$x(k) = \boldsymbol{h}_\mathrm{f}^\mathrm{T}\boldsymbol{s}_\mathrm{f}(k) + \boldsymbol{h}_\mathrm{t}^\mathrm{T}\boldsymbol{s}_\mathrm{t}(k-L_\mathrm{f}) + b(k)\,, \tag{6.27}$$

where

$$\begin{aligned} \boldsymbol{s}_\mathrm{t}(k-L_\mathrm{f}) = [&s(k-L_\mathrm{f})\;\; s(k-L_\mathrm{f}-1) \\ &\cdots\;\; s(k-L+1)]^\mathrm{T}\,. \end{aligned}$$

Substituting (6.27) into the cross-correlation vector, we obtain,

$$\boldsymbol{p}_\mathrm{f} = E\{\boldsymbol{s}_\mathrm{f}(k)x(k)\} = \mathbf{R}_\mathrm{f}\boldsymbol{h}_\mathrm{f} + \mathbf{R}_\mathrm{t}(L_\mathrm{f})\boldsymbol{h}_\mathrm{t}\,, \tag{6.28}$$

with $\mathbf{R}_\mathrm{t}(L_\mathrm{f}) = E\{\boldsymbol{s}_\mathrm{f}(k)\boldsymbol{s}_\mathrm{t}^\mathrm{T}(k-L_\mathrm{f})\}$. Finally, inserting the previous expression into the Wiener–Hopf equations (6.20), we obtain:

$$\hat{\boldsymbol{h}}_{\mathrm{f,o}} = \boldsymbol{h}_\mathrm{f} + \mathbf{R}_\mathrm{f}^{-1}\mathbf{R}_\mathrm{t}(L_\mathrm{f})\boldsymbol{h}_\mathrm{t}\,. \tag{6.29}$$

It is clear from (6.29) that the underestimation of the length of the impulse response in the Wiener method

will introduce a bias [equal to $\mathbf{R}_\mathrm{f}^{-1}\mathbf{R}_\mathrm{t}(L_\mathrm{f})\boldsymbol{h}_\mathrm{t}$] in the coefficients of the optimal filter. This bias depends on two things: the energy of the tail impulse response and the correlation of the input signal $s(k)$. If $s(k)$ is white, there is no bias since in this particular case the matrix $\mathbf{R}_\mathrm{t}(L_\mathrm{f})$ is zero. But for highly correlated signals like speech, $\mathbf{R}_\mathrm{t}(L_\mathrm{f})$ may not be negligible and the second term on the right-hand side of (6.29) may therefore be amplified if the energy of the tail is significative. As a consequence, it is important in practice to have a rough idea of the physics of the system, in order to choose an appropriate length for the modeling filter for good identification. As we can see, increasing the length of the filter will improve the accuracy of the solution. On the other hand, the complexity for solving the linear system will increase and the conditioning of $\mathbf{R}_\mathrm{f}$ will be worsened. Therefore, depending on the application, a reasonable balance has to be found.

For simplification, in the rest of this chapter, we will assume that $L_\mathrm{f} = L$ so that we can drop the subscript 'f' in all variables. In this scenario: $\mathbf{R}_\mathrm{f} = \mathbf{R}$, $\boldsymbol{s}_\mathrm{f}(k) = \boldsymbol{s}(k)$, $\hat{\boldsymbol{h}}_\mathrm{f,o} = \hat{\boldsymbol{h}}_\mathrm{o}$, etc.

## 6.5 Condition Number

The correlation matrix that appears in the Wiener–Hopf equations needs to be inverted to find the optimal filter. If this matrix is ill-conditioned and the data is perturbed, the accuracy of the solution will suffer a lot if the linear system is solved directly. One way to improve the accuracy is to regularize the covariance matrix. However, this regularization depends on the condition number: the higher the condition number, the larger the regularization. So it is important to be able to estimate this condition number in an efficient way, in order to use this information to improve the quality of the solution. Many other problems require the knowledge of this condition number for different reasons. For example, the performance of many adaptive algorithms depends on this number. Therefore, it is of great interest to have a detailed discussion of this topic here and to develop a practical algorithm to determine this condition number.

### 6.5.1 Decomposition of the Correlation Matrix

For a vector of length $L+1$,

$$\boldsymbol{s}_{L+1}(k) = [s(k)\ \ s(k-1)\ \ \cdots\ \ s(k-L)]^\mathrm{T}\,,$$

the covariance matrix of size $(L+1)\times(L+1)$ is:

$$\begin{aligned}\mathbf{R}_{L+1} &= E\left\{\boldsymbol{s}_{L+1}(k)\boldsymbol{s}_{L+1}^\mathrm{T}(k)\right\}\\ &= \begin{pmatrix} r(0) & \boldsymbol{r}_L^\mathrm{T} \\ \boldsymbol{r}_L & \mathbf{R}_L \end{pmatrix} = \begin{pmatrix} \mathbf{R}_L & \boldsymbol{r}_{\mathrm{b},L} \\ \boldsymbol{r}_{\mathrm{b},L}^\mathrm{T} & r(0) \end{pmatrix}\,, \end{aligned} \tag{6.30}$$

where $\mathbf{R}_L = E\{\boldsymbol{s}(k)\boldsymbol{s}^\mathrm{T}(k)\}$ and

$$\begin{aligned}\boldsymbol{r}_L &= [r(1)\ \ r(2)\ \ \cdots\ \ r(L)]^\mathrm{T}\,,\\ \boldsymbol{r}_{\mathrm{b},L} &= [r(L)\ \ r(L-1)\ \ \cdots\ \ r(1)]^\mathrm{T}\,.\end{aligned}$$

By using the Schur complements, it is easy to invert $\mathbf{R}_{L+1}$:

$$\mathbf{R}_{L+1}^{-1} = \begin{pmatrix} \mathbf{R}_L^{-1} + \varrho_L^{-1}\boldsymbol{b}_L\boldsymbol{b}_L^\mathrm{T} & -\varrho_L^{-1}\boldsymbol{b}_L \\ -\varrho_L^{-1}\boldsymbol{b}_L^\mathrm{T} & \varrho_L^{-1} \end{pmatrix}\,, \tag{6.31}$$

where

$$\begin{aligned}\boldsymbol{b}_L &= \mathbf{R}_L^{-1}\boldsymbol{r}_{\mathrm{b},L}\\ &= [b_{L,1}\ \ b_{L,2}\ \ \cdots\ \ b_{L,L}]^\mathrm{T}\end{aligned} \tag{6.32}$$

is the backward predictor of length $L$,

$$\begin{aligned}\varrho_L &= r(0) - \boldsymbol{r}_{\mathrm{b},L}^\mathrm{T}\boldsymbol{b}_L\\ &= r(0) - \boldsymbol{r}_L^\mathrm{T}\boldsymbol{a}_L\end{aligned} \tag{6.33}$$

is the prediction error energy, and $\boldsymbol{a}_L = \mathbf{J}_L\boldsymbol{b}_L$ is the forward predictor with $\mathbf{J}_L$ being the co-identity matrix. Equation (6.31) is important and will be used later for a fast computation of the condition number.

### 6.5.2 Condition Number with the Frobenius Norm

Usually, the condition number is computed by using the 2-norm matrix. However, in the context of Toeplitz matrices, it is more convenient to use the Frobenius norm as explained below and in [6.6, 7].

To simplify the notation, in this subsection we take $\mathbf{R}_{L+1} = \mathbf{R}$. This matrix is symmetric, positive, and assumed to be nonsingular. It can be diagonalized as follows:

$$\mathbf{Q}^\mathrm{T}\mathbf{R}\mathbf{Q} = \boldsymbol{\Lambda}\,, \tag{6.34}$$

where

$$\mathbf{Q}^\mathrm{T}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^\mathrm{T} = \mathbf{I}\,, \tag{6.35}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_{L+1}\}\,, \tag{6.36}$$

Amazon EX1010 Page 138

and $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{L+1}$. By definition, the square root of $\mathbf{R}$ is:

$$\mathbf{R}^{1/2} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{\mathrm{T}} . \tag{6.37}$$

The condition number of a matrix $\mathbf{R}$ is [6.8]:

$$\chi(\mathbf{R}) = \|\mathbf{R}\|\|\mathbf{R}^{-1}\| , \tag{6.38}$$

where $\|\cdot\|$ can be any matrix norm. Note that $\chi(\mathbf{R})$ depends on the underlying norm and subscripts will be used to distinguish the different condition numbers.

Consider the Frobenius norm:

$$\|\mathbf{R}\|_{\mathrm{F}} = \{\mathrm{tr}(\mathbf{R}^{\mathrm{T}}\mathbf{R})\}^{1/2} . \tag{6.39}$$

We can easily check that, indeed, $\|\cdot\|_{\mathrm{F}}$ is a matrix norm since for any real matrices $\mathbf{A}$ and $\mathbf{B}$ and a real scalar $c$, the following three conditions are satisfied:

- $\|\mathbf{A}\|_{\mathrm{F}} \geq 0$ and $\|\mathbf{A}\|_{\mathrm{F}} = 0$ if $\mathbf{A} = \mathbf{0}_{(L+1)\times(L+1)}$,
- $\|\mathbf{A}+\boldsymbol{B}\|_{\mathrm{F}} \leq \|\mathbf{A}\|_{\mathrm{F}} + \|\boldsymbol{B}\|_{\mathrm{F}}$,
- $\|c\mathbf{A}\|_{\mathrm{F}} = |c|\|\mathbf{A}\|_{\mathrm{F}}$.

We have:

$$\|\mathbf{R}^{1/2}\|_{\mathrm{F}} = \{\mathrm{tr}(\mathbf{R})\}^{1/2} = \left\{\sum_{l=1}^{L+1}\lambda_l\right\}^{1/2} \tag{6.40}$$

and

$$\|\mathbf{R}^{-1/2}\|_{\mathrm{F}} = \{\mathrm{tr}(\mathbf{R}^{-1})\}^{1/2} = \left\{\sum_{l=1}^{L+1}\frac{1}{\lambda_l}\right\}^{1/2} . \tag{6.41}$$

Hence, the condition number of $\mathbf{R}^{1/2}$ associated with $\|\cdot\|_{\mathrm{F}}$ is:

$$\chi_{\mathrm{F}}(\mathbf{R}^{1/2}) = \|\mathbf{R}^{1/2}\|_{\mathrm{F}}\|\mathbf{R}^{-1/2}\|_{\mathrm{F}} \geq L+1 . \tag{6.42}$$

(The inequality in the previous expression is easy to show by using the Cauchy–Schwartz inequality.) In this section, we choose to work on $\chi_{\mathrm{F}}(\mathbf{R}^{1/2})$ [rather than $\chi_{\mathrm{F}}(\mathbf{R})$], because efficient algorithms can be derived to estimate its value, as will be shown in the next subsection. As far as we know, it does not seem obvious how to estimate $\chi_{\mathrm{F}}(\mathbf{R})$ efficiently.

If $\chi(\mathbf{R}^{1/2})$ is large, then $\mathbf{R}^{1/2}$ is said to be an ill-conditioned matrix. Note that this is a norm-dependent property. However, according to [6.8], any two condition numbers $\chi_\alpha(\mathbf{R}^{1/2})$ and $\chi_\beta(\mathbf{R}^{1/2})$ are equivalent in that constants $c_1$ and $c_2$ can be found for which:

$$c_1\chi_\alpha(\mathbf{R}^{1/2}) \leq \chi_\beta(\mathbf{R}^{1/2}) \leq c_2\chi_\alpha(\mathbf{R}^{1/2}) . \tag{6.43}$$

For example, for the 1- and 2-norm matrices and for $\mathbf{R}$, we can show [6.8]

$$\frac{1}{(L+1)^2}\chi_2(\mathbf{R}) \leq \frac{1}{L+1}\chi_1(\mathbf{R}) \leq \chi_2(\mathbf{R}) . \tag{6.44}$$

We now show the same principle for the F- and 2-norm matrices and for $\mathbf{R}^{1/2}$. We recall that:

$$\chi_2(\mathbf{R}^{1/2}) = \sqrt{\frac{\lambda_{L+1}}{\lambda_1}} . \tag{6.45}$$

Since $\mathrm{tr}(\mathbf{R}^{-1}) \geq 1/\lambda_1$ and $\mathrm{tr}(\mathbf{R}) \geq \lambda_{L+1}$, we have

$$\mathrm{tr}(\mathbf{R})\mathrm{tr}(\mathbf{R}^{-1}) \geq \frac{\mathrm{tr}(\mathbf{R})}{\lambda_1} \geq \frac{\lambda_{L+1}}{\lambda_1} , \tag{6.46}$$

hence,

$$\chi_{\mathrm{F}}(\mathbf{R}^{1/2}) \geq \chi_2(\mathbf{R}^{1/2}) . \tag{6.47}$$

Also, since $\mathrm{tr}(\mathbf{R}) \leq (L+1)\lambda_{L+1}$ and $\mathrm{tr}(\mathbf{R}^{-1}) \leq (L+1)/\lambda_1$, we obtain:

$$\mathrm{tr}(\mathbf{R})\mathrm{tr}(\mathbf{R}^{-1}) \leq (L+1)\frac{\mathrm{tr}(\mathbf{R})}{\lambda_1} \leq (L+1)^2\frac{\lambda_{L+1}}{\lambda_1} , \tag{6.48}$$

thus,

$$\chi_{\mathrm{F}}(\mathbf{R}^{1/2}) \leq (L+1)\chi_2(\mathbf{R}^{1/2}) . \tag{6.49}$$

Therefore, we deduce that

$$\chi_2(\mathbf{R}^{1/2}) \leq \chi_{\mathrm{F}}(\mathbf{R}^{1/2}) \leq (L+1)\chi_2(\mathbf{R}^{1/2}) . \tag{6.50}$$

Moreover, by using the two inequalities,

$$\left(\sum_{l=1}^{L+1}\beta_l\right)^2 \geq \sum_{l=1}^{L+1}\beta_l^2 , \tag{6.51}$$

$$\left(\sum_{l=1}^{L+1}\beta_l\right)^2 \leq (L+1)\sum_{l=1}^{L+1}\beta_l^2 , \tag{6.52}$$

where $\beta_l > 0, \forall l$, it is easy to show that

$$\frac{1}{L+1}\chi_{\mathrm{F}}^2(\mathbf{R}^{1/2}) \leq \chi_{\mathrm{F}}(\mathbf{R}) \leq \chi_{\mathrm{F}}^2(\mathbf{R}^{1/2}) \leq (L+1)\chi_{\mathrm{F}}(\mathbf{R}) . \tag{6.53}$$

Note that $\chi_2(\mathbf{R}) = \chi_2^2(\mathbf{R}^{1/2})$ but $\chi_{\mathrm{F}}(\mathbf{R}) \neq \chi_{\mathrm{F}}^2(\mathbf{R}^{1/2})$. According to expressions (6.50) and (6.53), $\chi_{\mathrm{F}}(\mathbf{R}^{1/2})$ and $\chi_{\mathrm{F}}^2(\mathbf{R}^{1/2})$ are a good measure of the condition number of matrices $\mathbf{R}^{1/2}$ and $\mathbf{R}$, respectively. Basically, there is no difference in the trend of the condition numbers of $\mathbf{R}$ and $\mathbf{R}^{1/2}$. In other words, if $\mathbf{R}^{1/2}$ is ill-conditioned (resp. well-conditioned) so is $\mathbf{R}$. In the next subsection, we will show how to compute $\chi_{\mathrm{F}}^2(\mathbf{R}^{1/2})$ by using the Levinson–Durbin algorithm.

Amazon EX1010 Page 139

**Table 6.1** Computation of the condition number with the Levinson–Durbin algorithm

| | |
|---|---|
| Initialization: | $\varrho_0 = r(0)$ |
| Levinson-Durbin algorithm: | $k_l = \frac{1}{\varrho_{l-1}}\left[r(l) - \boldsymbol{a}_{l-1}^{\mathrm{T}} \boldsymbol{J}_{l-1} \boldsymbol{r}_{l-1}\right]$<br>$\boldsymbol{a}_l = \begin{pmatrix} \boldsymbol{a}_{l-1} \\ 0 \end{pmatrix} - k_l \boldsymbol{J}_l \begin{pmatrix} -1 \\ \boldsymbol{a}_{l-1} \end{pmatrix}$<br>$\varrho_l = \varrho_{l-1}(1-k_l^2)$<br>$l = 1, 2, \cdots, L$ |
| Condition number: | $\chi_{\mathrm{F}}^2\left(\mathbf{R}_{L+1}^{1/2}\right) = (L+1)r(0)\sum_{l=0}^{L} \varrho_l^{-1}\left[1 + \boldsymbol{a}_l^{\mathrm{T}} \boldsymbol{a}_l\right]$ |

### 6.5.3 Fast Computation of the Condition Number

In this subsection, we need to compute the two norms $\left\|\mathbf{R}_{L+1}^{1/2}\right\|_{\mathrm{F}}^2$ and $\left\|\mathbf{R}_{L+1}^{-1/2}\right\|_{\mathrm{F}}^2$ efficiently. The calculation of the first is straightforward. Indeed:

$$\left\|\mathbf{R}_{L+1}^{1/2}\right\|_{\mathrm{F}}^2 = \mathrm{tr}(\mathbf{R}_{L+1}) = (L+1)r(0)\,. \tag{6.54}$$

Expression (6.54) requires one multiplication only. Consider the matrix $\mathbf{G}_{L+1} = \mathbf{R}_{L+1}^{-1}$ where its diagonal elements are $g_{L+1,ii}$, $i = 1, 2, \cdots, L+1$. It is clear from (6.31) that the last diagonal component of $\mathbf{G}_{L+1}$ is $g_{L+1,(L+1)(L+1)} = \varrho_L^{-1}$. The $L$-th diagonal element of $\mathbf{G}_{L+1}$ is $g_{L+1,\mathrm{LL}} = \varrho_{L-1}^{-1} + \varrho_L^{-1} b_{L,L}^2$. Continuing the same process, we easily find:

$$g_{L+1,ii} = \varrho_{i-1}^{-1} + \sum_{l=i}^{L} \varrho_l^{-1} b_{l,i}^2\,, \tag{6.55}$$

with $\varrho_0 = r(0)$. Therefore, from (6.55) we deduce that:

$$\begin{aligned}\left\|\mathbf{R}_{L+1}^{-1/2}\right\|_{\mathrm{F}}^2 &= \mathrm{tr}(\mathbf{G}_{L+1}) \\ &= \sum_{l=0}^{L} \varrho_l^{-1}\left(1 + \boldsymbol{b}_l^{\mathrm{T}} \boldsymbol{b}_l\right) \\ &= \sum_{l=0}^{L} \varrho_l^{-1}\left(1 + \boldsymbol{a}_l^{\mathrm{T}} \boldsymbol{a}_l\right)\,,\end{aligned} \tag{6.56}$$

with $\boldsymbol{a}_0^{\mathrm{T}} \boldsymbol{a}_0 = \boldsymbol{b}_0^{\mathrm{T}} \boldsymbol{b}_0 = 0$.

Finally, the condition number is:

$$\chi_{\mathrm{F}}^2\left(\mathbf{R}_{L+1}^{1/2}\right) = (L+1)r(0)\sum_{l=0}^{L} \varrho_l^{-1}\left(1 + \boldsymbol{a}_l^{\mathrm{T}} \boldsymbol{a}_l\right)\,. \tag{6.57}$$

By using the Toeplitz structure, the Levinson–Durbin algorithm solves the linear prediction equation, $\boldsymbol{a}_L = \mathbf{R}_L^{-1} \boldsymbol{r}_L$, in $O(L^2)$ operations instead of $O(L^3)$. This algorithm computes all predictors $\boldsymbol{a}_l$, $l = 1, 2, \cdots, L$, and this is exactly what we need to compute (6.57). Expression (6.57) also shows a very nice link between the condition number and the predictors of all orders. This algorithm, which has roughly the same complexity as the Levinson–Durbin algorithm, is summarized in Table 6.1. Note that a very efficient algorithm was recently proposed by *Dias* and *Leitão* [6.9] to compute $\mathrm{tr}\{\mathbf{T}\mathbf{R}^{-1}\}$ (where $\mathbf{T}$ is a Toeplitz matrix, this form is a much more-general form than the one used in this section) with the Trench algorithm. Using these techniques here, we can further reduce the complexity [to $\mathcal{O}(L \ln L)$] for the estimation of the overall algorithm.

## 6.6 Adaptive Algorithms

Solving the Wiener–Hopf equations directly is not very practical, so adaptive algorithms are usually preferred to find the optimal Wiener filter. The aim of this section is to present a couple of basic adaptive algorithms that converge to the actual impulse response $\boldsymbol{h}$ and where the inversion of the correlation matrix $\mathbf{R}$ is avoided.

### 6.6.1 Deterministic Algorithm

The deterministic or steepest-descent algorithm is actually an iterative algorithm. It is summarized by the simple recursion,

$$\begin{aligned}\hat{\boldsymbol{h}}(k) &= \hat{\boldsymbol{h}}(k-1) + \mu[\boldsymbol{p} - \mathbf{R}\hat{\boldsymbol{h}}(k-1)]\,, \\ & k = 0, 1, 2, \cdots,\end{aligned} \tag{6.58}$$

where $\mu$ is a positive constant called the step-size parameter. In this algorithm, $\boldsymbol{p}$ and $\mathbf{R}$ are supposed to be known. The deterministic algorithm can be reformulated with the error signal:

$$e(k) = x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)\,, \tag{6.59}$$

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu E\{\boldsymbol{s}(k)e(k)\}\,. \tag{6.60}$$

Now the important question is: what are the conditions on $\mu$ to make the algorithm converge to the true impulse response $\boldsymbol{h}$? To answer this question, we will examine the *natural modes* of the algorithm [6.10].

We define the *misalignment vector* as,

$$\boldsymbol{m}(k) = \boldsymbol{h} - \hat{\boldsymbol{h}}(k)\,, \tag{6.61}$$

which is the difference between the true impulse response and the estimated one at time $k$. The positive quantity $\|\boldsymbol{m}(k)\|_2^2/\|\boldsymbol{h}\|_2^2$ is called the *normalized misalignment*. If we substitute (6.2) into the cross-correlation vector, we get,

$$\boldsymbol{p} = E\{\boldsymbol{s}(k)x(k)\} = \mathbf{R}\boldsymbol{h}\,. \tag{6.62}$$

Inserting (6.62) into (6.58) and subtracting $\boldsymbol{h}$ on both sides of the equation, we obtain:

$$\boldsymbol{m}(k) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{m}(k-1)\,, \tag{6.63}$$

where $\mathbf{I}$ is the identity matrix. Using the eigendecomposition of $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathrm{T}}$ in the previous expression, we get the equivalent form,

$$\boldsymbol{v}(k) = (\mathbf{I} - \mu\boldsymbol{\Lambda})\boldsymbol{v}(k-1)\,, \tag{6.64}$$

where

$$\boldsymbol{v}(k) = \mathbf{Q}^{\mathrm{T}}\boldsymbol{m}(k) = \mathbf{Q}^{\mathrm{T}}[\boldsymbol{h} - \hat{\boldsymbol{h}}(k)]\,. \tag{6.65}$$

Thus, for the $l$-th natural mode of the steepest-descent algorithm, we have [6.5]

$$v_l(k) = (1 - \mu\lambda_l)v_l(k-1)\,, \quad l = 1, 2, \cdots, L \tag{6.66}$$

or, equivalently,

$$v_l(k) = (1 - \mu\lambda_l)^k v_l(0)\,, \quad l = 1, 2, \cdots, L\,. \tag{6.67}$$

The algorithm converges if,

$$\lim_{k\to\infty} v_l(k) = 0\,, \quad \forall l\,. \tag{6.68}$$

In this case,

$$\lim_{k\to\infty} \hat{\boldsymbol{h}}(k) = \boldsymbol{h}\,. \tag{6.69}$$

It is straightforward to see from (6.67) that a necessary and sufficient condition for the stability of the deterministic algorithm is that,

$$-1 < 1 - \mu\lambda_l < 1\,, \quad \forall l\,, \tag{6.70}$$

which implies,

$$0 < \mu < \frac{2}{\lambda_l}\,, \quad \forall l\,, \tag{6.71}$$

or

$$0 < \mu < \frac{2}{\lambda_{\max}}\,, \tag{6.72}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

Let us evaluate the time needed for each natural mode to converge to a given value. Expression (6.67) gives:

$$\ln\frac{|v_l(k)|}{|v_l(0)|} = k\ln|1 - \mu\lambda_l|\,, \tag{6.73}$$

hence,

$$k = \frac{1}{\ln|1 - \mu\lambda_l|}\ln\frac{|v_l(k)|}{|v_l(0)|}\,. \tag{6.74}$$

The *time constant*, $\tau_l$, for the $l$-th natural mode is defined by taking $|v_l(k)|/|v_l(0)| = 1/e$ (where $e$ is the base of the natural logarithm) in (6.74). Therefore,

$$\tau_l = \frac{-1}{\ln|1 - \mu\lambda_l|}\,. \tag{6.75}$$

We can link the time constant with the condition number of the correlation matrix $\mathbf{R}$. First, let

$$\mu = \frac{\alpha}{\lambda_{\max}}\,, \tag{6.76}$$

where

$$0 < \alpha < 2\,, \tag{6.77}$$

to guaranty the convergence of the algorithm. $\alpha$ is called the normalized step-size parameter. Suppose that the smallest eigenvalue is $\lambda_1 = \lambda_{\min}$; in this case,

$$\tau_1 = \frac{-1}{\ln|1 - \alpha\lambda_{\min}/\lambda_{\max}|} = \frac{-1}{\ln|1 - \alpha/\chi_2(\mathbf{R})|}\,, \tag{6.78}$$

where $\chi_2(\mathbf{R}) = \lambda_{\max}/\lambda_{\min}$. We see that the convergence time of the slowest natural mode depends on the conditioning of $\mathbf{R}$.

Amazon EX1010 Page 141

From (6.65), we deduce that,

$$\begin{aligned}\boldsymbol{m}^{\mathrm{T}}(k)\boldsymbol{m}(k) &= \boldsymbol{v}^{\mathrm{T}}(k)\boldsymbol{v}(k)\\ &= \left\|\boldsymbol{h}-\hat{\boldsymbol{h}}(k)\right\|_2^2\\ &= \sum_{l=1}^{L}\lambda_l(1-\mu\lambda_l)^k v_l(0)\,. \end{aligned} \tag{6.79}$$

This value gives an idea on the global convergence of the filter to the true impulse response. This convergence is clearly governed by the smallest eigenvalues of $\mathbf{R}$.

We now examine the transient behavior of the MSE. Using (6.2), the error signal (6.59) can be rewritten as

$$\begin{aligned}e(k) &= x(k)-\hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)\\ &= b(k)+\boldsymbol{m}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)\,, \end{aligned} \tag{6.80}$$

so that the MSE is:

$$\begin{aligned}J(k) &= E\{e^2(k)\}\\ &= \sigma_b^2+\boldsymbol{m}^{\mathrm{T}}(k-1)\mathbf{R}\boldsymbol{m}(k-1)\\ &= \sigma_b^2+\boldsymbol{v}^{\mathrm{T}}(k-1)\boldsymbol{\Lambda}\,\boldsymbol{v}(k-1)\\ &= \sigma_b^2+\sum_{l=1}^{L}\lambda_l(1-\mu\lambda_l)^{2k-2}v_l^2(0)\,. \end{aligned} \tag{6.81}$$

A plot of $J(k)$ versus $k$ is called the *learning curve*. Note that the MSE decays exponentially. When the algorithm is convergent, we see that,

$$\lim_{k\to\infty} J(k) = \sigma_b^2\,. \tag{6.82}$$

This value corresponds to the MMSE, $J_{\min}$, obtained with the optimal Wiener filter when $L_{\mathrm{f}} = L$, which is what we assume in this section.

### 6.6.2 Stochastic Algorithm

The stochastic gradient or least-mean-square (LMS) algorithm, invented by *Widrow* and *Hoff* in the late 1950s [6.11], is certainly the most popular algorithm that we can find in the literature of adaptive filters. The popularity of the LMS is probably due to the fact that it is easy to understand, easy to implement, and robust in many respects.

One easy way to derive the stochastic gradient algorithm is by approximating the deterministic algorithm. Indeed, in practice, the two quantities $\boldsymbol{p} = E\{\boldsymbol{s}(k)x(k)\}$ and $\mathbf{R} = E\{\boldsymbol{s}(k)\boldsymbol{s}^{\mathrm{T}}(k)\}$ are in general not known. If we take their instantaneous estimates:

$$\hat{\boldsymbol{p}}(k) = \boldsymbol{s}(k)x(k)\,, \tag{6.83}$$

$$\hat{\mathbf{R}}(k) = \boldsymbol{s}(k)\boldsymbol{s}^{\mathrm{T}}(k)\,, \tag{6.84}$$

and replace them in the steepest-descent algorithm (6.58), we get:

$$\begin{aligned}\hat{\boldsymbol{h}}(k) &= \hat{\boldsymbol{h}}(k-1)+\mu[\hat{\boldsymbol{p}}(k)-\hat{\mathbf{R}}(k)\hat{\boldsymbol{h}}(k-1)]\\ &= \hat{\boldsymbol{h}}(k-1)+\mu\boldsymbol{s}(k)[x(k)-\boldsymbol{s}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}(k-1)]\,. \end{aligned} \tag{6.85}$$

This simple recursion is the LMS algorithm. Contrary to the deterministic algorithm, the LMS weight vector $\hat{\boldsymbol{h}}(k)$ is now a random vector. The three following equations summarize this algorithm [6.5],

$$\hat{x}(k) = \boldsymbol{s}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}(k-1)\,, \quad \text{filter output,} \tag{6.86}$$

$$e(k) = x(k)-\hat{x}(k)\,, \quad \text{error signal,} \tag{6.87}$$

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1)+\mu\boldsymbol{s}(k)e(k)\,, \quad \text{adaptation,} \tag{6.88}$$

which requires $2L$ additions and $2L+1$ multiplications at each iteration.

The stochastic gradient algorithm has been extensively studied and many theoretical results on its performance have been obtained [6.5, 10, 12]. In particular, we can show the convergence in the mean and mean square (see for example [6.13]), where under the independence assumption, the condition is remarkably the same as the one obtained for the deterministic algorithm, i. e.,

$$0 < \mu < \frac{2}{\lambda_{\max}}\,. \tag{6.89}$$

We can show that the asymptotic MSE for the LMS is:

$$\lim_{k\to\infty} J(k) = \sigma_b^2\left(1+\frac{\mu}{2}L\sigma_s^2\right)\,, \tag{6.90}$$

where $\sigma_s^2 = E\{s^2(k)\}$ is the variance of the input signal $s(k)$. If we compare (6.90) with the asymptotic MSE for the steepest-descent algorithm (6.82), we notice that a positive term,

$$J_{\mathrm{ex}}(\infty) = \frac{\mu}{2}L\sigma_s^2\sigma_b^2\,, \tag{6.91}$$

is added, called the *excess mean-square error*. This term, of course, has a negative effect on the final MSE and its appearance is due to the approximation discussed at the beginning of this subsection. We can reduce its effect by taking a very small $\mu$, but this will increase the convergence time of the LMS. This tradeoff between fast convergence and increased MSE is a well-known effect and is something to consider in any practical implementation.

A simple condition for the stability of LMS is that,

$$|\varepsilon(k)| < |e(k)|\,, \tag{6.92}$$

**Table 6.2** The normalized LMS (NLMS) algorithm

| | |
|---|---|
| Initialization: | $\hat{\boldsymbol{h}}(0) = \mathbf{0}_{L\times 1}$ |
| Parameters: | $0 < \alpha < 2$<br>$\delta > 0$ |
| Error: | $e(k) = x(k) - \boldsymbol{s}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}(k-1)$ |
| Update: | $\mu(k) = \dfrac{\alpha}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)+\delta}$<br>$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu(k)\boldsymbol{s}(k)e(k)$ |

where

$$\varepsilon(k) = x(k) - \boldsymbol{s}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}(k) \tag{6.93}$$

is the a posteriori error signal, computed after the filter is updated. This makes sense intuitively since $\varepsilon(k)$ contains more meaningful information than $e(k)$.

This condition is necessary for the LMS to converge to the true impulse response but not sufficient. However, it is very useful to use here and in many other algorithms to find the bounds for the step size $\mu$.

Inserting (6.88) into (6.93) and using the condition (6.92), we find:

$$0 < \mu < \frac{2}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} . \tag{6.94}$$

For $L$ large, $\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k) = L\sigma_s^2 = \mathrm{tr}(\mathbf{R})$. On the other hand, $\mathrm{tr}(\mathbf{R}) = \sum_{l=1}^{L}\lambda_l$ and this implies that $\mathrm{tr}(\mathbf{R}) \geq \lambda_{\max}$. Hence,

$$0 < \mu < \frac{2}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} \leq \frac{2}{\lambda_{\max}} . \tag{6.95}$$

If we now introduce the normalized step size $\alpha$ ($0 < \alpha < 2$), as we did in the previous subsection, the step size of the LMS will vary with time as follows,

$$\mu(k) = \frac{\alpha}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} , \tag{6.96}$$

and the LMS becomes the normalized LMS (NLMS):

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \frac{\alpha\boldsymbol{s}(k)e(k)}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} . \tag{6.97}$$

This algorithm is extremely helpful in practice, especially with nonstationary signals, since $\mu(k)$ can adjust itself at each new iteration. In order to avoid numerical difficulties when the energy of the input signal is small, we regularize the algorithm,

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \frac{\alpha\boldsymbol{s}(k)e(k)}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)+\delta} , \tag{6.98}$$

where $\delta > 0$ is the regularization factor. Table 6.2 summarizes this very important algorithm. (Note that the definition of $\mu(k)$ in this table is slightly modified in order to include the regularization parameter $\delta$.)

### 6.6.3 Variable-Step-Size NLMS Algorithm

The stability of the NLMS algorithm is governed by a step-size parameter. As already discussed, the choice of this parameter, within the stability conditions, reflects a tradeoff between fast convergence and good tracking ability on the one hand, and low misadjustment on the other hand. To meet these conflicting requirements, the step size needs to be controlled. While the formulation of this problem is straightforward, a good and reliable solution is not that easy to find. Many different schemes have been proposed in the last two decades [6.14–21]. In this subsection, we show how to derive in a very simple and elegant way a nonparametric variable-step-size NLMS algorithm.

We define the a priori and a posteriori error signals as, respectively,

$$\begin{aligned} e(k) &= x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\mathbf{s}(k) \\ &= \boldsymbol{s}^{\mathrm{T}}(k)[\boldsymbol{h} - \hat{\boldsymbol{h}}(k-1)] + b(k) , \end{aligned} \tag{6.99}$$

$$\begin{aligned} \varepsilon(k) &= x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k)\mathbf{s}(k) \\ &= \boldsymbol{s}^{\mathrm{T}}(k)[\boldsymbol{h} - \hat{\boldsymbol{h}}(k)] + b(k) . \end{aligned} \tag{6.100}$$

Consider the linear update equation:

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu(k)\boldsymbol{s}(k)e(k) . \tag{6.101}$$

One reasonable way to derive a $\mu(k)$ that makes (6.101) stable is to cancel the a posteriori error signal ([6.22] and references therein). Replacing (6.101) in (6.100) with the requirement $\varepsilon(k) = 0$ we easily find, assuming $e(k) \neq 0, \forall k$, that,

$$\mu_{\mathrm{NLMS}}(k) = \frac{1}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} . \tag{6.102}$$

Therefore, the obtained algorithm is the classical NLMS.

While this procedure makes sense in the absence of noise, finding the $\mu(k)$, in the presence of noise, that cancels (6.100) will introduce noise in $\hat{\boldsymbol{h}}(k)$ since

**Table 6.3** The nonparametric VSS-NLMS (NPVSS-NLMS) algorithm

| | |
|---|---|
| Initialization: | $\hat{\boldsymbol{h}}(0) = \boldsymbol{0}$ |
| | $\hat{\sigma}_e^2(0) = 0$ |
| Parameters: | $\lambda = 1 - \frac{1}{KL}$, exponential window with $K \geq 2$<br>$\sigma_b^2$, noise power known or estimated<br>$\delta = \text{cst} \cdot \sigma_s^2$, regularization<br>$\epsilon > 0$, very small number to avoid division by zero |
| Error: | $e(k) = x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)$ |
| Update: | $\hat{\sigma}_e^2(k) = \lambda\hat{\sigma}_e^2(k-1) + (1-\lambda)e^2(k)$<br>$\varsigma(k) = \left[\delta + \boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)\right]^{-1}\left[1 - \frac{\sigma_b}{\epsilon + \hat{\sigma}_e(k)}\right]$<br>$\mu_{\text{NPVSS}}(k) = \begin{cases} \varsigma(k) & \text{if } \hat{\sigma}_e(k) \geq \sigma_b \\ 0 & \text{otherwise} \end{cases}$<br>$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu_{\text{NPVSS}}(k)\boldsymbol{s}(k)e(k)$ |

$\boldsymbol{s}^{\mathrm{T}}(k)(\boldsymbol{h} - \hat{\boldsymbol{h}}(k)) = -b(k) \neq 0$, $\forall k$. What we would like, in fact, is to have $\boldsymbol{s}^{\mathrm{T}}(k)(\boldsymbol{h} - \hat{\boldsymbol{h}}(k)) = 0$, $\forall k$, which implies that $\varepsilon(k) = b(k)$. Hence, in this procedure we wish to find the step-size parameter $\mu(k)$ in such a way that

$$E\{\varepsilon^2(k)\} = \sigma_b^2 \,, \quad \forall k \,. \tag{6.103}$$

Using the approximation $\mathbf{s}^{\mathrm{T}}(k)\mathbf{s}(k) = L\sigma_s^2$ for $L \gg 1$, knowing that $\mu(k)$ is deterministic in nature, substituting (6.101) into (6.100), using (6.99) to eliminate $\hat{\mathbf{h}}(k-1)$, and equating to (6.103), we find:

$$\begin{aligned} E\{\varepsilon^2(k)\} &= [1 - \mu(k)L\sigma_s^2]^2\sigma_e^2(k) \\ &= \sigma_b^2 \,, \end{aligned} \tag{6.104}$$

where $\sigma_e^2(k) = E\{e^2(k)\}$ is the power of the error signal. Developing (6.104), we obtain a quadratic equation,

$$\mu^2(k) - \frac{2}{L\sigma_s^2}\mu(k) + \frac{1}{\left(L\sigma_s^2\right)^2}\left[1 - \frac{\sigma_b^2}{\sigma_e^2(k)}\right] = 0 \,, \tag{6.105}$$

for which the obvious solution is,

$$\begin{aligned} \mu_{\text{NPVSS}}(k) &= \frac{1}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)}\left[1 - \frac{\sigma_b}{\sigma_e(k)}\right] \\ &= \mu_{\text{NLMS}}(k)\alpha(k) \,, \end{aligned} \tag{6.106}$$

where $\alpha(k)$ $[0 \leq \alpha(k) \leq 1]$ is the normalized step size. Therefore, the nonparametric VSS-NLMS (NPVSS-NLMS) algorithm is [6.23],

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu_{\text{NPVSS}}(k)\boldsymbol{s}(k)e(k) \,, \tag{6.107}$$

where $\mu_{\text{NPVSS}}(k)$ is defined in (6.106).

We see from (6.106) that, before the algorithm converges, $\sigma_e(k)$ is large compared to $\sigma_b$, thus $\mu_{\text{NPVSS}}(k) \approx \mu_{\text{NLMS}}(k)$. On the other hand, when the algorithm starts to converge to the true solution, $\sigma_e(k) \approx \sigma_b$ and $\mu_{\text{NPVSS}}(k) \approx 0$. This is exactly what we desire in order to have both good convergence and low misadjustment. As we can notice, this approach was derived with almost no assumptions compared to all other algorithms belonging to the same family. Table 6.3 summarizes a practical version of the NPVSS-NLMS algorithm.

### 6.6.4 Proportionate NLMS Algorithms

In this subsection, we explain two very useful algorithms: the proportionate NLMS (PNLMS) and improved PNLMS (IPNLMS) algorithms.

It is well known that the NLMS algorithm converges and tracks slowly, especially for long impulse responses. In many situations where an adaptive algorithm is required, convergence and tracking are critical for a good performance of the entire system. While in the NLMS, the adaptation step is the same for all components of the filter, in the PNLMS [6.24], an adaptive individual step size is assigned to each filter coefficient. The step sizes are calculated from the last estimate of the filter coefficients in such a way that a larger coefficient receives a larger increment, thus increasing the convergence rate of that coefficient. This has the effect that active coefficients are adjusted faster than inactive coefficients (i. e., small or zero coefficients). Hence, PNLMS converges much faster than NLMS for sparse impulse responses. Unfortunately, PNLMS behaves much worse than NLMS when the impulse response is not sparse. This problem is due to the fact that the proportionate update is not very well refined. In [6.25], an IPNLMS was proposed where the adaptive individual step size has a better balance between the fixed step size of NLMS and

the large amount of proportionality in PNLMS. As a result, IPNLMS always converges and tracks better than NLMS and PNLMS, no matter how sparse the impulse response.

The error signal and the coefficient update equation of the two previously discussed algorithms can be written as:

$$e(k) = x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)\,, \tag{6.108}$$

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \frac{\alpha \mathbf{G}(k-1)\boldsymbol{s}(k)e(k)}{\boldsymbol{s}^{\mathrm{T}}(k)\mathbf{G}(k-1)\boldsymbol{s}(k) + \delta}\,, \tag{6.109}$$

where

$$\mathbf{G}(k-1) = \mathrm{diag}\{g_0(k-1)\ \ g_1(k-1) \cdots\ \ g_{L-1}(k-1)\} \tag{6.110}$$

is a diagonal matrix that adjusts the step sizes of the individual taps of the filter, $\alpha$ ($0 < \alpha < 2$) is the overall step-size factor, and $\delta$ is the regularization parameter.

The NLMS algorithm is obtained by taking:

$$\mathbf{G}(k) = \mathbf{I}\,, \tag{6.111}$$

$$\delta = \delta_{\mathrm{NLMS}} = \mathrm{cst}\sigma_s^2\,, \tag{6.112}$$

where $\mathbf{I}$, $\sigma_s^2$, and cst are the identity matrix, the power of the signal $s(k)$, and a small positive constant, respectively.

In the PNLMS, the diagonal elements of $\mathbf{G}(k) = \mathbf{G}_{\mathrm{p}}(k)$ are calculated as follows [6.24]:

$$\gamma_{\mathrm{p},l}(k) = \max\left\{\rho \max\left[\delta_{\mathrm{p}}, \left|\hat{h}_0(k)\right|, \cdots, \left|\hat{h}_{L-1}(k)\right|\right], \left|\hat{h}_l(k)\right|\right\}\,, \tag{6.113}$$

$$g_{\mathrm{p},l}(k) = \frac{\gamma_{\mathrm{p},l}(k)}{\|\boldsymbol{\gamma}_{\mathrm{p}}(k)\|_1}, \quad 0 \le l \le L-1\,, \tag{6.114}$$

where

$$\boldsymbol{\gamma}_{\mathrm{p}}(k) = [\gamma_{\mathrm{p},0}(k)\ \ \gamma_{\mathrm{p},1}(k)\ \cdots\ \gamma_{\mathrm{p},L-1}(k)]^{\mathrm{T}}\,.$$

The parameters $\delta_{\mathrm{p}}$ and $\rho$ are positive numbers with typical values $\delta_{\mathrm{p}} = 0.01$, $\rho = 0.01$. The first term in (6.113), $\rho$, prevents $\hat{h}_l(k)$ from stalling when its magnitude is much smaller than the magnitude of the largest coefficient and $\delta_{\mathrm{p}}$ regularizes the updating when all coefficients are zero at initialization. For the regularization parameter, we usually choose:

$$\delta_{\mathrm{PNLMS}} = \delta_{\mathrm{NLMS}}/L\,. \tag{6.115}$$

For the IPNLMS algorithm, the diagonal matrix, $\mathbf{G}(k) = \mathbf{G}_{\mathrm{ip}}(k)$, is computed in a more-elegant way [6.25]:

$$\gamma_{\mathrm{ip},l}(k) = (1-\beta)\frac{\|\hat{\boldsymbol{h}}(k)\|_1}{L} + (1+\beta)\left|\hat{h}_l(k)\right|\,, \tag{6.116}$$

$$g_{\mathrm{ip},l}(k) = \frac{\gamma_{\mathrm{ip},l}(k)}{\|\boldsymbol{\gamma}_{\mathrm{ip}}(k)\|_1} = \frac{1-\beta}{2L} + (1+\beta)\frac{\left|\hat{h}_l(k)\right|}{2\|\hat{\boldsymbol{h}}(k)\|_1}\,, \quad 0 \le l \le L-1\,, \tag{6.117}$$

where $\beta$ ($-1 \le \beta < 1$) is a parameter that controls the amount of proportionality in the IPNLMS. For $\beta = -1$, it can easily be checked that the IPNLMS and NLMS algorithms are identical. For $\beta$ close to 1, IPNLMS behaves like PNLMS. In practice, a good choice for $\beta$ is $-0.5$ or 0. With this choice and in simulations, IPNLMS always performs better than NLMS and PNLMS. As for the regularization parameter, it should be taken as:

$$\delta_{\mathrm{IPNLMS}} = \frac{1-\beta}{2L}\delta_{\mathrm{NLMS}}\,. \tag{6.118}$$

The IPNLMS algorithm is summarized in Table 6.4.

**Table 6.4** The improved PNLMS (IPNLMS) algorithm

| | |
|---|---|
| Initialization: | $\hat{h}_l(0) = 0, l = 0, 1, \cdots, L-1$ |
| Parameters: | $-1 \le \beta < 1$<br>$0 < \alpha < 2$<br>$\delta_{\mathrm{IPNLMS}} = \mathrm{cst}\cdot\sigma_s^2\frac{1-\beta}{2L}$<br>$\epsilon > 0$, very small number to avoid division by zero |
| Error: | $e(k) = x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}(k-1)\boldsymbol{s}(k)$ |
| Update: | $g_{\mathrm{ip},l}(k-1) = \frac{1-\beta}{2L} + (1+\beta)\frac{\left\|\hat{h}_l(k-1)\right\|}{2\left\|\hat{\boldsymbol{h}}(k-1)\right\|_1 + \epsilon}$<br>$\mu(k) = \frac{\alpha}{\sum_{j=0}^{L-1} s^2(k-j)g_{\mathrm{ip},j}(k-1) + \delta_{\mathrm{IPNLMS}}}$<br>$\hat{h}_l(k) = \hat{h}_l(k-1) + \mu(k)g_{\mathrm{ip},l}(k-1)s(k-l)e(k)$<br>$l = 0, 1, \cdots, L-1$ |

Before finishing this subsection, it is worth mentioning another variant of PNLMS, called PNLMS++ [6.26]. In this algorithm, the adaptation of the filter coefficients alternates between NLMS and PNLMS; as a result, PNLMS++ seems slightly less sensitive to the assumption of a sparse impulse response than PNLMS.

### 6.6.5 Sign Algorithms

Up to now, the only cost function that we have used has been the MSE. What makes this criterion so interesting is that an optimal solution (Wiener) can easily be derived as well as very powerful adaptive algorithms. An alternative to the MSE is the mean absolute error (MAE) [6.27],

$$J_{\mathrm{a}}(\hat{\boldsymbol{h}}) = E\{|e(k)|\} = E\{|x(k) - \hat{\boldsymbol{h}}^{\mathrm{T}}\boldsymbol{s}(k)|\} \,. \tag{6.119}$$

The gradient of this cost function is:

$$\frac{\partial J_{\mathrm{a}}(\hat{\boldsymbol{h}})}{\partial \hat{\boldsymbol{h}}} = -E\{\boldsymbol{s}(k)\mathrm{sgn}[e(k)]\} \,, \tag{6.120}$$

where

$$\mathrm{sgn}[e(k)] = \frac{e(k)}{|e(k)|} \,. \tag{6.121}$$

From the instantaneous value of the gradient of $J_{\mathrm{a}}(\hat{\boldsymbol{h}})$, we can derive the sign-error adaptive filter:

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu_{\mathrm{a}}\boldsymbol{s}(k)\mathrm{sgn}[e(k)] \,, \tag{6.122}$$

where $\mu_{\mathrm{a}}$ is the adaptation step of the algorithm. This algorithm is simplified compared to the LMS since the $L$ multiplications in the update equation are replaced by a sign change of the components of the signal vector $\boldsymbol{s}(k)$. Using the stability condition, $|\varepsilon(k)| < |e(k)|$, we deduce that:

$$0 < \mu_{\mathrm{a}} < \frac{2|e(k)|}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)} \,. \tag{6.123}$$

Another way to simplify the LMS filter is to replace $s(k)$ with its sign. We get the sign–data algorithm:

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu'_{\mathrm{a}}\mathrm{sgn}[\boldsymbol{s}(k)]e(k) \,, \tag{6.124}$$

where $\mu'_{\mathrm{a}}$ is the adaptation step of the algorithm and the stability condition is:

$$0 < \mu'_{\mathrm{a}} < \frac{2}{\boldsymbol{s}^{\mathrm{T}}(k)\mathrm{sgn}[\boldsymbol{s}(k)]} \,. \tag{6.125}$$

Combining the two previous approaches, we derive the sign–sign algorithm:

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}(k-1) + \mu''_{\mathrm{a}}\mathrm{sgn}[\boldsymbol{s}(k)]\mathrm{sgn}[e(k)] \,, \tag{6.126}$$

for which the stability condition is:

$$0 < \mu''_{\mathrm{a}} < \frac{2|e(k)|}{\boldsymbol{s}^{\mathrm{T}}(k)\mathrm{sgn}[\boldsymbol{s}(k)]} \,. \tag{6.127}$$

The algorithms derived in this subsection are very simple to implement and can be very useful in some applications. However, their convergence rate is usually slower than the LMS and their excess MSE is higher [6.28–30].

## 6.7 MIMO Wiener Filter

In this section, we consider a MIMO system with $M$ inputs and $N$ outputs (see Sect. 6.2 for more details):

$$\boldsymbol{x}(k) = \mathbf{H}\boldsymbol{s}(k) + \boldsymbol{b}(k) \,, \tag{6.128}$$

where

$$x_n(k) = \sum_{m=1}^{M} \boldsymbol{h}_{nm}^{\mathrm{T}}\boldsymbol{s}_m(k) + b_n(k) = \boldsymbol{h}_{n:}^{\mathrm{T}}\boldsymbol{s}(k) + b_n(k), \quad n = 1, 2, \cdots, N \,, \tag{6.129}$$

and

$$\mathbf{H} = \begin{pmatrix} \boldsymbol{h}_{11}^{\mathrm{T}} & \boldsymbol{h}_{12}^{\mathrm{T}} & \cdots & \boldsymbol{h}_{1M}^{\mathrm{T}} \\ \boldsymbol{h}_{21}^{\mathrm{T}} & \boldsymbol{h}_{22}^{\mathrm{T}} & \cdots & \boldsymbol{h}_{2M}^{\mathrm{T}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{h}_{N1}^{\mathrm{T}} & \boldsymbol{h}_{N2}^{\mathrm{T}} & \cdots & \boldsymbol{h}_{NM}^{\mathrm{T}} \end{pmatrix}_{N\times ML} = \begin{pmatrix} \boldsymbol{h}_{1:}^{\mathrm{T}} \\ \boldsymbol{h}_{2:}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{h}_{N:}^{\mathrm{T}} \end{pmatrix} \,. \tag{6.130}$$

We define the error signal at time $k$ at the $n$-th output as:

$$e_n(k) = x_n(k) - \hat{x}_n(k) = x_n(k) - \hat{\boldsymbol{h}}_{n:}^{\mathrm{T}}\boldsymbol{s}(k) \,, \quad n = 1, 2, \cdots, N \,, \tag{6.131}$$

Amazon EX1010 Page 146

**Table 6.5** The MISO NLMS algorithm

| | |
|---|---|
| Initialization: | $\hat{\boldsymbol{h}}_{n:}(0) = \mathbf{0}_{ML\times 1}$ |
| Parameters: | $0 < \alpha < 2$ |
| | $\delta > 0$ |
| Error: | $e_n(k) = x_n(k) - \boldsymbol{s}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}_{n:}(k-1)$ |
| Update: | $\mu(k) = \dfrac{\alpha}{\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{s}(k)+\delta}$ |
| | $\hat{\boldsymbol{h}}_{n:}(k) = \hat{\boldsymbol{h}}_{n:}(k-1) + \mu(k)\boldsymbol{s}(k)e_n(k)$ |

where $\hat{\boldsymbol{h}}_{n:}$ is an estimate of $\boldsymbol{h}_{n:}$. It is more convenient to define an error signal vector for all outputs:

$$\begin{aligned}\boldsymbol{e}(k) &= \boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k)\\ &= \boldsymbol{x}(k) - \hat{\mathbf{H}}\boldsymbol{s}(k)\,,\end{aligned} \tag{6.132}$$

where $\hat{\boldsymbol{H}}$ is an estimate of $\boldsymbol{H}$ and

$$\boldsymbol{e}(k) = [e_1(k)\;\; e_2(k)\;\; \cdots\;\; e_N(k)]^{\mathrm{T}}\,.$$

Having written the error signal, we now define the MIMO MSE with respect to the modeling filters as:

$$\begin{aligned}J(\hat{\mathbf{H}}) &= E\{\boldsymbol{e}^{\mathrm{T}}(k)\boldsymbol{e}(k)\}\\ &= \sum_{n=1}^{N} E\left\{e_n^2(k)\right\} = \sum_{n=1}^{N} J_n(\hat{\boldsymbol{h}}_{n:})\,.\end{aligned} \tag{6.133}$$

The minimization of (6.133) leads to the MIMO Wiener–Hopf equations:

$$\mathbf{R}_{ss}\hat{\mathbf{H}}_{\mathrm{o}}^{\mathrm{T}} = \mathbf{P}_{sx}\,, \tag{6.134}$$

where

$$\begin{aligned}\mathbf{R}_{ss} &= E\{\boldsymbol{s}(k)\boldsymbol{s}^{\mathrm{T}}(k)\}\\ &= \begin{pmatrix}\mathbf{R}_{s_1s_1} & \mathbf{R}_{s_1s_2} & \cdots & \mathbf{R}_{s_1s_M}\\ \mathbf{R}_{s_2s_1} & \mathbf{R}_{s_2s_2} & \cdots & \mathbf{R}_{s_2s_M}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{R}_{s_Ms_1} & \mathbf{R}_{s_Ms_2} & \cdots & \mathbf{R}_{s_Ms_M}\end{pmatrix}\end{aligned} \tag{6.135}$$

is the input signal covariance matrix (which has a block-Toeplitz structure) with $\mathbf{R}_{s_ms_i} = E\{\boldsymbol{s}_m(k)\boldsymbol{s}_i^{\mathrm{T}}(k)\}$, and

$$\begin{aligned}\mathbf{P}_{sx} &= E\{\boldsymbol{s}(k)\boldsymbol{x}^{\mathrm{T}}(k)\}\\ &= (\boldsymbol{p}_{sx_1}\;\; \boldsymbol{p}_{sx_2}\;\; \cdots\;\; \boldsymbol{p}_{sx_N})\end{aligned} \tag{6.136}$$

is the cross-correlation matrix between the input and output signals, with $\mathbf{p}_{sx_n} = E\{\boldsymbol{s}(k)x_n(k)\}$.

It can easily be seen that the MIMO Wiener–Hopf equations (6.134) can be decomposed into $N$ independent MISO Wiener–Hopf equations,

$$\mathbf{R}_{ss}\hat{\boldsymbol{h}}_{n:,\mathrm{o}} = \boldsymbol{p}_{sx_n}\,,\quad n = 1, 2, \cdots, N\,, \tag{6.137}$$

each one corresponding to an output signal of the system. In other words, minimizing $J(\hat{\mathbf{H}})$ or minimizing each $J_n(\hat{\boldsymbol{h}}_{n:})$ independently gives exactly the same results from an identification point of view. This observation is very important from a practical point of view when adaptive algorithms need to be designed. Indeed, any MIMO adaptive filter is simplified to $N$ MISO adaptive filters. As an example, we give the MISO NLMS algorithm in Table 6.5. We deduce from this discussion that, obviously, the identification of a SIMO system is equivalent to the identification of $N$ independent SISO systems. As a result, with a reference signal, the identification of any acoustic system simplifies to the identification of SISO or MISO systems.

### 6.7.1 Conditioning of the Covariance Matrix

The best possible case for the identification of a MISO system is when the input signals $s_m(k)$, $m = 1, 2, \cdots, M$, are uncorrelated. In this scenario, we have:

$$\mathbf{R}_{s_ms_i} = \mathbf{0}_{L\times L}\,,\quad \forall m, i = 1, 2, \cdots, M,\; m \neq i\,, \tag{6.138}$$

and the input signal covariance matrix $\mathbf{R}_{ss}$ is block-diagonal. Therefore, if $\mathbf{R}_{s_ms_m}$, $m = 1, 2, \cdots, M$, are nonsingular and well conditioned, the impulse responses of the MISO system are easy to estimate. This case, however, does not often occur in practice so it is of little interest.

The worst possible case, from an identification point of view, is when the signals $s_m(k)$ are generated from a unique source $s_{\mathrm{s}}(k)$, i. e.,

$$s_m(k) = \boldsymbol{g}_m^{\mathrm{T}}s_{\mathrm{s}}(k)\,,\quad m = 1, 2, \cdots, M\,, \tag{6.139}$$

where

$$\boldsymbol{g}_m = [g_{m,0}\;\; g_{m,1}\;\; \cdots\;\; g_{m,L-1}]^{\mathrm{T}}$$

is the impulse response between the source $s_{\mathrm{s}}(k)$ and the signal $s_m(k)$. In this scenario, it can be shown that matrix $\mathbf{R}_{ss}$ is rank-deficient by, at least, $(M-2)L+1$.

As a matter of fact, from (6.139), the input signal vector $\boldsymbol{s}_{\mathrm{s}}(k)$ can be written as

$$\begin{aligned}\mathbf{s}(k) &= \left[\boldsymbol{s}_1^{\mathrm{T}}(k)\ \boldsymbol{s}_2^{\mathrm{T}}(k)\ \cdots\ \boldsymbol{s}_M^{\mathrm{T}}(k)\right]^{\mathrm{T}} \\ &= \mathbf{G}\boldsymbol{s}_{\mathrm{s}}(k) \\ &= \begin{pmatrix} g_{1,0} & g_{1,1} & \cdots & g_{1,L-1} & 0 & 0 & \cdots & 0 \\ 0 & g_{1,0} & g_{1,1} & \cdots & g_{1,L-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & g_{1,0} & g_{1,1} & \cdots & & g_{1,L-1} \\ g_{2,0} & g_{2,1} & \cdots & g_{2,L-1} & 0 & 0 & \cdots & 0 \\ 0 & g_{2,0} & g_{2,1} & \cdots & g_{2,L-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & g_{2,0} & g_{2,1} & \cdots & & g_{2,L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ g_{M,0} & g_{M,1} & \cdots & g_{M,L-1} & 0 & 0 & \cdots & 0 \\ 0 & g_{M,0} & g_{M,1} & \cdots & g_{M,L-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & g_{M,0} & g_{M,1} & \cdots & & g_{M,L-1} \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} s_{\mathrm{s}}(k) \\ s_{\mathrm{s}}(k-1) \\ s_{\mathrm{s}}(k-2) \\ \vdots \\ s_{\mathrm{s}}(k-2L+3) \\ s_{\mathrm{s}}(k-2L+2) \end{pmatrix},\end{aligned}$$

where $\mathbf{G}$ is an $ML\times(2L-1)$ matrix, containing the impulse responses $g_m$ and is assumed to be of full column rank, and $\boldsymbol{s}_{\mathrm{s}}(k)$ is a $(2L-1)\times 1$ vector. We then have:

$$\begin{aligned}\mathbf{R}_{ss} &= E\{\mathbf{s}(k)\mathbf{s}^{\mathrm{T}}(k)\} \\ &= \mathbf{G}E\left\{\boldsymbol{s}_{\mathrm{s}}(k)\boldsymbol{s}_{\mathrm{s}}^{\mathrm{T}}(k)\right\}\mathbf{G}^{\mathrm{T}} \\ &= \mathbf{G}\mathbf{R}_{s_{\mathrm{s}}s_{\mathrm{s}}}\mathbf{G}^{\mathrm{T}}\,, \end{aligned} \tag{6.140}$$

where $\mathbf{R}_{s_{\mathrm{s}}s_{\mathrm{s}}}$ is the source signal covariance matrix of size $(2L-1)\times(2L-1)$, assumed to be full rank. We immediately see from (6.140), that:

$$\begin{aligned}\mathrm{Rank}[\mathbf{R}_{ss}] &= \min\{\mathrm{Rank}[\mathbf{G}],\ \mathrm{Rank}[\mathbf{R}_{s_{\mathrm{s}}s_{\mathrm{s}}}]\} \\ &= 2L-1\,, \end{aligned} \tag{6.141}$$

and

$$\begin{aligned}\mathrm{Null}[\mathbf{R}_{ss}] &= ML - \mathrm{Rank}[\mathbf{R}_{ss}] \\ &= (M-2)L+1\,, \end{aligned} \tag{6.142}$$

where Null[] and Rank[] denote the dimension of the null space and the rank of a matrix, respectively.

From this analysis, one can see that a MISO system is rank deficient if its inputs are the filtered version of the same source signal. Thus, the Wiener–Hopf equations do not have a unique solution.

In most practical situations, the signals $s_m(k)$, $m=1,2,\cdots,M$, are somehow related. If they are highly coherent, adaptive algorithms will be very slow to converge to the true solution and in some situations, they will converge to a solution that is far from the desired one.

We are now going to show in the particular case of a MISO system with two inputs how a high coherence between these signals affects the condition number of the covariance matrix:

$$\mathbf{R}_{ss} = \begin{pmatrix} \mathbf{R}_{s_1s_1} & \mathbf{R}_{s_1s_2} \\ \mathbf{R}_{s_2s_1} & \mathbf{R}_{s_2s_2} \end{pmatrix}.$$

For $L\to\infty$, a Toeplitz matrix is asymptotically equivalent to a circulant matrix if its elements are absolutely summable [6.31], which is the case for speech signals. In this situation, we can decompose

$$\mathbf{R}_{s_ms_i} = \mathbf{F}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{s_ms_i}\mathbf{F}\,, \quad m,i=1,2\,, \tag{6.143}$$

where $\mathbf{F}$ is the Fourier matrix and the diagonal matrix

$$\begin{aligned}\underset{\Rightarrow}{\mathbf{R}}_{s_ms_i} &= \mathrm{diag}\left\{\underset{\Rightarrow}{\mathbf{R}}_{s_ms_i}(0),\ \underset{\Rightarrow}{\mathbf{R}}_{s_ms_i}(1),\right. \\ &\qquad \left.\cdots,\ \underset{\Rightarrow}{\mathbf{R}}_{s_ms_i}(L-1)\right\} \end{aligned} \tag{6.144}$$

contains elements corresponding to the $L$ frequency bins that are formed from the discrete Fourier transform (DFT) of the first column of $\mathbf{R}_{s_ms_i}$. Letting $r_{s_ms_i}(l)$ be the auto- and cross-correlation for $m=i$ and $m\neq i$, respectively, we see that the spectral content between two signals is related to the correlation function by

$$\begin{aligned}\underset{\Rightarrow}{\mathbf{R}}_{s_ms_i}(f) &= \sum_{l=-\infty}^{\infty} r_{s_ms_i}(l)e^{-\mathrm{i}2\pi fl}\,, \\ &\quad f=0,1,\cdots,L-1\,. \end{aligned} \tag{6.145}$$

Amazon EX1010 Page 148

Using (6.143), $\mathbf{R}_{ss}$ can be expressed in terms of its spectra as:

$$\begin{aligned}\mathbf{R}_{ss} &= \mathbf{F}_{\mathrm{d}}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{ss}\mathbf{F}_{\mathrm{d}} \\ &= \begin{pmatrix}\mathbf{F}^{-1} & \mathbf{0}_{L\times L}\\ \mathbf{0}_{L\times L} & \mathbf{F}^{-1}\end{pmatrix}\begin{pmatrix}\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1} & \underset{\Rightarrow}{\mathbf{R}}_{s_1s_2}\\ \underset{\Rightarrow}{\mathbf{R}}_{s_2s_1} & \underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}\end{pmatrix} \\ &\quad\cdot\begin{pmatrix}\mathbf{F} & \mathbf{0}_{L\times L}\\ \mathbf{0}_{L\times L} & \mathbf{F}\end{pmatrix}. \end{aligned} \tag{6.146}$$

To compute the condition number $\chi^2_{\mathrm{F}}(\mathbf{R}_{ss}^{1/2})$ (Sect. 6.5), we need to compute $\mathrm{tr}(\mathbf{R}_{ss})$ and $\mathrm{tr}(\mathbf{R}_{ss}^{-1})$. The first trace is easy to compute. Indeed, using (6.146), we easily find:

$$\begin{aligned}\mathrm{tr}(\mathbf{R}_{ss}) &= \mathrm{tr}\left(\mathbf{F}_{\mathrm{d}}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{ss}\mathbf{F}_{\mathrm{d}}\right) = \mathrm{tr}\left(\underset{\Rightarrow}{\mathbf{R}}_{ss}\right) \\ &= \sum_{l=0}^{L-1}\left(\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}(l)+\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}(l)\right) . \end{aligned} \tag{6.147}$$

For the second trace, we have:

$$\mathrm{tr}\left(\mathbf{R}_{ss}^{-1}\right) = \mathrm{tr}\left(\mathbf{F}_{\mathrm{d}}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{ss}^{-1}\mathbf{F}_{\mathrm{d}}\right) = \mathrm{tr}\left(\underset{\Rightarrow}{\mathbf{R}}_{ss}^{-1}\right) . \tag{6.148}$$

Furthermore, it is easy to show that:

$$\begin{aligned}\underset{\Rightarrow}{\mathbf{R}}_{ss}^{-1} &= \begin{pmatrix}\underset{\Rightarrow}{\mathbf{R}}_1^{-1} & \mathbf{0}_{L\times L}\\ \mathbf{0}_{L\times L} & \underset{\Rightarrow}{\mathbf{R}}_2^{-1}\end{pmatrix} \\ &\quad\cdot\begin{pmatrix}\mathbf{I}_{L\times L} & -\underset{\Rightarrow}{\mathbf{R}}_{s_1s_2}\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}^{-1}\\ -\underset{\Rightarrow}{\mathbf{R}}_{s_2s_1}\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}^{-1} & \mathbf{I}_{L\times L}\end{pmatrix}, \end{aligned} \tag{6.149}$$

where

$$\underset{\Rightarrow}{\mathbf{R}}_1 = \left[\mathbf{I}_{L\times L} - \underset{\Rightarrow}{\mathbf{R}}_{s_1s_2}^2\left(\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}^{-1}\right)\right]\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1} , \tag{6.150}$$

$$\underset{\Rightarrow}{\mathbf{R}}_2 = \left[\mathbf{I}_{L\times L} - \underset{\Rightarrow}{\mathbf{R}}_{s_1s_2}^2\left(\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}^{-1}\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}^{-1}\right)\right]\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2} . \tag{6.151}$$

Hence,

$$\begin{aligned}\mathrm{tr}\left(\mathbf{R}_{ss}^{-1}\right) &= \mathrm{tr}\left(\underset{\Rightarrow}{\mathbf{R}}_1^{-1}+\underset{\Rightarrow}{\mathbf{R}}_2^{-1}\right) \\ &= \sum_{l=0}^{L-1}(1-|\gamma(l)|^2)^{-1} \\ &\quad\times\left[\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}^{-1}(l)+\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}^{-1}(l)\right] , \end{aligned} \tag{6.152}$$

where

$$\begin{aligned}|\gamma(f)|^2 &= \frac{\left|\underset{\Rightarrow}{\mathbf{R}}_{s_1s_2}(f)\right|^2}{\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}(f)\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}(f)} , \\ f &= 0, 1, \cdots, L-1 , \end{aligned} \tag{6.153}$$

is the squared interchannel coherence function of the $f$-th frequency bin.

We finally obtain the relationship between the interchannel coherence and the condition number based on the Frobenius norm:

$$\begin{aligned}\chi^2_{\mathrm{F}}\left(\mathbf{R}_{ss}^{1/2}\right) &= \left\{\sum_{l=0}^{L-1}\left[\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}(l)+\underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}(l)\right]\right\} \\ &\quad\times\left\{\sum_{l=0}^{L-1}[1-|\gamma(l)|^2]^{-1}\left[\underset{\Rightarrow}{\mathbf{R}}_{s_1s_1}^{-1}(l)\right.\right. \\ &\quad\left.\left.+\ \underset{\Rightarrow}{\mathbf{R}}_{s_2s_2}^{-1}(l)\right]\right\} . \end{aligned} \tag{6.154}$$

It is now evident from the previous expression that $\chi^2_{\mathrm{F}}(\mathbf{R}_{ss}^{1/2})$ increases with the squared interchannel coherence function, hence degrading the condition of $\mathbf{R}_{ss}$; as $\gamma \to 1$, $\chi^2_{\mathrm{F}}(\mathbf{R}_{ss}^{1/2}) \to \infty$ and the identification of the system is increasingly difficult, if not impossible.

## 6.8 Conclusions

In this chapter, we have explained the most important results of the Wiener theory in the context of system identification.

After discussing the four basic signal models, we derived the optimal Wiener filter for a SISO system and showed that this filter can be a very good approximation of the desired impulse response.

We discussed in details the condition number of the input signal correlation matrix. This matrix appears explicitly in the Wiener–Hopf equations and implicitly in all adaptive filters. A high condition number will perturb the accuracy of the solution of the Wiener–Hopf equations and will slow the rate of convergence of most adaptive algorithms. A fast, efficient algorithm to compute the conditional number was also developed.

We also discussed several important adaptive filters. In particular, the NLMS algorithm, which is extremely popular and useful in practice, was derived. Other emerging algorithms, such as the IPNLMS, were presented.

We generalized the Wiener principle to the MIMO system case. We showed that the MIMO Wiener–Hopf

equations can be decomposed into $N$ independent MISO Wiener–Hopf equations. As a result, adaptive filters for SISO and MISO systems, with a reference signal, cover all possible cases. A deep analysis of the conditioning of the input signal covariance matrix was given, showing that identification is not always obvious and depends on the interchannel coherence between the input signals.

## References


6.1 N. Wiener: *Extrapolation, Interpolation, and Smoothing of Stationary Time Series* (Wiley, New York 1949)

6.2 N. Wiener, E. Hopf: On a class of singular integral equations, Proc. Prussian Acad. Math.-Phys. Ser. (1931) p. 696

6.3 N. Levinson: The Wiener rms (root-mean-square) error criterion in filter design and prediction, J. Math. Phys. **25**, 261–278 (1947)

6.4 T. Kailath: A view of three decades of linear filtering theory, IEEE Trans. Inf. Theory **IT-20**, 146–181 (1974)

6.5 S. Haykin: *Adaptive Filter Theory*, 4th edn. (Prentice Hall, Upper Saddle River 2002)

6.6 J. Benesty, T. Gänsler: Computation of the condition number of a non-singular symmetric Toeplitz matrix with the Levinson–Durbin algorithm, IEEE Trans. Signal Process. **54**, 2362–2364 (2006)

6.7 J. Benesty, T. Gänsler: New insights into the RLS algorithm, EURASIP J. Appl. Signal Process. **2004**, 331–339 (2004)

6.8 G.H. Golub, C.F. Van Loan: *Matrix Computations* (The Johns Hopkins Univ. Press, Baltimore 1996)

6.9 J.M.B. Dias, J.M.N. Leitão: Efficient computation of $\mathrm{tr}\mathbf{TR}^{-1}$ for Toeplitz matrices, IEEE Signal Process. Let. **9**, 54–56 (2002)

6.10 B. Widrow: Adaptive filters. In: *Aspects of Network and System Theory*, ed. by R.E. Kalman, N. DeClaris (Holt Rinehart and Winston, New York 1970)

6.11 B. Widrow, M.E. Hoff Jr.: Adaptive switching circuits, IRE WESCON Conv. Rec. **4**, 96–104 (1960)

6.12 A. Feuer, E. Weinstein: Convergence analysis of LMS filters with uncorrelated Gaussian data, IEEE Trans. Acoust. Speech ASSP **33**, 222–230 (1985)

6.13 B. Widrow, S.D. Stearns: *Adaptive Signal Processing* (Prentice Hall, Englewood Cliffs 1985)

6.14 R.W. Harris, D.M. Chabries, F.A. Bishop: A variable step (VS) adaptive filter algorithm, IEEE Trans. Acoust. Speech ASSP **34**, 309–316 (1986)

6.15 R.H. Kwong, E.W. Johnston: A variable step size LMS algorithm, IEEE Trans. Signal Process. **40**, 1633–1642 (1992)

6.16 V.J. Mathews, Z. Xie: A stochastic gradient adaptive filter with gradient adaptive step size, IEEE Trans. Signal Process. **41**, 2075–2087 (1993)

6.17 J.B. Evans, P. Xue, B. Liu: Analysis and implementation of variable step size adaptive algorithms, IEEE Trans. Signal Process. **41**, 2517–2535 (1993)

6.18 T. Aboulnasr, K. Mayyas: A robust variable step-size LMS-type algorithm: analysis and simulations, IEEE Trans. Signal Process. **45**, 631–639 (1997)

6.19 D.I. Pazaitis, A.G. Constantinides: A novel kurtosis driven variable step-size adaptive algorithm, IEEE Trans. Signal Process. **47**, 864–872 (1999)

6.20 A. Mader, H. Puder, G.U. Schmidt: Step-size control for acoustic echo cancellation filters – An overview, Signal Process. **80**, 1697–1719 (2000)

6.21 H.-C. Shin, A.H. Sayed, W.-J. Song: Variable step-size NLMS and affine projection algorithms, IEEE Signal Process. Lett. **11**, 132–135 (2004)

6.22 D.R. Morgan, S.G. Kratzer: On a class of computationally efficient, rapidly converging, generalized NLMS algorithms, IEEE Signal Process. Lett. **3**, 245–247 (1996)

6.23 J. Benesty, H. Rey, L.R. Vega, S. Tressens: A nonparametric VSS-NLMS algorithm, IEEE Signal Process. Lett. **13**, 581–584 (2006), .

6.24 D.L. Duttweiler: Proportionate normalized least-mean-square adaptation in echo cancelers, IEEE Trans. Audio Speech **8**, 508–518 (2000)

6.25 J. Benesty, S.L. Gay: An improved PNLMS algorithm, Proc. IEEE ICASSP (2002) pp. 1881–1884

6.26 S.L. Gay: An efficient fast converging adaptive filter for network echo cancellation, Proc. Assilomar Conf. **1**, 394–398 (1998)

6.27 A. Gersho: Adaptive filtering with binary reinforcement, IEEE Trans. Inf. Theory **IT-30**, 191–199 (1984)

6.28 M.G. Bellanger: *Adaptive Digital Filters and Signal Analysis* (Marcel Dekker, New York 1987)

6.29 T. Claasen, W. Mecklenbrauker: Comparison of the convergence of two algorithms for adaptive FIR digital filters, IEEE Trans. Acoust. Speech ASSP **29**, 670–678 (1981)

6.30 N.J. Bershad: On the optimum data non-linearity in LMS adaptation, IEEE Trans. Acoust. Speech ASSP **34**, 69–76 (1986)

6.31 R. Gray: On the asymptotic eigenvalue distribution of Toeplitz matrices, IEEE Trans. Inform. Theory **IT-18**, 725–730 (1972)


Amazon EX1010 Page 150

# 7. Linear Prediction

J. Benesty, J. Chen, Y. Huang



Linear prediction plays a fundamental role in all aspects of speech. Its use seems natural and obvious in this context since for a speech signal the value of its current sample can be well modeled as a linear combination of its past values. In this chapter, we attempt to present the most important ideas on linear prediction. We derive the principal results, widely recognized by speech experts, in a very intuitive way without sacrificing mathematical rigor.

7.1 Fundamentals ..... 121
7.2 Forward Linear Prediction ..... 122
7.3 Backward Linear Prediction ..... 123
7.4 Levinson–Durbin Algorithm ..... 124
7.5 Lattice Predictor ..... 126
7.6 Spectral Representation ..... 127
7.7 Linear Interpolation ..... 128
7.8 Line Spectrum Pair Representation ..... 129
7.9 Multichannel Linear Prediction ..... 130
7.10 Conclusions ..... 133
References ..... 133

## 7.1 Fundamentals

Linear prediction (LP) is a fundamental tool in many diverse areas such as adaptive filtering, system identification, economics, geophysics, spectral estimation, and speech. Recently, a nice history of LP in the context of speech coding was written by *Atal* [7.1]. Readers are invited to consult this reference for more information about this topic and how it has evolved.

Linear prediction is widely used in speech applications (recognition, compression, modeling, etc.) [7.2,3]. This is due to the fact that the speech production process is well modeled with LP. Indeed, it is well recognized that a speech signal can be written in the following form [7.4,5],

$$x(k) = \sum_{l=1}^{L} a_l x(k-l) + Gu(k)\,, \tag{7.1}$$

where $k$ is the time index, $L$ represents the number of coefficients in the model (the order of the predictor), $a_l$, $l = 1, \cdots, L$, are defined as the linear prediction coefficients, $G$ is the gain of the system, and $u(k)$ is the excitation signal, which can be either a quasiperiodic train of impulses or a random noise source (also a combination of both signals for voiced fricatives such as 'v', 'z', and 'zh'). The periodic source produces voiced sounds such as vowels and nasals, and the noise source produces unvoiced or fricated sounds such as the fricatives. The parameters, $a_l$, determine the spectral characteristics of the particular sound for each of the two types of excitation and are widely used directly in many speech coding schemes and automatic speech recognition systems [7.4].

Equation (7.1) can be rewritten in the frequency domain, by using the $z$-transform. If $H(z)$ is the transfer function of the system, we have:

$$H(z) = \frac{G}{1 - \sum_{l=1}^{L} a_l z^{-l}} = \frac{G}{A(z)}\,, \tag{7.2}$$

which is an all-pole transfer function. This filter $[H(z)]$ is a good model of the human vocal tract [7.2]. Our main concern is to determine the predictor coefficients, $a_l, l = 1, 2, \cdots, L$, and to study the properties of the filter $A(z)$.

The applications of LP are numerous. Before addressing the estimation of LP coefficients, we give some examples to show the importance of LP. In many aspects of speech processing (noise reduction, speech separation, speech dereverberation, speech coding, etc.), it is of great interest to compare the closeness of the spectral envelope of two speech signals (the desired and

Amazon EX1010 Page 151

the processed ones) [7.6, 7]. One way of doing this is through comparing their LP coefficients. Consider the two speech signals $x(k)$ (desired) and $\hat{x}(k)$ (processed). Without entering too much into the details, one possible measure to evaluate the closeness of these two signals is the Itakura distance:

$$\mathrm{ID}_{x\hat{x}} = \ln \frac{E_x}{E_{\hat{x}}} , \tag{7.3}$$

where $E_x$ and $E_{\hat{x}}$ are the prediction-error powers of the signals $x(k)$ and $\hat{x}(k)$, respectively (see the following sections for more details). Note that the Itakura distance is not symmetric, i. e.,

$$\mathrm{ID}_{x\hat{x}} \neq \mathrm{ID}_{\hat{x}x} , \tag{7.4}$$

therefore, it is not a distance metric. However, asymmetry is usually not a problem for applications such as speech quality evaluation.

A more-powerful distance was proposed by Itakura and Saito in their formulation of linear prediction as an approximate maximum-likelihood estimation [7.8]. This distance between the two signals $x(k)$ and $\hat{x}(k)$ is defined as,

$$\mathrm{ISD}_{x\hat{x}} = \frac{E_{\hat{x}}}{E_x} - \ln \frac{E_{\hat{x}}}{E_x} - 1 . \tag{7.5}$$

Like the Itakura distance, this measure is not symmetric either; therefore, it is not a true metric.

The Itakura–Saito distance has many interesting properties. It has been shown that this measure is highly correlated with subjective quality judgements [7.6]. For example, a recent report on speech codec evaluation reveals that, if the Itakura–Saito measure between two speech signals is less than 0.5, the difference in their mean opinion score would be less than 1.6 [7.9]. Many other reported experiments also confirmed that when the Itakura–Saito distance between two speech signals is below 0.1, they would be perceived nearly identically by human ears. As a result, the Itakura–Saito distance, which is based on LP, is often used as an objective measure of speech quality. It is probably the most widely used measure of similarity between speech signals.

The two previous examples of the vocal-tract filter and the speech quality measure clearly show the importance of LP in speech applications.

In this chapter, we study the theory of linear prediction and derive the most important LP techniques that are often encountered in many speech applications. We assume here that all signals of interest are real, stationary, and zero mean.

## 7.2 Forward Linear Prediction

Consider a stationary random signal $x(k)$. The objective of the forward linear prediction is to predict the value of the sample $x(k)$ from its past values, i. e., $x(k-1)$, $x(k-2)$, etc. We define the forward prediction error as [7.10, 11],

$$\begin{aligned} e_{\mathrm{f},L}(k) &= x(k) - \hat{x}(k) \\ &= x(k) - \sum_{l=1}^{L} a_{L,l}\, x(k-l) \\ &= x(k) - \boldsymbol{a}_L^{\mathrm{T}} \boldsymbol{x}(k-1) , \end{aligned} \tag{7.6}$$

where the superscript 'T' denotes transposition, $\hat{x}(k)$ is the predicted sample,

$$\boldsymbol{a}_L = [a_{L,1} \;\; a_{L,2} \;\; \cdots \;\; a_{L,L}]^{\mathrm{T}}$$

is the forward predictor of length $L$, and

$$\boldsymbol{x}(k-1) = [x(k-1) \;\; x(k-2) \;\; \cdots \;\; x(k-L)]^{\mathrm{T}}$$

is a vector containing the $L$ most recent samples starting with and including $x(k-1)$.

We would like to find the optimal Wiener predictor. For that, we seek to minimize the mean-square error (MSE):

$$J_{\mathrm{f}}(\boldsymbol{a}_L) = E\left\{e_{\mathrm{f},L}^2(k)\right\} , \tag{7.7}$$

where $E\{\cdot\}$ denotes mathematical expectation. Taking the gradient of $J_{\mathrm{f}}(\boldsymbol{a}_L)$ with respect to $\boldsymbol{a}_L$ and equating to $\boldsymbol{0}_{L\times 1}$ (a vector of length $L$ containing only zeroes), we easily find the Wiener–Hopf equations:

$$\mathbf{R}_L \boldsymbol{a}_{\mathrm{o},L} = \boldsymbol{r}_{\mathrm{f},L} , \tag{7.8}$$

where the subscript 'o' in $\boldsymbol{a}_{\mathrm{o},L}$ stands for optimal,

$$\begin{aligned} \mathbf{R}_L &= E\{\boldsymbol{x}(k-1)\boldsymbol{x}^{\mathrm{T}}(k-1)\} \\ &= E\{\boldsymbol{x}(k)\boldsymbol{x}^{\mathrm{T}}(k)\} \\ &= \begin{pmatrix} r(0) & r(1) & \cdots & r(L-1) \\ r(1) & r(0) & \cdots & r(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(L-1) & r(L-2) & \cdots & r(0) \end{pmatrix} \end{aligned} \tag{7.9}$$

Amazon EX1010 Page 152

is the correlation matrix, and

$$\boldsymbol{r}_{\mathrm{f},L} = E\{\boldsymbol{x}(k-1)x(k)\} = [r(1)\ r(2)\ \cdots\ r(L)]^{\mathrm{T}} \tag{7.10}$$

is the correlation vector. The matrix $\mathbf{R}_L$ has a Toeplitz structure (i. e., all the entries along the diagonals are the same); assuming that it is nonsingular, we deduce the optimal forward predictor:

$$\boldsymbol{a}_{\mathrm{o},L} = \mathbf{R}_L^{-1}\boldsymbol{r}_{\mathrm{f},L}\ . \tag{7.11}$$

Expanding $e_{\mathrm{f},L}^2(k)$ in (7.7) and using (7.8) shows that the minimum mean-square error (MMSE),

$$J_{\mathrm{f,min}} = J_{\mathrm{f}}(\boldsymbol{a}_{\mathrm{o},L}) = r(0) - \boldsymbol{r}_{\mathrm{f},L}^{\mathrm{T}}\boldsymbol{a}_{\mathrm{o},L} = E_{\mathrm{f},L}\ . \tag{7.12}$$

This is also called the forward prediction-error power.

Define the augmented correlation matrix:

$$\mathbf{R}_{L+1} = \begin{pmatrix} r(0) & \boldsymbol{r}_{\mathrm{f},L}^{\mathrm{T}} \\ \boldsymbol{r}_{\mathrm{f},L} & \mathbf{R}_L \end{pmatrix}\ , \tag{7.13}$$

equations (7.8) and (7.12) may be combined in a convenient way:

$$\mathbf{R}_{L+1}\begin{pmatrix} 1 \\ -\boldsymbol{a}_{\mathrm{o},L} \end{pmatrix} = \begin{pmatrix} E_{\mathrm{f},L} \\ \boldsymbol{0}_{L\times 1} \end{pmatrix}\ . \tag{7.14}$$

We refer to (7.14) as the augmented Wiener–Hopf equations of a forward predictor of order $L$. From (7.13) we derive that,

$$\det(\mathbf{R}_{L+1}) = E_{\mathrm{f},L}\det(\mathbf{R}_L)\ , \tag{7.15}$$

where 'det' stands for determinant.

Let us now write the forward prediction errors for the optimal predictors of orders $L$ and $L-i$:

$$e_{\mathrm{f,o},L}(k) = x(k) - \sum_{l=1}^{L} a_{\mathrm{o},L,l}\,x(k-l)\ , \tag{7.16}$$

$$e_{\mathrm{f,o},L-i}(k) = x(k) - \sum_{l=1}^{L-i} a_{\mathrm{o},L-i,l}\,x(k-l)\ . \tag{7.17}$$

From the principle of orthogonality [7.11], we know that:

$$E\{e_{\mathrm{f,o},L}(k)\boldsymbol{x}(k-1)\} = \boldsymbol{0}_{L\times 1}\ . \tag{7.18}$$

For $1 \le i \le L$, we can verify by using (7.18), that:

$$E\{e_{\mathrm{f,o},L}(k)e_{\mathrm{f,o},L-i}(k-i)\} = 0\ . \tag{7.19}$$

As a result,

$$\lim_{L\to\infty} E\{e_{\mathrm{f,o},L}(k)e_{\mathrm{f,o},L-i}(k-i)\} = E\{e_{\mathrm{f,o}}(k)e_{\mathrm{f,o}}(k-i)\} = 0\ . \tag{7.20}$$

This indicates that the signal $e_{\mathrm{f,o}}(k)$ is a white noise. So the optimal forward predictor has this important property of being able to whiten a stationary random process, provided that the order of the predictor is high enough.

## 7.3 Backward Linear Prediction

The aim of the backward linear prediction is to predict the value of the sample $x(k-L)$ from its future values, i. e., $x(k)$, $x(k-1)$, $\cdots$, $x(k-L+1)$. We define the backward prediction error as,

$$e_{\mathrm{b},L}(k) = x(k-L) - \hat{x}(k-L) = x(k-L) - \sum_{l=1}^{L} b_{L,l}\,x(k-l+1) = x(k-L) - \boldsymbol{b}_L^{\mathrm{T}}\boldsymbol{x}(k)\ , \tag{7.21}$$

where $\hat{x}(k-L)$ is the predicted sample,

$$\boldsymbol{b}_L = [b_{L,1}\ \ b_{L,2}\ \cdots\ b_{L,L}]^{\mathrm{T}}$$

is the backward predictor of order $L$, and

$$\boldsymbol{x}(k) = [x(k)\ \ x(k-1)\ \cdots\ x(k-L+1)]^{\mathrm{T}}\ .$$

The minimization of the MSE,

$$J_{\mathrm{b}}(\boldsymbol{b}_L) = E\left\{e_{\mathrm{b},L}^2(k)\right\}\ , \tag{7.22}$$

leads to the Wiener–Hopf equations:

$$\mathbf{R}_L\boldsymbol{b}_{\mathrm{o},L} = \boldsymbol{r}_{\mathrm{b},L}\ , \tag{7.23}$$

where

$$\boldsymbol{r}_{\mathrm{b},L} = E\{\boldsymbol{x}(k)x(k-L)\} = [r(L)\ \ r(L-1)\ \cdots\ r(1)]^{\mathrm{T}}\ . \tag{7.24}$$

Therefore, the optimal backward predictor is:

$$\boldsymbol{b}_{\mathrm{o},L} = \mathbf{R}_L^{-1}\boldsymbol{r}_{\mathrm{b},L}\ . \tag{7.25}$$

The MMSE for backward prediction,

$$J_{\mathrm{b,min}} = J_{\mathrm{b}}(\boldsymbol{b}_{\mathrm{o},L}) = r(0) - \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}}\boldsymbol{b}_{\mathrm{o},L} = E_{\mathrm{b},L}\ , \tag{7.26}$$

is also called the backward prediction-error power.

Define the augmented correlation matrix:

$$\mathbf{R}_{L+1} = \begin{pmatrix} \mathbf{R}_L & \boldsymbol{r}_{\mathrm{b},L} \\ \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}} & r(0) \end{pmatrix}\ , \tag{7.27}$$



Amazon EX1010 Page 153

equations (7.23) and (7.26) may be combined in a convenient way:

$$\mathbf{R}_{L+1}\begin{pmatrix}-\boldsymbol{b}_{\mathrm{o},L}\\1\end{pmatrix}=\begin{pmatrix}\boldsymbol{0}_{L\times 1}\\E_{\mathrm{b},L}\end{pmatrix}\,. \tag{7.28}$$

We refer to this expression as the augmented Wiener–Hopf equations of a backward predictor of order $L$.

One important property of backward prediction is that the error signals of different orders with the optimal predictors are uncorrelated, i. e., $E\{e_{\mathrm{b,o},i}(k)e_{\mathrm{b,o},l}(k)\}=0, i\neq l, i,l=0,1,\cdots,L-1$. To prove this, let us rewrite the error signal in vector form:

$$\boldsymbol{e}_{\mathrm{b,o}}(k)=\mathbf{L}\boldsymbol{x}(k)\,, \tag{7.29}$$

where

$$\boldsymbol{e}_{\mathrm{b,o}}(k)=[e_{\mathrm{b,o},0}(k)\;\; e_{\mathrm{b,o},1}(k)\;\;\cdots\;\; e_{\mathrm{b,o},L-1}(k)]^{\mathrm{T}} \tag{7.30}$$

and

$$\mathbf{L}=\begin{pmatrix}1 & 0 & 0 & \cdots & 0\\ -\boldsymbol{b}_{\mathrm{o},1}^{\mathrm{T}} & 1 & 0 & \cdots & 0\\ -\boldsymbol{b}_{\mathrm{o},2}^{\mathrm{T}} & & 1 & \cdots & 0\\ \vdots & \vdots & & \vdots & \vdots\\ -\boldsymbol{b}_{\mathrm{o},L-1}^{\mathrm{T}} & & & & 1\end{pmatrix} \tag{7.31}$$

is a lower triangular matrix with 1s along its main diagonal. The covariance matrix corresponding to the vector signal $\boldsymbol{e}_{\mathrm{b,o}}(k)$ is:

$$E\left\{\boldsymbol{e}_{\mathrm{b,o}}(k)\boldsymbol{e}_{\mathrm{b,o}}^{\mathrm{T}}(k)\right\}=\mathbf{L}\mathbf{R}_L\mathbf{L}^{\mathrm{T}}\,. \tag{7.32}$$

By definition, the previous matrix is symmetric. The matrix product $\mathbf{R}_L\mathbf{L}^{\mathrm{T}}$ is a lower triangular matrix because of (7.28) and the main diagonal contains the backward prediction-error powers $E_{\mathrm{b},l}$ $(0\leq l\leq L-1)$. Since $\mathbf{L}$ is also a lower triangular matrix, the product between the two matrices $\mathbf{L}$ and $\mathbf{R}_L\mathbf{L}^{\mathrm{T}}$ should have the same structure and, since it has to be symmetric, the only possibility is that this resulting matrix is diagonal:

$$E\left\{\boldsymbol{e}_{\mathrm{b,o}}(k)\boldsymbol{e}_{\mathrm{b,o}}^{\mathrm{T}}(k)\right\}=\mathrm{diag}[E_{\mathrm{b},0},E_{\mathrm{b},1},\cdots,E_{\mathrm{b},L-1}]\,, \tag{7.33}$$

and hence the prediction errors are uncorrelated.

Furthermore,

$$\mathbf{L}\mathbf{R}_L\mathbf{L}^{\mathrm{T}}=\mathrm{diag}[E_{\mathrm{b},0},E_{\mathrm{b},1},\cdots,E_{\mathrm{b},L-1}]\,, \tag{7.34}$$

taking the inverse of the previous equation,

$$\mathbf{L}^{-\mathrm{T}}\mathbf{R}_L^{-1}\mathbf{L}^{-1}=\mathrm{diag}\left[E_{\mathrm{b},0}^{-1},E_{\mathrm{b},1}^{-1},\cdots,E_{\mathrm{b},L-1}^{-1}\right]\,, \tag{7.35}$$

we finally get:

$$\mathbf{R}_L^{-1}=\mathbf{L}^{\mathrm{T}}\mathrm{diag}\left[E_{\mathrm{b},0}^{-1},E_{\mathrm{b},1}^{-1},\cdots,E_{\mathrm{b},L-1}^{-1}\right]\mathbf{L}\,. \tag{7.36}$$

Expression (7.36) defines the Cholesky factorization of the inverse matrix $\mathbf{R}_L^{-1}$ [7.10, 12].

## 7.4 Levinson–Durbin Algorithm

The Levinson–Durbin algorithm is an efficient way to solve the Wiener–Hopf equations for the forward and backward prediction coefficients. This efficient method can be derived thanks to the Toeplitz structure of the correlation matrix $\mathbf{R}_L$. This algorithm was first invented by *Levinson* [7.13] and independently reformulated at a later date by *Durbin* [7.14, 15]. *Burg* gave a more-elegant presentation [7.16]. Before describing this algorithm, we first need to show some important relations between the forward and backward predictors.

We define the co-identity matrix as:

$$\mathbf{J}_L=\begin{pmatrix}0 & 0 & \cdots & 0 & 1\\ 0 & 0 & \cdots & 1 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 1 & \cdots & 0 & 0\\ 1 & 0 & \cdots & 0 & 0\end{pmatrix}\,.$$

We can easily check that:

$$\mathbf{R}_L\mathbf{J}_L=\mathbf{J}_L\mathbf{R}_L\,. \tag{7.37}$$

The matrix $\mathbf{R}_L$ is said to be persymmetric. We also have, $\boldsymbol{r}_{\mathrm{f},L}=\mathbf{J}_L\boldsymbol{r}_{\mathrm{b},L}$. If we left-multiply both sides of the Wiener–Hopf equations (7.23) by $\mathbf{J}_L$, we get:

$$\begin{aligned}\mathbf{J}_L\mathbf{R}_L\boldsymbol{b}_{\mathrm{o},L}&=\mathbf{J}_L\boldsymbol{r}_{\mathrm{b},L}=\boldsymbol{r}_{\mathrm{f},L}\\&=\mathbf{R}_L\mathbf{J}_L\boldsymbol{b}_{\mathrm{o},L}=\mathbf{R}_L\boldsymbol{a}_{\mathrm{o},L}\,,\end{aligned} \tag{7.38}$$

and, assuming that $\mathbf{R}_L$ is nonsingular, we see that:

$$\boldsymbol{a}_{\mathrm{o},L}=\mathbf{J}_L\boldsymbol{b}_{\mathrm{o},L}\,. \tag{7.39}$$

Amazon EX1010 Page 154

Furthermore,

$$\begin{aligned} E_{\mathrm{b},L} &= r(0) - \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}} \boldsymbol{b}_{\mathrm{o},L} \\ &= r(0) - \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}} \mathbf{J}_L \mathbf{J}_L \boldsymbol{b}_{\mathrm{o},L} \\ &= r(0) - \boldsymbol{r}_{\mathrm{f},L}^{\mathrm{T}} \boldsymbol{a}_{\mathrm{o},L} \\ &= E_{\mathrm{f},L} = E_L \,. \end{aligned} \tag{7.40}$$

Therefore, for a stationary process, the forward and backward prediction-error powers are equal and the coefficients of the optimal forward predictor are the same as those of the optimal backward predictor, but in a reverse order.

The Levinson–Durbin algorithm is based on recursions of the orders of the prediction equations. Consider the following expression,

$$\begin{pmatrix} \mathbf{R}_L & \boldsymbol{r}_{\mathrm{b},L} \\ \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}} & r(0) \end{pmatrix} \begin{pmatrix} 1 \\ -\boldsymbol{a}_{\mathrm{o},L-1} \\ 0 \end{pmatrix} = \begin{pmatrix} E_{L-1} \\ \boldsymbol{0}_{(L-1)\times 1} \\ K_L \end{pmatrix} , \tag{7.41}$$

where

$$\begin{aligned} K_L &= r(L) - \boldsymbol{a}_{\mathrm{o},L-1}^{\mathrm{T}} \boldsymbol{r}_{\mathrm{b},L-1} \\ &= r(L) - \boldsymbol{a}_{\mathrm{o},L-1}^{\mathrm{T}} \mathbf{J}_{L-1} \boldsymbol{r}_{\mathrm{f},L-1} \,. \end{aligned} \tag{7.42}$$

We define the reflection coefficient as,

$$\kappa_L = \frac{K_L}{E_{L-1}} \,. \tag{7.43}$$

From backward linear prediction, we have:

$$\begin{pmatrix} r(0) & \boldsymbol{r}_{\mathrm{f},L}^{\mathrm{T}} \\ \boldsymbol{r}_{\mathrm{f},L} & \mathbf{R}_L \end{pmatrix} \begin{pmatrix} 0 \\ -\boldsymbol{b}_{\mathrm{o},L-1} \\ 1 \end{pmatrix} = \begin{pmatrix} K_L \\ \boldsymbol{0}_{(L-1)\times 1} \\ E_{L-1} \end{pmatrix} \,. \tag{7.44}$$

Multiplying both sides of the previous equation by $\kappa_L$, we get,

$$\mathbf{R}_{L+1} \begin{pmatrix} 0 \\ -\kappa_L \boldsymbol{b}_{\mathrm{o},L-1} \\ \kappa_L \end{pmatrix} = \begin{pmatrix} \kappa_L^2 E_{L-1} \\ \boldsymbol{0}_{(L-1)\times 1} \\ K_L \end{pmatrix} \,. \tag{7.45}$$

If we now subtract (7.45) from (7.41), we obtain,

$$\mathbf{R}_{L+1} \begin{pmatrix} 1 \\ \kappa_L \boldsymbol{b}_{\mathrm{o},L-1} - \boldsymbol{a}_{\mathrm{o},L-1} \\ -\kappa_L \end{pmatrix} = \begin{pmatrix} E_{L-1}(1-\kappa_L^2) \\ \boldsymbol{0}_{L\times 1} \end{pmatrix} \,. \tag{7.46}$$

Assuming that $\mathbf{R}_{L+1}$ is nonsingular and identifying (7.46) with (7.14), we can deduce the recursive equations:

$$\boldsymbol{a}_{\mathrm{o},L} = \begin{pmatrix} \boldsymbol{a}_{\mathrm{o},L-1} \\ 0 \end{pmatrix} - \kappa_L \begin{pmatrix} \boldsymbol{b}_{\mathrm{o},L-1} \\ -1 \end{pmatrix} \,, \tag{7.47}$$

$$E_L = E_{L-1}(1-\kappa_L^2) \,, \tag{7.48}$$

$$a_{\mathrm{o},L,L} = \kappa_L \,. \tag{7.49}$$

Iterating on the prediction-error power given in (7.48), we find that,

$$E_L = r(0) \prod_{l=1}^{L} (1-\kappa_l^2) \,, \tag{7.50}$$

and since $E_L \geq 0$, this implies that,

$$|\kappa_l| \leq 1, \quad \forall l \geq 1 \,. \tag{7.51}$$

Also, from (7.48) we see that we have,

$$0 \leq E_l \leq E_{l-1}, \quad \forall l \geq 1 \,, \tag{7.52}$$

so, as the order of the predictors increases, the prediction-error power decreases.

Table 7.1 summarizes the Levinson–Durbin algorithm, whose arithmetic complexity is proportional to $L^2$. This algorithm is much more efficient than standard methods such as the Gauss elimination technique, whose complexity is on the order of $L^3$. The saving in number of operations to find the optimal Wiener predictor can be very important, especially when $L$ is large. The other advantage of the Levinson–Durbin algorithm is that it gives the predictors of all orders and the algorithm can be stopped if the prediction-error power is under a threshold, which can be very useful in practice when the choice of the predictor order is not easy to get in advance. A slightly more-efficient approach, called the split Levinson algorithm, can be found in [7.17]. This algorithm requires roughly half the number of multiplications and the same number of additions as the classical Levison–Durbin algorithm. Even more-efficient algorithms have been proposed (see, for example, [7.18]) but they are numerically unstable, which is not acceptable in most speech applications.

**Table 7.1** Levinson–Durbin algorithm

**Initialization:** $E_0 = r(0)$

**For** $1 \leq l \leq L$

$$\kappa_l = \frac{1}{E_{l-1}} \left[ r(l) - \boldsymbol{a}_{\mathrm{o},l-1}^{\mathrm{T}} \mathbf{J}_{l-1} \boldsymbol{r}_{\mathrm{f},l-1} \right]$$

$$\boldsymbol{a}_{\mathrm{o},l} = \begin{pmatrix} \boldsymbol{a}_{\mathrm{o},l-1} \\ 0 \end{pmatrix} - \kappa_l \mathbf{J}_l \begin{pmatrix} -1 \\ \boldsymbol{a}_{\mathrm{o},l-1} \end{pmatrix}$$

$$E_l = E_{l-1}\left(1-\kappa_l^2\right)$$



Amazon EX1010 Page 155

## 7.5 Lattice Predictor

In this section, we will show that the order-recursive structure of the forward and backward prediction errors has the form of a ladder, which is called a lattice predictor.

Inserting (7.47) into the forward prediction error for the optimal predictor of order $L$,

$$e_{\mathrm{f,o},L}(k) = x(k) - \boldsymbol{a}_{\mathrm{o},L}^{\mathrm{T}}\boldsymbol{x}(k-1)\,, \tag{7.53}$$

we obtain,

$$\begin{aligned} e_{\mathrm{f,o},L}(k) &= e_{\mathrm{f,o},L-1}(k) \\ &\quad - \kappa_L \begin{pmatrix} -\boldsymbol{b}_{\mathrm{o},L-1}^{\mathrm{T}} & 1 \end{pmatrix} \boldsymbol{x}(k-1)\,. \end{aligned} \tag{7.54}$$

The second term (without the reflection coefficient) on the right-hand side of (7.54) is the backward prediction error, at time $k-1$, for the optimal predictor of order $L-1$. Therefore, (7.54) can be rewritten

$$e_{\mathrm{f,o},L}(k) = e_{\mathrm{f,o},L-1}(k) - \kappa_L e_{\mathrm{b,o},L-1}(k-1)\,. \tag{7.55}$$

If we insert (7.47) again into the backward prediction error for the optimal predictor of order $L$,

$$\begin{aligned} e_{\mathrm{b,o},L}(k) &= x(k-L) - \boldsymbol{b}_{\mathrm{o},L}^{\mathrm{T}}\boldsymbol{x}(k) \\ &= x(k-L) - \boldsymbol{a}_{\mathrm{o},L}^{\mathrm{T}}\boldsymbol{J}_L\boldsymbol{x}(k)\,, \end{aligned} \tag{7.56}$$

we get,

$$e_{\mathrm{b,o},L}(k) = e_{\mathrm{b,o},L-1}(k-1) - \kappa_L e_{\mathrm{f,o},L-1}(k)\,. \tag{7.57}$$

If we put (7.55) and (7.57) into a matrix form, we have,

$$\begin{aligned} \begin{pmatrix} e_{\mathrm{f,o},L}(k) \\ e_{\mathrm{b,o},L}(k) \end{pmatrix} &= \begin{pmatrix} 1 & -\kappa_L \\ -\kappa_L & 1 \end{pmatrix} \begin{pmatrix} e_{\mathrm{f,o},L-1}(k) \\ e_{\mathrm{b,o},L-1}(k-1) \end{pmatrix} \\ &= \prod_{l=1}^{L} \begin{pmatrix} 1 & -\kappa_l \\ -\kappa_l & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ x(k-1) \end{pmatrix}\,, \end{aligned} \tag{7.58}$$

where we have taken for initial conditions (order 0), $e_{\mathrm{f,o},0}(k) = x(k)$ and $e_{\mathrm{b,o},0}(k-1) = x(k-1)$. Figure 7.1 depicts the $l$-th stage of a lattice predictor. For the whole lattice predictor, $L$ of these stages are needed and are connected in cascade, one to each other, starting from order 0 to order $L$.

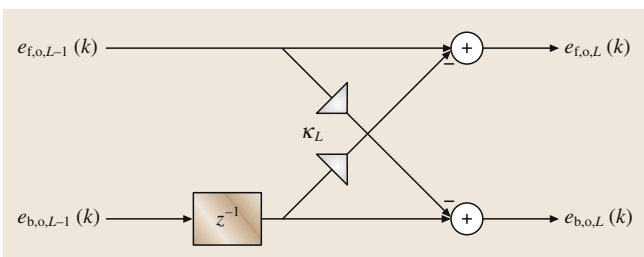



**Fig. 7.1** Stage $l$ of a lattice predictor

Now let us compute the variance of $e_{\mathrm{b,o},L}(k)$ from (7.57),

$$\begin{aligned} E\{e_{\mathrm{b,o},L}^2(k)\} &= E_L \\ &= E_{L-1} + \kappa_L^2 E_{L-1} \\ &\quad - 2\kappa_L E\{e_{\mathrm{f,o},L-1}(k)e_{\mathrm{b,o},L-1}(k-1)\} \\ &= E_{L-1}\left(1-\kappa_L^2\right)\,. \end{aligned} \tag{7.59}$$

Developing the previous expression, we obtain,

$$\begin{aligned} \kappa_L &= \frac{E\{e_{\mathrm{f,o},L-1}(k)e_{\mathrm{b,o},L-1}(k-1)\}}{E_{L-1}} \\ &= \frac{E\{e_{\mathrm{f,o},L-1}(k)e_{\mathrm{b,o},L-1}(k-1)\}}{\sqrt{E\{e_{\mathrm{f,o},L-1}^2(k)\}E\{e_{\mathrm{b,o},L-1}^2(k-1)\}}}\,. \end{aligned} \tag{7.60}$$

We see from (7.60) that the reflection coefficients are also the normalized cross-correlation coefficients between the forward and backward prediction errors, which is why they are also often called partial correlation (PARCOR) coefficients [7.3, 19, 20]. These coefficients are linked to the zeroes of the forward prediction-error FIR filter of order $L$, whose transfer function is

$$A_{\mathrm{o},L}(z) = 1 - \sum_{l=1}^{L} a_{\mathrm{o},L,l} z^{-l} = \prod_{l=1}^{L} \left(1 - z_{\mathrm{o},l} z^{-1}\right)\,, \tag{7.61}$$

where $z_{\mathrm{o},l}$ are the roots of $A_{\mathrm{o},L}(z)$. Since $\kappa_L = a_{\mathrm{o},L,L}$, we have,

$$\kappa_L = (-1)^{L+1} \prod_{l=1}^{L} z_{\mathrm{o},l}\,. \tag{7.62}$$

The filter $A_{\mathrm{o},L}(z)$ can be shown to be minimum phase, i. e., $|z_{\mathrm{o},l}| \le 1$, $\forall l$. As a result [because of the relation (7.39)], the filter $B_{\mathrm{o},L}(z)$ corresponding to the backward predictor is maximum phase. We will now show this very important property that the forward predictor is minimum phase. As far as we know, this simple and elegant proof was first shown by M. Mohan Sondhi but was never been published. A similar proof can be found in [7.21] and [7.22].

To avoid cumbersome notation, redefine the coefficients $w_l = -a_{\mathrm{o},L,l}$, with $w_0 = 1$, so that the polynomial

Amazon EX1010 Page 156

becomes,

$$A_{o,L}(z) = \sum_{l=0}^{L} w_l z^{-l} . \tag{7.63}$$

Also, define the vector,

$$\boldsymbol{w} = [w_0 \;\; w_1 \;\; \cdots \;\; w_L]^{\mathrm{T}} .$$

We know that,

$$\mathbf{R}_{L+1}\boldsymbol{w} = \begin{pmatrix} E_L \\ \boldsymbol{0}_{L\times 1} \end{pmatrix} . \tag{7.64}$$

If $\lambda$ is a root of the polynomial, it follows that,

$$A_{o,L}(z) = (1-\lambda z^{-1}) \sum_{l=0}^{L-1} g_l z^{-l}, \text{ with } g_0 = 1 . \tag{7.65}$$

(Note that since $\lambda$ can be complex, the coefficients $g_l$ are, in general, complex.) Thus the vector $\boldsymbol{w}$ can be written

$$\boldsymbol{w} = \boldsymbol{g} - \lambda \tilde{\boldsymbol{g}} , \tag{7.66}$$

where

$$\boldsymbol{g} = [1 \;\; g_1 \;\; g_2 \;\; \cdots \;\; g_{L-1} \;\; 0]^{\mathrm{T}} = [\boldsymbol{g}'^{\mathrm{T}} \;\; 0]^{\mathrm{T}} ,$$
$$\tilde{\boldsymbol{g}} = [0 \;\; 1 \;\; g_1 \;\; g_2 \;\; \cdots \;\; g_{L-1}]^{\mathrm{T}} = [0 \;\; \boldsymbol{g}'^{\mathrm{T}}]^{\mathrm{T}} .$$

Substituting (7.66) in (7.64), we obtain,

$$\mathbf{R}_{L+1}\boldsymbol{g} = \lambda \mathbf{R}_{L+1}\tilde{\boldsymbol{g}} + \begin{pmatrix} E_L \\ \boldsymbol{0}_{L\times 1} \end{pmatrix} . \tag{7.67}$$

Now, premultiplying by $\tilde{\boldsymbol{g}}^{\mathrm{H}}$ (where the superscript $^{\mathrm{H}}$ denotes conjugate transpose) gives,

$$\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g} = \lambda \tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}} . \tag{7.68}$$

Thus,

$$\left|\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g}\right|^2 = |\lambda|^2 \left(\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}}\right)^2 . \tag{7.69}$$

Using the Schwartz inequality,

$$\left|\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g}\right|^2 \le \left(\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}}\right)\left(\boldsymbol{g}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g}\right) . \tag{7.70}$$

However,

$$\begin{aligned}\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}} &= \begin{pmatrix} 0 & \boldsymbol{g}'^{\mathrm{H}} \end{pmatrix} \begin{pmatrix} r(0) & \boldsymbol{r}_{\mathrm{f},L}^{\mathrm{T}} \\ \boldsymbol{r}_{\mathrm{f},L} & \mathbf{R}_L \end{pmatrix} \begin{pmatrix} 0 \\ \boldsymbol{g}' \end{pmatrix} \\ &= \boldsymbol{g}'^{\mathrm{H}}\mathbf{R}_L\boldsymbol{g}' ,\end{aligned} \tag{7.71}$$

Similarly,

$$\begin{aligned}\boldsymbol{g}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g} &= \begin{pmatrix} \boldsymbol{g}'^{\mathrm{H}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R}_L & \boldsymbol{r}_{\mathrm{b},L} \\ \boldsymbol{r}_{\mathrm{b},L}^{\mathrm{T}} & r(0) \end{pmatrix} \begin{pmatrix} \boldsymbol{g}' \\ 0 \end{pmatrix} \\ &= \boldsymbol{g}'^{\mathrm{H}}\mathbf{R}_L\boldsymbol{g}' .\end{aligned} \tag{7.72}$$

Therefore, $\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}} = \boldsymbol{g}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g}$, and the Schwartz inequality becomes,

$$\left|\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\boldsymbol{g}\right|^2 \le \left(\tilde{\boldsymbol{g}}^{\mathrm{H}}\mathbf{R}_{L+1}\tilde{\boldsymbol{g}}\right)^2 . \tag{7.73}$$

From (7.69) we see that $|\lambda|^2 \le 1$. This completes the proof.

This property allows one easily to ensure that the all-pole system in (7.2) is stable (when the correlation matrix is positive definite) by simply imposing the constraint that the PARCOR coefficients are less than 1 in magnitude. As a result, in speech communication, transmitting PARCOR coefficients is more advantageous than directly transmitting linear predication coefficients.

## 7.6 Spectral Representation

It is important to understand the link between the spectrum of a speech signal and its prediction coefficients. Let us again take the speech model given in Sect. 7.1,

$$x(k) = \sum_{l=1}^{L} a_l x(k-l) + Gu(k) , \tag{7.74}$$

where we now assume that $u(k)$ is a white random signal with variance $\sigma_u^2 = 1$. Since $x(k)$ is the output of the filter $H(z)$ (see Sect. 7.1), whose input is $u(k)$, its spectrum is [7.11],

$$S_x(\omega) = \left|H(\mathrm{e}^{\mathrm{i}\omega})\right|^2 S_u(\omega) , \tag{7.75}$$

where $\omega$ is the angular frequency, $H(\mathrm{e}^{\mathrm{i}\omega})$ is the frequency response of the filter $H(z)$, and $S_u(\omega)$ is the spectrum of $u$. We have $S_u(\omega) = 1$ ($u$ is white). Using (7.2), we deduce the spectrum of $x$,

$$\begin{aligned} S_x(\omega) &= \frac{G^2}{|A(\mathrm{e}^{\mathrm{i}\omega})|^2} \\ &= \frac{G^2}{\left|1 - \sum_{l=1}^{L} a_l \mathrm{e}^{-\mathrm{i}l\omega}\right|^2} . \end{aligned} \tag{7.76}$$

Therefore, the spectrum of a speech signal can be modeled by the frequency response of an all-pole filter,



Amazon EX1010 Page 157

whose elements are the prediction coefficients [7.23–25].

Consider the prediction error signal,

$$e_L(k) = x(k) - \boldsymbol{a}_L^{\mathrm{T}} \boldsymbol{x}(k-1) \,. \tag{7.77}$$

Taking the $z$-transform of (7.77) and setting $z = \mathrm{e}^{\mathrm{i}\omega}$, we obtain:

$$S_{x,L}(\omega) = \frac{|E_L(\mathrm{e}^{\mathrm{i}\omega})|^2}{|A_L(\mathrm{e}^{\mathrm{i}\omega})|^2} \,, \tag{7.78}$$

where $|E_L(\mathrm{e}^{\mathrm{i}\omega})|^2$ is the spectrum of $e_L(k)$. From Sect. 7.2, we know that, for a large order $L$, linear prediction tends to whiten the signal, so the power spectrum $|E_L(\mathrm{e}^{\mathrm{i}\omega})|^2$ of the error signal, $e_L(k)$, will tend to be flat. Hence,

$$\lim_{L\to\infty} \left|E_L(\mathrm{e}^{\mathrm{i}\omega})\right|^2 = G^2 \,. \tag{7.79}$$

As a result,

$$\lim_{L\to\infty} S_{x,L}(\omega) = \frac{G^2}{\left|1 - \sum_{l=1}^{\infty} a_l \mathrm{e}^{-\mathrm{i}l\omega}\right|^2} \,. \tag{7.80}$$

This confirms that (7.76) can be a very good approximation of the spectrum of a speech signal, as long as the order of the predictor is large enough.

## 7.7 Linear Interpolation

Linear interpolation can be seen as a straightforward generalization of forward and backward linear predictions. Indeed, in linear interpolation, we try to predict the value of the sample $x(k-i)$ from its past and future values [7.26, 27]. We define the interpolation error as

$$\begin{aligned} e_i(k) &= x(k-i) - \hat{x}(k-i) \\ &= x(k-i) - \sum_{l=0, l\neq i}^{L} c_{i,l}\, x(k-l) \\ &= \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{x}_{L+1}(k), \quad i = 0, 1, \cdots, L, \end{aligned} \tag{7.81}$$

where $\hat{x}(k-i)$ is the interpolated sample,

$$\boldsymbol{c}_i = [-c_{i,0} \;\; -c_{i,1} \;\; \cdots \;\; c_{i,i} \;\; \cdots \;\; -c_{i,L}]^{\mathrm{T}}$$

is a vector of length $L+1$ containing the interpolation coefficients, with $c_{i,i} = 1$, and

$$\boldsymbol{x}_{L+1}(k) = [x(k) \;\; x(k-1) \;\; \cdots \;\; x(k-L)]^{\mathrm{T}} \,.$$

The special cases $i = 0$ and $i = L$ are the forward and backward prediction errors, respectively.

To find the optimal Wiener interpolator, we need to minimize the cost function,

$$\begin{aligned} J_i(\boldsymbol{c}_i) &= E\left\{e_i^2(k)\right\} \\ &= \boldsymbol{c}_i^{\mathrm{T}} \mathbf{R}_{L+1} \boldsymbol{c}_i \,, \end{aligned} \tag{7.82}$$

subject to the constraint

$$\boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{v}_i = c_{i,i} = 1 \,, \tag{7.83}$$

where

$$\boldsymbol{v}_i = [0 \;\; 0 \;\; \cdots \;\; 0 \;\; 1 \;\; 0 \;\; \cdots \;\; 0]^{\mathrm{T}}$$

is a vector of length $L+1$ with its $i$-th component equal to one and all others equal to zero. By using a Lagrange multiplier, it is easy to see that the solution to this optimization problem is

$$\mathbf{R}_{L+1} \boldsymbol{c}_{\mathrm{o},i} = E_i \boldsymbol{v}_i \,, \tag{7.84}$$

where

$$\begin{aligned} E_i &= \boldsymbol{c}_{\mathrm{o},i}^{\mathrm{T}} \mathbf{R}_{L+1} \boldsymbol{c}_{\mathrm{o},i} \\ &= \frac{1}{\boldsymbol{v}_i^{\mathrm{T}} \mathbf{R}_{L+1}^{-1} \boldsymbol{v}_i} \end{aligned} \tag{7.85}$$

is the interpolation-error power.

From (7.84) we find,

$$\frac{\boldsymbol{c}_{\mathrm{o},i}}{E_i} = \mathbf{R}_{L+1}^{-1} \boldsymbol{v}_i \,, \tag{7.86}$$

hence the $i$-th column of $\mathbf{R}_{L+1}^{-1}$ is $\boldsymbol{c}_{\mathrm{o},i}/E_i$. We can now see that $\mathbf{R}_{L+1}^{-1}$ can be factorized as follows [7.28]:

$$\begin{aligned} \mathbf{R}_{L+1}^{-1} &= \begin{pmatrix} 1 & -c_{\mathrm{o},1,0} & \cdots & -c_{\mathrm{o},L,0} \\ -c_{\mathrm{o},0,1} & 1 & \cdots & -c_{\mathrm{o},L,1} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{\mathrm{o},0,L} & -c_{\mathrm{o},1,L} & \cdots & 1 \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} 1/E_0 & 0 & \cdots & 0 \\ 0 & 1/E_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/E_{L-1} \end{pmatrix} \\ &= \mathbf{C}_{\mathrm{o}}^{\mathrm{T}} \mathbf{D}_{\mathrm{e}}^{-1} \,. \end{aligned} \tag{7.87}$$

Furthermore, since $\mathbf{R}_{L+1}^{-1}$ is a symmetric matrix, (7.87) can be written as,

$$\mathbf{R}_{L+1}^{-1} = \begin{pmatrix} 1/E_0 & 0 & \cdots & 0 \\ 0 & 1/E_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/E_{L-1} \end{pmatrix} \cdot \begin{pmatrix} 1 & -c_{\mathrm{o},0,1} & \cdots & -c_{\mathrm{o},0,L} \\ -c_{\mathrm{o},1,0} & 1 & \cdots & -c_{\mathrm{o},1,L} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{\mathrm{o},L,0} & -c_{\mathrm{o},L,1} & \cdots & 1 \end{pmatrix} = \mathbf{D}_{\mathrm{e}}^{-1}\mathbf{C}_{\mathrm{o}} \,. \tag{7.88}$$

Therefore, we deduce that,

$$\frac{c_{\mathrm{o},i,l}}{E_i} = \frac{c_{\mathrm{o},l,i}}{E_l} \,, \quad i,l = 0,1,\cdots,L \,. \tag{7.89}$$

The first and last columns of $\mathbf{R}_{L+1}^{-1}$ contain, respectively, the normalized forward and backward predictors and all the columns between contain the normalized interpolators.

We are now going to show how the condition number of the correlation matrix depends on the interpolators. The condition number of the matrix $\mathbf{R}_{L+1}$ is defined as [7.29]:

$$\chi(\mathbf{R}_{L+1}) = \|\mathbf{R}_{L+1}\| \left\|\mathbf{R}_{L+1}^{-1}\right\| \,, \tag{7.90}$$

where $\|\cdot\|$ can be any matrix norm. Note that $\chi(\mathbf{R})$ depends on the underlying norm. Let us compute $\chi(\mathbf{R}_{L+1})$ using the Frobenius norm:

$$\begin{aligned} \|\mathbf{R}_{L+1}\|_{\mathrm{F}} &= \left[\mathrm{tr}\left(\mathbf{R}_{L+1}^{\mathrm{T}}\mathbf{R}_{L+1}\right)\right]^{1/2} \\ &= \left[\mathrm{tr}\left(\mathbf{R}_{L+1}^{2}\right)\right]^{1/2} \end{aligned} \tag{7.91}$$

and

$$\left\|\mathbf{R}_{L+1}^{-1}\right\|_{\mathrm{F}} = \left[\mathrm{tr}\left(\mathbf{R}_{L+1}^{-2}\right)\right]^{1/2} \,. \tag{7.92}$$

From (7.86), we have,

$$\frac{\boldsymbol{c}_{\mathrm{o},i}^{\mathrm{T}}\boldsymbol{c}_{\mathrm{o},i}}{E_i^2} = \boldsymbol{v}_i^{\mathrm{T}}\mathbf{R}_{L+1}^{-2}\boldsymbol{v}_i \,, \tag{7.93}$$

which implies that,

$$\begin{aligned} \sum_{i=0}^{L}\frac{\boldsymbol{c}_{\mathrm{o},i}^{\mathrm{T}}\boldsymbol{c}_{\mathrm{o},i}}{E_i^2} &= \sum_{i=0}^{L}\boldsymbol{v}_i^{\mathrm{T}}\mathbf{R}_{L+1}^{-2}\boldsymbol{v}_i \\ &= \mathrm{tr}\left(\mathbf{R}_{L+1}^{-2}\right) \,. \end{aligned} \tag{7.94}$$

Also, we can easily check that,

$$\mathrm{tr}\left(\mathbf{R}_{L+1}^{2}\right) = (L+1)r^2(0) + 2\sum_{l=1}^{L}(L+1-l)r^2(l) \,. \tag{7.95}$$

Therefore, the square of the condition number of the correlation matrix associated with the Frobenius norm is

$$\chi_{\mathrm{F}}^2(\mathbf{R}_{L+1}) = \left[(L+1)r^2(0) + 2\sum_{l=1}^{L}(L+1-l)r^2(l)\right] \times \sum_{i=0}^{L}\frac{\boldsymbol{c}_{\mathrm{o},i}^{\mathrm{T}}\boldsymbol{c}_{\mathrm{o},i}}{E_i^2} \,. \tag{7.96}$$

Some other interesting relations between the forward predictors and the condition number can be found in [7.30].

To conclude this section, we would like to let readers know that several algorithms exist to compute the optimal predictors efficiently, see for example, [7.31] and [7.32]. All these algorithms are based on Levinson–Durbin recursions.

## 7.8 Line Spectrum Pair Representation

Line spectrum pair (LSP) representation, first introduced by *Itakura* [7.33], is a more-robust way to represent the coefficients of linear predictive models. The LSP polynomials have some very interesting properties shown in [7.34].

A polynomial $P(z)$ of order $L$ is said to be symmetric if

$$P(z) = z^{-L}P(z^{-1}) \tag{7.97}$$

and a polynomial $Q(z)$ is antisymmetric if

$$Q(z) = -z^{-L}Q(z^{-1}) \,. \tag{7.98}$$

Let

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_L z^{-L} \tag{7.99}$$

be the optimal polynomial predictor of order $L$. It is well known that in speech compression the coefficients of this polynomial are inappropriate for quantization because of their relatively large dynamic range and also because, as stated earlier, quantization can change a stable LPC filter into an unstable one [7.35]. From (7.99), we can construct two artificial $(L+1)$-th-order (symmetric and antisymmetric) polynomials by setting the $(L+1)$-th

reflection coefficient, $\kappa_{L+1}$, to be $+1$ and $-1$. These two cases correspond, respectively, to an entirely closed or to an entirely open end at the last section of an acoustic tube of $L+1$ piecewise-uniform sections [7.35],

$$P(z) = A(z) + z^{-L}A(z^{-1}) , \tag{7.100}$$

$$Q(z) = A(z) - z^{-L}A(z^{-1}) . \tag{7.101}$$

The polynomial $A(z)$ can be easily reconstructed from $P(z)$ and $Q(z)$ by

$$A(z) = \frac{1}{2}[P(z) + Q(z)] . \tag{7.102}$$

It was proved in [7.36] and [7.34] that the LSP polynomials, $P(z)$ and $Q(z)$, have the following important properties:

- all zeros of LSP polynomials are on the unit circle,
- the zeros of $P(z)$ and $Q(z)$ are interlaced, and
- the minimum-phase property of $A(z)$ can be easily preserved if the first two properties are intact after quantization.

Now, define the two prediction error signals:

$$e^{+}(k) = x(n) - \frac{1}{2}[\boldsymbol{x}(n-1) + \boldsymbol{x}(n)]^{\mathrm{T}}\boldsymbol{a}^{+}, \tag{7.103}$$

$$e^{-}(k) = x(n) - \frac{1}{2}[\boldsymbol{x}(n-1) - \boldsymbol{x}(n)]^{\mathrm{T}}\boldsymbol{a}^{-} . \tag{7.104}$$

It is shown in [7.37] and [7.38] that the LSP polynomials, whose trivial zeroes have been removed, are equivalent to the two optimal Wiener predictors $\boldsymbol{a}_{\mathrm{o}}^{+}$ and $\boldsymbol{a}_{\mathrm{o}}^{-}$. This is easy to see if we rewrite, $e^{+}(k)$ for example as

$$\begin{aligned} e^{+}(k) &= \left[1 \quad -\frac{\boldsymbol{a}^{+T}}{2}\right]\boldsymbol{x}_{L+1}(n) \\ &\quad + \left[-\frac{\boldsymbol{a}^{+T}}{2}\ 0\right]\boldsymbol{x}_{L+1}(n) \\ &= \boldsymbol{g}^{\mathrm{T}}\mathbf{I}_{\mathrm{d}}^{\mathrm{T}}\boldsymbol{x}_{L+1}(n) , \end{aligned} \tag{7.105}$$

where

$$\mathbf{I}_{\mathrm{d}} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{7.106}$$

and

$$\boldsymbol{g} = \begin{pmatrix} 1 \\ -\dfrac{\boldsymbol{a}^{+}}{2} \end{pmatrix} . \tag{7.107}$$

By minimizing the MSE, $E\{e^{+2}(k)\}$, with respect to $\boldsymbol{g}$, with the constraint $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{v}_1 = 1$, one can find the most important results. For readers who are interested in more details on the properties of LSP polynomials, we recommend the paper by *Bäckström* and *Magi* [7.39].

## 7.9 Multichannel Linear Prediction

Multichannel linear prediction can be very useful in stereo or multichannel speech compression. In an increasing number of speech or audio applications, we have at least two channels available, which are often highly correlated with each other. Therefore, it makes sense to take this interchannel correlation into account in order to obtain more-efficient compression schemes. Multichannel linear prediction is the best way to do this.

Let

$$\chi(k) = \left[x_1(k) \ \ x_2(k) \ \cdots \ x_M(k)\right]^{\mathrm{T}}$$

be a real, zero-mean, stationary $M$-channel time series. We define the multichannel forward prediction error vector as,

$$\begin{aligned} \boldsymbol{e}_{\mathrm{f},L}(k) &= \chi(k) - \hat{\chi}(k) \\ &= \chi(k) - \sum_{l=1}^{L}\mathbf{A}_{L,l}\chi(k-l) \\ &= \chi(k) - \mathbf{A}_{L}^{\mathrm{T}}\boldsymbol{x}(k-1) , \end{aligned} \tag{7.108}$$

where

$$\mathbf{A}_L = [\mathbf{A}_{L,1} \ \ \mathbf{A}_{L,2} \ \cdots \ \mathbf{A}_{L,L}]^{\mathrm{T}}$$

is the forward predictor matrix of size $ML \times M$, each one of the square matrices $\mathbf{A}_{L,l}$ is of size $M \times M$, and

$$\boldsymbol{x}(k-1) = [\chi^{\mathrm{T}}(k-1) \ \ \chi^{\mathrm{T}}(k-2) \ \cdots \ \chi^{\mathrm{T}}(k-L)]^{\mathrm{T}}$$

is a vector of length $ML$. (For convenience, some of the notation used in this section is the same as that in the previous sections.)

Amazon EX1010 Page 160

To derive the optimal Wiener forward predictors, we need to minimize the MSE,

$$J_{\mathrm{f}}(\mathbf{A}_L) = E\left\{\boldsymbol{e}_{\mathrm{f},L}^{\mathrm{T}}(k)\boldsymbol{e}_{\mathrm{f},L}(k)\right\} . \tag{7.109}$$

We find the multichannel Wiener–Hopf equations:

$$\mathbf{R}_L\mathbf{A}_{\mathrm{o},L} = \mathbf{R}_{\mathrm{f}}(1/L) , \tag{7.110}$$

where

$$\begin{aligned}\mathbf{R}_L &= E\{\boldsymbol{x}(k-1)\boldsymbol{x}^{\mathrm{T}}(k-1)\} \qquad (7.111)\\ &= E\{\boldsymbol{x}(k)\boldsymbol{x}^{\mathrm{T}}(k)\}\\ &= \begin{pmatrix} \mathbf{R}(0) & \mathbf{R}(1) & \cdots & \mathbf{R}(L-1)\\ \mathbf{R}^{\mathrm{T}}(1) & \mathbf{R}(0) & \cdots & \mathbf{R}(L-2)\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{R}^{\mathrm{T}}(L-1) & \mathbf{R}^{\mathrm{T}}(L-2) & \cdots & \mathbf{R}(0)\end{pmatrix} \qquad (7.112)\end{aligned}$$

is the block-Toeplitz covariance matrix of size ML×ML,

$$\begin{aligned}&\mathbf{R}(l) = E\{\chi(k)\chi^{\mathrm{T}}(k-l)\}, \quad l = 0, 1, \cdots, L-1 ,\\ &\mathbf{R}(-l) = E\{\chi(k-l)\chi^{\mathrm{T}}(k)\} = \mathbf{R}^{\mathrm{T}}(l) ,\end{aligned}$$

and

$$\begin{aligned}\mathbf{R}_{\mathrm{f}}(1/L) &= [\mathbf{R}(1)\ \mathbf{R}(2)\ \cdots\ \mathbf{R}(L)]^{\mathrm{T}}\\ &= E\{\boldsymbol{x}(k-1)\chi^{\mathrm{T}}(k)\}\end{aligned}$$

is the intercorrelation matrix of size ML×M.

Using the augmented block-Toeplitz covariance matrix of size (ML+M)×(ML+M):

$$\mathbf{R}_{L+1} = \begin{pmatrix} \mathbf{R}(0) & \mathbf{R}_{\mathrm{f}}^{\mathrm{T}}(1/L)\\ \mathbf{R}_{\mathrm{f}}(1/L) & \mathbf{R}_L\end{pmatrix} , \tag{7.113}$$

we deduce the augmented multichannel Wiener–Hopf equations:

$$\mathbf{R}_{L+1}\begin{pmatrix}\mathbf{I}_{M\times M}\\ -\mathbf{A}_{\mathrm{o},L}\end{pmatrix} = \begin{pmatrix}\mathbf{E}_{\mathrm{f},L}\\ \mathbf{0}_{\mathrm{ML}\times M}\end{pmatrix} , \tag{7.114}$$

where $\mathbf{I}_{M\times M}$ is the identity matrix of size $M\times M$ and

$$\begin{aligned}\mathbf{E}_{\mathrm{f},L} &= E\left\{\boldsymbol{e}_{\mathrm{f,o},L}(k)\boldsymbol{e}_{\mathrm{f,o},L}^{\mathrm{T}}(k)\right\}\\ &= \mathbf{R}(0) - \mathbf{R}_{\mathrm{f}}^{\mathrm{T}}(1/L)\mathbf{A}_{\mathrm{o},L} \qquad (7.115)\end{aligned}$$

is the forward error covariance matrix of size $M\times M$, with

$$\boldsymbol{e}_{\mathrm{f,o},L}(k) = \chi(k) - \mathbf{A}_{\mathrm{o},L}^{\mathrm{T}}\boldsymbol{x}(k-1) . \tag{7.116}$$

We will proceed with the same philosophy to derive important equations for the multichannel backward prediction. We define the multichannel backward prediction error vector as

$$\begin{aligned}\boldsymbol{e}_{\mathrm{b},L}(k) &= \chi(k-L) - \hat{\chi}(k-L)\\ &= \chi(k-L) - \sum_{l=1}^{L}\mathbf{B}_{L,l}\chi(k-l+1)\\ &= \chi(k-L) - \mathbf{B}_L^{\mathrm{T}}\boldsymbol{x}(k) , \qquad (7.117)\end{aligned}$$

where

$$\mathbf{B}_L = [\mathbf{B}_{L,1}\ \mathbf{B}_{L,2}\ \cdots\ \mathbf{B}_{L,L}]^{\mathrm{T}}$$

is the backward predictor matrix of size ML×M with each one of the square submatrices $\mathbf{B}_{L,l}$ being of size $M\times M$.

The minimization of the MSE,

$$J_{\mathrm{b}}(\mathbf{B}_L) = E\left\{\boldsymbol{e}_{\mathrm{b},L}^{\mathrm{T}}(k)\boldsymbol{e}_{\mathrm{b},L}(k)\right\} , \tag{7.118}$$

leads to the multichannel Wiener–Hopf equations for the backward prediction:

$$\mathbf{R}_L\mathbf{B}_{\mathrm{o},L} = \mathbf{R}_{\mathrm{b}}(1/L) , \tag{7.119}$$

where

$$\begin{aligned}\mathbf{R}_{\mathrm{b}}(1/L) &= E\{\boldsymbol{x}(k)\chi^{\mathrm{T}}(k-L)\} \qquad (7.120)\\ &= [\mathbf{R}^{\mathrm{T}}(L)\ \mathbf{R}^{\mathrm{T}}(L-1)\ \cdots\ \mathbf{R}^{\mathrm{T}}(1)]^{\mathrm{T}} .\end{aligned}$$

By using the augmented block-Toeplitz covariance matrix:

$$\mathbf{R}_{L+1} = \begin{pmatrix}\mathbf{R}_L & \mathbf{R}_{\mathrm{b}}(1/L)\\ \mathbf{R}_{\mathrm{b}}^{\mathrm{T}}(1/L) & \mathbf{R}(0)\end{pmatrix} , \tag{7.121}$$

we find the augmented multichannel Wiener–Hopf equations:

$$\mathbf{R}_{L+1}\begin{pmatrix}-\mathbf{B}_{\mathrm{o},L}\\ \mathbf{I}_{M\times M}\end{pmatrix} = \begin{pmatrix}\mathbf{0}_{\mathrm{ML}\times M}\\ \mathbf{E}_{\mathrm{b},L}\end{pmatrix} , \tag{7.122}$$

where

$$\begin{aligned}\mathbf{E}_{\mathrm{b},L} &= E\left\{\boldsymbol{e}_{\mathrm{b,o},L}(k)\boldsymbol{e}_{\mathrm{b,o},L}^{\mathrm{T}}(k)\right\}\\ &= \mathbf{R}(0) - \mathbf{R}_{\mathrm{b}}^{\mathrm{T}}(1/L)\mathbf{B}_{\mathrm{o},L} \qquad (7.123)\end{aligned}$$

is the backward error covariance matrix of size $M\times M$, with

$$\boldsymbol{e}_{\mathrm{b,o},L}(k) = \chi(k-L) - \mathbf{B}_{\mathrm{o},L}^{\mathrm{T}}\boldsymbol{x}(k) . \tag{7.124}$$

Amazon EX1010 Page 161

To solve the multichannel Wiener–Hopf equations efficiently, we need to derive some important relations [7.40]. Consider the following system,

$$\begin{pmatrix} \mathbf{R}_L & \mathbf{R}_{\mathrm{b}}(1/L) \\ \mathbf{R}_{\mathrm{b}}^{\mathrm{T}}(1/L) & \mathbf{R}(0) \end{pmatrix} \begin{pmatrix} \mathbf{I}_{M\times M} \\ -\mathbf{A}_{\mathrm{o},L-1} \\ \mathbf{0}_{M\times M} \end{pmatrix} = \begin{pmatrix} \mathbf{E}_{\mathrm{f},L-1} \\ \mathbf{0}_{(ML-M)\times M} \\ \mathbf{K}_{\mathrm{f},L} \end{pmatrix} , \tag{7.125}$$

where

$$\mathbf{K}_{\mathrm{f},L} = \mathbf{R}^{\mathrm{T}}(L) - \mathbf{R}_{\mathrm{b}}^{\mathrm{T}}(1/L-1)\mathbf{A}_{\mathrm{o},L-1} . \tag{7.126}$$

Consider the other system,

$$\begin{pmatrix} \mathbf{R}(0) & \mathbf{R}_{\mathrm{f}}^{\mathrm{T}}(1/L) \\ \mathbf{R}_{\mathrm{f}}(1/L) & \mathbf{R}_L \end{pmatrix} \begin{pmatrix} \mathbf{0}_{M\times M} \\ -\mathbf{B}_{\mathrm{o},L-1} \\ \mathbf{I}_{M\times M} \end{pmatrix} = \begin{pmatrix} \mathbf{K}_{\mathrm{b},L} \\ \mathbf{0}_{(ML-M)\times M} \\ \mathbf{E}_{\mathrm{b},L-1} \end{pmatrix} , \tag{7.127}$$

where

$$\mathbf{K}_{\mathrm{b},L} = \mathbf{R}(L) - \mathbf{R}_{\mathrm{f}}^{\mathrm{T}}(1/L-1)\mathbf{B}_{\mathrm{o},L-1} . \tag{7.128}$$

If we post-multiply both sides of (7.127) by $\mathbf{E}_{\mathrm{b},L-1}^{-1}\mathbf{K}_{\mathrm{f},L}$, we get:

$$\mathbf{R}_{L+1} \begin{pmatrix} \mathbf{0}_{M\times M} \\ -\mathbf{B}_{\mathrm{o},L-1} \\ \mathbf{I}_{M\times M} \end{pmatrix} \mathbf{E}_{\mathrm{b},L-1}^{-1}\mathbf{K}_{\mathrm{f},L} = \begin{pmatrix} \mathbf{K}_{\mathrm{b},L}\mathbf{E}_{\mathrm{b},L-1}^{-1}\mathbf{K}_{\mathrm{f},L} \\ \mathbf{0}_{(ML-M)\times M} \\ \mathbf{K}_{\mathrm{f},L} \end{pmatrix} . \tag{7.129}$$

Subtracting (7.129) from (7.125) and identifying the resulting system with the augmented multichannel Wiener–Hopf equations for forward prediction [eq. (7.114)], we deduce the two recursions:

$$\mathbf{E}_{\mathrm{f},L} = \mathbf{E}_{\mathrm{f},L-1} - \mathbf{K}_{\mathrm{b},L}\mathbf{E}_{\mathrm{b},L-1}^{-1}\mathbf{K}_{\mathrm{f},L} , \tag{7.130}$$

$$\mathbf{A}_{\mathrm{o},L} = \begin{pmatrix} \mathbf{A}_{\mathrm{o},L-1} \\ \mathbf{0}_{M\times M} \end{pmatrix} - \begin{pmatrix} \mathbf{B}_{\mathrm{o},L-1} \\ -\mathbf{I}_{M\times M} \end{pmatrix} \mathbf{E}_{\mathrm{b},L-1}^{-1}\mathbf{K}_{\mathrm{f},L} . \tag{7.131}$$

Similarly, if we post-multiply both sides of (7.125) by $\mathbf{E}_{\mathrm{f},L-1}^{-1}\mathbf{K}_{\mathrm{b},L}$, we obtain:

$$\mathbf{R}_{L+1} \begin{pmatrix} \mathbf{I}_{M\times M} \\ -\mathbf{A}_{\mathrm{o},L-1} \\ \mathbf{0}_{M\times M} \end{pmatrix} \mathbf{E}_{\mathrm{f},L-1}^{-1}\mathbf{K}_{\mathrm{b},L} = \begin{pmatrix} \mathbf{K}_{\mathrm{b},L} \\ \mathbf{0}_{(ML-M)\times M} \\ \mathbf{K}_{\mathrm{f},L}\mathbf{E}_{\mathrm{f},L-1}^{-1}\mathbf{K}_{\mathrm{b},L} \end{pmatrix} . \tag{7.132}$$

Subtracting (7.132) from (7.127) and identifying the resulting system with the augmented multichannel Wiener–Hopf equations for backward prediction (7.122), we deduce the two recursions:

$$\mathbf{E}_{\mathrm{b},L} = \mathbf{E}_{\mathrm{b},L-1} - \mathbf{K}_{\mathrm{f},L}\mathbf{E}_{\mathrm{f},L-1}^{-1}\mathbf{K}_{\mathrm{b},L} , \tag{7.133}$$

$$\mathbf{B}_{\mathrm{o},L} = \begin{pmatrix} \mathbf{0}_{M\times M} \\ \mathbf{B}_{\mathrm{o},L-1} \end{pmatrix} - \begin{pmatrix} -\mathbf{I}_{M\times M} \\ \mathbf{A}_{\mathrm{o},L-1} \end{pmatrix} \mathbf{E}_{\mathrm{f},L-1}^{-1}\mathbf{K}_{\mathrm{b},L} . \tag{7.134}$$

Relations (7.130), (7.131), (7.133), and (7.134) were independently discovered by *Whittle* [7.41] and *Wiggins* and *Robinson* [7.42].

Another important relation needs to be found. Indeed, using (7.116) and (7.124), we can easily verify,

$$E\left\{\boldsymbol{e}_{\mathrm{f,o},L-1}(k)\boldsymbol{e}_{\mathrm{b,o},L-1}^{\mathrm{T}}(k-1)\right\} = \mathbf{K}_{\mathrm{b},L} , \tag{7.135}$$

$$E\left\{\boldsymbol{e}_{\mathrm{b,o},L-1}(k-1)\boldsymbol{e}_{\mathrm{f,o},L-1}^{\mathrm{T}}(k)\right\} = \mathbf{K}_{\mathrm{f},L} , \tag{7.136}$$

which implies that,

$$\mathbf{K}_{\mathrm{b},L} = \mathbf{K}_{\mathrm{f},L}^{\mathrm{T}} . \tag{7.137}$$

Table 7.2 summarizes the Levinson–Wiggins–Robinson algorithm [7.42–44], which is a generalization of the Levinson–Durbin algorithm to the multichannel case.

**Table 7.2** Levinson–Wiggins–Robinson algorithm

**Initialization:** $\mathbf{E}_{\mathrm{f},0} = \mathbf{E}_{\mathrm{b},0} = \mathbf{R}(0)$

**For** $1 \le l \le L$

$$\mathbf{K}_{\mathrm{b},l} = \mathbf{R}(l) - \mathbf{R}_{\mathrm{f}}^{\mathrm{T}}(1:l-1)\mathbf{B}_{\mathrm{o},l-1}$$

$$\mathbf{A}_{\mathrm{o},l} = \begin{bmatrix} \mathbf{A}_{\mathrm{o},l-1} \\ \mathbf{0}_{M\times M} \end{bmatrix} - \begin{bmatrix} \mathbf{B}_{\mathrm{o},l-1} \\ -\mathbf{I}_{M\times M} \end{bmatrix} \mathbf{E}_{\mathrm{b},l-1}^{-1}\mathbf{K}_{\mathrm{b},l}^{\mathrm{T}}$$

$$\mathbf{B}_{\mathrm{o},l} = \begin{bmatrix} \mathbf{0}_{M\times M} \\ \mathbf{B}_{\mathrm{o},l-1} \end{bmatrix} - \begin{bmatrix} -\mathbf{I}_{M\times M} \\ \mathbf{A}_{\mathrm{o},l-1} \end{bmatrix} \mathbf{E}_{\mathrm{f},l-1}^{-1}\mathbf{K}_{\mathrm{b},l}$$

$$\mathbf{E}_{\mathrm{f},l} = \mathbf{E}_{\mathrm{f},l-1} - \mathbf{K}_{\mathrm{b},l}\mathbf{E}_{\mathrm{b},l-1}^{-1}\mathbf{K}_{\mathrm{b},l}^{\mathrm{T}}$$

$$\mathbf{E}_{\mathrm{b},l} = \mathbf{E}_{\mathrm{b},l-1} - \mathbf{K}_{\mathrm{b},l}^{\mathrm{T}}\mathbf{E}_{\mathrm{f},l-1}^{-1}\mathbf{K}_{\mathrm{b},l}$$

Amazon EX1010 Page 162

## 7.10 Conclusions

In this chapter, we have tried to present the most important results in linear prediction for speech. We have explained the principle of forward linear prediction and have shown that the optimal prediction error signal tends to be a white signal. We have extended the principle of forward linear prediction to backward linear prediction and derived the Cholesky factorization of the inverse correlation matrix. We have developed the classical Levinson–Durbin algorithm, which is a very efficient way to solve the Wiener–Hopf equations for the forward and backward prediction coefficients. We have explained the idea behind the lattice predictor. We have shown how the spectrum of a speech signal can easily be estimated thanks to the prediction coefficients. We have given some notions of linear interpolation and have demonstrated how the condition number of the correlation matrix is related to the optimal interpolators. We have also presented some notions of line spectrum pair polynomials. Finally, in the last section, we have generalized some of these ideas to the multichannel case.



### References


7.1 B.S. Atal: The history of linear prediction, IEEE Signal Proc. Mag. **23**, 154–161 (2006)

7.2 L.R. Rabiner, R.W. Schaffer: *Digital Processing of Speech Signals* (Prentice-Hall, Englewood Cliffs 1976)

7.3 J.D. Markel, A.H. Gray Jr.: *Linear Prediction of Speech* (Springer, New York 1976)

7.4 J. Makhoul: Linear prediction: a tutorial review, Proc. IEEE **63**, 561–580 (1975)

7.5 J.W. Picone: Signal modeling techniques in speech recognition, Proc. IEEE **81**, 1215–1247 (1993)

7.6 S.R. Quackenbush, T.P. Barnwell, M.A. Clements: *Objective Measures of Speech Quality* (Prentice-Hall, Englewood Cliffs 1988)

7.7 Y. Huang, J. Benesty, J. Chen: *Acoustic MIMO Signal Processing* (Springer, New York 2006)

7.8 F. Itakura, S. Saito: A statistical method for estimation of speech spectral density and formant frequencies, Electron. Commun. Jpn. **53**(A), 36–43 (1970)

7.9 G. Chen, S.N. Koh, I.Y. Soon: Enhanced Itakura measure incorporating masking properties of human auditory system, Signal Process. **83**, 1445–1456 (2003)

7.10 M.G. Bellanger: *Adaptive Digital Filters and Signal analysis* (Marcel Dekker, New York 1987)

7.11 S. Haykin: *Adaptive Filter Theory*, 4th edn. (Prentice-Hall, Upper Saddle River 2002)

7.12 C.W. Therrien: On the relation between triangular matrix decomposition and linear prediction, Proc. IEEE **71**, 1459–1460 (1983)

7.13 N. Levinson: The Wiener RMS (root mean square) error criterion in filter design and prediction, J. Math. Phys. **25**(4), 261–278 (1947), Also Appendix B, in N. Wiener, Extrapolation, Interpolation and Smoothing of Stationary Time Series (MIT, Cambridge 1949)

7.14 J. Durbin: Efficient estimation of parameters in moving-average models, Biometrika **46**, 306–316 (1959), Parts 1 and 2

7.15 J. Durbin: The fitting of time-series models, Rev. Inst. Int. Stat. **28**(3), 233–243 (1960)

7.16 J.P. Burg: *Maximum Entropy Spectral Analysis Ph.D. Dissertation* (Stanford Univ., Stanford 1975)

7.17 P. Delsarte, Y.V. Genin: The split Levinson algorithm, IEEE Trans. Acoust. Speech **ASSP-34**, 470–478 (1986)

7.18 R. Kumar: A fast algorithm for solving a Toeplitz system of equations, IEEE Trans. Acoust. Speech **ASSP-33**, 254–265 (1985)

7.19 J. Makhoul: Stable and efficient lattice methods for linear prediction, IEEE Trans. Acoust. Speech **ASSP-25**, 423–428 (1977)

7.20 F. Itakura, S. Saito: Digital filtering techniques for speech analysis and synthesis, Proc. 7th Int. Conf. Acoust. **25**(C-1), 261–264 (1971)

7.21 S.W. Lang, J.H. McClellan: A simple proof of stability for all-pole linear prediction models, Proc. IEEE **67**, 860–861 (1979)

7.22 M.H. Hayes: *Statistical Digital Signal Processing and Modeling* (Wiley, New York 1996)

7.23 J. Makhoul: Spectral analysis of speech by linear prediction, IEEE Trans. Acoust. Speech **AU-21**(3), 140–148 (1973)

7.24 S.M. Kay, S.L. Marple Jr.: Spectrum analysis – a modern perspective, Proc. IEEE **69**, 1380–1419 (1981)

7.25 J. M.Cadzow: Spectral estimation: an overdetermined rational model equation approach, Proc. IEEE **70**, 907–939 (1982)

7.26 S. Kay: Some results in linear interpolation theory, IEEE Trans. Acoust. Speech **ASSP-31**, 746–749 (1983)

7.27 B. Picinbono, J.-M. Kerilis: Some properties of prediction and interpolation errors, IEEE Trans. Acoust. Speech **ASSP-36**, 525–531 (1988)

7.28 J. Benesty, T. Gaensler: New insights into the RLS algorithm, EURASIP J. Appl. Si. Pr. **2004**, 331–339 (2004)

7.29 G.H. Golub, C.F. Van Loan: *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore 1996)

7.30 J. Benesty, T. Gaensler: Computation of the condition number of a non-singular symmetric Toeplitz matrix with the Levinson-Durbin algorithm, IEEE Trans. Signal Proces. **54**, 2362–2364 (2006)


Amazon EX1010 Page 163

7.31 C.K. Coursey, J.A. Stuller: Linear interpolation lattice, IEEE Trans. Signal Proces. **39**, 965–967 (1991)

7.32 M.R.K. Khansari, A. Leon-Garcia: A fast algorithm for optimal linear interpolation, IEEE Trans. Signal Process. **41**, 2934–2937 (1993)

7.33 F. Itakura: Line spectrum representation of linear predictive coefficients of speech signal, J. Acoust. Soc. Am. **57**(1), 35 (1975)

7.34 F.K. Soong, B.-H. Juang: Line spectrum pair (LSP) and speech data compression, Proc. ICASSP (1984) pp. 1.10.1–1.10.4

7.35 F.K. Soong, B.-H. Juang: Optimal quantization of LSP paramaters, IEEE Trans. Speech Audio Process. **1**, 15–24 (1993)

7.36 S. Sagayama: Stability condition of LSP speech synthesis didgital filter, Proc. Acoust. Soc. Jpn. (1982) pp. 153–154, (in Japanese)

7.37 B. Kleijn, T. Bäckström, P. Alku: On line spectral frequencies, IEEE Signal Proc. Lett. **10**, 75–77 (2003)

7.38 T. Bäckström, P. Alku, T. Paatero, B. Kleijn: A time-domain interpretation for the LSP decomposition, IEEE Trans. Speech Audio Process. **12**, 554–560 (2004)

7.39 T. Bäckström, C. Magi: Properties of line spectrum pair polynomials – A review, Elsevier Signal Process. **86**, 3286–3298 (2006)

7.40 T. Kailath: A view of three decades of linear filtering theory, IEEE Trans. Inf. Theory **IT-20**, 146–181 (1974)

7.41 P. Whittle: On the fitting of multivariate autoregressions and the approximate canonical factorization of a spectral density matrix, Biometrika **50**, 129–134 (1963)

7.42 R.A. Wiggins, E.A. Robinson: Recursive solution to the multichannel filtering problem, J. Geophys. Res. **70**, 1885–1891 (1965)

7.43 O.N. Strand: Multichannel complex maximum entropy (autoregressive) spectral analysis, IEEE T. Automat. Contr. **AC-22**, 634–640 (1977)

7.44 P. Delsarte, Y.V. Genin: Multichannel singular predictor polynomials, IEEE Trans. Circuits Syst. **35**, 190–200 (1988)


Amazon EX1010 Page 164

# 8. The Kalman Filter

S. Gannot, A. Yeredor

The Kalman filter and its variants are some of the most popular tools in statistical signal processing and estimation theory. In this chapter, we introduce the Kalman filter, providing a succinct, yet rigorous derivation thereof, which is based on the orthogonality principle. We also introduce several important variants of the Kalman filter, namely various Kalman smoothers, a Kalman predictor, a nonlinear extension (the extended Kalman filter), and adaptation to cases of temporally correlated measurement noise.

The application of the Kalman filter to two important speech processing problems, namely, speech enhancement and speaker localization is demonstrated.

8.1 **Derivation of the Kalman Filter** ..... 136
8.1.1 The Minimum Mean Square Linear Optimal Estimator ..... 136
8.1.2 The Estimation Error: Necessary and Sufficient Conditions for Optimality 137
8.1.3 The Kalman Filter ..... 138

8.2 **Examples: Estimation of Parametric Stochastic Process from Noisy Observations** ..... 141
8.2.1 Autoregressive (AR) Process ..... 142
8.2.2 Moving-Average (MA) Process ..... 143
8.2.3 Autoregressive Moving-Average (ARMA) Process ..... 143
8.2.4 The Case of Temporally Correlated Noise ..... 143

8.3 **Extensions of the Kalman Filter** ..... 144
8.3.1 The Kalman Predictor ..... 144
8.3.2 The Kalman Smoother ..... 145
8.3.3 The Extended Kalman Filter ..... 148

8.4 **The Application of the Kalman Filter to Speech Processing** ..... 149
8.4.1 Literature Survey ..... 149
8.4.2 Speech Enhancement ..... 151
8.4.3 Speaker Tracking ..... 154

8.5 **Summary** ..... 157

**References** ..... 157



The Kalman filter, together with its basic variants, are some of the most widely applied tools in fields related to statistical signal processing, especially in the context of causal, (nearly) real-time applications. In its fundamental, classical form, Kalman filtering is aimed at sequential (recursive) linear estimation of the state of a linear dynamic system from noisy measurements, linearly related to the unobserved state variables. The general statistical framework of the Kalman filter is Bayesian, namely a priori knowledge of statistical properties (up to second order) of the underlying process is assumed. This knowledge is specified through the mean and covariance of the initial state, and through the recursive model equation, which can be used to propagate these properties through time.

At each time instant, the Kalman filter's output is the optimal *linear* minimum mean square error (MSE) estimator of the state from measurements up to that time instant; moreover, it is the optimal minimum MSE estimator in general (i. e., among all linear and nonlinear estimators), when the both the states and the measurements share a jointly Gaussian probability distribution, specified by the model equations and by the measurement equations. The recursive structure of the filter (dwelling on the recursive structure of the model and measurement equations) is computationally appealing, as it enables to automatically (and optimally) take all past measurements into account, without the need to explicitly remember (namely, spend memory resources on) and account for these measurements. As an important byproduct, the Kalman filter equations also provide expressions for the MSE in the resulting estimates.

The Kalman filter has found numerous applications in fields related to control of dynamic systems. It is also used for estimating and predicting the trajectories of

moving objects, ranging from celestial bodies and missiles to microscopic particles. Since the late 1980s the Kalman filter was adopted for the estimation of speech signals contaminated by additive noise.

Rudolf Emil Kalman was born on 19 May 1930 in Budapest, Hungary (These historical notes are based on *Grewal* and *Andrews*' book [8.1] ). During the second world war his family emigrated to the USA. In November 1958, while working in the Research Institute for Advanced Studies (RIAS), Kalman came up with the fundamental idea of applying state-space notation to the Wiener filter problem. This turned out to be the basis of the celebrated Kalman filter [8.2, 3]. The first known application of the Kalman filter, a trajectory estimation for the Apollo mission, was the work of *Schmidt* [8.4, 5] conducted at the Ames Research Center of the National Aeronautics and Space Administration (NASA). Since then the Kalman filter is an essential part of nearly every trajectory estimation task.

The Kalman filter and its variants (predictor, smoother and nonlinear forms) are widely addressed in the statistical signal processing and control theory literature. There is a plethora of articles and text books on both theoretical and practical aspects. The interested reader is referred to the books [8.1, 6, 7] and [8.8], to name just a few. A survey of associated speech-related papers can be found in Sect. 8.4.

This chapter is aimed at providing an overview of the basic Kalman filter, as well as some of its most popular variants, including concise, yet rigorous derivations of the resulting expressions. In addition, it provides emphasis on the applicability of Kalman filtering in speech-processing-related problems, such as speech enhancement (denoising) and speaker tracking. Although the general framework of Kalman filtering allows for complex-valued state vectors and measurements, we chose to concentrate, for simplicity, on the real-valued case, which is most commonly encountered in speech-related applications.

This chapter is structured as follows. In the next section, we provide the formulation of the estimation problem, as well as a rigorous derivation of the basic Kalman filter equations. In Sect. 8.2, we demonstrate the use of the Kalman filter in common stochastic denoising problems. Section 8.3 offers an overview of classical, immediate extensions of the Kalman filter, namely the Kalman predictor, the Kalman smoother (in various forms), and the extended Kalman filter (which addresses nonlinear models). Section 8.4 is dedicated to applications of Kalman filter in the context of speech processing; it provides a thorough literature survey of recent research activities in this context, as well as detailed examples of two popular applications, namely speech enhancement and speaker tracking.

## 8.1 Derivation of the Kalman Filter

Several approaches can be taken in the derivation of the Kalman filter. Following specification of the model equations and measurement equations, one can assume a Gaussian distribution of the driving processes, as well as of the initial state, and then derive the posterior distributions of the states given the observations, taking the mean of the resulting distributions as the states' estimates, namely as the filter's outputs. Another option is to take a recursive weighted least-squares (WLS) approach combined with special weighting of the previous estimate of the states in the role of additional *measurements*. Both of these approaches, while perfectly legitimate, tend to be somewhat cumbersome in the exposition of the associated derivations.

In this section, we will take a slightly different approach, dwelling on the well-known orthogonality principle, which is a basic property of the optimal linear estimator's estimation error. By showing that the associated optimality properties are automatically inherited from one recursion phase to the next, we would establish the claimed optimality of the entire scheme.

In the following subsection, we will review the optimal linear estimator (in the Bayesian framework) and the associated necessary and sufficient conditions for optimality. We will then exploit these conditions in the derivation of the Kalman filter in the subsequent subsection.

### 8.1.1 The Minimum Mean Square Linear Optimal Estimator

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two random vectors with an arbitrary distribution having finite moments up to second order. Denoting by $\boldsymbol{z} = (\boldsymbol{x}^{\mathrm{T}} \ \boldsymbol{y}^{\mathrm{T}})^{\mathrm{T}}$ the concatenation of the two vectors into a single vector, the mean and covariance of $\boldsymbol{z}$ are then given by

$$\boldsymbol{\eta}_z = E(\boldsymbol{z}) = \begin{pmatrix} \boldsymbol{\eta}_x \\ \boldsymbol{\eta}_y \end{pmatrix} \tag{8.1a}$$

Amazon EX1010 Page 166

$$\mathbf{C}_{zz} = E\big[(z-\boldsymbol{\eta}_z)(z-\boldsymbol{\eta}_z)^{\mathrm{T}}\big] = \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{pmatrix} \tag{8.1b}$$

with obvious partitions according to the respective dimensions, where $\boldsymbol{\eta}_x$ and $\boldsymbol{\eta}_y$ denote the means of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, $\mathbf{C}_{xx}$, $\mathbf{C}_{yy}$, denote the covariance matrices of $\boldsymbol{x}$, $\boldsymbol{y}$, respectively, and $\mathbf{C}_{xy} = \mathbf{C}_{yx}^{\mathrm{T}}$ denotes the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Assume now that it is desired to estimate $\boldsymbol{x}$ from observation of a $\boldsymbol{y}$ using linear estimation of the form

$$\hat{\boldsymbol{x}} = \mathbf{A}\boldsymbol{y} + \boldsymbol{b}\,, \tag{8.2}$$

where $\hat{\boldsymbol{x}}$ denotes the estimate of $\boldsymbol{x}$, and $\mathbf{A}$ and $\boldsymbol{b}$ are some fixed matrix and vector, respectively, of the appropriate dimensions. Note that any linear estimator can be represented in such a form, and its properties are uniquely determined by $\mathbf{A}$ and $\boldsymbol{b}$.

Assume further that it is desired to find the linear estimator that attains the smallest (matrix) MSE among all linear estimators. More specifically, denoting

$$\boldsymbol{\epsilon} = \hat{\boldsymbol{x}} - \boldsymbol{x} \tag{8.3}$$

as the estimation error, the (matrix) MSE is defined as

$$\mathbf{P} = E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}})\,. \tag{8.4}$$

An estimator $\hat{\boldsymbol{x}}_1$ is said to attain a smaller (matrix) MSE than another estimator, say $\hat{\boldsymbol{x}}_2$, when the matrix $\mathbf{P}_1$ attained by $\hat{\boldsymbol{x}}_1$ is smaller than the matrix $\mathbf{P}_2$ attained by $\hat{\boldsymbol{x}}_2$, in the sense that the difference matrix $\mathbf{P}_2 - \mathbf{P}_1$ is positive (semi-)definite.

Let us find $\mathbf{A}$ and $\boldsymbol{b}$ which minimize the MSE. For any $\mathbf{A}$, $\boldsymbol{b}$, the estimation error is given by

$$\begin{aligned} \boldsymbol{\epsilon} &= \hat{\boldsymbol{x}} - \boldsymbol{x} = \mathbf{A}\boldsymbol{y} + \boldsymbol{b} - \boldsymbol{x} \\ &= \mathbf{A}(\boldsymbol{y} - \boldsymbol{\eta}_y) - (\boldsymbol{x} - \boldsymbol{\eta}_x) + \boldsymbol{c}\,, \end{aligned} \tag{8.5}$$

where $\boldsymbol{c}$ is a constant vector, defined as

$$\boldsymbol{c} = \boldsymbol{b} + \mathbf{A}\boldsymbol{\eta}_y - \boldsymbol{\eta}_x\,. \tag{8.6}$$

We thus obtain

$$\begin{aligned} \mathbf{P} &= E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}}) = E\big[(\mathbf{A}(\boldsymbol{y} - \boldsymbol{\eta}_y) - (\boldsymbol{x} - \boldsymbol{\eta}_x) + \boldsymbol{c}) \\ &\quad (\mathbf{A}(\boldsymbol{y} - \boldsymbol{\eta}_y) - (\boldsymbol{x} - \boldsymbol{\eta}_x) + \boldsymbol{c})^{\mathrm{T}}\big] \\ &= \mathbf{A}\mathbf{C}_{yy}\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{C}_{yx} - \mathbf{C}_{xy}\mathbf{A}^{\mathrm{T}} + \mathbf{C}_{xx} + \boldsymbol{c}\boldsymbol{c}^{\mathrm{T}}\,. \end{aligned} \tag{8.7}$$

Since the term $\boldsymbol{c}\boldsymbol{c}^{\mathrm{T}}$ is positive semidefinite, $\mathbf{P}$ would definitely be minimized with respect to $\boldsymbol{c}$ (controlled by $\boldsymbol{b}$) by setting $\boldsymbol{c} = \mathbf{0}$, obtained by choosing

$$\boldsymbol{b} = \boldsymbol{\eta}_x - \mathbf{A}\boldsymbol{\eta}_y\,. \tag{8.8}$$

It now remains to minimize

$$\mathbf{P} = \mathbf{A}\mathbf{C}_{yy}\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{C}_{yx} - \mathbf{C}_{xy}\mathbf{A}^{\mathrm{T}} + \mathbf{C}_{xx} \tag{8.9}$$

with respect to $\mathbf{A}$. Assuming that $\mathbf{C}_{yy}$ is invertible, we note that $\mathbf{P}$ in (8.9) can also be written as

$$\begin{aligned} \mathbf{P} &= \big(\mathbf{A} - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\big)\mathbf{C}_{yy}\cdot\big(\mathbf{A} - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\big)^{\mathrm{T}} \\ &\quad + \mathbf{C}_{xx} - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\,. \end{aligned} \tag{8.10}$$

If $\mathbf{C}_{yy}$ is singular, this means that the measurements vector $\boldsymbol{y}$ contains some redundancy, as one or more linear combinations of its elements are zero (in the mean-squared sense). This means in turn, that $\boldsymbol{y}$ can be reduced into a smaller measurements vector with non-singular covariance, without loss in the attainable MSE in estimating $\boldsymbol{x}$. The first term in (8.10) is again positive semi-definite, and may be set to zero by selecting

$$\mathbf{A} = \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\,. \tag{8.11}$$

Since the other two terms do not depend on $\mathbf{A}$ (or on $\boldsymbol{b}$), this is the global minimizing solution, and the remainder is the residual (minimal) MSE matrix. Combining (8.8) and (8.11), we obtain the optimal linear estimator

$$\hat{\boldsymbol{x}} = \boldsymbol{\eta}_x + \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\eta}_y)\,, \tag{8.12}$$

whose MSE is given by (8.10):

$$\mathbf{P} = \mathbf{C}_{xx} - \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\,. \tag{8.13}$$

### 8.1.2 The Estimation Error: Necessary and Sufficient Conditions for Optimality

Recall the estimation error (8.5) of any linear estimator:

$$\boldsymbol{\epsilon} = \mathbf{A}(\boldsymbol{y} - \boldsymbol{\eta}_y) - (\boldsymbol{x} - \boldsymbol{\eta}_x) + \boldsymbol{c}\,. \tag{8.14}$$

We obviously have

$$E(\boldsymbol{\epsilon}) = \boldsymbol{c} \tag{8.15}$$

and

$$E(\boldsymbol{\epsilon}\boldsymbol{y}^{\mathrm{T}}) = \mathbf{A}\mathbf{C}_{yy} - \mathbf{C}_{xy} + \boldsymbol{c}\boldsymbol{\eta}_y^{\mathrm{T}}\,. \tag{8.16}$$

When substituting (8.8) and (8.11) into (8.15) and (8.16) we easily obtain $E(\boldsymbol{\epsilon}) = \mathbf{0}$ and $E(\boldsymbol{\epsilon}\boldsymbol{y}^{\mathrm{T}}) = \mathbf{0}$. Moreover, it can be seen from (8.15) and (8.16) that, for any linear estimator, $E(\boldsymbol{\epsilon}) = \mathbf{0}$ and $E(\boldsymbol{\epsilon}\boldsymbol{y}^{\mathrm{T}}) = \mathbf{0}$ imply (respectively) (8.8) and (8.11), which in turn imply the MSE optimality of the estimator. We may therefore conclude with the following necessary and sufficient conditions for the optimal linear estimator.



Amazon EX1010 Page 167

*Theorem 8.1*

*Let $\hat{\boldsymbol{x}}$ denote an estimator of a random vector $\boldsymbol{x}$ from the random vector (*measurements*) $\boldsymbol{y}$, and let $\boldsymbol{\epsilon} = \hat{\boldsymbol{x}} - \boldsymbol{x}$ denote the estimation error. $\hat{\boldsymbol{x}}$ is the optimal linear estimator of $\boldsymbol{x}$ from $\boldsymbol{y}$ in the sense of minimum MSE if and only if the following three conditions hold:*

1. *$\hat{\boldsymbol{x}}$ is a linear function of $\boldsymbol{y}$,*
2. *$E(\boldsymbol{\epsilon}) = \mathbf{0}$,*
3. *$E(\boldsymbol{\epsilon}\boldsymbol{y}^T) = \mathbf{0}$.*

The condition of linearity is trivial. The zero-mean condition asserts that there is no constant bias in the optimal linear estimation, since any such bias can be removed in a linear operation, which would thereby reduce the MSE. The third condition asserts that there should be no correlation between any of the estimation error's elements and any of the measurements. This is a fundamental condition for MSE optimality of the optimal linear estimator: if any of its estimation error's elements were correlated with any of the measurements, then this would (intuitively) mean that by observing these measurements one could deduce the mean direction of departure of the respective errors from their (zero) mean. Such knowledge would in turn imply that a linear estimator of the respective estimation errors can be constructed and *appended* to the original estimator, so as to further reduce the MSE without breeching the linearity framework. If such an operation is possible, the original estimator cannot be MSE-optimal.

Therefore, the orthogonality condition, together with the zero-mean and linearity conditions, imply that all linear operations that may potentially reduce the MSE have been exhausted, and no further linear operations may be able to reduce the MSE further. Naturally, this implies MSE optimality of the respective linear estimator. In the sequel, we shall exploit these three conditions in constructively designing the estimator to satisfy all three, thereby inducing its optimality.

### 8.1.3 The Kalman Filter

Quite commonly, it is required to estimate time-dependent random vectors $\boldsymbol{x}_n$ from measurements $\boldsymbol{y}_n$ obtained sequentially in time (where $n$ is the time index). Although the expressions derived above for the linear estimator's parameters (the matrix $\mathbf{A}$ and vector $\boldsymbol{b}$) can always be used, the associated computation of covariance matrices (and their inversion) may become prohibitively computationally demanding in such applications, especially as the dimension of the accumulated measurements vector placed in a matrix $\mathbf{Y} = (\boldsymbol{y}_1\ \boldsymbol{y}_2\ \cdots\ \boldsymbol{y}_n)$ increases. Even if these matrices are computed offline, the required storage space has to grow with the increase in the dimension of $\boldsymbol{y}$, as more measurements become available.

Often, however, the propagation of the statistics of the process of interest (together with the measurements) can be deduced from a recursive description of the model. It would therefore be computationally appealing if it were possible to apply a recursive scheme, in which the optimal linear estimator takes a similar form, with simplified computation of the associated matrices. Such a recursive (linear, optimal) estimation scheme is known as Kalman filtering.

Assume that the underlying process of interest satisfies the following recursive *model equations*:

$$\boldsymbol{x}_n = \boldsymbol{\Phi}_n \boldsymbol{x}_{n-1} + \boldsymbol{w}_n, \quad n = 1, 2, \ldots, \tag{8.17}$$

where $\boldsymbol{\Phi}_n$ denotes the (known) transition matrix from the state at time instant $n-1$ to the state at time instant $n$ and $\boldsymbol{w}_n$ denotes the stochastic *driving noise* at time instant $n$, assumed to be of zero mean with known covariance $\mathbf{Q}_n$. The initial conditions for this set of equation are the mean $\boldsymbol{m}_0 = E(\boldsymbol{x}_0)$ and covariance $\mathbf{P}_0 = E\left((\boldsymbol{x}_0 - \boldsymbol{m}_0)(\boldsymbol{x}_0 - \boldsymbol{m}_0)^{\mathrm{T}}\right)$ of the initial state $\boldsymbol{x}_0$.

Assume further that the measurements $\boldsymbol{y}_n$ are related to the states $\boldsymbol{x}_n$ via the following *measurement equations*:

$$\boldsymbol{y}_n = \mathbf{H}_n \boldsymbol{x}_n + \boldsymbol{v}_n, \quad n = 1, 2, \ldots, \tag{8.18}$$

where $\mathbf{H}_n$ denotes the (known) transformation matrix from the state $\boldsymbol{x}_n$ to the measurement $\boldsymbol{y}_n$, and $\boldsymbol{v}_n$ denotes the stochastic *measurement noise* at time instant $n$, assumed to be of zero mean with known covariance $\mathbf{R}_n$.

It is further assumed that the initial state $\boldsymbol{x}_0$, the driving noise $\boldsymbol{w}_n$, and the measurement noise $\boldsymbol{v}_n$ are all uncorrelated, namely:

$$\begin{aligned}
&E\left(\boldsymbol{x}_0 \boldsymbol{w}_n^{\mathrm{T}}\right) = \mathbf{0} && E\left(\boldsymbol{x}_0 \boldsymbol{v}_n^{\mathrm{T}}\right) = \mathbf{0} && \forall n \\
&E\left(\boldsymbol{w}_n \boldsymbol{v}_m^{\mathrm{T}}\right) = \mathbf{0} && && \forall n, m \\
&E\left(\boldsymbol{w}_n \boldsymbol{w}_m^{\mathrm{T}}\right) = \mathbf{0} && E\left(\boldsymbol{v}_n \boldsymbol{v}_m^{\mathrm{T}}\right) = \mathbf{0} && \forall n \neq m \\
&E\left(\boldsymbol{w}_n \boldsymbol{w}_n^{\mathrm{T}}\right) = \mathbf{Q}_n && E\left(\boldsymbol{v}_n \boldsymbol{v}_n^{\mathrm{T}}\right) = \mathbf{R}_n\,. && \forall n
\end{aligned} \tag{8.19}$$

In the classical causal filtering framework, it is desired to compute, at each time instant $n$, the optimal linear estimator of $\boldsymbol{x}_n$ from all the measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ up to the same time instant. We shall now propose a recursive scheme for this estimation process.

Amazon EX1010 Page 168

To this end, let us introduce a general simplified notation, denoting by $\hat{\boldsymbol{x}}_{k|\ell}$ the optimal linear estimator of $\boldsymbol{x}_k$ from $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\ell$. Likewise, we denote by $\boldsymbol{\epsilon}_{k|\ell} = \hat{\boldsymbol{x}}_{k|\ell} - \boldsymbol{x}_k$ this estimator's estimation error, and by $\mathbf{P}_{k|\ell} = E(\boldsymbol{\epsilon}_{k|\ell}\boldsymbol{\epsilon}_{k|\ell}^{\mathrm{T}})$ its (matrix) MSE.

Assume now, that at time instant $n$ we are given $\hat{\boldsymbol{x}}_{n-1|n-1}$. Let us proceed in the following two-step scheme:

1. Obtain $\hat{\boldsymbol{x}}_{n|n-1}$ from

$$\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{\Phi}_n \hat{\boldsymbol{x}}_{n-1|n-1} \,, \tag{8.20}$$

2. Using the new measurement $\boldsymbol{y}_n$, obtain $\hat{\boldsymbol{x}}_{n|n}$ from

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \mathbf{K}_n(\boldsymbol{y}_n - \mathbf{H}_n \hat{\boldsymbol{x}}_{n|n-1}) \,, \tag{8.21}$$

where $\mathbf{K}_n$ is a special *weighting matrix* to be derived shortly, often termed the *Kalman gain*.

These steps have the following intuitive interpretation:

- In the first step (often called the *propagation* step), we compute the estimate of $\boldsymbol{x}_n$ based on all the previous measurements up to $\boldsymbol{y}_{n-1}$. Since we already have the optimal estimate of $\boldsymbol{x}_{n-1}$ based on these measurements, the best we can do, in the absence of new information, is to apply the model equation to $\hat{\boldsymbol{x}}_{n-1|n-1}$, ignoring the unknown, zero-mean driving noise $\boldsymbol{w}_n$.
- In the second step (often called the *update* step), we use the new measurement $\boldsymbol{y}_n$ to update the estimate of $\boldsymbol{x}_n$ from the first step, which was only based on previous measurements. The update consists of an additive term, which is proportional to the *innovation* contained in the new measurement $\boldsymbol{y}_n$. This innovation reflects the *additional* information in $\boldsymbol{y}_n$ regarding the estimate $\hat{\boldsymbol{x}}_{n|n-1}$: if the measurement is equal to what could be expected from that estimate, namely to $\mathbf{H}_n\hat{\boldsymbol{x}}_{n|n-1}$, then there is no innovation (the term is zero), and consequently $\hat{\boldsymbol{x}}_{n|n-1}$ is deemed *good enough* to become $\hat{\boldsymbol{x}}_{n|n}$, having been *confirmed* by $\boldsymbol{y}_n$. However, usually this is not the case, and a nonzero innovation has to be incorporated when updating $\hat{\boldsymbol{x}}_{n|n-1}$ to $\hat{\boldsymbol{x}}_{n|n}$. The proper weighting in the transformation from the innovation to the update is prescribed by the Kalman gain matrix $\mathbf{K}_n$.

Natural and intuitively appealing as it may be, this explanation can by no means serve as a proof of optimality. We shall now derive a rigorous proof of optimality, based on the three necessary and sufficient conditions mentioned above, namely linearity, zero-mean of the error, and orthogonality of the error to all measurements on which the estimate is based. Our proof will be based on recursive induction of these conditions, dwelling on the recursive structure of the estimator. An important byproduct of the proof will be an expression for the Kalman gain matrix $\mathbf{K}_n$, obtained by enforcing the optimality conditions. We will also obtain important expressions for the associated error covariances $\mathbf{P}_{n|n-1}$ and $\mathbf{P}_{n|n}$.

As mentioned above, let us assume that at time instant $n$ we have $\hat{\boldsymbol{x}}_{n-1|n-1}$, the optimal linear estimator of $\boldsymbol{x}_{n-1}$ from $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n-1}$. Its associated estimation error is $\boldsymbol{\epsilon}_{n-1|n-1}$, which by virtue of the optimality of $\hat{\boldsymbol{x}}_{n-1|n-1}$, has zero mean and is orthogonal to $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n-1}$.

We will now show that the proposed scheme propagates the optimality conditions from $\hat{\boldsymbol{x}}_{n-1|n-1}$ to the resulting estimate $\hat{\boldsymbol{x}}_{n|n}$ (and, also, along the way, to the intermediate estimate $\hat{\boldsymbol{x}}_{n|n-1}$). Considering the linearity condition first, it is obvious that, since $\hat{\boldsymbol{x}}_{n-1|n-1}$ is a linear function of all $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n-1}$, so is $\hat{\boldsymbol{x}}_{n|n-1}$ obtained from (8.20) and thus also $\hat{\boldsymbol{x}}_{n|n}$ obtained from (8.21).

To examine the two remaining conditions, let us monitor the propagation of the estimation error. Subtracting $\boldsymbol{x}_n$ from (8.20), we obtain, by substituting (8.17),

$$\begin{aligned}
\boldsymbol{\epsilon}_{n|n-1} &= \hat{\boldsymbol{x}}_{n|n-1} - \boldsymbol{x}_n \\
&= \boldsymbol{\Phi}_n \hat{\boldsymbol{x}}_{n-1|n-1} - (\boldsymbol{\Phi}_n \boldsymbol{x}_{n-1} + \boldsymbol{w}_n) \\
&= \boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1} - \boldsymbol{x}_{n-1}) - \boldsymbol{w}_n \\
&= \boldsymbol{\Phi}_n \boldsymbol{\epsilon}_{n-1|n-1} - \boldsymbol{w}_n \,.
\end{aligned} \tag{8.22}$$

Likewise, subtracting $\boldsymbol{x}_n$ from (8.21), we get, by substituting (8.18),

$$\begin{aligned}
\boldsymbol{\epsilon}_{n|n} &= \hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_n \\
&= \hat{\boldsymbol{x}}_{n|n-1} + \mathbf{K}_n(\boldsymbol{y}_n - \mathbf{H}_n \hat{\boldsymbol{x}}_{n|n-1}) - \boldsymbol{x}_n \\
&= \boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n(\mathbf{H}_n \boldsymbol{x}_n + \boldsymbol{v}_n - \mathbf{H}_n \hat{\boldsymbol{x}}_{n|n-1}) \\
&= \boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n(-\mathbf{H}_n \boldsymbol{\epsilon}_{n|n-1} + \boldsymbol{v}_n) \\
&= (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n \boldsymbol{v}_n \,.
\end{aligned} \tag{8.23}$$

It is now easily observed that from (8.22)

$$E(\boldsymbol{\epsilon}_{n|n-1}) = \boldsymbol{\Phi}_n E(\boldsymbol{\epsilon}_{n-1|n-1}) - E(\boldsymbol{w}_n) = \mathbf{0} \,, \tag{8.24}$$

and hence from (8.23),

$$E(\boldsymbol{\epsilon}_{n|n}) = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n) E(\boldsymbol{\epsilon}_{n|n-1}) + \mathbf{K}_n E(\boldsymbol{v}_n) = \mathbf{0} \,, \tag{8.25}$$

thus the zero-mean condition is seen to propagate simply by construction of the proposed estimation scheme. It now remains to verify the orthogonality condition. Let



Amazon EX1010 Page 169

us first examine the orthogonality of $\boldsymbol{\epsilon}_{n|n-1}$ to all the measurements on which $\hat{\boldsymbol{x}}_{n|n-1}$ is based, namely to all $\boldsymbol{y}_\ell$, $\ell = 1, 2, \ldots, n-1$. Using (8.22) again we get

$$\begin{aligned} E\big(\boldsymbol{\epsilon}_{n|n-1}\boldsymbol{y}_\ell^{\mathrm{T}}\big) &= E\big[(\boldsymbol{\Phi}_n\boldsymbol{\epsilon}_{n-1|n-1} - \boldsymbol{w}_n)\boldsymbol{y}_\ell^{\mathrm{T}}\big] \\ &= \boldsymbol{\Phi}_n E\big(\boldsymbol{\epsilon}_{n-1|n-1}\boldsymbol{y}_\ell^{\mathrm{T}}\big) - E\big(\boldsymbol{w}_n\boldsymbol{y}_\ell^{\mathrm{T}}\big) \\ &= \mathbf{0}\,, \qquad \ell = 1, 2, \ldots, n-1, \end{aligned} \tag{8.26}$$

where the first term is zeroed-out due to the optimality of $\hat{\boldsymbol{x}}_{n-1|n-1}$, and the second term is zeroed-out since all of the measurements $\boldsymbol{y}_\ell$ up to $\ell = n-1$ are essentially linear functions of the random vectors $\boldsymbol{x}_0$, $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{n-1}$, $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n-1}$, which are all orthogonal to $\boldsymbol{w}_n$ – thus these measurements are all orthogonal to $\boldsymbol{w}_n$ as well. Note that this already implies that the intermediate estimate $\hat{\boldsymbol{x}}_{n|n-1}$ of (8.20) is the optimal linear estimator of $\boldsymbol{x}_n$ from all measurements up to the previous time instant $n-1$.

We now wish to verify a similar orthogonality condition for $\boldsymbol{\epsilon}_{n|n}$. Note that for optimality of $\hat{\boldsymbol{x}}_{n|n}$, $\boldsymbol{\epsilon}_{n|n}$ has to be orthogonal to all $\boldsymbol{y}_\ell$ for $\ell = 1, 2, \ldots, n$, namely to the same measurements as $\boldsymbol{\epsilon}_{n|n-1}$, plus the new measurement $\boldsymbol{y}_n$ used in the update stage. Let us first examine the orthogonality to past measurements only, up to $\ell = n-1$. Using (8.23) we get

$$\begin{aligned} E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{y}_\ell^{\mathrm{T}}\big) &= E\big\{[(\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n\boldsymbol{v}_n]\boldsymbol{y}_\ell^{\mathrm{T}}\big\} \\ &= (\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)E\big(\boldsymbol{\epsilon}_{n|n-1}\boldsymbol{y}_\ell^{\mathrm{T}}\big) + \mathbf{K}_n E\big(\boldsymbol{v}_n\boldsymbol{y}_\ell^{\mathrm{T}}\big) \\ &= \mathbf{0}\,, \qquad \ell = 1, 2, \ldots, n-1, \end{aligned} \tag{8.27}$$

where the first term is zeroed-out following (8.26), and the second is zeroed-out due to a similar argument as used above for (8.26).

Note that all of the conditions up to this point were propagated *automatically* by the structure of the estimation scheme, regardless of the value of $\mathbf{K}_n$. By imposing the remaining condition of orthogonality of $\boldsymbol{\epsilon}_{n|n}$ to $\boldsymbol{y}_n$, we would now get an explicit expression for $\mathbf{K}_n$. We have

$$\begin{aligned} E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{y}_n^{\mathrm{T}}\big) &= E\{[(\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n\boldsymbol{v}_n] \\ &\quad \cdot (\mathbf{H}_n\boldsymbol{x}_n + \boldsymbol{v}_n)^{\mathrm{T}}\} \\ &= E\big\{[(\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1} + \mathbf{K}_n\boldsymbol{v}_n] \\ &\quad \cdot (\mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1} \pm \boldsymbol{\epsilon}_{n|n-1}) + \boldsymbol{v}_n)^{\mathrm{T}}\big\}\,. \end{aligned} \tag{8.28}$$

This expression involves six types of expectations that have to be computed, as follows:

$$E\big(\boldsymbol{\epsilon}_{n|n-1}\hat{\boldsymbol{x}}_{n|n-1}^{\mathrm{T}}\big) = \mathbf{0}\,, \tag{8.29a}$$

since $\hat{\boldsymbol{x}}_{n|n-1}$ is a linear function of measurements $\boldsymbol{y}_\ell$ up to $\ell = n-1$, all of which are orthogonal to $\boldsymbol{\epsilon}_{n|n-1}$ [by (8.26)];

$$E\big(\boldsymbol{\epsilon}_{n|n-1}\boldsymbol{\epsilon}_{n|n-1}^{\mathrm{T}}\big) = \mathbf{P}_{n|n-1}\,, \tag{8.29b}$$

by definition;

$$E\big(\boldsymbol{\epsilon}_{n|n-1}\boldsymbol{v}_n^{\mathrm{T}}\big) = \mathbf{0}\,, \tag{8.29c}$$

since $\boldsymbol{\epsilon}_{n|n-1}$ is a linear function of past measurements and of $\boldsymbol{x}_n$, all of which are in turn linear functions of $\boldsymbol{x}_0$, $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n-1}$, which are all orthogonal to $\boldsymbol{v}_n$;

$$E\big(\boldsymbol{v}_n\hat{\boldsymbol{x}}_{n|n-1}^{\mathrm{T}}\big) = \mathbf{0}\,, \tag{8.29d}$$

due to a similar argument;

$$E\big(\boldsymbol{v}_n\boldsymbol{\epsilon}_{n|n-1}^{\mathrm{T}}\big) = \mathbf{0}\,, \tag{8.29e}$$

[same as (8.29c)]; and

$$E\big(\boldsymbol{v}_n\boldsymbol{v}_n^{\mathrm{T}}\big) = \mathbf{R}_n\,, \tag{8.29f}$$

from the measurement equation (8.18). To conclude, we obtain, substituting (8.29a)–(8.29f) into (8.28)

$$E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{y}_n^{\mathrm{T}}\big) = (\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}} + \mathbf{K}_n\mathbf{R}_n\,, \tag{8.30}$$

which has to equal $\mathbf{0}$ to secure all optimality conditions. This leads to the requirement

$$\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}} = \mathbf{K}_n\big(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}} + \mathbf{R}_n\big)\,, \tag{8.31}$$

which is easily satisfied by setting the Kalman gain $\mathbf{K}_n$ to be

$$\mathbf{K}_n = \mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}\big(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}} + \mathbf{R}_n\big)^{-1}\,. \tag{8.32}$$

A remaining practical issue, which is also of interest for performance evaluation, is the computation of the error covariance $\mathbf{P}_{n|n-1}$, required in the computation of (8.32). We take a similar recursive approach in the computation of these matrices: assume that along with $\hat{\boldsymbol{x}}_{n-1|n-1}$ (the previous estimate) we are also given $\mathbf{P}_{n-1|n-1}$. From (8.22) we have

$$\begin{aligned} &\mathbf{P}_{n|n-1} \\ &= E\big(\boldsymbol{\epsilon}_{n|n-1}\boldsymbol{\epsilon}_{n|n-1}^{\mathrm{T}}\big) \\ &= E\big[(\boldsymbol{\Phi}_n\boldsymbol{\epsilon}_{n-1|n-1} - \boldsymbol{w}_n)(\boldsymbol{\Phi}_n\boldsymbol{\epsilon}_{n-1|n-1} - \boldsymbol{w}_n)^{\mathrm{T}}\big] \\ &= \boldsymbol{\Phi}_n\mathbf{P}_{n-1|n-1}\boldsymbol{\Phi}_n^{\mathrm{T}} + \mathbf{Q}_n\,, \end{aligned} \tag{8.33}$$

in which we used the orthogonality of $\boldsymbol{w}_n$ to $\boldsymbol{\epsilon}_{n-1|n-1}$, the latter being a linear function of random vectors from the past, which are all uncorrelated with $\boldsymbol{w}_n$.

Amazon EX1010 Page 170

Although $\mathbf{P}_{n|n-1}$ is sufficient for computing $\mathbf{K}_n$, we are also interested in $\mathbf{P}_{n|n}$, both for obtaining the estimate's MSE and for passing that matrix onto the following recursion phase. From (8.23) we get

$$\begin{aligned}\mathbf{P}_{n|n} &= E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{\epsilon}_{n|n}^{\mathrm{T}}\big)\\ &= E\big\{[(\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1}+\mathbf{K}_n\boldsymbol{v}_n]\\ &\quad\cdot(\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\boldsymbol{\epsilon}_{n|n-1}+\mathbf{K}_n\boldsymbol{v}_n)^{\mathrm{T}}\big\}\\ &= (\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}(\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\\ &\quad+\mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^{\mathrm{T}}\,, \end{aligned} \tag{8.34}$$

again having used the orthogonality of $\boldsymbol{v}_n$ to $\boldsymbol{\epsilon}_{n|n-1}$, a linear function of past measurements and of $\boldsymbol{x}_n$, which are all orthogonal to $\boldsymbol{v}_n$. Although (8.34) can indeed serve to compute $\mathbf{P}_{n|n}$, considerable simplification of this expression is possible. In fact, it can be shown that this expression is equivalent to the following:

$$\mathbf{P}_{n|n} = (\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}\,. \tag{8.35}$$

To show this, observe that the difference between (8.34) and (8.35) can be expressed as

$$\begin{aligned}&-(\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}\mathbf{K}_n^{\mathrm{T}}+\mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^{\mathrm{T}}\\ &=\big(-\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}+\mathbf{K}_n\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}+\mathbf{K}_n\mathbf{R}_n\big)\mathbf{K}_n^{\mathrm{T}}\\ &=\big[-\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}+\mathbf{K}_n\big(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}+\mathbf{R}_n\big)\big]\mathbf{K}_n^{\mathrm{T}}\,. \end{aligned} \tag{8.36}$$

Substituting (8.31) we easily observe that this difference vanishes, implying the equivalence of the two expressions.

This concludes the derivation of the expressions necessary to carry out a single recursion phase, in which the optimal linear estimate and its MSE matrix, provided from the previous recursion phase, are used, together with a new measurement, to compute an updated optimal linear estimate and its associated MSE matrix. In such a recursive structure, the output of each recursion phase serves as the input to the next recursion phase, and optimality is maintained throughout. A small remaining issue is the initialization of the process: we first need $\hat{\boldsymbol{x}}_{0|0}$, the MSE-optimal linear estimate of $\boldsymbol{x}_0$ based on no measurements at all. Naturally, that MSE-optimal estimator is the known mean of $\boldsymbol{x}_0$, namely $\boldsymbol{m}_0$. Moreover, the required covariance of that estimate is given by

$$\begin{aligned}\mathbf{P}_{0|0} &= E\big[(\hat{\boldsymbol{x}}_{0|0}-\boldsymbol{x}_0)(\hat{\boldsymbol{x}}_{0|0}-\boldsymbol{x}_0)^{\mathrm{T}}\big]\\ &= E\big[(\boldsymbol{m}_0-\boldsymbol{x}_0)(\boldsymbol{m}_0-\boldsymbol{x}_0)^{\mathrm{T}}\big]=\mathbf{P}_0\,. \end{aligned} \tag{8.37}$$

We are now ready to summarize the recipe for applying the Kalman filter:
*Initialization*:

$$\begin{aligned}&\text{Let } \hat{\boldsymbol{x}}_{0|0} := \boldsymbol{m}_0\,,\\ &\text{and } \mathbf{P}_{0|0} := \mathbf{P}_0\,.\end{aligned}$$

Proceed for n=1,2, . . . :
*Propagation equations*:

$$\hat{\boldsymbol{x}}_{n|n-1} := \boldsymbol{\Phi}_n\hat{\boldsymbol{x}}_{n-1|n-1}\,, \tag{p1}$$

$$\mathbf{P}_{n|n-1} := \boldsymbol{\Phi}_n\mathbf{P}_{n-1|n-1}\boldsymbol{\Phi}_n^{\mathrm{T}}+\mathbf{Q}_n\,. \tag{p2}$$

*Update equations*:

$$\mathbf{K}_n := \mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}\big(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}+\mathbf{R}_n\big)^{-1}\,, \tag{u1}$$

$$\hat{\boldsymbol{x}}_{n|n} := \hat{\boldsymbol{x}}_{n|n-1}+\mathbf{K}_n(\boldsymbol{y}_n-\mathbf{H}_n\hat{\boldsymbol{x}}_{n|n-1})\,, \tag{u2}$$

$$\mathbf{P}_{n|n} := (\mathbf{I}-\mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}\,. \tag{u3}$$

Note that (p2), (u1), and (u3) are all data independent, so wherever applicable, they can be computed in advance before the data sequence arrives. Only (p1) and (u2) need to be computed online. This appealing feature not only enables to save real-time computations, but also enables one to assess the propagation of MSE in time beforehand, as this MSE, expressed by the sequence of $\mathbf{P}_{n|n}$, is data independent.

## 8.2 Examples: Estimation of Parametric Stochastic Process from Noisy Observations

Formulating the Kalman filter for the estimation of a stochastic process contaminated by additive noise often boils down to the definition of the state-space equations. In this section, we will exemplify the procedure for noisy observations of several quite common stochastic processes. Although we shall consider stationary processes, the derivation of the Kalman filter does not necessitate any stationarity assumption. In Sect. 8.4 we will consider speech signals, which can only be assumed to be quasistationary.

Consider a difference equation with time-invariant coefficients,

$$x(n) = -\sum_{k=1}^{p}\alpha_k x(n-k)+\sum_{k=0}^{q}\beta_k w(n-k), \tag{8.38}$$



Amazon EX1010 Page 171

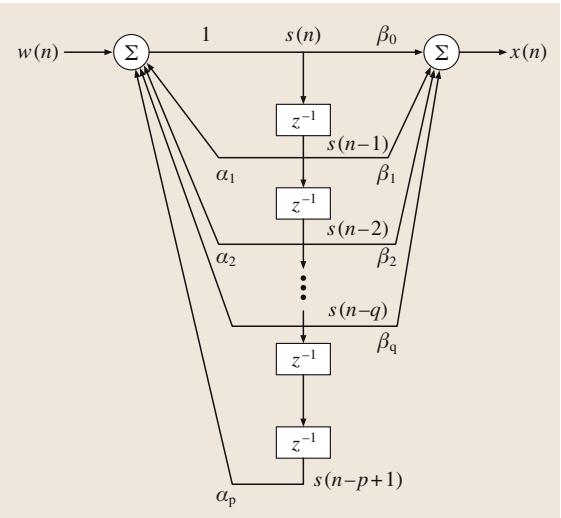



**Fig. 8.1** A block diagram presenting an autoregressive moving-average process

where $w(n)$ is some zero-mean white-noise input, with variance $E[w^2(n)] = \sigma_w^2$. The output signal is usually termed an autoregressive moving-average (ARMA) process. The process $x(n)$ is a sum of two terms. The first term is a weighted sum of past output signal values [hence the term autoregressive (AR)]. The time-invariant AR coefficients, $\alpha_1, \ldots, \alpha_p$ are the weight values. The second term is a weighted average of past and current input samples [hence the term moving-average (MA)]. The time-invariant MA coefficients, $\beta_0, \ldots, \beta_q$, are the weight values. A possible block diagram of the process generation is depicted in Fig. 8.1; this form is often called *direct form II*, and is one of several alternative forms.

The block $z^{-1}$ denotes a unit delay. It is desired to estimate $x(n)$ from noisy observations thereof, $y(n)$, given by

$$y(n) = x(n) + v(n), \tag{8.39}$$

where $v(n)$ denotes additive background noise with variance $\sigma_v^2$.

We will now turn to a formulation of the state-space representation of three special cases of the difference equation given in (8.38), namely, AR, MA, and ARMA processes. The state variables will be defined, in all three cases, as the inputs of the delay units, constituting the $\max(p, q+1)\times 1$ state vector

$$\boldsymbol{x}_n^{\mathrm{T}} = (s(n-p+1)\; s(n-p+2)\; \ldots\; s(n))\,. \tag{8.40}$$

We assume without loss of generality that $p > q$. If this inequality does not hold, extra, zero-valued coefficients can be appended to the existing coefficients.

### 8.2.1 Autoregressive (AR) Process

In the autoregressive case, the coefficients $\beta_1, \ldots, \beta_q$ are assumed to be zero, while $\beta_0 = 1$. The resulting process is given by

$$x(n) = -\sum_{k=1}^{p} \alpha_k x(n-k) + w(n)\,. \tag{8.41}$$

The $p\times 1$ state-vector is given by (8.40), where the state variable $s(n)$ is essentially the output signal $x(n)$. In some cases it might be useful to define a longer state-vector with $p+1$ elements $x(n-p), \ldots, x(n)$. We will elaborate on such a case when discussing the application to speech signals in Sect. 8.4. However, continuing with the $p\times 1$ formulation, define the $p\times p$ transition matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ -\alpha_p & -\alpha_{p-1} & \cdots & -\alpha_2 & -\alpha_1 \end{pmatrix}. \tag{8.42}$$

Note that the transition matrix is time invariant in this case. Define also the vector $(p\times 1)$ and matrix $(1\times p)$

$$\boldsymbol{w}_n^{\mathrm{T}} = (0\; \ldots\; 0\; w(n))\,, \tag{8.43}$$

$$\mathbf{H} = (0\; \ldots\; 0\; 1)\,. \tag{8.44}$$

Then (8.41) and (8.39) can be rewritten as

$$\begin{aligned} \boldsymbol{x}_n &= \boldsymbol{\Phi}\boldsymbol{x}_{n-1} + \boldsymbol{w}_n \\ y(n) &= \mathbf{H}\boldsymbol{x}_n + v(n)\,. \end{aligned} \tag{8.45}$$

Accordingly, the driving noise correlation matrix is given by

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \sigma_w^2 \end{pmatrix} \tag{8.46}$$

and, since in this case the measurement vector is a scalar, the measurement noise correlation matrix is the scalar $\mathbf{R} = \sigma_v^2$. Using the state-space representation in (8.45) the Kalman filter is readily applied.

Amazon EX1010 Page 172

### 8.2.2 Moving-Average (MA) Process

In the moving-average case, the coefficients $\alpha_1, \ldots, \alpha_p$ are assumed to be identically zero, hence the process model simplifies to

$$x(n) = \sum_{k=0}^{q} \beta_k w(n-k) \,. \tag{8.47}$$

The $(q+1)\times 1$ state-vector is accordingly given in a form similar to (8.40) with $p$ replaced by $q+1$. Now the state variable $s(n)$ is essentially the input signal $w(n)$. Using the above definitions, we can reformulate (8.47) in a state-space representation. The $(q+1)\times(q+1)$ transition matrix becomes a simple time-shift matrix,

$$\boldsymbol{\Phi} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} . \tag{8.48}$$

Define the vector $[(q+1)\times 1]$ and matrix $[1\times(q+1)]$

$$\boldsymbol{w}_n^{\mathrm{T}} = (0 \ \ldots \ 0 \ w(n)) \,, \tag{8.49}$$

$$\mathbf{H} = (\beta_q \ \ldots \ \beta_1 \ \beta_0) \,. \tag{8.50}$$

Then (8.47) and (8.39) can again be rewritten as

$$\begin{aligned} \boldsymbol{x}_n &= \boldsymbol{\Phi}\boldsymbol{x}_{n-1} + \boldsymbol{w}_n \\ y(n) &= \mathbf{H}\boldsymbol{x}_n + v(n) \,. \end{aligned} \tag{8.51}$$

As in the previous case (the AR process), the driving noise correlation matrix is given by (8.46) and the measurement noise correlation matrix by the scalar $\mathbf{R} = \sigma_v^2$. The state-space representation (8.51) can be used in the Kalman formulation.

### 8.2.3 Autoregressive Moving-Average (ARMA) Process

The state vector of the ARMA process is also given by (8.40) (assuming $p > q$), but now it is not directly related to either the input or the output signals. The transition matrix is identical to the $p\times p$ AR transition matrix in (8.42). The $p\times 1$ measurement matrix is equal to the corresponding MA measurement matrix padded with $p-q-1$ zeros

$$\mathbf{H} = (\beta_q \ \ldots \ \beta_1 \ \beta_0 \ 0 \ \ldots \ 0) \,. \tag{8.52}$$

The driving noise vector is identical to both the AR and MA processes and is given by (8.43). As in the special cases, the driving noise correlation matrix is given by (8.46) and the measurement noise correlation matrix is the scalar $\mathbf{R} = \sigma_v^2$.

### 8.2.4 The Case of Temporally Correlated Noise

The assumption that the measurement noise is temporally uncorrelated, i. e., $E(\boldsymbol{v}_n \boldsymbol{v}_m^{\mathrm{T}}) = \mathbf{0}$, $\forall n \neq m$ is not always realistic. To circumvent the need to account explicitly for correlated noise in the Kalman filter equations, the following procedure is commonly used.

Assume that the measurement noise $\boldsymbol{v}_n$ can be described using a state-space model as well. Let $\boldsymbol{x}_n^{\mathrm{v}}$ denote the respective noise state vector. Then

$$\boldsymbol{x}_n^{\mathrm{v}} = \boldsymbol{\Phi}_n^{\mathrm{v}} \boldsymbol{x}_{n-1}^{\mathrm{v}} + \boldsymbol{w}_n^{\mathrm{v}} \,, \quad n = 1, 2, \ldots \,, \tag{8.53}$$

where $\boldsymbol{\Phi}_n^{\mathrm{v}}$ is a known matrix, and $\boldsymbol{w}_n^{\mathrm{v}}$ is a temporally uncorrelated zero-mean driving noise (for the noise signal) with correlation matrix $E[\boldsymbol{w}_n^{\mathrm{v}}(\boldsymbol{w}_n^{\mathrm{v}})^{\mathrm{T}}] = \mathbf{Q}_n^{\mathrm{v}}$. The measurement noise $\boldsymbol{v}_n$ is related to the noise state vector $\boldsymbol{x}_n^{\mathrm{v}}$ by:

$$\boldsymbol{v}_n = \mathbf{H}_n^{\mathrm{v}} \boldsymbol{x}_n^{\mathrm{v}} \,, \quad n = 1, 2, \ldots \,, \tag{8.54}$$

where $\mathbf{H}_n^{\mathrm{v}}$ is known.

Assume that the recursive model in (8.17) and the measurement equations (8.18) still hold. We restate these equation with a slight change of notations. The signal state-space model is given by:

$$\boldsymbol{x}_n^{\mathrm{s}} = \boldsymbol{\Phi}_n^{\mathrm{s}} \boldsymbol{x}_{n-1}^{\mathrm{s}} + \boldsymbol{w}_n^{\mathrm{s}} \quad n = 1, 2, \ldots \tag{8.55}$$

and the signal measurement model is given by:

$$\boldsymbol{y}_n = \mathbf{H}_n^{\mathrm{s}} \boldsymbol{x}_n^{\mathrm{s}} + \boldsymbol{v}_n = \mathbf{H}_n^{\mathrm{s}} \boldsymbol{x}_n^{\mathrm{s}} + \mathbf{H}_n^{\mathrm{v}} \boldsymbol{x}_n^{\mathrm{v}} \,, \quad n = 1, 2, \ldots \tag{8.56}$$

Therefore, concatenating both state vectors will enable the use of the *standard* Kalman filter. Let

$$\boldsymbol{x}_n^{\mathrm{T}} = \left( (\boldsymbol{x}_n^{\mathrm{s}})^{\mathrm{T}} \ (\boldsymbol{x}_n^{\mathrm{v}})^{\mathrm{T}} \right)$$

be the augmented state vector, and define

$$\boldsymbol{w}_n^{\mathrm{T}} = \left( (\boldsymbol{w}_n^{\mathrm{s}})^{\mathrm{T}} \ (\boldsymbol{w}_n^{\mathrm{v}})^{\mathrm{T}} \right)$$

to be the driving noise vector. The augmented transition matrix is then given by

$$\boldsymbol{\Phi}_n = \begin{pmatrix} \boldsymbol{\Phi}_n^{\mathrm{s}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_n^{\mathrm{v}} \end{pmatrix}$$

where $\mathbf{0}$ is an all-zeros matrix of the proper dimensions. Then the concatenated recursive model is given by

$$\boldsymbol{x}_n = \boldsymbol{\Phi}_n \boldsymbol{x}_{n-1} + \boldsymbol{w}_n, \quad n = 1, 2, \ldots \tag{8.57}$$



Amazon EX1010 Page 173

The respective driving noise covariance matrix is given by

$$\mathbf{Q}_n = \begin{pmatrix} \mathbf{Q}_n^{\mathrm{s}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_n^{\mathrm{v}} \end{pmatrix} .$$

Accordingly, the measurement equation can be reformulated as

$$\boldsymbol{y}_n = \mathbf{H}_n \boldsymbol{x}_n , \quad n = 1, 2, \ldots , \tag{8.58}$$

where

$$\mathbf{H}_n = \left(\mathbf{H}_n^{\mathrm{s}} \mathbf{H}_n^{\mathrm{v}}\right) .$$

Note, that as the measurement equation (8.58) is noise-free, the noise correlation matrix is zero, i. e., $\mathbf{R}_n = \mathbf{0}$.

## 8.3 Extensions of the Kalman Filter

### 8.3.1 The Kalman Predictor



In various applications it is desirable to predict the value of a future state, say $\boldsymbol{x}_m$, from measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots , \boldsymbol{y}_n$ up to some time instant $n < m$. The optimal linear estimator (*predictor*) in the sense of minimum MSE is the Kalman predictor, which is based on the Kalman filter and has a simple, intuitively appealing structure.

Let us first consider a one-step predictor, which predicts (estimates) $\boldsymbol{x}_{n+1}$ from $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots , \boldsymbol{y}_n$. The prediction equation is given by

$$\hat{\boldsymbol{x}}_{n+1|n} = \boldsymbol{\Phi}_{n+1} \hat{\boldsymbol{x}}_{n|n} , \tag{8.59}$$

where $\hat{\boldsymbol{x}}_{n|n}$ is the Kalman filter's output at time instant $n$, namely the optimal linear estimate of $\boldsymbol{x}_n$ from the same measurements. As shown in Sect. 8.1, in order to establish optimality of the proposed predictor, we need to verify linearity, zero-mean estimation errors, and orthogonality of the estimation errors to all of the measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots , \boldsymbol{y}_n$ on which the prediction is based. The linearity property is trivially satisfied from the linearity of the expression (8.59) and of the Kalman filter. To examine the other two properties, let us first establish a recursive relation for the prediction error. Subtracting $\boldsymbol{x}_{n+1}$ from both sides of (8.59), we get

$$\begin{aligned} \boldsymbol{\epsilon}_{n+1|n} &= \boldsymbol{\Phi}_{n+1} \hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_{n+1} \\ &= \boldsymbol{\Phi}_{n+1} \hat{\boldsymbol{x}}_{n|n} - (\boldsymbol{\Phi}_{n+1} \boldsymbol{x}_n + \boldsymbol{w}_{n+1}) \\ &= \boldsymbol{\Phi}_{n+1} \boldsymbol{\epsilon}_{n|n} - \boldsymbol{w}_{n+1} , \end{aligned} \tag{8.60}$$

where we have substituted the model equation (8.17) for $\boldsymbol{x}_{n+1}$. It is now easily verified that the optimality of the Kalman filter induces optimality of the one-step predictor, as

$$E(\boldsymbol{\epsilon}_{n+1|n}) = \boldsymbol{\Phi}_{n+1} E(\boldsymbol{\epsilon}_{n|n}) - E(\boldsymbol{w}_{n+1}) = \mathbf{0} \tag{8.61}$$

and

$$E(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{y}_\ell) = \boldsymbol{\Phi}_{n+1} E(\boldsymbol{\epsilon}_{n|n} \boldsymbol{y}_\ell) - E(\boldsymbol{w}_{n+1} \boldsymbol{y}_\ell) = \mathbf{0} \tag{8.62}$$

for all $\ell = 1, 2, \ldots , n$, where we have also exploited the orthogonality of $\boldsymbol{w}_{n+1}$ to past measurements (which are linear functions of $\boldsymbol{x}_0$, past of $\boldsymbol{w}_k$ and past of $\boldsymbol{v}_k$, all of which are orthogonal to $\boldsymbol{w}_{n+1}$).

This establishes the optimality of the proposed one-step Kalman predictor. The associated MSE is also easily derived from (8.60),

$$\mathbf{P}_{n+1|n} = \boldsymbol{\Phi}_{n+1} \mathbf{P}_{n|n} \boldsymbol{\Phi}_{n+1|n}^{\mathrm{T}} + \mathbf{Q}_{n+1} , \tag{8.63}$$

where we have exploited the orthogonality of $\boldsymbol{w}_{n+1}$ to $\boldsymbol{\epsilon}_{n|n}$. In fact, the expressions obtained so far are reminiscent of the prediction steps (p1) and (p2) of the Kalman filter, in which the optimal one-step prediction $\hat{\boldsymbol{x}}_{n|n-1}$ is computed. Indeed, with one-step forward shifts of the indices we naturally get similar expressions.

Now suppose that we have the optimal linear $k$-step prediction and are interested in the $k+1$-step prediction. The optimal linear $k+1$-step predictor would have a structure similar to the one-step predictor, but would be based on the $k$-step predictor, rather than on the Kalman filter output:

$$\hat{\boldsymbol{x}}_{n+k+1|n} = \boldsymbol{\Phi}_{n+k+1} \hat{\boldsymbol{x}}_{n+k|n} . \tag{8.64}$$

Optimality can then be easily verified in the same way as for the one-step predictor, as the recursive relation of the prediction errors is essentially the same:

$$\begin{aligned} &\boldsymbol{\epsilon}_{n+k+1|n} \\ &= \boldsymbol{\Phi}_{n+k+1} \hat{\boldsymbol{x}}_{n+k|n} - \boldsymbol{x}_{n+k+1} \\ &= \boldsymbol{\Phi}_{n+k+1} \hat{\boldsymbol{x}}_{n+k|n} - (\boldsymbol{\Phi}_{n+k+1} \boldsymbol{x}_{n+k} + \boldsymbol{w}_{n+k+1}) \\ &= \boldsymbol{\Phi}_{n+k+1} \boldsymbol{\epsilon}_{n+k|n} - \boldsymbol{w}_{n+k+1} . \end{aligned} \tag{8.65}$$

The optimality propagates in exactly the same way, as orthogonality of $\boldsymbol{w}_{n+k+1}$ to past measurements and prediction errors is maintained. Likewise, the MSE can be

Amazon EX1010 Page 174

expressed as

$$\mathbf{P}_{n+k+1|n} = \boldsymbol{\Phi}_{n+k+1}\mathbf{P}_{n+k|n}\boldsymbol{\Phi}^{\mathrm{T}}_{n+k+1} + \mathbf{Q}_{n+k+1} \,. \tag{8.66}$$

We have therefore established, by induction, the optimality of a *propagating predictor* based on the Kalman filter output. If the intermediate predictors are not required, the general $d$-step predictor can be computed directly from the Kalman filter's output as

$$\hat{\boldsymbol{x}}_{m|n} = \boldsymbol{\Phi}_m\boldsymbol{\Phi}_{m-1}\cdots\boldsymbol{\Phi}_{n+1}\hat{\boldsymbol{x}}_{n|n} \,, \tag{8.67}$$

which simply propagates the model equations from time instant $n$ to time instant $m$. The matrix products $\boldsymbol{\Phi}_m\boldsymbol{\Phi}_{m-1}\cdots\boldsymbol{\Phi}_{n+1}$ can be computed in advance, so that only one matrix multiplication per prediction is required in real time.

As to computation of the MSE, the recursive form of (8.66) must be maintained, beginning with $\mathbf{P}_{n+1|n}$ and ending with $\mathbf{P}_{m|n}$. Naturally, the Kalman predictor's MSE $\mathbf{P}_{m|n}$ in estimating $\boldsymbol{x}_m$ from measurements up to time instant $n$ will always be greater than or equal to (usually greater than) the Kalman filter's error $\mathbf{P}_{m|m}$ in estimating the same state from measurements up to time instant $m$.

### 8.3.2 The Kalman Smoother

As opposed to the Kalman *predictor*, the Kalman *smoother* is aimed at estimating the state $\boldsymbol{x}_n$ from measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m$, where $m > n$, namely to base the estimate of $\boldsymbol{x}_n$ on *future* measurements, as well as on past and present measurements. The output of the optimal linear (Kalman) *smoother* must always be at least as good as (and is often much better than) the output of the optimal linear (Kalman) *filter* in terms of estimation MSE, since it optimally accounts for more measurements for estimation of $\boldsymbol{x}_n$. Naturally, such a *smoothing* scheme is not feasible in a causal real-time system. However, in view of the potential performance gain, it is sometime desirable to trade affordable latency for improved accuracy, namely to *delay* the output of the filter until some further measurements are obtained, thereby making the filter smoother. In addition, in non-real-time applications it is sometimes possible to work in an *offline* batch mode, in which it is possible to *go back in time* and improve all of the causal filter's estimates based on the entire batch of measurements.

It is common practice to distinguish between three different paradigms for smoothing, depending on the associated application: fixed-interval smoothing, fixed-point smoothing, and fixed-lag smoothing. We shall now discuss each type separately, with emphasis on the fixed-interval smoother, which can be viewed as the most general of the three, as it can also be used in the context of the other two.

#### Fixed-Interval Smoothing

A fixed-interval smoother is usually (but not necessarily) used in the context of batch-mode processing: once a batch of $N$ measurements $\boldsymbol{y}_1, \boldsymbol{y}_2 \ldots, \boldsymbol{y}_N$ has been obtained, and the Kalman filter output $\hat{\boldsymbol{x}}_{n|n}$ has been computed for all $n = 1, 2, \ldots, N$, it is desired to go back and compute, for each $n$, a *smoothed* estimate of $\boldsymbol{x}_n$, based on the entire batch of measurements, which is therefore denoted by $\hat{\boldsymbol{x}}_{n|N}$. The fixed-interval Kalman smoother offers a backwards-recursive scheme for obtaining the desired estimates. Some fixed-interval smoothers (see, e.g., [8.8]) employ three-pass smoothing: in the first pass, the ordinary Kalman filter is applied; in the second pass, a backwards Kalman filter is applied; in the third pass the results of both passes are optimally combined to attain the smoothed output.

However, herein we shall review the Rauch–Tung–Striebel two-pass smoother (derived by *H. Rauch*, *K. Tung* and *C. Striebel* in 1965 [8.9]). The two-pass smoother employs just one (the second) pass on the data in addition to the original (Kalman filter) pass. This second pass begins with the estimate of the last state $\hat{\boldsymbol{x}}_{N|N}$ (which is both the filtered and the smoothed estimate for $n = N$, since $N$ is the last sample in the batch), and propagates backwards as $n = N-1, N-2, \ldots, 1$, generating the smoothed estimate $\hat{\boldsymbol{x}}_{n|N}$ at each time instant $n$ from the previously obtained Kalman filter's outputs $\hat{\boldsymbol{x}}_{n|n}$ and $\hat{\boldsymbol{x}}_{n+1|n}$ and from the smoothed output of the succeeding time instant, $\hat{\boldsymbol{x}}_{n+1|N}$.

For simplicity of notation, we shall denote the Kalman smoother's output (at time instant $n$) as $\hat{\boldsymbol{x}}_n$, rather than $\hat{\boldsymbol{x}}_{n|N}$. The form $\hat{\boldsymbol{x}}_{n|m}$ will still be used to denote the Kalman filter's output. As already mentioned, for $n = N$ we have $\hat{\boldsymbol{x}}_N = \hat{\boldsymbol{x}}_{N|N}$, namely the smoother's output equals the Kalman filter's output, which is already the optimal linear estimate of $\boldsymbol{x}_N$ based on the entire batch. For $n = N-1, N-2, \ldots, 1$, the following backwards-recursive scheme is proposed:

$$\hat{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_{n|n} + \mathbf{A}_n(\hat{\boldsymbol{x}}_{n+1} - \hat{\boldsymbol{x}}_{n+1|n}) \,, \qquad n = N-1, N-2, \ldots, 1 \,, \tag{8.68}$$

where $\mathbf{A}_n$ is a gain matrix given by

$$\mathbf{A}_n = \mathbf{P}_{n|n}\boldsymbol{\Phi}^{\mathrm{T}}_{n+1}\mathbf{P}^{-1}_{n+1|n} \,. \tag{8.69}$$

At each time instant $n$, the smoother *corrects* the filter by adding a term that is proportional (via $\mathbf{A}_n$) to the difference between the smoother's output and the filter's predicted output for $n+1$. We shall now show the optimality of this scheme by verifying the three optimality conditions outlined in Sect. 8.1, namely: linearity, zero mean of the estimation error, and orthogonality of the estimation errors to the measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$ on which the estimates are based.

The linearity condition is trivially satisfied by the proposed structure, dwelling on the linearity of the Kalman filter. To verify the other two conditions, we first obtain recursion relations for the estimation error, denoted by $\boldsymbol{\epsilon}_n = \hat{\boldsymbol{x}}_n - \boldsymbol{x}_n$. By subtracting $\boldsymbol{x}_n$ from both sides of (8.68) and $\boldsymbol{x}_{n+1}$ from both terms in the parentheses we obtain

$$\boldsymbol{\epsilon}_n = \boldsymbol{\epsilon}_{n|n} + \mathbf{A}_n(\boldsymbol{\epsilon}_{n+1} - \boldsymbol{\epsilon}_{n+1|n}) \,, \tag{8.70}$$

where $\boldsymbol{\epsilon}_{n|n}$ and $\boldsymbol{\epsilon}_{n+1|n}$ are the Kalman filter errors, defined in Sect. 8.1. Due to the optimality of the Kalman filter these errors have zero mean and are orthogonal to all $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$.

Exploiting the zero-mean property in (8.70), we obtain $E(\boldsymbol{\epsilon}_n) = \mathbf{A}_n E(\boldsymbol{\epsilon}_{n+1})$. Recalling that $\boldsymbol{\epsilon}_N = \boldsymbol{\epsilon}_{N|N}$ (whose mean is zero), the zero-mean property is thereby induced backwards for all $n$.

We now consider the orthogonality property. From (8.70) we obtain

$$\begin{aligned} E(\boldsymbol{\epsilon}_n \boldsymbol{y}_\ell^{\mathrm{T}}) &= E(\boldsymbol{\epsilon}_{n|n} \boldsymbol{y}_\ell^{\mathrm{T}}) + \mathbf{A}_n E(\boldsymbol{\epsilon}_{n+1} \boldsymbol{y}_\ell^{\mathrm{T}}) \\ &\quad - \mathbf{A}_n E(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{y}_\ell^{\mathrm{T}}) \;\; \forall \ell \,. \end{aligned} \tag{8.71}$$

Due to the optimality of the Kalman filter and to the assumed optimality of $\hat{\boldsymbol{x}}_{n+1}$, we immediately obtain $E(\boldsymbol{\epsilon}_n \boldsymbol{y}_\ell^{\mathrm{T}}) = \mathbf{0}$ for all $\ell \le n$. Obviously, to establish the smoother's optimality it now remains to show the orthogonality of $\boldsymbol{\epsilon}_n$ to $\boldsymbol{y}_\ell$ for $n < \ell \le N$. Let us therefore consider the case $\ell = n+k$ for positive values of $k$. Based on the assumed optimality of $\hat{\boldsymbol{x}}_{n+1}$, we obtain

$$\begin{aligned} &E\big(\boldsymbol{\epsilon}_n \boldsymbol{y}_{n+k}^{\mathrm{T}}\big) \\ &= E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{y}_{n+k}^{\mathrm{T}}\big) - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{y}_{n+k}^{\mathrm{T}}\big) \\ &= E\big[\boldsymbol{\epsilon}_{n|n} (\mathbf{H}_{n+k} \boldsymbol{x}_{n+k} + \boldsymbol{v}_{n+k})^{\mathrm{T}}\big] \\ &\quad - \mathbf{A}_n E\big[\boldsymbol{\epsilon}_{n+1|n} (\mathbf{H}_{n+k} \boldsymbol{x}_{n+k} + \boldsymbol{v}_{n+k})^{\mathrm{T}}\big] \\ &= E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) \mathbf{H}_{n+k}^{\mathrm{T}} - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) \mathbf{H}_{n+k}^{\mathrm{T}} \\ &= \big[E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big)\big] \mathbf{H}_{n+k}^{\mathrm{T}} \,, \end{aligned} \tag{8.72}$$

where the third equality is due to orthogonality of $\boldsymbol{\epsilon}_{n|n}$ and $\boldsymbol{\epsilon}_{n+1|n}$ to future measurement noise $\boldsymbol{v}_{n+k}$ for all positive $k$. We shall now show that the term in parentheses in (8.72) is zero. Consider first the case $k = 1$. The term in parentheses is then given by

$$\begin{aligned} &E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+1}^{\mathrm{T}}\big) - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+1}^{\mathrm{T}}\big) \\ &= E\big(\boldsymbol{\epsilon}_{n|n} (\boldsymbol{\Phi}_{n+1} \boldsymbol{x}_n + \boldsymbol{w}_{n+1})^{\mathrm{T}}\big) \\ &\quad - \mathbf{A}_n E\big[\boldsymbol{\epsilon}_{n+1|n} (\hat{\boldsymbol{x}}_{n+1|n} - \boldsymbol{\epsilon}_{n+1|n})^{\mathrm{T}}\big] \\ &= E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_n^{\mathrm{T}}\big) \boldsymbol{\Phi}_{n+1}^{\mathrm{T}} + \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{\epsilon}_{n+1|n}^{\mathrm{T}}\big) \\ &= E\big[\boldsymbol{\epsilon}_{n|n} (\hat{\boldsymbol{x}}_{n|n} - \boldsymbol{\epsilon}_{n|n})^{\mathrm{T}}\big] \boldsymbol{\Phi}_{n+1}^{\mathrm{T}} + \mathbf{A}_n \mathbf{P}_{n+1|n} \\ &= -\mathbf{P}_{n|n} \boldsymbol{\Phi}_{n+1}^{\mathrm{T}} + \mathbf{A}_n \mathbf{P}_{n+1|n} = \mathbf{0} \,, \end{aligned} \tag{8.73}$$

where the transitions rely on orthogonality of $\boldsymbol{\epsilon}_{n|n}$ to $\boldsymbol{w}_{n+1}$, as well as on the orthogonality of both $\boldsymbol{\epsilon}_{n|n}$ and $\boldsymbol{\epsilon}_{n+1|n}$ to the Kalman filter's outputs $\hat{\boldsymbol{x}}_{n|n}$ and $\hat{\boldsymbol{x}}_{n+1|n}$, which are linear functions of measurements to which these errors are orthogonal. The last equality is obtained by substituting $\mathbf{A}_n$ from (8.69).

Now consider, by induction, the case of $k > 1$. Assume that the expression in parentheses in (8.72) is zero for some $k$, and consider $k+1$, for which the expression can be written as

$$\begin{aligned} &E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+k+1}^{\mathrm{T}}\big) - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+k+1}^{\mathrm{T}}\big) \\ &= E\big[\boldsymbol{\epsilon}_{n|n} (\boldsymbol{\Phi}_{n+k+1} \boldsymbol{x}_{n+k} + \boldsymbol{w}_{n+k+1})^{\mathrm{T}}\big] \\ &\quad - \mathbf{A}_n E\big[\boldsymbol{\epsilon}_{n+1|n} (\boldsymbol{\Phi}_{n+k+1} \boldsymbol{x}_{n+k} + \boldsymbol{w}_{n+k+1})^{\mathrm{T}}\big] \\ &= E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) \boldsymbol{\Phi}_{n+k+1}^{\mathrm{T}} \\ &\quad - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) \boldsymbol{\Phi}_{n+k+1}^{\mathrm{T}} \\ &= \big[E\big(\boldsymbol{\epsilon}_{n|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big) - \mathbf{A}_n E\big(\boldsymbol{\epsilon}_{n+1|n} \boldsymbol{x}_{n+k}^{\mathrm{T}}\big)\big] \boldsymbol{\Phi}_{n+k+1}^{\mathrm{T}} \\ &= \mathbf{0} \,, \end{aligned} \tag{8.74}$$

where the second equality is due to the orthogonality of $\boldsymbol{\epsilon}_{n|n}$ and $\boldsymbol{\epsilon}_{n+1|n}$ to $\boldsymbol{w}_{n+k+1}$. This established the optimality of the proposed smoother. A remaining important issue is the resulting MSE, expressed by $\mathbf{P}_n = E(\boldsymbol{\epsilon}_n \boldsymbol{\epsilon}_n^{\mathrm{T}})$ (we maintain the notation simplification in using $\mathbf{P}_n$ instead of $\mathbf{P}_{n|N}$). Note that, while in the Kalman filter the computation of $\mathbf{P}_{n|n}$ as well as $\mathbf{P}_{n|n-1}$ is inherent to the filtering equation and is therefore obtained as a byproduct, the smoother does not require computation of $\mathbf{P}_n$. Nevertheless, we shall now derive a recursive expression for this important measure of performance.

Obviously, we have $\mathbf{P}_N = \mathbf{P}_{N|N}$. Let us now rewrite (8.70) as:

$$\boldsymbol{\epsilon}_n - \mathbf{A}_n \boldsymbol{\epsilon}_{n+1} = \boldsymbol{\epsilon}_{n|n} - \mathbf{A}_n \boldsymbol{\epsilon}_{n+1|n} \,. \tag{8.75}$$

Multiplying each side by its transpose and taking the mean, we obtain

Amazon EX1010 Page 176

$$\begin{aligned}&\mathbf{P}_n+\mathbf{A}_n\mathbf{P}_{n+1}\mathbf{A}_n^{\mathrm{T}}-E\big(\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big)\mathbf{A}_n^{\mathrm{T}}-\mathbf{A}_nE\big(\boldsymbol{\epsilon}_{n+1}\boldsymbol{\epsilon}_n^{\mathrm{T}}\big)\\&=\mathbf{P}_{n|n}+\mathbf{A}_n\mathbf{P}_{n+1|n}\mathbf{A}_n^{\mathrm{T}}\\&\quad-E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{\epsilon}_{n+1|n}^{\mathrm{T}}\big)\mathbf{A}_n^{\mathrm{T}}-\mathbf{A}_nE\big(\boldsymbol{\epsilon}_{n+1|n}\boldsymbol{\epsilon}_{n|n}^{\mathrm{T}}\big)\,.\end{aligned}\tag{8.76}$$

In order to obtain a closed-form recursive expression, we need to evaluate the terms $E(\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}})$ and $E(\boldsymbol{\epsilon}_{n|n}\boldsymbol{\epsilon}_{n+1|n}^{\mathrm{T}})$. Regarding the latter, we observe that [using the recursion in (8.22)]

$$\begin{aligned}E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{\epsilon}_{n+1|n}^{\mathrm{T}}\big)&=E\big[\boldsymbol{\epsilon}_{n|n}(\boldsymbol{\Phi}_{n+1}\boldsymbol{\epsilon}_{n|n}-\boldsymbol{w}_{n+1})^{\mathrm{T}}\big]\\&=\mathbf{P}_{n|n}\boldsymbol{\Phi}_{n+1}^{\mathrm{T}}=\mathbf{A}_n\mathbf{P}_{n+1|n}\,,\end{aligned}\tag{8.77}$$

where we have exploited the orthogonality of $\boldsymbol{\epsilon}_{n|n}$ to $\boldsymbol{w}_{n+1}$, as well as the expression for $\mathbf{A}_n$ (8.69). Regarding the other term, let us first prove the identity

$$E\big[(\boldsymbol{\epsilon}_n-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1})\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big]=\mathbf{0}\,.\tag{8.78}$$

To show this, note that

$$\begin{aligned}&E\big[(\boldsymbol{\epsilon}_n-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1})\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big]\\&=E\big[(\boldsymbol{\epsilon}_n-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1})(\hat{\boldsymbol{x}}_{n+1}-\boldsymbol{x}_{n+1})^{\mathrm{T}}\big]\\&=-E\big[(\boldsymbol{\epsilon}_n-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1})\boldsymbol{x}_{n+1}^{\mathrm{T}}\big]\\&=-E\big[(\boldsymbol{\epsilon}_{n|n}-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1|n})\boldsymbol{x}_{n+1}^{\mathrm{T}}\big]\,,\end{aligned}\tag{8.79}$$

where we have used the orthogonality of $\boldsymbol{\epsilon}_n$ and $\boldsymbol{\epsilon}_{n+1}$ to $\hat{\boldsymbol{x}}_{n+1}$ (a linear function of all the measurements, to which both $\boldsymbol{\epsilon}_n$ and $\boldsymbol{\epsilon}_{n+1}$ are orthogonal), as well as the relation (8.75). Now,

$$\begin{aligned}&E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{x}_{n+1}^{\mathrm{T}}\big)=E\big[\boldsymbol{\epsilon}_{n|n}(\boldsymbol{\Phi}_{n+1}\boldsymbol{x}_n+\boldsymbol{w}_{n+1})^{\mathrm{T}}\big]\\&=E\big(\boldsymbol{\epsilon}_{n|n}\boldsymbol{x}_n^{\mathrm{T}}\big)\boldsymbol{\Phi}_{n+1}^{\mathrm{T}}=E\big[\boldsymbol{\epsilon}_{n|n}(\hat{\boldsymbol{x}}_{n|n}-\boldsymbol{\epsilon}_{n|n})^{\mathrm{T}}\big]\boldsymbol{\Phi}_{n+1}^{\mathrm{T}}\\&=-\mathbf{P}_{n|n}\boldsymbol{\Phi}_{n+1}^{\mathrm{T}}\end{aligned}\tag{8.80}$$

(again, due to orthogonality of $\boldsymbol{\epsilon}_{n|n}$ to $\boldsymbol{w}_{n+1}$ and to $\hat{\boldsymbol{x}}_{n|n}$), and

$$\begin{aligned}&E\big(\mathbf{A}_n\boldsymbol{\epsilon}_{n+1|n}\boldsymbol{x}_{n+1}^{\mathrm{T}}\big)\\&=\mathbf{A}_nE\big[\boldsymbol{\epsilon}_{n+1|n}(\hat{\boldsymbol{x}}_{n+1|n}-\boldsymbol{\epsilon}_{n+1|n})^{\mathrm{T}}\big]\\&=-\mathbf{A}_n\mathbf{P}_{n+1|n}=-\mathbf{P}_{n|n}\boldsymbol{\Phi}_{n+1}^{\mathrm{T}}\end{aligned}\tag{8.81}$$

[due to the orthogonality of $\boldsymbol{\epsilon}_{n+1|n}$ to $\hat{\boldsymbol{x}}_{n+1|n}$ and due to (8.69)]. Substituting (8.80) and (8.81) into (8.79), we obtain the desired identity (8.78). Now, the expression on the left-hand side of (8.78) can also be written as

$$\begin{aligned}&E\big[(\boldsymbol{\epsilon}_n-\mathbf{A}_n\boldsymbol{\epsilon}_{n+1})\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big]\\&=E\big(\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big)-\mathbf{A}_nE\big[\boldsymbol{\epsilon}_{n+1}\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big]\,.\end{aligned}\tag{8.82}$$

Since this term has just been shown to vanish, we deduce that

$$E\big(\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_{n+1}^{\mathrm{T}}\big)=\mathbf{A}_n\mathbf{P}_{n+1}\,.\tag{8.83}$$

We can now substitute the missing expressions (8.77) and (8.83) (and their transposed forms) into (8.76), obtaining

$$\begin{aligned}&\mathbf{P}_n+\mathbf{A}_n\mathbf{P}_{n+1}\mathbf{A}_n^{\mathrm{T}}-2\mathbf{A}_n\mathbf{P}_{n+1}\mathbf{A}_n^{\mathrm{T}}\\&=\mathbf{P}_{n|n}+\mathbf{A}_n\mathbf{P}_{n+1|n}\mathbf{A}_n^{\mathrm{T}}-2\mathbf{A}_n\mathbf{P}_{n+1|n}\mathbf{A}_n^{\mathrm{T}}\,.\end{aligned}\tag{8.84}$$

We thus obtain the following (backwards-)recursive expression for $\mathbf{P}_n$:

$$\begin{aligned}\mathbf{P}_n&=\mathbf{P}_{n|n}-\mathbf{A}_n(\mathbf{P}_{n+1|n}-\mathbf{P}_{n+1})\mathbf{A}_n^{\mathrm{T}}\,,\\&\qquad n=N-1,N-2,\ldots,1\,,\end{aligned}\tag{8.85}$$

in which it is nicely observed that, as expected, for each $n$ the smoother's MSE $\mathbf{P}_n$ is smaller than or equal to (usually smaller than) the filter's MSE $\mathbf{P}_{n|n}$. This is because the smoothed estimate is optimally based on more measurements than the filtered estimate. To observe this relation, note that for each $n<N$ we have $\mathbf{P}_{n+1|n}\geq\mathbf{P}_{n+1|n+1}$, and, since for $n=N$ we have $\mathbf{P}_{N|N}=\mathbf{P}_N$, we get, by backwards induction, $\mathbf{P}_{n+1|n}\geq\mathbf{P}_{n+1|n+1}\geq\mathbf{P}_{n+1}$ for all $n<N$.

### Fixed-Lag Smoothing

A fixed-lag smoother is aimed at estimating the state at time $n-d$ from measurements up to time $n$, where $d$ is some fixed positive integer (the *lag*). As $n$ progresses, the smoother generates the estimates $\hat{\boldsymbol{x}}_{n-d|n}$, trading a delay of $d$ measurements (in real-time contexts) for improved accuracy in the estimation of $\boldsymbol{x}_{n-d}$. We shall not specify the specific smoothing equations in here; note only that the fixed-interval smoother derived above can also be used for fixed-lag smoothing by applying $d$ backwards-recursions at each time instant. However, such a scheme is computationally more expensive than applying a true fixed-lag smoother, especially for large values of $d$.

It is important to note, that in some applications (mainly signal denoising) the state vector $\boldsymbol{x}$ contains past-samples of the underlying signal — see, e.g., the case of AR sources in Sect. 8.2. In such cases, estimates of some fixed-lag past samples of *the underlying signal(s)* (as opposed to past samples of *the entire state*) are readily available from the Kalman filter's state estimate, eliminating the need for a smoother. The maximum available lag is determined by the oldest sample contained in $\boldsymbol{x}_n$. For example, in the case of estimating AR signals in white noise, it is the AR order minus 1. If desired, it is theoretically possible to increase the size of the state vector artificially so as to include more past samples that would automatically be estimated at the filter's



Amazon EX1010 Page 177

output, thus eliminating the need for explicit smoothing, but at the cost of increased dimensionality of the Kalman filter. However, usually this trade-off has more conceptual than computational appeal.

#### Fixed-Point Smoothing

A fixed-point smoother is aimed at estimating the state at a fixed time instant $m$ from measurements up to time $n$, extending beyond $m$. Namely, as $n$ progresses, the smoother keeps updating its estimate $\hat{\boldsymbol{x}}_{m|n}$. Again, we shall not include the explicit fixed-point smoothing equation in here; note only that, in principle, the fixed-interval smoother can again be used to compute the fixed-point smoothing, but at significantly higher computational cost, especially as $n$ departs farther from $m$.

### 8.3.3 The Extended Kalman Filter

In many scenarios the assumption of linear transition and/or measurement models is not realistic. In fact, the first full implementation that transformed the Kalman filter into a useful working tool was done in the Ames Research Center of the NASA in Mountain View, California in 1960 [8.10], where the trajectory estimation for the Apollo project (the first manned mission to the moon) was studied at the time. In that context, the linear model was observed to be inadequate for describing the trajectory of interest. The resulting extended Kalman filter (EKF) was derived by Stanley F. Schmidt after Kalman's visit. This extension of the Kalman filter is still the most popular tool for dealing with nonlinear applications. However, it should be stressed that, as opposed to the case of a linear model, no claims of optimality hold, and that the EKF is merely a heuristic extension of the linear Kalman filter.

Consider the nonlinear recursive system:

$$\boldsymbol{x}_n = \boldsymbol{\phi}_n(\boldsymbol{x}_{n-1}) + \boldsymbol{w}_n, \quad n = 1, 2, \ldots , \tag{8.86}$$

where $\boldsymbol{\phi}_n(\cdot)$ is a nonlinear vector function of a vector input. $\boldsymbol{w}_n$ is the temporally uncorrelated driving noise. Consider also the nonlinear measurement equation

$$\boldsymbol{y}_n = \boldsymbol{h}_n(\boldsymbol{x}_n) + \boldsymbol{v}_n, \quad n = 1, 2, \ldots , \tag{8.87}$$

where $\boldsymbol{h}_n(\cdot)$ is a nonlinear vector function of a vector input. $\boldsymbol{v}_n$ is the temporally uncorrelated measurement noise.

The simple idea behind the EKF is to approximate the nonlinear functions by their first-order Taylor series expansion about the estimated trajectory [8.4, 5]. Let,

$$\boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) = \frac{\partial \boldsymbol{\phi}_n(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{n-1|n-1}}$$

be the derivative matrix of the nonlinear transition function with respect to its argument at the current estimate, namely, the $(k, \ell)$-th element of $\boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1})$ is the derivative of the $k$-th element of $\boldsymbol{\phi}_n(\boldsymbol{x})$ with respect to the $\ell$-th element of $\boldsymbol{x}$, evaluated at $\boldsymbol{x} = \hat{\boldsymbol{x}}_{n-1|n-1}$. Then, using a first-order Taylor approximation:

$$\begin{aligned}\boldsymbol{x}_n \approx\ & \boldsymbol{\phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) \\ & \cdot (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{w}_n, \quad n = 1, 2, \ldots\end{aligned} \tag{8.88}$$

In the *propagation* step, based on the rationale of the linear Kalman filter, we compute the estimate of $\boldsymbol{x}_n$ by applying the model equation to $\hat{\boldsymbol{x}}_{n-1|n-1}$, ignoring the unknown, zero-mean driving noise $\boldsymbol{w}_n$.

$$\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{\phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) \, . \tag{8.89}$$

To calculate the corresponding covariance we first derive the error term. Let,

$$\begin{aligned}\boldsymbol{\epsilon}_{n|n-1} &= \hat{\boldsymbol{x}}_{n|n-1} - \boldsymbol{x}_n = \boldsymbol{\phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) - \boldsymbol{x}_n \\ &\approx \boldsymbol{\phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) - (\boldsymbol{\phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1}) \\ &\quad + \boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{w}_n) \\ &= \boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1})\boldsymbol{\epsilon}_{n-1|n-1} - \boldsymbol{w}_n \, .\end{aligned} \tag{8.90}$$

In the context of nonlinear models, it is common practice to assume that the driving noise $\boldsymbol{w}_n$ and the measurement noise $\boldsymbol{v}_n$ are not only uncorrelated sequences, but are also statistically independent sequences and are also statistically independent of the initial state $\boldsymbol{x}_0$. Using this assumption, the estimation error $\boldsymbol{\epsilon}_{n-1|n-1}$ can be assumed statistically independent of the *future* driving noise $\boldsymbol{w}_n$, hence $E(\boldsymbol{\epsilon}_{n-1|n-1}\boldsymbol{w}_n^{\mathrm{T}}) = \mathbf{0}$. Therefore, the covariance is approximately given, under the small-estimation-error assumption, as:

$$\begin{aligned}&\mathbf{P}_{n|n-1} \\ &\approx \boldsymbol{\Phi}_n(\hat{\boldsymbol{x}}_{n-1|n-1})\mathbf{P}_{n-1|n-1}\boldsymbol{\Phi}_n^{\mathrm{T}}(\hat{\boldsymbol{x}}_{n-1|n-1}) + \mathbf{Q}_n \, .\end{aligned} \tag{8.91}$$

Now, let

$$\mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1}) = \frac{\partial \boldsymbol{h}_n(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{n|n-1}}$$

be the derivative of the nonlinear measurement function with respect to its argument at the current estimate. Then,

$$\begin{aligned}\boldsymbol{y}_n \approx\ & \boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n|n-1}) + \mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1}) \\ & \cdot (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{v}_n, \quad n = 1, 2, \ldots\end{aligned} \tag{8.92}$$



Amazon EX1010 Page 178

This equation can be reformulated as

$$\begin{aligned}\tilde{\boldsymbol{y}}_n &= \boldsymbol{y}_n - (\boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n|n-1}) - \mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1})\hat{\boldsymbol{x}}_{n|n-1}) \\ &\approx \mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1})\boldsymbol{x}_n + \boldsymbol{v}_n, \quad n = 1, 2, \ldots\end{aligned} \tag{8.93}$$

Note, that $\tilde{\boldsymbol{y}}_n$ depends both on the available estimate $\hat{\boldsymbol{x}}_{n|n-1}$ and on $\boldsymbol{y}_n$. Using (8.21) the update equation becomes,

$$\begin{aligned}\hat{\boldsymbol{x}}_{n|n} &= \hat{\boldsymbol{x}}_{n|n-1} + \mathbf{K}_n(\tilde{\boldsymbol{y}}_n - \mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1})\hat{\boldsymbol{x}}_{n|n-1}) \\ &= \hat{\boldsymbol{x}}_{n|n-1} + \mathbf{K}_n(\boldsymbol{y}_n - \boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n|n-1}))\,,\end{aligned} \tag{8.94}$$

where the last transition is due to (8.93). The respective covariance is approximated by

$$\mathbf{P}_{n|n} \approx [\mathbf{I} - \mathbf{K}_n(\hat{\boldsymbol{x}}_{n|n-1})\mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1})]\mathbf{P}_{n|n-1}\,. \tag{8.95}$$

The Kalman gain is similar to the linear case, using the linearized equations.

$$\begin{aligned}\mathbf{K}_n = &\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}(\hat{\boldsymbol{x}}_{n|n-1}) \\ &[\mathbf{H}_n(\hat{\boldsymbol{x}}_{n|n-1})\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}(\hat{\boldsymbol{x}}_{n|n-1}) + \mathbf{R}_n]^{-1}\,,\end{aligned} \tag{8.96}$$

Note that, as opposed to the case of the linear Kalman filter, the Kalman gain and the associated covariances cannot be calculated in advance (offline), since the matrices $\boldsymbol{\Phi}$ and $\mathbf{H}$ are, in general, data dependent.



## 8.4 The Application of the Kalman Filter to Speech Processing

### 8.4.1 Literature Survey

Speech quality and intelligibility might significantly deteriorate in the presence of background noise, especially when the speech signal is subject to subsequent processing. In particular, speech coders, and automatic speech recognition (ASR) systems, which were designed or trained to act on clean speech signals, might be rendered useless in the presence of background noise. Speech enhancement algorithms have therefore attracted a great deal of interest over the past three decades.

Among these speech enhancement algorithms there are numerous algorithms based on *Wiener* [8.11] or Kalman filtering [8.2], both of which require knowledge of the parameters involved. For example, for the application of the Kalman filter, it is common to model the speech as a quasistationary AR process [in the speech context, the AR parameters are usually called linear predictive coefficients (LPC)], which requires knowledge of the AR parameters. The noise level (and structure) should also be available for the Kalman formulation. Since these parameters are usually unknown, the problem of joint estimation of signal and parameters arises. A typical procedure, applied to this type of problems, is the estimate–maximize (EM) algorithm [8.12]. Under the associated model assumptions, this procedure is guaranteed to converge to the maximum-likelihood (ML) estimate of the problem's parameters, or at least to a local maximum of the likelihood function, and to yield an estimate of the speech signal, computed at the parameter that maximizes the likelihood.

*Paliwal* and *Basu* [8.13] were, perhaps, the first to use the Kalman filter in the context of speech enhancement. Their experimental results reveal its distinct advantage over the Wiener filter, due to the ability of the Kalman filter to account for the quasistationarity of the speech signal. However, in their experiment the unknown speech parameters were not estimated from the noisy speech signal, but rather from the clean signal, which is obviously unavailable in most practical situations. *Gibson* et al. [8.14, 15] proposed to extend the use of the Kalman filter by incorporating a colored-noise model for improving the enhancement performance for a certain class of noise sources. The proposed algorithm iterates between Kalman filtering of the given corrupted speech measurements, and estimation of the speech parameters given the enhanced speech waveform. Since the authors suggest using the ordinary Yule–Walker equations [8.16] for estimating the speech AR parameters, the resulting algorithm is only an approximated version of the EM algorithm. The estimated speech LPC parameters were shown to improve speech coding systems that rely on AR modeling of the speech signal.

A comprehensive study of the use of the EM algorithm in diverse problems of joint estimation of signals and parameters is given in a series of works by Weinstein et al. In [8.17], the noise cancelation problem presented by *Widrow* [8.18] is addressed using the EM algorithm in the frequency domain. The more-general two-channel noise cancelation problem is addressed in [8.19]. Both are comprised of iterations between parameter estimation and Wiener filtering. In [8.20], a time-domain formulation of the single-microphone speech enhancement problem is presented. The approach consists of representing the signal model using linear dynamic state equations, and applying the EM method. The resulting algorithm is similar in structure to the *Lim* and *Oppenheim* [8.21] algorithm, except that the noncausal

Wiener filter is replaced by the Kalman smoothing equations. Sequential speech enhancement algorithms are presented as well. These sequential algorithms are characterized by a forward Kalman filter whose parameters are continuously updated by gradient-descent search on the likelihood function. In [8.22, 23] sequential approximations to the EM algorithm are elaborated on in the context of two-channels noise cancelation. The related problem of single-microphone active noise cancelation (ANC) is presented in [8.24].

*Lee* et al. [8.25, 26] extend the sequential single sensor algorithm of Weinstein et al. by considering an alternative speech-generation model, in which the white Gaussian excitation sequence is replaced with a Gaussian-mixture distributed sequence, that may account for the presence of an impulse train in the excitation sequence of voiced speech. A recursive gradient-based approach is applied to the parameter estimation. Lee et al. examined the signal-to-noise ratio (SNR) improvement of the algorithm when applied to synthetic speech input. *Goh* et al. [8.27] propose different modeling of the speech excitation sequence. The proposed model is comprised of both Gaussian white noise (modeling the unvoiced part) and an impulse train (presenting the voiced part). The latter is modeled as a long-term AR process. The resulting high-dimensional Kalman filter is efficiently implemented by exploiting the sparsity of the associated matrices. The parameter estimation is conducted via EM iterations. When the standard Kalman filter gain is recursively computed, one needs to estimate the speech and noise gains. To avoid this estimation stage, *Gabrea* et al. [8.28] propose checking the whiteness of the innovation sequence to test whether the asymptotically optimal solution has been reached. Since the estimation of the AR parameters cannot be avoided, Gabrea et al. propose to use the modified Yule–Walker procedure [8.29]. Another extension to the work of Weinstein et al. was proposed by *Gannot* et al. [8.30]. In this work both iterative batch and sequential versions of the EM algorithm are considered. The estimate stage (E-step) is implemented by applying the Kalman filter. Higher-order statistics (HOS) are employed to obtain a robust initialization to the parameter estimation stage (called the maximization stage – M-step). *Fujimoto* and *Ariki* [8.31] use Kalman filtering in the frequency domain (without using the AR model). Initialization of their algorithm is obtained by the classical spectral subtraction [8.32] algorithm (see also the relevant Chap. 44 in this Handbook).

Several nonlinear extensions to the standard Kalman filter exist. *Lee* et al. [8.33] propose the application of a robust Kalman filter. Similar to other contributions, iterations between signal enhancement and parameter estimation are conducted. The novelty of that paper stems from the use of nonlinear estimation procedures. Both the parameters and signals are estimated in a robust manner by introducing a saturation function into the cost function, rather than using the standard squared cost function. *Ma* et al. [8.34] introduce the application of a postfilter based on masking properties of the human auditory system to further enhance the resulting auditory quality of the speech signal. However, not much attention is paid in this work to the estimation of the associated AR parameters.

*Shen* and *Deng* [8.35] present a new and interesting approach to speech enhancement based on $H_\infty$ filtering. This approach differs from the traditional Kalman filtering approach in the definition of the error criterion for the filter design. Rather than minimizing a squared error term, as in the standard Kalman filter, their procedure consists of calculating a filter that minimizes the worst possible amplification of the estimation error in terms of the modeling errors and additive noises. The parameters are estimated in parallel using the $H_\infty$ criterion as well. The authors claim that their resulting minimax estimation method is highly robust and more appropriate in practical speech enhancement. It should be noted, however, that the implementation of the minimax criterion in the parameter estimation stage of the algorithm seems to be much more complicated than the conventional estimation procedure.

*Wan* et al. [8.36] assume a nonlinear model of the speech production, i. e., that the speech utterance is the output of a neural network (NN) with unknown parameters. Their algorithm is comprised of iterations between parameter estimation and signal enhancement. The nonlinearity inherent to the NN is addressed by the application of the EKF. The recently proposed unscented transform (UT), suggested by *Julier* et al. [8.37], is a novel method for calculating first- and second-order statistics of a random variable undergoing a nonlinear transformation, that was used for constructing a Kalman filter for the nonlinear case. The resulting filter, named the unscented Kalman filter (UKF), was shown to be superior to the well-established EKF in many application of interest. *Wan* et al. [8.38] propose to replace the EKF in [8.36] by the UKF, resulting in improved performance. *Gannot* and *Moonen* [8.39] use the UKF in the speech enhancement application (as well as speech dereverberation), where the nonlinearity arises from the multiplication of the speech and the parameters. Their proposed method is only applied to a synthetic AR process. *Fong* and *God-*

*sill* [8.40] use the particle Kalman filter for the speech enhancement task. The speech signal gain is given a random walk model, while its partial correlation coefficients (PARCOR) [8.41] are given a constrained random walk model (as their absolute value must be smaller than 1). Monte Carlo filtering is applied for estimating these parameters, whereas a linear Kalman filter is applied in parallel for enhancing the speech signal.

A distinct family of algorithms, also using the Kalman filter, employs a training stage for estimation of the associated parameters. In this approach a hidden Markov model (HMM) for the clean signals is estimated from the training data, and the clean signal is estimated from the noisy signal by applying Bayesian estimators. This method was first proposed by *Ephraim* et al. [8.42, 43], where a bank of HMM state-related Wiener filters was used. The Wiener filter was later replaced by the Kalman filter in [8.44, 45]. The problem of unknown speech gain contours and noise parameters is alleviated by using EM iterations. This method, proposed by *Lee* and *Jung* [8.46], use Kalman filter in the E-step, based on the trained AR parameters and the estimated noise parameters. In the M-step the noise and gain parameters are recursively estimated. An interacting multiple model (IMM) algorithm, in which the Kalman filters in the different states interact with one another, is applied for enhancing speech contaminated by additive white or colored noise by *Kim* et al. [8.47]. Finally, a nonlinear extension of the HMM concept is proposed by *Lee* et al. [8.48]. The speech is assumed to be an output of a NN with time-varying parameters controlled by a hidden Markov chain. Both the training and the enhancement stage become nonlinear. The nonlinearity is addressed by application of the EKF.

## 8.4.2 Speech Enhancement

In the following derivation we model both the speech and noise signals as AR processes. Hence, the case of temporally correlated noise, derived in Sect. 8.2.4, is considered. Note, however, that the speech and noise parameters are not known and should be estimated. We propose the framework of the EM algorithm for jointly estimating the signals and their parameters. For the application of EM we have to specify the probability distribution of the processes involved. In order to obtain a tractable estimation scheme, we shall assume Gaussian distribution models. Recall that in a Gaussian framework the Kalman filter's output is not only the optimal linear estimate, but also the general optimal estimate, namely, $\hat{\boldsymbol{x}}_{n|n} = E(\boldsymbol{x}_n|\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$.

For the full derivation of the related EM estimation scheme the reader is referred to [8.49].

### The Signal Model

Consider a speech signal received by a single microphone and contaminated by a colored noise signal. Let the signal measured by the microphone be given by:

$$y(n) = x(n) + v(n)\,, \tag{8.97}$$

where $x(n)$ represents the sampled speech signal, and $v(n)$ represents additive background noise. We assume the standard LPC modeling for the speech signal over an analysis frame; i. e., $x(n)$ is modeled as a Gaussian stochastic AR process:

$$x(n) = -\sum_{k=1}^{p} \alpha_k x(n-k) + w^{\mathrm{s}}(n)\,, \tag{8.98}$$

where the excitation $w^{\mathrm{s}}(n)$ is a zero-mean white Gaussian noise with variance $E\{[w^{\mathrm{s}}(n)]^2\} = \sigma^2_{w^{\mathrm{s}}}$, and $\alpha_1, \ldots, \alpha_p$ are the AR coefficients, assumed to be time invariant over an analysis frame, due to the quasistationarity assumption. We may incorporate the more-detailed voiced speech model suggested in [8.50] in which the excitation process is composed of a weighted linear combination of an impulse train and a white-noise sequence to represent voiced and unvoiced speech, respectively. However, in our experiments, this approach did not yield any significant performance improvements over the standard LPC modeling. In reformulating (8.98) in the state-space representation, we take an approach that slightly differs from the state-space formulation of an AR process, defined in Sect. 8.2. The state vector, defined here, is a $(p+1)\times 1$ vector. The extra term will allow for more-convenient parameter estimation. Define

$$\boldsymbol{x}^{\mathrm{s}}_n = (x(n-p)\; x(n-p+1)\; \ldots\; x(n))^{\mathrm{T}}\,.$$

Define also the $(p+1)\times(p+1)$ speech transition matrix

$$\boldsymbol{\Phi}^{\mathrm{s}} = \begin{pmatrix} 0 & 1 & 0 & & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & -\alpha_p & -\alpha_{p-1} & \cdots & -\alpha_2 & -\alpha_1 \end{pmatrix}\,.$$

Let $\boldsymbol{w}^{\mathrm{s}}_n$ be the $(p+1)\times 1$ driving noise vector and $\mathbf{H}^{\mathrm{s}}$ be the $1\times(p+1)$ the measurement matrix

$$\boldsymbol{w}^{\mathrm{s}}_n = (0\; \ldots\; 0\; w^{\mathrm{s}}(n))^{\mathrm{T}}\,,$$
$$\mathbf{H}^{\mathrm{s}} = (0\; \ldots\; 0\; 1)\,.$$

Then (8.98) can be rewritten as

$$\begin{aligned} \boldsymbol{x}_n^{\mathrm{s}} &= \boldsymbol{\Phi}^{\mathrm{s}} \boldsymbol{x}_{n-1}^{\mathrm{s}} + \boldsymbol{w}_n^{\mathrm{s}} \,, \\ y(n) &= \mathbf{H}^{\mathrm{s}} \boldsymbol{x}_n^{\mathrm{s}} + v(n) \,. \end{aligned} \tag{8.99}$$

The additive noise $v(n)$ is also assumed to be a realization from a zero-mean, possibly nonwhite Gaussian AR process:

$$v(n) = -\sum_{k=1}^{q} \beta_k v(n-k) + w^{\mathrm{v}}(n) \,, \tag{8.100}$$

where $\beta_1, \ldots, \beta_q$ are the AR parameters of the noise process, and the excitation $w^{\mathrm{v}}(n)$ is a white Gaussian noise with variance $E\{[w^{\mathrm{v}}(n)]^2\} = \sigma_{w^{\mathrm{v}}}^2$. Many common noise sources may indeed be closely approximated as low-order AR processes. Significant improvement may be obtained by incorporating the colored-noise model into the estimation process [8.15, 30].

Equation (8.100) can be rewritten in a state-space formulation as well. Define the $(q+1)\times 1$ state vector $\boldsymbol{x}_n^{\mathrm{v}} = (v(n-q) \;\; v(n-q+1) \;\; \ldots \;\; v(n))^{\mathrm{T}}$ and the $(q+1)\times(q+1)$ noise transition matrix

$$\boldsymbol{\Phi}^{\mathrm{v}} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 0 \;\; 1 \\ 0 & -\beta_q & -\beta_{q-1} & \cdots & -\beta_2 \;\; -\beta_1 \end{pmatrix} .$$

Let $\boldsymbol{w}_n^{\mathrm{v}}$ be the $(q+1)\times 1$ driving noise vector and $\mathbf{H}^{\mathrm{v}}$ be the $1\times(q+1)$ measurement vector

$$\begin{aligned} \boldsymbol{w}_n^{\mathrm{v}} &= (0 \;\ldots\; 0 \; w^{\mathrm{v}}(n))^{\mathrm{T}} \,, \\ \mathbf{H}^{\mathrm{v}} &= (0 \;\ldots\; 0 \; 1) \,. \end{aligned}$$

Using these definitions (8.100) can be rewritten as

$$\begin{aligned} \boldsymbol{x}_n^{\mathrm{v}} &= \boldsymbol{\Phi}^{\mathrm{v}} \boldsymbol{x}_{n-1}^{\mathrm{v}} + \boldsymbol{w}_n^{\mathrm{v}} \,, \\ y(n) &= s(n) + \mathbf{H}^{\mathrm{v}} \boldsymbol{x}_n^{\mathrm{v}} \,. \end{aligned} \tag{8.101}$$

Equations (8.99) and (8.101) can be concatenated into a one, larger, state-space equation.

$$\begin{aligned} \boldsymbol{x}_n &= \boldsymbol{\Phi} \boldsymbol{x}_{n-1} + \boldsymbol{w}_n \,, \\ y(n) &= \mathbf{H} \boldsymbol{x}_n \,. \end{aligned} \tag{8.102}$$

The augmented $(p+q+2)\times 1$ state vector $\boldsymbol{x}_n$ is defined by

$$\begin{aligned} \boldsymbol{x}_n^{\mathrm{T}} &= \left( (\boldsymbol{x}_n^{\mathrm{s}})^{\mathrm{T}} \; (\boldsymbol{x}_n^{\mathrm{v}})^{\mathrm{T}} \right) \\ &= \left( (\bar{\boldsymbol{x}}_n^{\mathrm{s}})^{\mathrm{T}} \; x(n) \; (\bar{\boldsymbol{x}}_n^{\mathrm{v}})^{\mathrm{T}} \; v(n) \right) , \end{aligned} \tag{8.103}$$

where $\bar{\boldsymbol{x}}_n^{\mathrm{s}}$ and $\bar{\boldsymbol{x}}_n^{\mathrm{s}}$ are $p\times 1$ and $q\times 1$ regression vectors, respectively, defined as

$$\begin{aligned} \bar{\boldsymbol{x}}_n^{\mathrm{s}} &= (x(n-p) \; x(n-p+1) \; \ldots \; x(n-1))^{\mathrm{T}} \,, \\ \bar{\boldsymbol{x}}_n^{\mathrm{v}} &= (v(n-q) \; v(n-q+1) \; \ldots \; v(n-1))^{\mathrm{T}} \,. \end{aligned}$$

The augmented state-transition matrix $\boldsymbol{\Phi}$ is given by

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\Phi}^{\mathrm{s}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}^{\mathrm{v}} \end{pmatrix} .$$

The driving noise vector is given by

$$\boldsymbol{w}_n^{\mathrm{T}} = \left( (\boldsymbol{w}_n^{\mathrm{s}})^{\mathrm{T}} \; (\boldsymbol{w}_n^{\mathrm{v}})^{\mathrm{T}} \right)$$

and the measurement vector is given by

$$\mathbf{H} = (\mathbf{H}^{\mathrm{s}} \;\; \mathbf{H}^{\mathrm{v}}) \,.$$

The respective driving noise covariance matrices are given by the $(p+1)\times(p+1)$ matrix $\mathbf{Q}^{\mathrm{s}}$ and the $(q+1)\times(q+1)$ matrix $\mathbf{Q}^{\mathrm{v}}$ as

$$\mathbf{Q}^{\mathrm{s}} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \sigma_{w^{\mathrm{s}}}^2 \end{pmatrix}$$

$$\mathbf{Q}^{\mathrm{v}} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \sigma_{w^{\mathrm{v}}}^2 \end{pmatrix}$$

and the augmented covariance matrix

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}^{\mathrm{s}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^{\mathrm{v}} \end{pmatrix} .$$

Note that the augmented state-space representation (8.102) is noise free. The signal $v(n)$ is treated as another source signal, rather than as a noise signal.

Assuming that all signal and noise parameters are known, which implies that $\boldsymbol{\Phi}$, $\mathbf{H}$, and $\mathbf{Q}$ are known, the optimal MMSE linear state estimate of $\boldsymbol{x}_n$ (consisting of the desired speech signal $s(n)$), given the frame samples $\{y(0), y(1), \ldots, y(N-1)\}$, is obtained by using the Kalman smoothing equations. However, since the signal and noise parameters are not known a priori, they must be estimated within the algorithm as well. If the casual Kalman filter is used, rather than the Kalman smoother, only a suboptimal estimation is obtained, but the performance loss is usually not too significant.

Amazon EX1010 Page 182

### EM-Based Algorithm

The following (simplified) iterative algorithm was derived in [8.30, 49]. The algorithm that iterates between state and parameter estimation can be viewed as a simplified version of the EM [8.12] procedure for the problem at hand.

Let $\boldsymbol{\theta}$ be the vector of unknown parameters in the extended model

$$\boldsymbol{\theta}^{\mathrm{T}} = \left(\boldsymbol{\alpha}^{\mathrm{T}}\ \sigma^2_{w^{\mathrm{s}}}\ \boldsymbol{\beta}^{\mathrm{T}}\ \sigma^2_{w^{\mathrm{v}}}\right), \tag{8.104}$$

where

$$\boldsymbol{\alpha}^{\mathrm{T}} = (\alpha_p\ \alpha_{p-1}\ \ldots\ \alpha_1),$$
$$\boldsymbol{\beta}^{\mathrm{T}} = (\beta_q\ \beta_{q-1}\ \ldots\ \beta_1).$$

The EM algorithm is an iterative procedure for computing the ML estimate of the parameters. In our application, the noisy signal $y(n)$ is divided into non-overlapping frames comprising the samples $\{y(0), y(1)\ldots, y(N-1)\}$, where $N$ is the frame length. The EM algorithm is applied to each frame separately. Denote by $\ell$ the iteration index in each frame. Let $\hat{\boldsymbol{\theta}}^{(\ell)}$ be the estimate of $\boldsymbol{\theta}$ after $\ell$ iterations of the algorithm in the current frame. Define, $\boldsymbol{y}_0^n = \{y(0), y(1), \ldots, y(n)\}$, the samples available for casual estimation. We use the notations

$$\hat{\boldsymbol{x}}_{n|n} = E_{\hat{\boldsymbol{\theta}}^{(\ell)}}\left(\boldsymbol{x}_n|\boldsymbol{y}_0^n\right) \tag{8.105}$$
$$\widehat{\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}} = E_{\hat{\boldsymbol{\theta}}^{(\ell)}}\left(\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}|\boldsymbol{y}_0^n\right)$$

for designating estimates of the first- and second-order statistics of the state vector, based on observations $\boldsymbol{y}_0^n$, and using the current parameter estimate $\hat{\boldsymbol{\theta}}^{(\ell)}$. In the Gaussian case, the term $\hat{\boldsymbol{x}}_{n|n}$ of (8.105) is exactly the output of the Kalman filter (operating with the parameters set $\hat{\boldsymbol{\theta}}^{(\ell)}$) bearing the same notation. Note, however, that in the non-Gaussian case the Kalman filter's output is merely the optimal *linear* estimate of $\boldsymbol{x}_n$ from $\boldsymbol{y}_0^n$, which differ from the conditional mean, namely from the optimal *general* estimate used in (8.105). The second-moment term $\widehat{\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}}$ is not directly available from the Kalman filter, however it may be obtained indirectly as follows.

The two terms in (8.105) are related via the corresponding conditional covariance matrix

$$\widehat{\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}} - \hat{\boldsymbol{x}}_{n|n}\hat{\boldsymbol{x}}_{n|n}^{\mathrm{T}} = E_{\hat{\boldsymbol{\theta}}^{(\ell)}}\left(\hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_n\right) \cdot (\hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_n)^{\mathrm{T}}|\boldsymbol{y}_0^n\big). \tag{8.106}$$

Recall that the Kalman filter's error covariance $\mathbf{P}_{n|n}$ is essentially the unconditional MSE

$$\mathbf{P}_{n|n} = E_{\hat{\boldsymbol{\theta}}^{(\ell)}}\left[(\hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_n)(\hat{\boldsymbol{x}}_{n|n} - \boldsymbol{x}_n)^{\mathrm{T}}\right], \tag{8.107}$$

Luckily, however, in the Gaussian case the conditional MSE of the optimal estimate $\boldsymbol{x}_{n|n}$ does not depend on the data $\boldsymbol{y}_0^n$. This implies, by taking the mean of the conditional MSE with respect to $\boldsymbol{y}_0^n$, that the conditional MSE equals the (constant) unconditional MSE. We therefore conclude that the conditional covariance in (8.106) is given by the Kalman filter's output MSE $\mathbf{P}_{n|n}$.

Hence, the following relation can be used (in the Gaussian case) for estimating the second-order statistics of the state vector based on the observations $\boldsymbol{y}_0^n$ and the current parameter estimate

$$\widehat{\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}} = \mathbf{P}_{n|n} + \hat{\boldsymbol{x}}_{n|n}\hat{\boldsymbol{x}}_{n|n}^{\mathrm{T}}. \tag{8.108}$$

The terms $\hat{\boldsymbol{x}}_{n|n}$ and $\mathbf{P}_{n|n}$ will be calculated in the sequel using a forward Kalman filtering recursion.

To obtain the parameter estimation in the next iteration, $\hat{\theta}^{(\ell+1)}$, we use the following EM-based two-stage iterative procedure. The Kalman filter constitutes the estimation stage (also known as the E-step) of the EM algorithm. The Yule–Walker-based parameter estimation constitutes the maximization stage of the algorithm, known as the M-step.

E-step:

For $n = 0, 1, 2, \ldots, N-1$:

Propagation equations:

$$\hat{\boldsymbol{x}}_{n|n-1} = \hat{\boldsymbol{\Phi}}^{(\ell)}\hat{\boldsymbol{x}}_{n-1|n-1}$$
$$\mathbf{P}_{n|n-1} = \hat{\boldsymbol{\Phi}}^{(\ell)}\mathbf{P}_{n-1|n-1}(\hat{\boldsymbol{\Phi}}^{(\ell)})^{\mathrm{T}} + \hat{\mathbf{Q}}^{(\ell)}$$

Update equations:

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{k}_n[y(n) - \mathbf{H}\hat{\boldsymbol{x}}_{n|n-1}]$$
$$\mathbf{P}_{n|n} = \mathbf{P}_{n|n-1} - \boldsymbol{k}_n\mathbf{H}\mathbf{P}_{n|n-1}$$

Kalman gain:

$$\boldsymbol{k}_n = \frac{1}{\mathbf{H}\mathbf{P}_{n|n-1}\mathbf{H}^{\mathrm{T}}}\mathbf{P}_{n|n-1}\mathbf{H}^{\mathrm{T}}$$

$\hat{\boldsymbol{\Phi}}^{(\ell)}$ and $\hat{\mathbf{Q}}^{(\ell)}$ are estimates of the matrices $\boldsymbol{\Phi}$ and $\mathbf{Q}$, respectively, at the current iteration stage $\ell$.

The estimated parameters in the next iteration $\ell+1$ are computed by using equation sets similar to the standard Yule–Walker solution for estimating the coefficients of an AR process, except that the correlation values are replaced by their a posteriori empirical value. This calculation constitutes the M-step of the algorithm.



Amazon EX1010 Page 183

M-step:

$$\hat{\boldsymbol{\alpha}}^{(l+1)} = -\left[\sum_{n=0}^{N-1} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{s}}(\bar{\boldsymbol{x}}_n^{\mathrm{s}})^{\mathrm{T}}}\right]^{-1} \sum_{n=0}^{N-1} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{s}} x(n)}$$

$$\widehat{\sigma_{w^{\mathrm{s}}}^2}^{(l+1)} = \frac{1}{N} \sum_{n=0}^{N-1} \left[\widehat{x^2(n)} + (\hat{\boldsymbol{\alpha}}^{(l+1)})^{\mathrm{T}} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{s}} x(n)}\right]$$

$$\hat{\boldsymbol{\beta}}^{(l+1)} = -\left[\sum_{n=0}^{N-1} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{v}}(\bar{\boldsymbol{x}}_n^{\mathrm{v}})^{\mathrm{T}}}\right]^{-1} \sum_{n=0}^{N-1} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{v}} v(n)}$$

$$\widehat{\sigma_{w^{\mathrm{v}}}^2}^{(l+1)} = \frac{1}{N} \sum_{n=0}^{N-1} \left[\widehat{v^2(n)} + (\hat{\boldsymbol{\beta}}^{(l+1)})^{\mathrm{T}} \widehat{\bar{\boldsymbol{x}}_n^{\mathrm{v}} v(n)}\right]$$

We note that $\widehat{\bar{\boldsymbol{x}}_n^{\mathrm{s}}(\bar{\boldsymbol{x}}_n^{\mathrm{s}})^{\mathrm{T}}}$ is the upper left $p \times p$ submatrix of $\widehat{\boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}}}$. $\widehat{\bar{\boldsymbol{x}}_n^{\mathrm{s}} x(n)}$, $\widehat{x^2(n)}$, $\widehat{\bar{\boldsymbol{x}}_n^{\mathrm{v}}(\bar{\boldsymbol{x}}_n^{\mathrm{v}})^{\mathrm{T}}}$, $\widehat{\bar{\boldsymbol{x}}_n^{\mathrm{v}} v(n)}$, and $\widehat{v^2(n)}$ may similarly be extracted from $\widehat{\boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}}}$. In fact, this is the reason for choosing $p+1$- and $q+1$-dimensional state vectors for this estimation scheme, although, as seen in Sect. 8.2, $p$- and $q$-dimensional state vectors are sufficient for Kalman filtering when the parameters are known.

Although the forward Kalman filter was used instead of the optimal Kalman smoother, it is empirically shown that the EM-based method maintains a monotonic convergence behavior towards the ML estimate of all unknown parameters or at least to a local maximum of the likelihood function. Each iteration increases the likelihood of the estimate of the parameters and improves the signal state-space estimator in MMSE sense.

The iterative algorithm [designated the Kalman EM iterative (KEMI) method in [8.30]] is summarized in Fig. 8.2. Note, that two other blocks are depicted in the figure. The first block, denoted *framing*, is responsible for the segmentation of the noisy speech signal. The second block is the *initialization* block. The initialization procedure, based on the use of HOS, is beyond the scope of this summary.

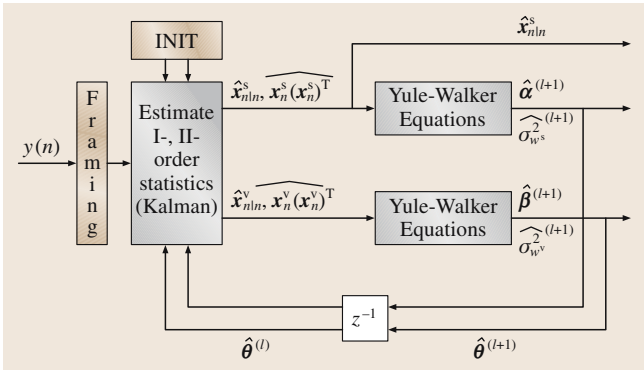



**Fig. 8.2** Iterative Kalman algorithm based on the EM procedure

### Demonstration

The performance of the KEMI algorithm is demonstrated using the sonograms depicted in Fig. 8.3. We used a speech signal drawn from TIMIT [8.51] and downsampled to 8 kHz. The speech signal was degraded by additive noise at SNR = 5 dB. The noise source was the speech-like noise signals drawn from the NOISEX-92 [8.52] database. In applying the algorithm we used an AR model with order $p = 10$ for the speech signal and for modeling the nonwhite noise signal we used an AR model with order $q = 4$. In this experiment, we used five iterations.

It is clearly shown in Fig. 8.3 that the noise signal is attenuated while imposing only minor distortion on the speech signal. For a more-comprehensive and comparative experimental study, including objective distortion measures as well as recognition rate of a speech recognition algorithm, please refer to [8.49].

## 8.4.3 Speaker Tracking

Determining the spatial position of a speaker finds a growing interest in video conference scenarios where automated camera steering and tracking are required. Acoustic source localization might also be used as a preprocessor stage for speech enhancement algorithms, which are based on microphone array beamformers.

Usually, methods for speaker localization consist of two stages. In the first stage, a microphone array is used to extract the time difference between arrivals of the speech signal at each pair of microphones. For each pair the time difference of arrival (TDOA) is estimated using spatially separated microphone pairs. A classical method for estimating the TDOA is the generalized cross-correlation (GCC) algorithm [8.53].

In the second stage the noisy TDOA readings from all pairs are combined in order to produce the source location estimate. In a dynamic setting the source location may be time-varying, thus its estimate must be constantly updated (tracked) as new TDOA readings become available. It has become common practice to employ Kalman filtering in such tracking problems, so as to properly weigh the effect of incoming TDOA readings on the estimated source location, accounting for a smooth propagation model of the source's trajectory (when applicable). Although not directly related to the processing of the speech signal itself, the use of Kalman

Amazon EX1010 Page 184

filtering in this problem serves to demonstrate the diversity and versatility of this remarkable and practical estimation tool.

### Algorithm Derivation

Consider an $M+1$-microphone array as depicted in Fig. 8.4.

The microphones are placed at the Cartesian coordinates $\boldsymbol{m}_i = (x_i \;\; y_i \;\; z_i)^{\mathrm{T}}$; $i = 0, \ldots, M$. To simplify the exposition, the location of a reference microphone $\boldsymbol{m}_0$ is set at the origin of the axes $\boldsymbol{m}_0 = (0 \;\; 0 \;\; 0)^{\mathrm{T}}$. Define the source coordinates at time instant $n$ by $\boldsymbol{s}_n = (x_{\mathrm{s}}(n) \;\; y_{\mathrm{s}}(n) \;\; z_{\mathrm{s}}(n))^{\mathrm{T}}$. Each of the $M$ microphones, combined with the reference microphone, is used at time instant $n$ to extract a TDOA measurement $\tau_i(n)$; $i = 1, \ldots, M$. Denote the $i$-th range difference measurement by $r_i(n) = c\tau_i(n)$, where $c$ is the sound propagation speed (approximately 340 m/s in air). It can easily be verified from simple geometrical considerations (see Fig. 8.4) that this range difference is related to the source and the microphone location by the nonlinear equation

$$r_i(n) = \|\boldsymbol{s}_n - \boldsymbol{m}_i\| - \|\boldsymbol{s}_n\|; \; i = 1, \ldots, M\,, \qquad (8.109)$$

where $\|\boldsymbol{s}_n\| = \sqrt{x_s^2(n) + y_s^2(n) + z_s^2(n)}$ denote the Euclidean norm of the source coordinates.

Usually, only estimates of the true TDOAs are available. Thus, concatenating $M$ estimates of the quantity in (8.109), a nonlinear measurement model is obtained:

$$\boldsymbol{r}_n = \begin{pmatrix} \|\boldsymbol{s}_n - \boldsymbol{m}_1\| - \|\boldsymbol{s}_n\| \\ \vdots \\ \|\boldsymbol{s}_n - \boldsymbol{m}_M\| - \|\boldsymbol{s}_n\| \end{pmatrix} + \boldsymbol{v}_n = \boldsymbol{h}(\boldsymbol{s}_n) + \boldsymbol{v}_n\,. \qquad (8.110)$$

Here, $\boldsymbol{v}_n^{\mathrm{T}} = (v_1(n) \;\; v_2(n) \;\; \ldots \;\; v_M(n))$ is a vector of estimation errors with zero mean and covariance $\mathbf{R}_n = E(\boldsymbol{v}_n \boldsymbol{v}_n^{\mathrm{T}})$. The goal of the localization task is to extract the speaker's trajectory $\boldsymbol{s}_n$ from the measurements vector $\boldsymbol{r}_n$. Any estimation procedure (e.g., [8.53]) could be used for the TDOA estimation. The method introduced in the sequel, constituting the second stage of the localization procedure, is independent of the choice of the first stage.

We propose a dynamic model for the source trajectory. As the actual track is unknown, a simplified random walk model is used instead.

$$\boldsymbol{s}_{n+1} = \boldsymbol{\Phi}\boldsymbol{s}_n + \boldsymbol{w}_n\,, \qquad (8.111)$$

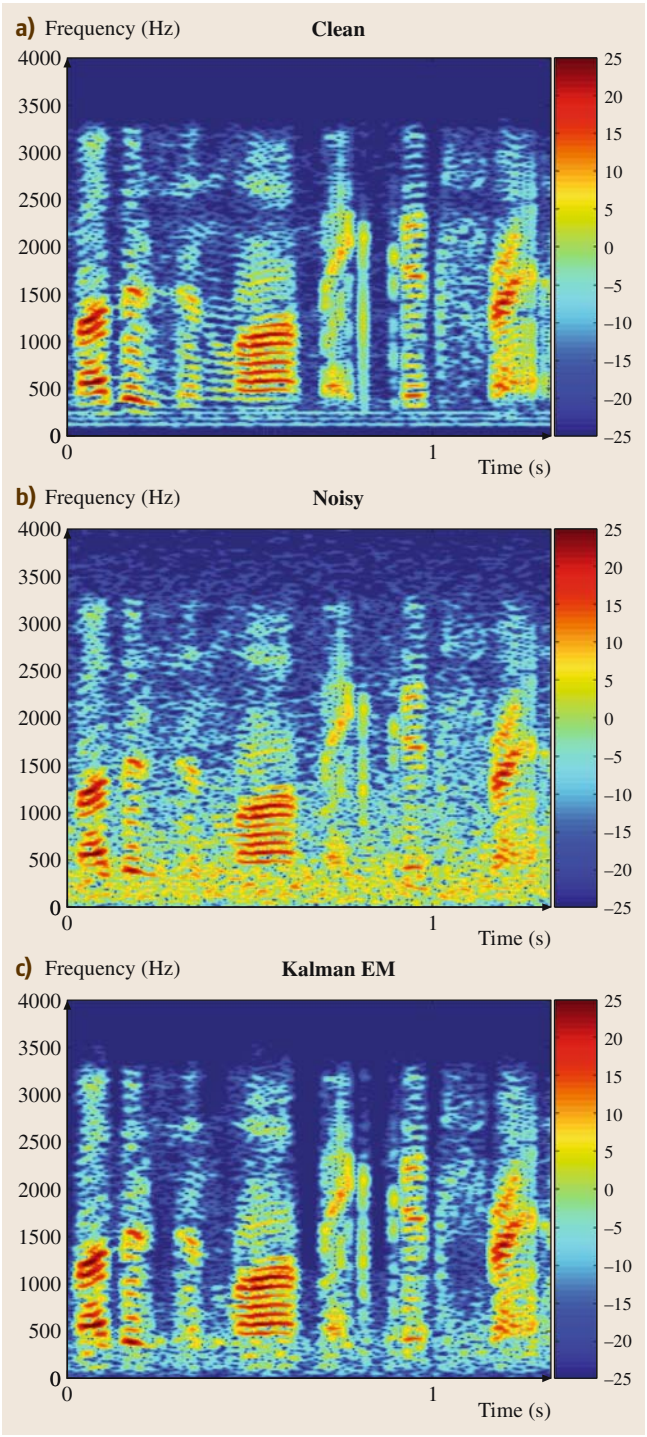



**Fig. 8.3a–c** Iterative Kalman algorithm based on the EM procedure



Amazon EX1010 Page 185

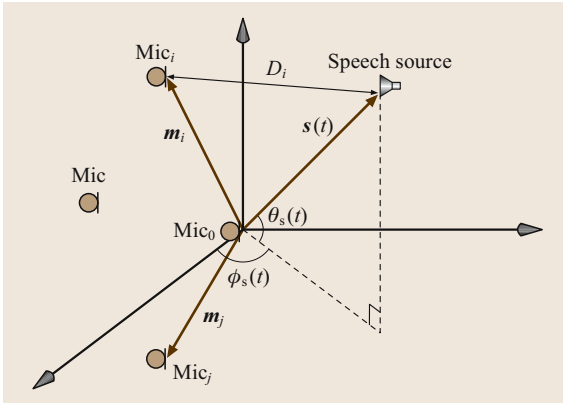



**Fig. 8.4** Microphone array. Speaker location at time instant $n$ is $\boldsymbol{s}_n$ with azimuth angle $\phi_s(n)$ and elevation angle $\theta_s(n)$. Microphone position notated by $\boldsymbol{m}_i$; $i = 0, \ldots, M$

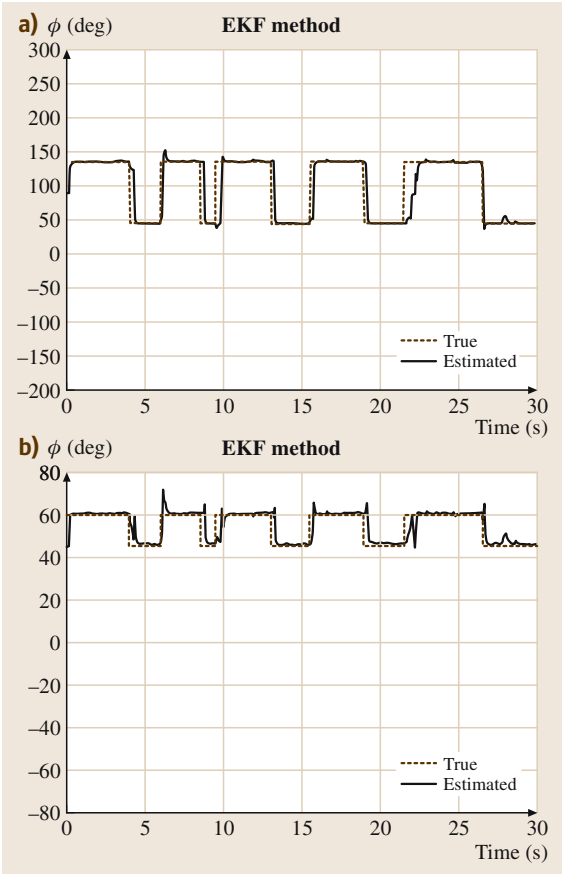



**Fig. 8.5a,b** Azimuth angle, $\phi$ **(a)**, and elevation angle, $\theta$ **(b)** estimation results in a switching scenario

where $\boldsymbol{w}_n$ is the coordinate-wise, zero-mean, temporally white driving noise with covariance matrix $\mathbf{Q}_n = E(\boldsymbol{w}_n \boldsymbol{w}_n^{\mathrm{T}})$. $\boldsymbol{\Phi}$ is a transition matrix assumed to be close to the identity matrix. This dynamic model together with the nonlinear measurement model given in (8.110) constitutes the state-space equation.

Since this model is nonlinear (due to the measurement equation) the classical Kalman filter cannot be used to estimate the state vector. Hence, a nonlinear extension thereof is required. We propose to use the EKF, derived in Sect. 8.3.3. This procedure only yields a suboptimal solution to the problem at hand. We provide here, for the completeness of the exposition, the calculations involved in the EKF aiming to solve the localization problem. Note, that in our case, (8.111) is already linear. However the measurement model in (8.110) still needs to be linearized. Assume that an estimate $\hat{\boldsymbol{s}}_{n-1|n-1}$ of the speaker location at time instant $n-1$, is known, as well as its corresponding error covariance matrix, $\mathbf{P}_{n-1|n-1}$. Then, recalling that the model equation is linear, the EKF recursion takes the following form.

Propagation equations:

$$\hat{\boldsymbol{s}}_{n|n-1} = \boldsymbol{\Phi}\hat{\boldsymbol{s}}_{n-1|n-1} \tag{8.112a}$$

$$\mathbf{P}_{n|n-1} = \boldsymbol{\Phi}\mathbf{P}_{n-1|n-1}\boldsymbol{\Phi}^{\mathrm{T}} + \mathbf{Q}_n \tag{8.112b}$$

Update equations:

$$\hat{\boldsymbol{s}}_{n|n} = \hat{\boldsymbol{s}}_{n|n-1} + \mathbf{K}_n(\boldsymbol{r}_n - \boldsymbol{h}(\hat{\boldsymbol{s}}_{n|n-1})) \tag{8.112c}$$

$$\mathbf{H}_n = \nabla_{\boldsymbol{s}_n}\boldsymbol{h}(\hat{\boldsymbol{s}}_{n|n-1}) \tag{8.112d}$$

$$= \begin{pmatrix} \left(\frac{\hat{\boldsymbol{s}}_{n|n-1}-\boldsymbol{m}_1}{\|\hat{\boldsymbol{s}}_{n|n-1}-\boldsymbol{m}_1\|} - \frac{\hat{\boldsymbol{s}}_{n|n-1}}{\|\hat{\boldsymbol{s}}_{n|n-1}\|}\right)^{\mathrm{T}} \\ \vdots \\ \left(\frac{\hat{\boldsymbol{s}}_{n|n-1}-\boldsymbol{m}_M}{\|\hat{\boldsymbol{s}}_{n|n-1}-\boldsymbol{m}_M\|} - \frac{\hat{\boldsymbol{s}}_{n|n-1}}{\|\hat{\boldsymbol{s}}_{n|n-1}\|}\right)^{\mathrm{T}} \end{pmatrix}$$

$$\mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\,\mathbf{P}_{n|n-1} \tag{8.112e}$$

Kalman gain:

$$\mathbf{K}_n = \mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}}\left(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^{\mathrm{T}} + \mathbf{R}_n\right)^{-1} \tag{8.112f}$$

with the initialization $\hat{\boldsymbol{s}}_{0|0}$ and its respective covariance $P_{0|0}$.

### Demonstration

Consider the following simulation, which is typical of a video conference scenario. Two speakers, located at two different and fixed locations, speak alternately. The camera should be able to maneuver from one person to the other. For this scenario,

Amazon EX1010 Page 186

simulation is conducted with one speaker located at the polar position ($\phi = \frac{\pi}{4}$ rad $\theta = \frac{\pi}{4}$ rad $R = 1.5$ m) and the other at ($\phi = \frac{3\pi}{4}$ rad $\theta = \frac{\pi}{3}$ rad $R = 1.5$ m). A directional interference is placed at the position ($\phi = \frac{\pi}{2}$ rad $\theta = \frac{\pi}{4}$ rad $R = 1.0$ m). Six microphones were mounted at the following positions (Cartesian coordinates, in meters), relative to the reference microphone (which is at the axes origin):

$$\begin{aligned} \boldsymbol{m}_1^{\mathrm{T}} &= (0.3 \quad 0 \quad 0), \quad \boldsymbol{m}_2^{\mathrm{T}} = (-0.3 \quad 0 \quad 0), \\ \boldsymbol{m}_3^{\mathrm{T}} &= (\ 0 \ 0.3 \quad 0), \quad \boldsymbol{m}_4^{\mathrm{T}} = (\ 0 \quad -0.3 \quad 0), \\ \boldsymbol{m}_5^{\mathrm{T}} &= (\ 0 \quad 0 \ 0.3), \quad \boldsymbol{m}_6^{\mathrm{T}} = (\ 0 \quad 0 \quad -0.3). \end{aligned} \tag{8.113}$$

We used TDOA readings estimated from the noisy microphones data by the RS1 algorithm described in [8.54]. For this estimation stage, room reverberation (set to a reverberation time of $T_{60} = 0.25$ s) and the directional interferer were taken into account in simulation. Room reverberation was simulated by the image method [8.55]. The mean SNR level was set to 10 dB. Figure 8.5 presents the azimuth and elevation angles estimates obtained by the EKF method. The tracking ability of the algorithm is evident, even with abrupt changes of the desired speaker location.

## 8.5 Summary

In this chapter, we have introduced one of the most important contributions to the field of statistical signal processing, namely, the Kalman filter. We provided a rigorous derivation of the filter, based on the orthogonality principle. We also introduced several important variants of the Kalman filter: various Kalman smoothers, the Kalman predictor, a nonlinear extension (the EKF), and the treatment of temporally correlated measurement noise. We showed that the Kalman filter can easily deal with quite common parametric families of processes ( AR, MA, and ARMA).

Finally, we demonstrated the application of the Kalman filter to two important speech processing problems, namely, speech enhancement and speaker localization. Despite the fact that the Kalman filter has already been used for almost two decades in speech enhancement applications, there is still much to do to improve the performance of such Kalman-filter-based algorithms. The main advantage of these algorithms stems from the fact that the Kalman filter may be continuously updated. The speech obtained has a natural sound and the residual noise is clean of annoying artifacts. However, the obtained noise suppression seems limited. This disadvantage, in our opinion, stems from the linear processing regime in which the Kalman filter is applied. Incorporating nonlinearity might yield better noise reduction, while maintaining the low distortion and keeping the low computational load. In localization problems, however, the Kalman filter still finds a leading position, more than 40 years after it was first introduced.



### References


8.1 M.S. Grewal, A.P. Andrews: *Kalman Filtering, Theory and Practice*, Information and System Sciences series (Prentice-Hall, New Jersey 1993)

8.2 R.E. Kalman: A new approach to linear filtering and prediction problems, ASME J. Basic Eng. **82**(Series D), 35–45 (1960)

8.3 R.E. Kalman: New methods and results in linear prediction and filtering theory, Proc. Symposium on Engineering Applications of Random Function Theory and Probability (Wiley, New York 1961)

8.4 S.F. Schmidt: *Computational techniques in Kalman filtering* (NATO Advisory Group for Aerospace Research and Development, London 1970)

8.5 S.F. Schmidt: *Practical Aspects of Kalman filtering implementation* (NATO Advisory Group for Aerospace Research and Development, London 1976)

8.6 P.S. Maybeck: *Stochastic Models, Estimation and Control*, Vol. 2 (Academic, San Diego 1982)

8.7 T. Kailath: *Lectures on Wiener and Kalman Filtering* (Springer, New York 1981)

8.8 S. Haykin (Ed.): *Kalman filtering and neural networks*, Adaptive and Learning Systems for Signal Processing (Wiley, New York 2001)

8.9 H.E. Rauch, F. Tung, C.T. Striebel: Maximum likelihood estimates of linear dynamic systems, AIAA J. **3**, 1445–1450 (1965)

8.10 L.A. McGee, S.F. Schmidt: *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*, National Aeronautics and Space Administration (NASA) Technical Memorandum 86847, Nov. 1985

8.11 N. Wiener: *The Extrapolation, Interpolation and Smoothing of Stationary Time Series* (Wiley, New York 1949)

8.12 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B **39**(1), 1–38 (1977)


Amazon EX1010 Page 187

8.13 K.K. Paliwal, A. Basu: A speech enhancement method based on Kalman filtering, Proc. IEEE ICASSP (IEEE, Dallas 1987) pp. 177–180

8.14 B. Koo, J.D. Gibson, S.D. Gray: Filtering of colored noise for speech enhancement and coding, Proc. IEEE ICASSP (IEEE, Glasgow 1989) pp. 349–352

8.15 J.D. Gibson, B. Koo, S.D. Gray: Filtering of colored noise for speech enhancement and coding, IEEE Trans. Acoust. Speech **39**(6), 1732–1742 (1991)

8.16 S. Haykin: *Adaptive Filter Theory*, Information and System Sciences, 4th edn. (Prentice-Hall, Upper Saddle River 2002)

8.17 M. Feder, A.V. Oppenheim, E. Weinstein: Methods for noise cancellation based on the EM algorithm, Proc. IEEE ICASSP (IEEE, Dallas 1987) pp. 201–204

8.18 B. Widrow, J.R. Glover Jr., J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeider, E. Dong Jr., R.C. Goodlin: Adaptive noise cancelling: principals and applications, Proc. IEEE **63**(12), 1692–1716 (1975)

8.19 M. Feder, A.V. Oppenheim, E. Weinstein: Maximum likelihood noise cancellation using the EM algorithm, IEEE Trans. Acoust. Speech **37**(2), 204–216 (1989)

8.20 E. Weinstein, A.V. Oppenheim, M. Feder: *Signal Enhancement Using Single and Multi-Sensor Measurements* (MIT, Cambridge 1990)

8.21 J.S. Lim, A.V. Oppenheim: All-pole modeling of degraded speech, IEEE Trans. Acoust. Speech **26**(3), 197–210 (1978)

8.22 E. Weinstein, A.V. Oppenheim, M. Feder, J.R. Buck: Iterative and sequential algorithms for multisensor signal enhancement, IEEE Trans. Signal Process. **42**(4), 846–859 (1994)

8.23 M. Feder, E. Weinstein, A.V. Oppenheim: A new class of sequential and adaptive algorithms with application to noise cancellation, Proc. IEEE ICASSP (IEEE, New York 1988) pp. 557–560

8.24 A.V. Oppenheim, E. Weinstein, K.C. Zangi, M. Feder, D. Gauger: Single-sensor active noise cancellation, IEEE Trans. Speech Audio Process. **2**(2), 285–290 (1994)

8.25 B.-G. Lee, K.Y. Lee, S. Ann: An EM-based approach for parameter enhancement with an application to speech signals, Signal Process. **46**, 1–14 (1995)

8.26 K.Y. Lee, B.-G. Lee, S. Ann: Adaptive filtering for speech enhancement in colored noise, IEEE Signal Process. Lett. **4**(10), 277–279 (1997)

8.27 Z. Goh, K.-C. Tan, B.T.G. Tan: Kalman-Filtering Speech Enhancement Method Based on a Voiced-Unvoiced Speech Model, IEEE Trans. Speech Audio Process. **7**(5), 510–524 (1999)

8.28 M. Gabrea, E. Grivel, M. Najim: A single microphone Kalman filter-based noise canceller, IEEE Signal Process. Lett. **6**(3), 55–57 (1999)

8.29 B.D.O. Anderson, J.B. Moore: *Optimal Filtering*, Information and System Sciences Series (Prentice-Hall, Englewood Cliffs 1979)

8.30 S. Gannot, D. Burshtein, E. Weinstein: Iterative and sequential Kalman filter-based speech enhancement algorithms, IEEE Trans. Speech Audio Process. **6**(4), 373–385 (1998)

8.31 M. Fujimoto, Y. Ariki: Noisy speech recognition using noise reduction method based nn Kalman filter, Proc. IEEE ICASSP (IEEE, Istanbul 2000) pp. 1727–1730

8.32 S.F. Boll: Suppression of acoustic noise in speech using spectral subtraction, IEEE Trans. Acoust. Speech **27**(2), 113–120 (1979)

8.33 K.Y. Lee, B.-G. Lee, I. Song, S. Ann: Robust estimation of ar parameters and its application for speech enhancement, Proc. IEEE ICASSP (IEEE, San Francisco 1992) pp. 309–312

8.34 N. Ma, M. Bouchard, R.A. Goubran: Perceptual Kalman filtering for speech enhancement in colored noise, Proc. IEEE ICASSP, Vol. 1 (IEEE, Montreal 2004) pp. 717–720

8.35 X. Shen, L. Deng: A dynamic system approach to speech enhancement using the $H_\infty$ filtering algorithm, IEEE Trans. Speech Audio Process. **27**(4), 391–399 (1999)

8.36 E.A. Wan, A.T. Nelson: Removal of noise from speech using the dual Ekf algorithm, Proc. IEEE ICASSP (IEEE, Seattle, Washington 1998)

8.37 S.J. Julier, J.K. Uhlmann: Unscented filtering and nonlinear estimation, Proc. IEEE **92**(3), 401–422 (2004)

8.38 E.A. Wan, R. van der Merwe: The unscented Kalman filter for nonlinear estimation, Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC) (IEEE, Lake Louise 2000)

8.39 S. Gannot, M. Moonen: On the application of the unscented Kalman filter to sppech processing, The International Workshop on Acoustic Echo and Noise Control (IWAENC) (Kyoto 2003) pp. 27–30

8.40 W. Fong, S. Godsill: Monte Carlo smoothing with application to audio signal enhancement, IEEE SSP Workshop (IEEE, Singapore 2001) pp. 18–210

8.41 T.W. Parsons: *Voice and Speech Processing* (McGraw-Hill, USA 1987)

8.42 Y. Ephraim, D. Malah, B.H. Juang: On the application of hidden Markov models for enhancing noisy speech, IEEE Trans. Acoust. Speech **37**, 1846–1856 (1989)

8.43 Y. Ephraim: A bayesian estimation approach for speech enhancement using hidden markov models, IEEE Trans. Signal Process. **40**, 725–735 (1992)

8.44 Y. Ephraim: Speech enhancement using state dependent dynamical system model, Proc. IEEE ICASSP (IEEE, San Francisco 1992) pp. 289–292

8.45 K.Y. Lee, K. Shirai: Efficient recursive estimation for speech enhancement in colored noise, IEEE Signal Process. Lett. **3**, 196–199 (1996)

8.46 K.Y. Lee, S. Jung: Time-domain approach using multiple Kalman filters and em algorithm to speech




Amazon EX1010 Page 188

enhancement with nonstationary noise, IEEE Trans. Speech Audio Process. **8**(3), 373–385 (2000)

8.47 J.B. Kim, K.Y. Lee, C.W. Lee: On the applications of the interacting multiple model algorithm for enhancing noisy speech, IEEE Trans. Speech Audio Process. **8**(3), 349–352 (2000)

8.48 K.Y. Lee, S. McLaughlin, K. Shirai: Speech enhancement based on extended Kalman filter and neural predictive hidden Markov model, Proc. Int. Workshop Neural Networks for Signal Processing (IEEE, Kyoto 1996) pp. 302–310

8.49 S. Gannot: Speech Enhancement: Appliction of the Kalman filter in the estimate-maximize (EM) framework. In: *Speech Enhancement*, Signals and Communication Technology, ed. by S. Makino, J. Benesty, J. Chen (Springer, Berlin 2005), pp. 161–198

8.50 D. Burshtein: Joint modeling and maximum-likelihood estimation of pitch and linear prediction coefficient parameters, J. Acoust. Soc. Am. **3**, 1531–1537 (1992)

8.51 J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, V. Zue: Acoustic-Phonetic Continuous Speech Corpus (TIMIT), CD-ROM (Oct. 1991)

8.52 A. Varga, H.J.M. Steeneken: Assessment for automatic speech recognition: II. NOISEX-92: a database and an experiment to study the effect of additive noise on speech recognition systems, Speech Commun. **12**, 247–251 (1993)

8.53 C.H. Knapp, G.C. Carter: The generalized correlation method for estimation of time delay, IEEE Trans. Acoust. Speech **24**(4), 320–327 (1976)

8.54 T.G. Dvorkind, S. Gannot: Time difference of arrival estimation of speech source in a noisy and reverberant environment, Signal Process. **85**(1), 177–204 (2005)

8.55 J.B. Allen, D.A. Berkley: Image method for efficiently simulating small-room acoustics, J. Acoust. Soc. Am. **65**(4), 943–950 (1979)



Amazon EX1010 Page 189

# 9. Homomorphic Systems and Cepstrum Analysis of Speech

R. W. Schafer

In 1963, *Bogert*, *Healy*, and *Tukey* published a chapter with one of the most unusual titles to be found in the literature of science and engineering [9.1]. In this chapter, they observed that the logarithm of the power spectrum of a signal plus its echo (delayed and scaled replica) consists of the logarithm of the signal spectrum plus a periodic component due to the echo. They suggested that further spectrum analysis of the log spectrum could highlight the periodic component in the log spectrum and thus lead to a new indicator of the occurrence of an echo. Specifically they made the following observation:

> *In general, we find ourselves operating on the frequency side in ways customary on the time side and vice versa.*

As an aid in formalizing this new point of view, they introduced a number of paraphrased words. For example, they defined the *cepstrum* of a signal as the power spectrum of the logarithm of the power spectrum of a signal. (In fact, they used discrete-time spectrum estimates based on the discrete Fourier transform.) Similarly, the term *quefrency* was introduced for the independent variable of the cepstrum [9.1].

In this chapter we will explore why the cepstrum has emerged as a central concept in digital speech processing. We will start with definitions appropriate for discrete-time signal processing and develop some of the general properties and computational approaches for the cepstrum of speech. Using this basis, we will explore the many ways that the cepstrum has been used in speech processing applications.


9.1 **Definitions** ..... 161
9.1.1 Definition of the Cepstrum ..... 162
9.1.2 Homomorphic Systems ..... 162
9.1.3 Numerical Computation of Cepstra ..... 163

9.2 ***Z*-Transform Analysis** ..... 164

9.3 **Discrete-Time Model for Speech Production** ..... 165

9.4 **The Cepstrum of Speech** ..... 166
9.4.1 Short-Time Cepstrum of Speech ..... 166
9.4.2 Homomorphic Filtering of Speech ..... 168

9.5 **Relation to LPC** ..... 169
9.5.1 LPC Versus Cepstrum Smoothing ..... 169
9.5.2 Cepstrum from LPC Model ..... 170
9.5.3 Minimum Phase and Recursive Computation ..... 170

9.6 **Application to Pitch Detection** ..... 171

9.7 **Applications to Analysis/Synthesis Coding** 172
9.7.1 Homomorphic Vocoder ..... 173
9.7.2 Homomorphic Formant Vocoder ..... 174
9.7.3 Analysis-by-Synthesis Vocoder ..... 175

9.8 **Applications to Speech Pattern Recognition** ..... 176
9.8.1 Compensation for Linear Filtering ..... 177
9.8.2 Weighted Distance Measures ..... 177
9.8.3 Group Delay Spectrum ..... 178
9.8.4 Mel-Frequency Cepstrum Coefficients (MFCC) ..... 179

9.9 **Summary** ..... 180

**References** ..... 180




## 9.1 Definitions

As stated above, *Bogert* et al. based their definition of the cepstrum on a rather loose interpretation of the power spectrum of an analog signal. Since effective application of the cepstrum concept required digital processing, it was necessary, early on, to develop a solid definition in terms of discrete-time signal theory [9.2, 3].

Amazon EX1010 Page 190

### 9.1.1 Definition of the Cepstrum

For discrete-time signals, a definition in the spirit of [9.1] is that the *cepstrum* of a signal is the inverse discrete-time Fourier transform (IDTFT) of the logarithm of the magnitude of the DTFT of the signal. That is, the cepstrum of a signal $x[n]$ is defined as

$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(\mathrm{e}^{\mathrm{i}\omega})|\,\mathrm{e}^{\mathrm{i}\omega n}\,\mathrm{d}\omega\,, \tag{9.1a}$$

where the DTFT of the signal is defined as [9.4],

$$X(\mathrm{e}^{\mathrm{i}\omega}) = \sum_{n=-\infty}^{\infty} x[n]\,\mathrm{e}^{-\mathrm{i}\omega n}\,. \tag{9.1b}$$

Equation (9.1a) is the definition of the cepstrum that we shall use throughout this chapter. Note that $c[n]$, being an IDTFT, is nominally a function of a discrete index $n$. If the input sequence is obtained by sampling an analog signal, i. e., $x[n] = x_\mathrm{a}(n/f_\mathrm{s})$, then it would be natural to associate *time* with the index $n$ in the cepstrum. However, elaborating on the cepstrum conceit, *Bogert* et al. introduced the term *quefrency* for the name of the independent variable of the cepstrum [9.1]. This new term is often useful in describing the fundamental properties of the cepstrum. For example, low quefrencies correspond to slowly varying components in the log magnitude spectrum, while high quefrencies correspond to rapidly vary components of the log magnitude. Isolated peaks at multiples of a quefrency $P_0$ in the cepstrum correspond to a periodic component in the log magnitude with period $2\pi/P_0$ in normalized radian frequency $\omega$ or $f_\mathrm{s}/P_0$ in cyclic analog frequency.

### 9.1.2 Homomorphic Systems

Contemporaneously with the introduction of the cepstrum concept, *Oppenheim* [9.5] developed a new theory of nonlinear systems that was based on the mathematical theory of linear vector spaces. The essence of this theory was that certain operations of signal combination (convolution and multiplication in particular) satisfy the same postulates as does vector addition in the theory of linear vector spaces. From this observation, Oppenheim showed that classes of nonlinear systems could be defined on the basis of a generalized principle of superposition. He termed such systems *homomorphic*. Of particular importance for our present discussion is the class of homomorphic systems for which the input and output are combined by convolution. Such systems are represented by the canonic form shown in Fig. 9.1, where the operator $D_*\{\ \}$ is called the *characteristic system for convolution* and $D_*^{-1}\{\ \}$ is its inverse. This characteristic system is defined by the property that when $x[n] = x_1[n] * x_2[n]$, the corresponding output is

$$\begin{aligned}\hat{x}[n] &= D_*\{x_1[n] * x_2[n]\}\\ &= D_*\{x_1[n]\} + D_*\{x_2[n]\}\\ &= \hat{x}_1[n] + \hat{x}_2[n]\,.\end{aligned} \tag{9.2}$$

That is, the characteristic system transforms a combination by convolution into a corresponding combination by addition.

The middle system in Fig. 9.1 is the system denoted $L\{\ \}$, which is an ordinary linear system satisfying the principle of superposition with addition as both the input and output operation for signal combination [9.4]. Note that Fig. 9.1 shows the input and output operations at the top and just outside of each block.

Finally, the inverse characteristic system must transform a sum into a convolution so that the overall system transformation satisfies $H\{x_1[n] * x_2[n]\} = H\{x_1[n]\} * H\{x_2[n]\} = y_1[n] * y_2[n]$.

From Fig. 9.1, all homomorphic systems for convolution differ only in the (ordinary) linear part $L\{\ \}$. The key to the definition is the characteristic system $D_*\{\ \}$, which turns convolution into addition, and as we will see, it is $D_*\{\ \}$ that makes the connection between the cepstrum and the theory of homomorphic systems for convolution. Indeed, Fig. 9.2 shows a sequence of mathematical operators that has the desired property in (9.2). Specifically, the output $\hat{x}[n]$ can be represented by the discrete-time Fourier transform equations

$$X(\mathrm{e}^{\mathrm{i}\omega}) = \sum_{n=-\infty}^{\infty} x[n]\,\mathrm{e}^{-\mathrm{i}\omega n} \tag{9.3a}$$

$$\hat{X}(\mathrm{e}^{\mathrm{i}\omega}) = \log[X(\mathrm{e}^{\mathrm{i}\omega})] \tag{9.3b}$$

$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(\mathrm{e}^{\mathrm{i}\omega})\,\mathrm{e}^{\mathrm{i}\omega n}\,\mathrm{d}\omega\,. \tag{9.3c}$$

Equation (9.3a) and (9.3c) are the DTFT and IDTFT, respectively, while (9.3b) is the *complex* logarithm of the

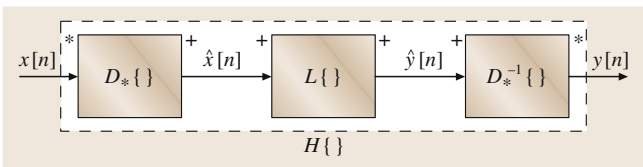



**Fig. 9.1** Canonic form for homomorphic systems for convolution



Amazon EX1010 Page 191

complex function $X(e^{i\omega})$. It is easily seen that the cascade of these three operations has the desired property of turning convolution into addition. First, the DTFT of a convolution of two sequences is the product of their DTFTs [9.4]. Appropriately defined, the complex logarithm of a product of two functions is the sum of the complex logarithms of the individual functions. Finally, the IDTFT is a linear operator in the conventional sense, so the IDTFT of a sum is the sum of the corresponding inverse transforms [9.4]. (Interestingly, the DTFT is also linear in the conventional sense, but it is also a homomorphic operator with input operation convolution and output operation multiplication. Likewise, the IDTFT operator is both conventionally linear and homomorphic with input operation multiplication and output operation convolution.) Hence, if the complex logarithm in Fig. 9.2 is appropriately defined and computed, then if the input is $x[n] = x_1[n] * x_2[n]$, it follows that the output in in Fig. 9.2 is $\hat{x}[n] = \hat{x}_1[n] + \hat{x}_2[n]$.

The inverse of the characteristic system for convolution is depicted in Fig. 9.3. It is obtained by simply using the complex exponential to invert the effect of the complex logarithm.

The representation of Figs. 9.2 and 9.3 was developed by *Oppenheim*, *Schafer*, and *Stockham*, and first published in [9.2, 3]. Because of the obvious close relationship of the characteristic system for convolution as defined by (9.3a), (9.3b), and (9.3c) to the definition of the cepstrum in (9.1a) and (9.1b), they called the output of the characteristic system for convolution, the *complex cepstrum*. This is not because the complex cepstrum is complex; indeed, if $x[n]$ is real, then $\hat{x}[n]$ will also be real. Rather, the modifier *complex* is used to imply that the *complex logarithm* is used in the computation of the complex cepstrum. The complex logarithm in (9.3b) is defined as

$$\begin{aligned}\hat{X}(e^{i\omega}) &= \log\{X(e^{i\omega})\} \\ &= \log|X(e^{i\omega})| + i\arg\{X(e^{i\omega})\}\,, \end{aligned} \tag{9.4}$$

where if $X(e^{i\omega}) = X_1(e^{i\omega}) \cdot X_2(e^{i\omega})$, as for the convolution $x[n] = x_1[n] * x_2[n]$, then the following must hold:

$$\log|X(e^{i\omega})| = \log|X_1(e^{i\omega})| + \log|X_2(e^{i\omega})| \tag{9.5a}$$

$$\arg\left\{X(e^{i\omega})\right\} = \arg\{X_1(e^{i\omega})\} + \arg\left\{X_2(e^{i\omega})\right\}\,. \tag{9.5b}$$

Satisfying (9.5a) presents no serious difficulty, but (9.5b) is more problematic as we discuss below.

The relationship between the cepstrum and complex cepstrum can be obtained by noting that, if $x[n]$ is real,

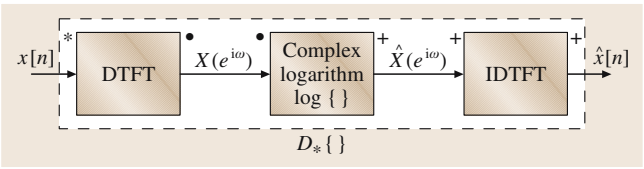



**Fig. 9.2** DTFT representation of the characteristic system for convolution

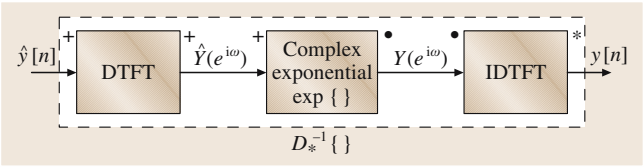



**Fig. 9.3** DTFT representation of the inverse characteristic system for convolution

then $\log|X(e^{i\omega})|$ is a real and even function of $\omega$, while $\arg\{X(e^{i\omega})$, the imaginary part of $\hat{X}(e^{i\omega})$, is a real and odd function of $\omega$. This implies that [9.4],

$$c[n] = \mathrm{Ev}\{\hat{x}[n]\} = \frac{\hat{x}[n] + \hat{x}[-n]}{2}\,. \tag{9.6}$$

That is, as we have defined them, the cepstrum is the *even part* of the complex cepstrum.

### 9.1.3 Numerical Computation of Cepstra

So far, we have merely defined the cepstrum and complex cepstrum in terms of mathematical operators. To be useful for speech processing, we must replace these operators by computable operations. This can be done by noting that the discrete Fourier transform (DFT) [computed with a fast Fourier transform (FFT) algorithm] is a sampled (in frequency) version of the DTFT of a finite-length sequence, i. e., $X[k] = X(e^{i2\pi k/N})$ [9.4]. Figure 9.4 depicts the operations

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-i(2\pi k/N)n} \tag{9.7a}$$

$$\hat{X}[k] = \log|X[k]| + i\arg\{X[k]\} \tag{9.7b}$$

$$\tilde{\hat{x}}[n] = \frac{1}{N}\sum_{n=0}^{N-1} \hat{X}[k]\, e^{i(2\pi k/N)n}\,, \tag{9.7c}$$

where the DTFT has been replaced by the finite DFT computation. Note the *tilde* ($\tilde{}$) symbol above $c[n]$ and $\hat{x}[n]$ in Fig. 9.4 and in (9.7c), which is used to emphasize that the cepstra computed using the DFT suffer from time-domain aliasing due to the sampling of the log

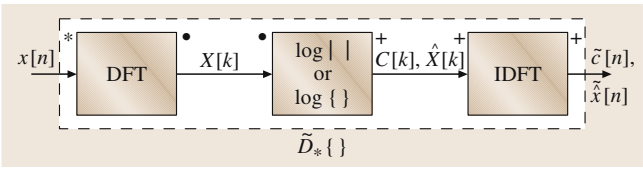



**Fig. 9.4** Computing the cepstrum or complex cepstrum using the DFT

of the DTFT [9.4]. Specifically, the complex cepstrum computed by the DFT is related to the complex cepstrum defined by the DTFT by the equation

$$\tilde{\hat{x}}[n] = \sum_{r=-\infty}^{\infty} \hat{x}[n+rN] \,. \tag{9.8}$$

An identical equation holds for the time-aliased cepstrum $\tilde{c}[n]$.

The effect of time-domain aliasing can be made negligible by using a large value for $N$. A more-serious problem in computation of the complex cepstrum is the computation of the complex logarithm. As discussed above, we must compute samples of $\arg\{X(e^{i\omega})\} = \arg\{X_1(e^{i\omega}) \cdot X_2(e^{i\omega})\}$ such that

$$\arg\{X[k]\} = \arg\{X_1[k]\} + \arg\{X_2[k]\} \,, \tag{9.9}$$

and this requires special care. Standard arctan functions return the principal value of the phase angle, which we denote, $-\pi < \mathrm{ARG}\{X[k]\} \leq \pi$, and in general it must be assumed that

$$\mathrm{ARG}\{X[k]\} \neq \mathrm{ARG}\{X_1[k]\} + \mathrm{ARG}\{X_2[k]\} \,, \tag{9.10}$$

when $X[k] = X_1[k]X_2[k]$. It is not that the principal value gives the wrong phase for the product of two DFTs, but that it is not equal to the sum of the two individual principal value phases. The problem is the discontinuities that result from the modulo $2\pi$ computation. If the principal value phase can be *unwrapped* by finding a sequence $R[k]$ such that

$$\arg\{X[k]\} = \mathrm{ARG}\{X[k]\} + 2\pi R[k] \,, \tag{9.11}$$

then the resulting phase would have the desired property of (9.9). The sequence $R[k]$ will have integer values that are constant over subintervals of $0 \leq k < N/2$. Abrupt changes will occur at the *discontinuities* of $\mathrm{ARG}\{X[k]\}$. The earliest phase unwrapping algorithm for cepstrum computation took the straightforward approach of searching for discontinuities of size $2\pi$ in $\mathrm{ARG}\{X[k]\}$ [9.2,3]. This approach requires a very small frequency sample spacing ($N$ large) so as not to miss discontinuities that occur when the phase has a steep slope. *Tribolet* gave an improved algorithm that uses both the principal value phase and the phase derivative (which can be computed directly from $X[k]$) [9.6].

While phase unwrapping can be problematic in some situations involving the complex cepstrum, it is not a problem in computing the cepstrum, where it is not used, and as we will see in the next section, phase unwrapping can be avoided by using the $z$-transform.

## 9.2 Z-Transform Analysis

The characteristic system for convolution can also be represented by the two-sided $z$-transform, as depicted in Fig. 9.5. This representation is very useful for theoretical investigations, and recent developments in polynomial root finding have made the $z$-transform representation a viable computational basis as well. For this purpose, we assume that the input signal $x[n]$ has a rational $z$-transform of the form

$$X(z) = X_{\max}(z) \cdot X_{\mathrm{uc}}(z) \cdot X_{\min}(z) \,, \tag{9.12}$$

where

$$\begin{aligned} X_{\max}(z) &= z^{M_o} \prod_{k=1}^{M_o} \left(1 - a_k z^{-1}\right) \\ &= \prod_{k=1}^{M_o} (-a_k) \prod_{k=1}^{M_o} \left(1 - a_k^{-1} z\right) \,, \end{aligned} \tag{9.13a}$$

$$X_{\mathrm{uc}}(z) = \prod_{k=1}^{M_{\mathrm{uc}}} \left(1 - e^{i\theta_k} z^{-1}\right) \,, \tag{9.13b}$$

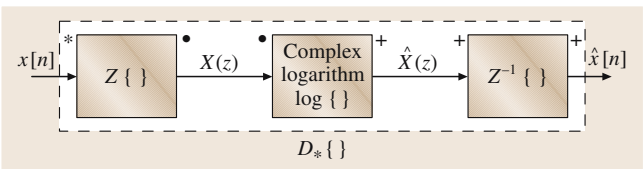



**Fig. 9.5** $z$-transform representation of the characteristic system for convolution



Amazon EX1010 Page 193

$$X_{\min}(z) = A \frac{\prod_{k=1}^{M_i} \left(1 - b_k z^{-1}\right)}{\prod_{k=1}^{N_i} \left(1 - c_k z^{-1}\right)} . \tag{9.13c}$$

The zeros of $X_{\max}(z)$, i. e., $z_k = a_k$, are zeros of $X(z)$ outside of the unit circle ($|a_k| > 1$). The factor $X_{\max}(z)$ is thus the *maximum-phase* part of $X(z)$. The factor $X_{\text{uc}}(z)$ contains all the zeros (with angles $\theta_k$) *on* the unit circle. The *minimum-phase* part is $X_{\min}(z)$, where $b_k$ and $c_k$ are zeros and poles, respectively, that are inside the unit circle ($|b_k| < 1$ and $|c_k| < 1$). The factor $z^{M_o}$ implies a shift of $M_o$ samples to the left. It is included to simplify the results in (9.16).

The complex cepstrum of $x[n]$ is determined by assuming that the complex logarithm $\log[X(z)]$ results in the sum of logarithms of each of the product terms, i. e.,

$$\begin{aligned}\hat{X}(z) = {} & \log \left| \prod_{k=1}^{M_o} (-a_k) \right| + \sum_{k=1}^{M_o} \log \left(1 - a_k^{-1} z\right) \\ & + \sum_{k=1}^{M_{\text{uc}}} \log(1 - \mathrm{e}^{\mathrm{i}\theta_k} z^{-1}) \\ & + \log |A| + \sum_{k=1}^{M_i} \log \left(1 - b_k z^{-1}\right) \\ & - \sum_{k=1}^{N_i} \log \left(1 - c_k z^{-1}\right) . \end{aligned} \tag{9.14}$$

Applying the power series expansion

$$\log(1 - a) = -\sum_{n=1}^{\infty} \frac{a^n}{n} , \quad |a| < 1 \tag{9.15}$$

to each of the terms in (9.14) and collecting the coefficients of the positive and negative powers of $z$ gives

$$\hat{x}[n] = \begin{cases} \displaystyle\sum_{k=1}^{M_o} \frac{a_k^n}{n} & n < 0 \\ \displaystyle \log |A| + \log \left| \prod_{k=1}^{m_o} (-a_k) \right| & n = 0 \\ \displaystyle \left( - \sum_{k=1}^{M_{\text{uc}}} \frac{\mathrm{e}^{\mathrm{i}\theta_k n}}{n} \right. & \\ \displaystyle \left. - \sum_{k=1}^{M_i} \frac{b_k^n}{n} + \sum_{k=1}^{N_i} \frac{c_k^n}{n} \right) & n > 0 \end{cases} . \tag{9.16}$$

Given all the poles and zeros of a $z$-transform $X(z)$, (9.16) allows us to compute a finite set of samples of the complex cepstrum with no approximation. This is clearly the case in theoretical analysis where the poles and zeros are specified, but (9.16) is also useful as the basis for computation. All that is needed is a process for obtaining the $z$-transform as a rational function and a process for finding the zeros of the numerator and denominator. This has become more feasible with increasing computational power and with new algorithmic advances in finding roots of large polynomials [9.7].

One method of obtaining a $z$-transform is simply to select a finite-length sequence of samples of a signal. The $z$-transform is then simply a polynomial with the samples $x[n]$ as coefficients, i. e.,

$$\begin{aligned} X(z) &= \sum_{n=0}^{M} x[n] z^{-n} \\ &= A \prod_{k=1}^{M_o} \left(1 - a_k z^{-1}\right) \prod_{k=1}^{M_i} \left(1 - b_k z^{-1}\right) . \end{aligned} \tag{9.17}$$

A second method that yields a $z$-transform is the method of linear predictive analysis, which we will discuss briefly in Sect. 9.5.

## 9.3 Discrete-Time Model for Speech Production

A short segment of a sampled speech waveform is shown in Fig. 9.6. Note that about the first 90 ms (720 samples at 8000 samples/s) look like random noise, while the remaining samples in view appear to be almost periodic. Such a speech signal is the output of a physical system that produces an acoustic wave whose properties vary in time to encode a message. To create a speech signal, humans perform a coordinated sequence of physical gestures involving the lungs, vocal cords, tongue, and lips. Assuming that we have a sampled speech signal, it is helpful to have a discrete-time system model upon which to base analysis [9.8, 9]. A simple model is depicted in Fig. 9.7. The block on the right labeled 'time-varying digital filter' represents a linear system with a slowly time-varying frequency response (or impulse response). Its purpose is to model the frequency



Amazon EX1010 Page 194

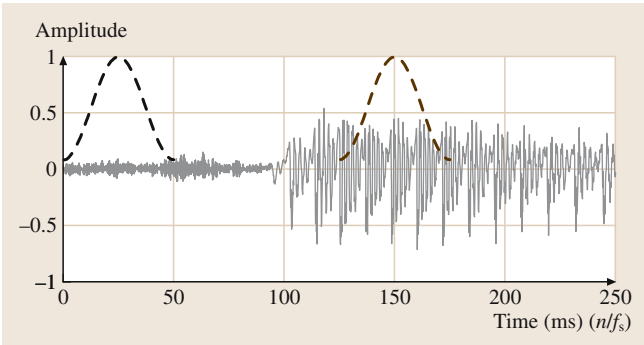



**Fig. 9.6** Segment of sampled speech waveform (samples connected by straight lines for plotting)

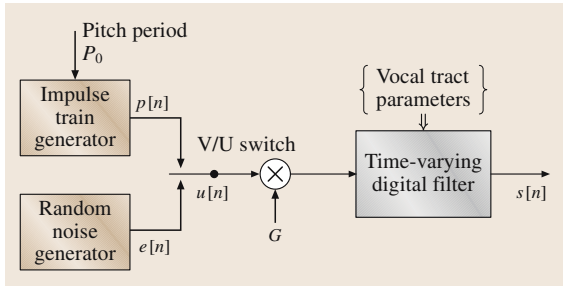



**Fig. 9.7** Simplified discrete-time model for the speech signal

resonances (formants) of the human vocal tract. It is common to assume that, over short time intervals, the system is not time-varying, and that it is characterized by an all-pole rational system function of the form

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} = \frac{G}{\prod_{k=1}^{p} \left(1 - c_k z^{-1}\right)} , \quad (9.18)$$

where $|c_k| < 1$ for stability. The blocks labeled 'impulse train generator' and 'random noise generator' represent the glottal pulse excitation and fricative (constriction) excitation of the vocal tract, respectively. The multiplicative factor $G$ is lumped with $H(z)$ in (9.18), but it could be considered part of the excitation since it simply applies a gain to the excitation. By properly choosing the parameters of the excitations and the linear system, it is possible to create discrete-time signals that are perceived to be very close, if not identical, to a given sampled human-produced speech signal. Therefore, the model of Fig. 9.7 is the basis for most analysis/synthesis speech coders and automatic speech recognition systems, and much work has been done on estimating the time-varying parameters of this model from sampled speech signals. Homomorphic filtering and cepstrum analysis are important techniques for such analysis.

## 9.4 The Cepstrum of Speech

A fundamental tenet of digital speech processing is that the properties of the speech signal change slowly relative to the sampling rate of the signal, and furthermore, that these properties can be *sampled* by a process of *short-time analysis* in which speech properties such as pitch and vocal tract response are assumed to hold constant over a short time interval (called a window or frame) of duration on the order of 20–30 ms. This is illustrated in Fig. 9.6 by the 401-sample signal analysis windows placed at sample index 0 and 1000 (0 and 125 ms). Typically for the 8000 samples/s sampling rate of the speech in Fig. 9.6, the analysis window would be moved in steps of 80–160 samples (10–20 ms).

### 9.4.1 Short-Time Cepstrum of Speech

Following [9.9], we assume that the model of Fig. 9.7 represents the sampled speech signal $s[n]$, and that over the analysis interval the output of the model equals the speech signal. Furthermore, over the length of the window $L$, we assume that

$$s[n] = u[n] * h[n] , \quad 0 \le n \le L-1 , \quad (9.19)$$

where $h[n]$ is the impulse response corresponding to the system function in (9.18). The impulse response $h[n]$ models the combined effects of the gain $G$, the vocal-tract frequency response, the glottal wave shape, and radiation of sound at the lips, but we will refer to it simply as the *vocal-tract impulse response*. Also, we assume that the impulse response $h[n]$ is short compared to the window so that the windowed segment can be represented as

$$\begin{aligned} x[n] &= w[n](u[n] * h[n]) \\ &\approx u_w[n] * h[n] , \quad 0 \le n \le L-1 , \end{aligned} \quad (9.20)$$



Amazon EX1010 Page 195

where $u_w[n] = w[n]u[n]$, i. e., any tapering due to the analysis window is incorporated into the excitation. In the case of voiced speech, $u[n]$ is an impulse train of the form

$$u[n] = p[n] = \sum_{k=0}^{N_p-1} \delta[n - kP_0] \,, \tag{9.21}$$

where $\delta[n]$ represents the unit sample sequence and $P_0$ is the discrete-time *pitch period*. In the case of unvoiced speech, the excitation would be a white noise sequence, which is spectrally shaped by the linear filter.

For voiced speech, the windowed excitation is

$$u_w[n] = w[n]p[n] = \sum_{k=0}^{N_p-1} w_{P_0}[k]\delta[n - kP_0] \,, \tag{9.22}$$

where $w_{P_0}[k]$ is the *time-sampled* window sequence defined as

$$w_{P_0}[k] = \begin{cases} w[kP_0] & k = 0, 1, \ldots, N_p - 1 \\ 0 & \text{otherwise} \end{cases} . \tag{9.23}$$

From (9.22), the DTFT of $u_w[n]$ is

$$U_w(\mathrm{e}^{\mathrm{i}\omega}) = \sum_{k=0}^{N_p-1} w_{P_0}[k]\mathrm{e}^{-\mathrm{i}\omega kP_0} = W_{P_0}(\mathrm{e}^{\mathrm{i}\omega P_0}) \,, \tag{9.24}$$

and from (9.24) it follows that $U_w(\mathrm{e}^{\mathrm{i}\omega})$ is periodic in $\omega$ with period $2\pi/P_0$. Therefore,

$$\hat{X}(\mathrm{e}^{\mathrm{i}\omega}) = \log\{H(\mathrm{e}^{\mathrm{i}\omega})\} + \log\left\{U_w(\mathrm{e}^{\mathrm{i}\omega})\right\} \tag{9.25}$$

has two components: (i) $\log\{H(\mathrm{e}^{\mathrm{i}\omega})\}$, due to the vocal-tract frequency response and slowly varying in $\omega$, and (ii) $\log\{W_{P_0}(\mathrm{e}^{\mathrm{i}\omega P_0})\}$, due to the excitation and periodic with period $2\pi/P_0$. (For signals sampled with sampling rate $f_s$, this period corresponds to $f_s/P_0$ Hz in cyclic analog frequency.) The complex cepstrum of the windowed speech segment $x[n]$ is therefore

$$\hat{x}[n] = \hat{h}[n] + \hat{u}_w[n] \,. \tag{9.26}$$

For voiced speech, the cepstral component due to the excitation has the form

$$\hat{u}_w[n] = \begin{cases} \hat{w}_{P_0}[n/P_0] & n = 0, \pm P_0, \pm 2P_0, \ldots \\ 0 & \text{otherwise} \end{cases} . \tag{9.27}$$

(Note that $\hat{w}_{P_0}[n]$ corresponds to $\log\{W_{P_0}(\mathrm{e}^{\mathrm{i}\omega})\}$, so through the upsampling theorem [9.4], $\log\{W_{P_0}(\mathrm{e}^{\mathrm{i}\omega P_0})\}$ corresponds to (9.27).) That is, in keeping with the $2\pi/P_0$ periodicity of $\log\{U_w(\mathrm{e}^{\mathrm{i}\omega})\} = \log\{W_{P_0}(\mathrm{e}^{\mathrm{i}\omega P_0})\}$, the corresponding complex cepstrum (or cepstrum) has impulses (isolated samples) at quefrencies that are multiples of $P_0$. For unvoiced speech, no such periodicity occurs in the logarithm, and therefore no cepstral peaks occur.

To illustrate the cepstrum analysis of speech, a 401-sample segment of voiced speech was selected by a Hamming window positioned at sample 1000 in Fig. 9.6. The resulting windowed $x[n]$ is shown in Fig. 9.8a. The log magnitude and unwrapped phase for $X(\mathrm{e}^{\mathrm{i}\omega})$ are shown as the rapidly oscillating curves in Figs. 9.9a and 9.9b, respectively. The resulting complex cepstrum is shown in Fig. 9.10a and the cepstrum is shown in Fig. 9.10b, i. e., Fig. 9.10a is the IDTFT of $\log\{X(\mathrm{e}^{\mathrm{i}\omega})\}$ with Fig. 9.9a as the real part and Fig. 9.9b as the imaginary part, while Fig. 9.10b is the IDTFT of $\log|X(\mathrm{e}^{\mathrm{i}\omega})|$ with zero imaginary part. [From (9.6), Figure 9.10b is also the even part of Fig. 9.10a.]

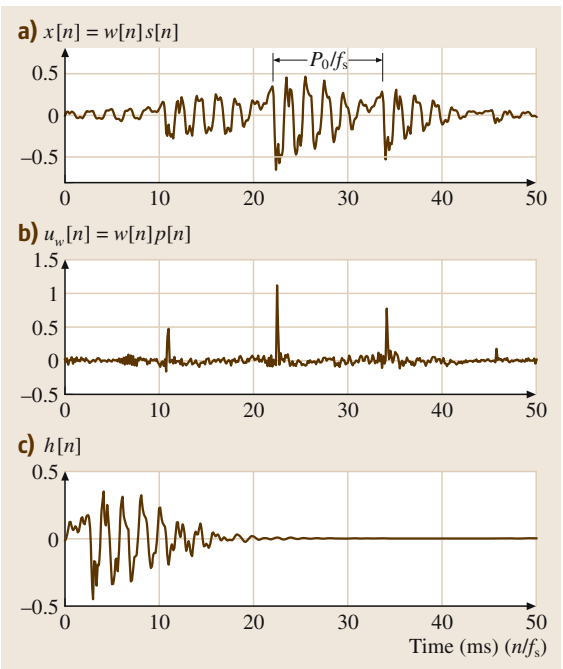



**Fig. 9.8a–c** Time waveforms: **(a)** windowed speech segment, **(b)** output corresponding to high-quefrency components of the complex cepstrum, and **(c)** output corresponding to low-quefrency components of the complex cepstrum



Amazon EX1010 Page 196

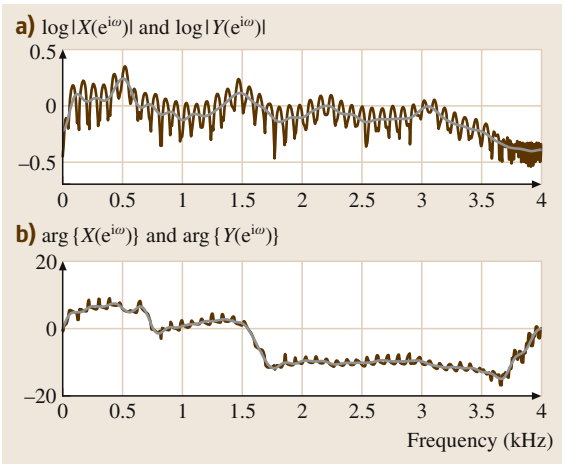



**Fig. 9.9** **(a)** Log magnitude (the *light-gray curve* corresponds to low-quefrency components), and **(b)** unwrapped phase

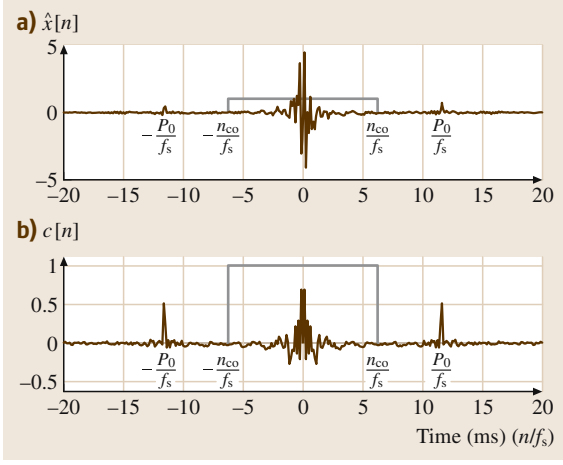



**Fig. 9.10** **(a)** Complex cepstrum. **(b)** Cepstrum

Observe the rapidly varying periodic components in Fig. 9.9a,b, and the corresponding peaks in Fig. 9.10 located at about quefrency 11.5 ms ($P_0 = 92$ samples at $f_s = 8000$ samples/s rate). These peaks in the cepstrum and complex cepstrum are due to the voiced nature of the speech segment, and their quefrency locations correspond to the *pitch period* in Fig. 9.8a. Furthermore, note that when expressed in Hz, the period of the ripples in Fig. 9.9 is $8000/92 = 87$ Hz. Thus, we see from the cepstrum that the fundamental voice frequency for this segment of speech is about 87 Hz.

## 9.4.2 Homomorphic Filtering of Speech

Homomorphic filtering can be used to separate the components of the speech model. In the terminology of Bogert et al., we *lifter* the signal by multiplying its complex cepstrum (or cepstrum) by a sequence $\ell[n]$, i. e., we form

$$\hat{y}[n] = \ell[n]\hat{x}[n] \,. \tag{9.28}$$

This is one choice for the linear system $L\{\ \}$ shown in the canonic system of Fig. 9.1. The corresponding *liftered output* is obtained by implementing the inverse characteristic system of Fig. 9.3. Note that the corresponding operation on the complex logarithm is periodic convolution in the frequency domain, i. e.,

$$\hat{Y}(\mathrm{e}^{\mathrm{i}\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(\mathrm{e}^{\mathrm{i}\theta})\hat{L}(\mathrm{e}^{\mathrm{i}(\omega-\theta)})\,\mathrm{d}\theta \,. \tag{9.29}$$

In words, liftering is equivalent to *frequency-invariant* linear filtering of the complex logarithm. For example, multiplication by a function such as the light-gray curve in Fig. 9.10a would have the effect of removing the periodic component from the log magnitude and unwrapped phase in Fig. 9.9a or b or in the case of the cepstrum, the periodic component would be removed from only the log magnitude. Specifically, if we multiply all components of the complex cepstrum of Fig. 9.10a above quefrency 50 (6.25 ms) by zero, and compute the DTFT of the result, we obtain $\hat{Y}(\mathrm{e}^{\mathrm{i}\omega})$, as shown by the smooth curves in Fig. 9.9a,b. If we send the resulting *liftered* complex logarithm through the exponential and inverse DTFT of the inverse characteristic system of Fig. 9.3, the result is as shown in Fig. 9.8c, which we can interpret as an estimate of $h[n]$. Similarly, we can remove the low quefrencies by retaining cepstrum components above some cutoff quefrency, and we obtain an output that depends mainly on the excitation components of the complex cepstrum. Such a result is shown in Fig. 9.8b, which is an estimate of $u_w[n]$. Notice that the impulses at multiples of the pitch period retain the shape of the Hamming window as suggested by (9.22).

The previous example illustrates that homomorphic filtering can be used to separate the components of a convolution, and this technique can be used to *deconvolve* short segments of voiced speech. Specific applications of this approach will be considered in the following sections.



Amazon EX1010 Page 197

## 9.5 Relation to LPC

The technique of linear predictive analysis, usually referred to as linear predictive coding (LPC), is widely used in digital speech processing systems. It is therefore of interest to consider the relationship between cepstrum analysis and linear predictive analysis. In LPC, an all-pole model such as (9.18) is obtained by first assuming that the model output $s[n]$ is equal to the given speech signal $x[n]$, and then computing a set of predictor coefficients $\alpha_k$ that minimize the mean-squared prediction error

$$E = \left\langle e^2[n] \right\rangle = \left\langle \left( x[n] - \sum_{k=1}^{p} \alpha_k x[n-k] \right)^2 \right\rangle . \quad (9.30)$$

The optimum coefficients satisfy the set of $p$ linear equations in the $p$ unknown coefficients $\alpha_k$

$$\sum_{k=1}^{p} \alpha_k r[|k-i|] = r[i] \; i = 1, 2, \ldots , p , \quad (9.31a)$$

and the gain constant $G$ is given by

$$G = r[0] - \sum_{k=1}^{p} \alpha_k r[k] , \quad (9.31b)$$

where $r[k] =< x[n]x[n+k] >$ with $< >$ denoting averaging over a finite time window. Thus, $p+1$ values of the autocorrelation function $r[k]$ are sufficient to determine all the $p+1$ parameters of the all-pole model of (9.18) [9.8].

### 9.5.1 LPC Versus Cepstrum Smoothing

Figure 9.11 shows the autocorrelation function and the cepstrum of the segment of speech in Fig. 9.8a. (As is common in LPC analysis [9.8], the high frequencies were pre-emphasized by a first difference operator prior to LPC and cepstral analysis.) Note that the autocorrelation function $r[k]$ is the IDTFT of $|X(e^{i\omega})|^2$, while the cepstrum $c[n]$ is the IDTFT of $\log|X(e^{i\omega})|$, so it is not surprising that they should be related and have similar properties. Like the cepstrum, the autocorrelation shows a peak at the time corresponding to the pitch period of the speech signal; however, this peak is not nearly as distinct as the peak in the cepstrum. In LPC analysis, the periodic variations of the short-time Fourier transform are removed by taking $p$ to be much less than the pitch period $P_0$. The faint gray line in Fig. 9.11a shows the autocorrelation values selected to determine a system model with $p = 12$, which is much less than the value $P_0 = 92$ determined by cepstrum analysis to be the period of the segment of speech under analysis. Such a value for $p$ is sufficient to represent the combined effects of the vocal tract, glottal spectrum, and radiation at the lips for a sampling rate of 8 kHz, but too small to represent the fine detail in the short-time spectrum. If $H(z)$ in (9.18) is evaluated on the unit circle of the $z$-plane, we get an estimate of the frequency response of the linear system of the speech model, i. e., the function $\log|H(e^{i\omega/f_s})|$, which is plotted in Fig. 9.12 as the solid curve. It is superimposed with the original $\log|X(e^{i\omega/f_s})|$, shown as the rapidly varying light-gray plot.

For comparison, the dashed line is the result of liftering the short-time spectrum with a cepstrum cutoff quefrency of $n_{co} = 50$ samples and the dash-dot line shows liftering with a lower cutoff quefrency of $n_{co} = 13$. Thus, both the LPC model and cepstrum liftering can smooth out the periodic component of the short-time spectrum of a segment of voiced speech. Notice that the peaks of the LPC spectrum tend be biased toward a nearby harmonic peak of the short-time spectrum. This can cause undesirable variability in LPC representations. Generally, the LPC order $p$ is chosen to provide one resonance (two complex-conjugate poles) per kilohertz of bandwidth with 2–4 extra poles to represent overall spectral tilt, etc. The liftered spectra, on the other hand, are essentially local averages of the log of the short-time Fourier transform magnitude. Thus, they follow the slow variations of the log of the short-time spectrum, and the lower the cutoff quefrency, the smoother the resulting spectrum estimate.

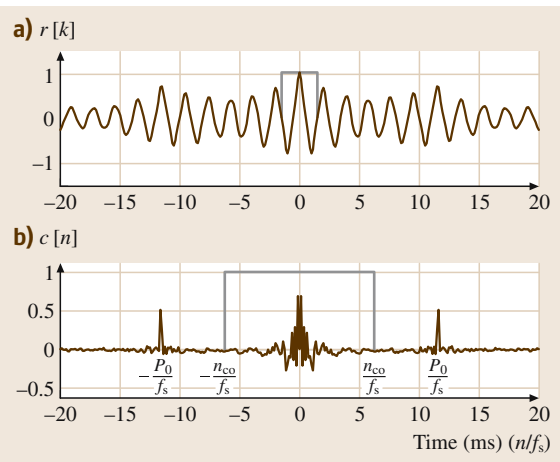



**Fig. 9.11** **(a)** Autocorrelation function showing points used in LPC analysis. **(b)** Cepstrum showing points used in homomorphic smoothing of the short-time spectrum



Amazon EX1010 Page 198

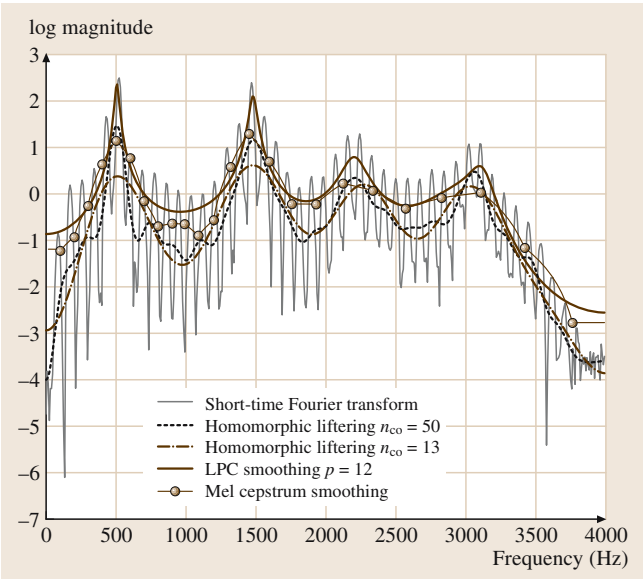



**Fig. 9.12** Comparison of spectral smoothing methods

### 9.5.2 Cepstrum from LPC Model

In the LPC method, a short segment of windowed speech is represented first by $p+1$ autocorrelation values $r[k]$, $k = 0, 1, \ldots, p$, then by $G$ and the predictor coefficients $\alpha_k$, $k = 1, 2, \ldots, p$, and finally by the $z$-transform $H(z)$ given by (9.18). The truncation of the autocorrelation function to $p+1$ values causes the smoothing that we seek in estimating the vocal tract system. If we realize that, corresponding to $H(z)$, there is an impulse response $h[n]$, then it follows that the LPC analysis has computed the parameters of the $z$-transform of $h[n]$, and thus, we can think of LPC analysis as an implementation of the Z{ } operator in Fig. 9.5 when the input is $h[n]$. In Sect. 9.2, we showed that, given the rational $z$-transform of a sequence, we can compute the complex cepstrum directly from the zeros of the numerator and denominator polynomials. From (9.18) and (9.16) and the fact that the LPC model estimate is minimum phase (all poles inside the unit circle) [9.8], it follows that the complex cepstrum of the impulse response of the LPC model is

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log G & n = 0 \\ \sum_{k=1}^{M_i} \frac{c_k^n}{n} & n > 0 \end{cases} \tag{9.32}$$

and the cepstrum would be

$$c[n] = \mathrm{Ev}\{\hat{h}[n]\} = \begin{cases} \log G & n = 0 \\ \frac{1}{2} \sum_{k=1}^{M_i} \frac{c_k^{|n|}}{|n|} & n \neq 0 \end{cases} . \tag{9.33}$$

### 9.5.3 Minimum Phase and Recursive Computation

Minimum-phase sequences like the LPC model impulse response have a number of important properties. Some of these properties are themselves sufficient to define the minimum-phase condition. For example, all the poles and zeros of the $z$-transform of a minimum-phase sequence are inside the unit circle. An equivalent statement is that the complex cepstrum is causal, i. e., $\hat{h}[n] = 0$ for $n < 0$ as in (9.32).

The special properties of a minimum-phase sequence can be used to derive additional computational algorithms that can be useful in speech processing. For example, using the derivative theorem for $z$-transforms [9.4] and the causality of minimum-phase sequences and their complex cepstra, it can be shown [9.3] that the impulse response of a minimum-phase system and its corresponding complex cepstrum are related by the recursion formula

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log G & n = 0 \\ \frac{h[n]}{h[0]} - \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \frac{\hat{h}[k]h[n-k]}{h[0]} & n \geq 1 \end{cases} \tag{9.34}$$

If we write the system function of the LPC model of (9.18) as

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} = \frac{G}{A(z)} , \tag{9.35}$$

we can define

$$a[n] = \begin{cases} 1 & n = 0 \\ -\alpha_n & 1 \leq n \leq p \\ 0 & \text{otherwise} \end{cases} \tag{9.36}$$

as the sequence whose $z$-transform is the denominator polynomial $A(z)$. Since the complex cepstrum is the

inverse $z$-transform of $\hat{H}(z) = \log G - \log\{A(z)\}$, it follows that $\hat{h}[n] = (\log G)\delta[n] - \hat{a}[n]$, and using (9.34) and the fact that $\hat{h}[n] = -\hat{a}[n]$ for $n \geq 1$, we can express the complex cepstrum $\hat{h}[n]$ directly in terms of the predictor coefficients as

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log G & n = 0 \\ \alpha_n + \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]\alpha_{n-k} & n > 0 \end{cases} , \quad (9.37)$$

From (9.37) we see that the predictor coefficients can be obtained from the complex cepstrum through

$$\alpha_n = \hat{h}[n] - \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]\alpha_{n-k} \quad 1 \leq n \leq p \, . \quad (9.38)$$

From (9.38), it follows that $p+1$ values of the complex cepstrum are sufficient to fully determine the LPC model since all the predictor coefficients and $G$ can be computed from $\hat{h}[n]$ for $n = 0, 1, \ldots, p$.

*Kopec* et al. [9.10] showed several other ways that linear prediction and cepstrum analysis can be combined to separate the components of more-complicated models than the all-pole model usually assumed in LPC analysis.

## 9.6 Application to Pitch Detection

The cepstrum was first applied in the speech processing field to determining the excitation parameters for the discrete-time speech model of Fig. 9.7. According to *Noll* [9.11], his colleague M. R. Schroeder suggested to him that the periodic structure of voiced speech was very similar to the echo structure that motivated their Bell Telephone Laboratories colleagues to define the cepstrum. He applied the cepstrum using short-time Fourier analysis as we have described in Sect. 9.4.1, and like *Bogert* et al. [9.1] he was only interested in *detecting* periodicity, not in extracting the details of the speech model. Noll proposed a pitch-detection algorithm that was based on cepstrum analysis implemented as in Fig. 9.4 and applied to successive short-time windowed segments of a speech signal. This is illustrated in Fig. 9.6 for two window positions, one clearly unvoiced speech and the later one voiced speech.

Figure 9.13 shows a plot that is very similar to the plot first published by *Noll* [9.11]. On the left is a sequence of log short-time spectra (rapidly varying curves) and on the right is the corresponding sequence of cepstra computed from the log spectra on the left. Spectrum and cepstrum number 1 are from the segment of speech selected by the leftmost window in Fig. 9.6. The successive spectra and cepstra are for 50 ms segments obtained by moving the window in steps of 12.5 ms (100 samples at 8000 samples/s). Thus, the second window position in Fig. 9.6 corresponds to spectrum and cepstrum number 11 in Fig. 9.13. From the given window length and time increment, we can see from Fig. 9.6 that for positions 1–5, the window will only include unvoiced speech, while for positions 6 and 7 the signal will be partly voiced and partly unvoiced. For positions 8–15 the window only includes voiced speech. Note that the rapid variations of the unvoiced spectra appear random with no periodic structure. This is typical of Fourier transforms (periodograms) of short segments of random signals. On the other hand, the spectra for voiced segments have a structure of periodic ripples due to the harmonic structure of the quasiperiodic segment of voiced speech. As can be seen from the plots on the right, the cepstrum peak at about quefrency 11–12 ms strongly signals voiced speech. As we have shown, the quefrency of the peak is an accurate estimate of the pitch period during the corresponding speech interval. As shown in Fig. 9.11, the autocorrelation function also displays an indication of periodicity, but not not nearly as clearly as does the cepstrum.

The essence of the pitch-detection algorithm proposed by Noll is to compute a sequence of short-time cepstra and search each successive cepstrum for a peak in the quefrency region of the expected pitch period. The presence of a strong peak implies voiced speech, and the quefrency location of the peak gives an estimate of the pitch period. As in most signal processing applications such as this, the algorithm includes many features designed to handle cases that do not fit the underlying model very well. For example, for frames 6 and 7 the cepstrum peak is weak, corresponding to the transition from unvoiced to voiced speech. In other problematic cases, the peak at twice the pitch period may be stronger than the peak at the quefrency of the pitch period. Noll applied temporal continuity constraints to prevent such errors. The reader should consult *Noll*'s paper [9.11] for the full details of his cepstrum pitch-detection algorithm.



Amazon EX1010 Page 200

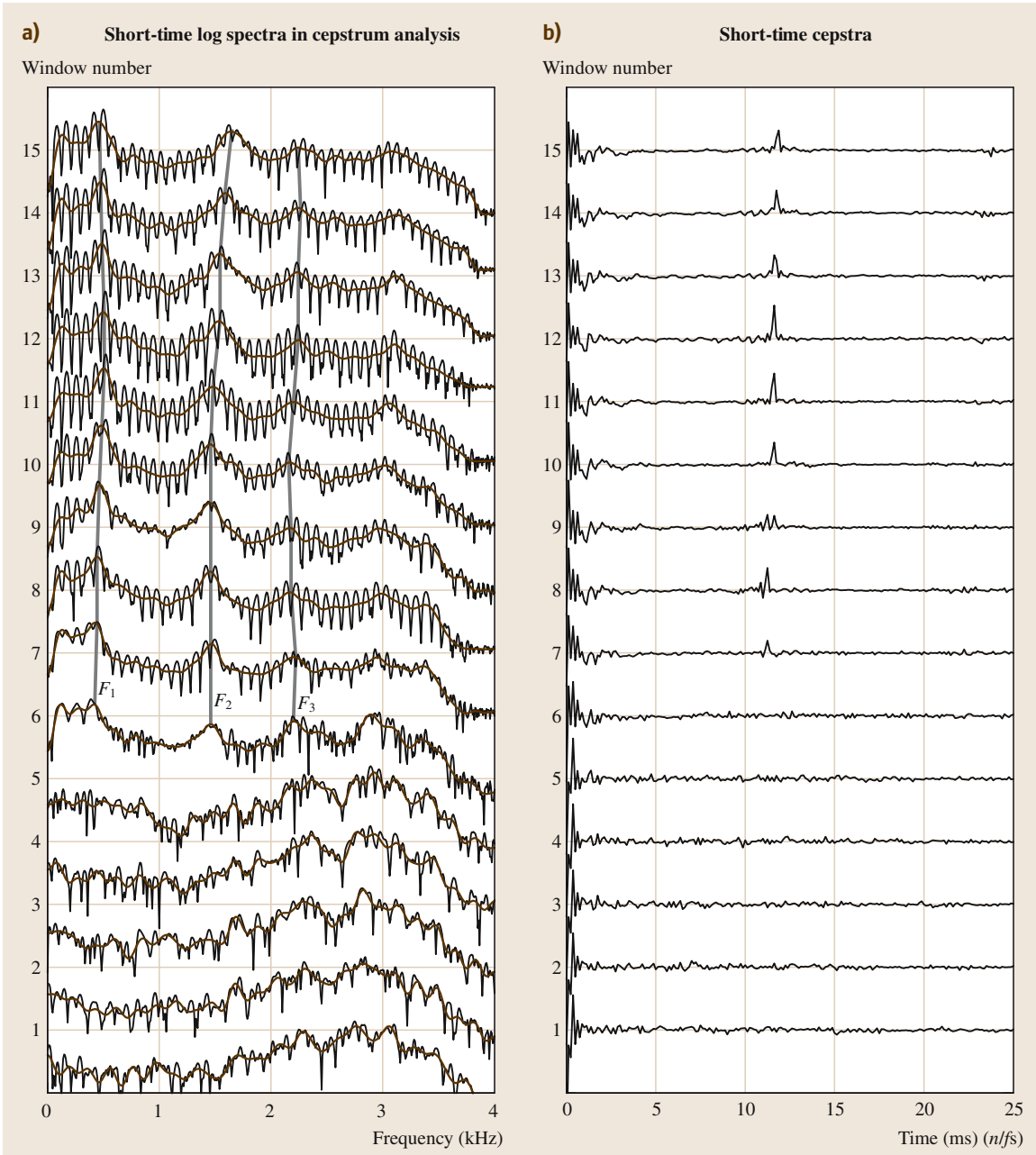



**Fig. 9.13** (a) Short-time spectra and (b) cepstra of segment of speech in Fig. 9.6

## 9.7 Applications to Analysis/Synthesis Coding

In analysis/synthesis speech coding, the sampled speech signal is divided into frames (blocks of samples), and the excitation parameters and impulse response are determined for each successive frame. The cepstrum can



Amazon EX1010 Page 201

be used to estimate the parameters of the simple excitation model. For compression, the excitation parameters (voicing decision and pitch period) are quantized together with a quantized representation of the impulse response of the vocal-tract system. The composite quantized representation is then used for transmission or storage. To reconstitute the speech signal, the model of Fig. 9.7 is used with model parameters updated at the frame rate from the compressed data. For example, LPC vocoders use LPC analysis techniques to obtain the vocal-tract system model in the form of (9.18). This all-pole representation implies that the vocal-tract model can be implemented by a recursive digital filter for synthesis [9.12].

The cepstrum has also been used in analysis/synthesis speech coding to extract an estimate of the linear system in the speech model of Fig. 9.7. We shall discuss three approaches based on homomorphic methods.

### 9.7.1 Homomorphic Vocoder

A homomorphic vocoder is an analysis/synthesis speech coding system that uses a cepstrum pitch detector as described in Sect. 9.6 to estimate the excitation parameters and short-time homomorphic deconvolution to estimate the time varying impulse response of the speech model of Fig. 9.7 [9.13]. More specifically, the vocal-tract impulse response is extracted by homomorphic filtering from the same short-time cepstrum that is used for pitch detection. Each frame of the speech signal is represented by quantized versions of voiced/unvoiced decision, pitch period, gain, and a set of low-quefrency cepstrum values to represent the vocal tract impulse response. Since the cepstrum is an even function of $n$, only $c[n]$ for $n = 1, \ldots, n_{co}$ need be retained in order to construct $c[-n]$. Furthermore, we can assume that $c[0] = \log G$.

Synthesis is done by discrete convolution of an impulse response (reconstructed from the quantized low-quefrency cepstrum values) with an excitation signal constructed from the voicing, pitch period, and gain information. For voiced speech frames, the excitation is a train of unit impulses with spacing $P_0$ (as estimated from the cepstrum), while for unvoiced frames, the excitation is a discrete random noise sequence. The gain $G$ is used to provide the correct signal amplitude after synthesis.

Since the cepstrum (not the complex cepstrum) is used to represent the linear filter for each frame, it is not possible to estimate the fully fledged mixed-phase impulse response as in the example of Fig. 9.8c. The cepstrum only represents the log magnitude of the short-time spectrum. Therefore, it is necessary to generate an appropriate phase to pair with the smoothed log magnitude function. This can be achieved as part of the liftering that extracts the low-quefrency values of the cepstrum. The simplest approach is to simply assign zero for the phase at each frequency. This is the case if we define

$$\hat{h}_z[n] = \ell_z[n]c[n] \,, \tag{9.39}$$

where

$$\ell_z[n] = \begin{cases} 1 & |n| \le n_{co} \\ 0 & \text{otherwise} \end{cases} \,. \tag{9.40}$$

This situation is depicted in Fig. 9.10b. Since $\hat{h}_z[n] = \hat{h}_z[-n]$, it follows that $\hat{H}_z(e^{i\omega})$ is purely real. Therefore, if $\hat{h}_z[n]$ is the input to the inverse characteristic system $D_*^{-1}\{\,\}$ (implemented with the DFT), then the resulting output $h_z[n]$ will have even symmetry as well. As an example, Fig. 9.14a shows the *zero-phase* impulse response that has the same log magnitude function as the mixed-phase impulse response of Fig. 9.8c, i. e., the smooth function in Fig. 9.9a .(The zero-phase impulse response has been truncated to the range $-80 \le n \le 80$

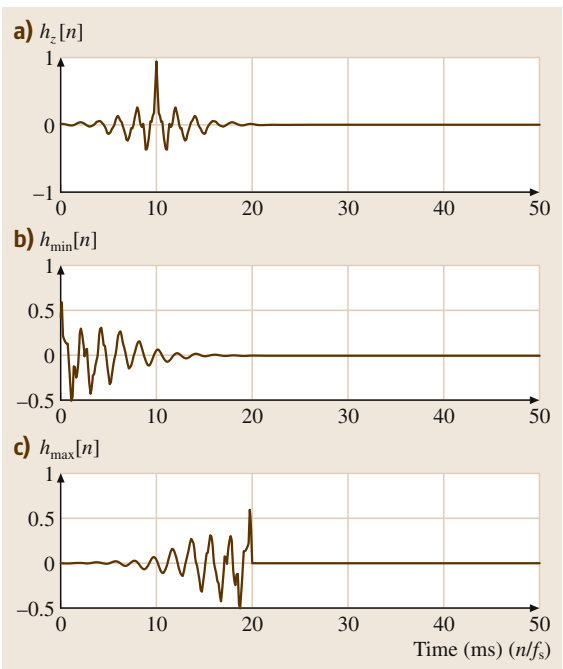



**Fig. 9.14** **(a)** Minimum-phase impulse response. **(b)** Maximum-phase impulse response. **(c)** Zero-phase impulse response



Amazon EX1010 Page 202

and a delay of 80 samples has been included to make the impulse response causal.)

Another way of pairing a phase function with the smoothed log magnitude is to impose the minimum-phase condition, i. e., the complex cepstrum is zero for $n < 0$. Specifically, if $\hat{h}_{\min}[n] = \ell_{\min}[n]c[n]$, where

$$\ell_{\min}[n] = \begin{cases} 2 & 1 \le n \le n_{co} \\ 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{9.41}$$

then the resulting impulse response $h_{\min}[n]$ at the output of the inverse characteristic system is a minimum-phase impulse response whose Fourier transform has the same log magnitude as does $h_z[n]$ corresponding to (9.39), and whose phase is the minimum-phase function that would be obtained through the discrete *Hilbert* transform [9.4]. Figure 9.14b shows the minimum-phase impulse response that has the same Fourier transform log magnitude as the impulse responses in Figs. 9.8c and 9.14a.

Still another way to obtain a nonzero phase is to impose a *maximum-phase* condition; i. e., the complex cepstrum is zero for $n > 0$. The maximum-phase impulse response reconstructed from the cepstrum with $\ell_{\max}[n] = \ell_{\min}[-n]$ will be simply $h_{\max}[n] = h_{\min}[-n]$. For our example the maximum-phase impulse response is shown in Fig. 9.14c. [Since $h_{\max}[n] = h_{\min}[-n]$ is noncausal, we have included a delay of 160 samples (20 ms) to make it causal.]

The complex cepstrum is not generally used in the homomorphic vocoder because it would be necessary to quantize and encode the complex cepstrum values for $-n_{co} \le n \le n_{co}$ as opposed to just $0 \le n \le n_{co}$ for the zero-phase, minimum-phase, and maximum-phase representations. *Oppenheim* [9.13] reported on experiments that showed that synthetic speech output using minimum-phase impulse responses was preferred slightly over synthesis with zero-phase impulse responses. Both minimum- and zero-phase synthesis were preferred to maximum-phase impulse response synthesis, even though the maximum-phase impulse response has the same wave shape and energy distribution (except time-reversed) as the minimum-phase impulse response. Oppenheim speculated that this was because the minimum-phase approach most closely matches the short-time phase of the natural vocal-tract system.

### 9.7.2 Homomorphic Formant Vocoder

Homomorphic processing has also been proposed as the basis for a formant vocoder [9.14]. Here the approach was like the homomorphic vocoder of Sect. 9.7.1 in that the excitation parameters were derived from a cepstrum pitch detector, but instead of extracting an impulse response at each frame, the first three formant (resonance) frequencies were estimated and tracked in time from the liftered short-time log magnitude spectrum. These three (quantized) formant frequency estimates were used to represent the vocal tract filter for each analysis frame. For synthesis, the three formant frequencies were used to control the complex pole locations of three second-order sections of a 10th-order cascade-form recursive digital filter; a sampling rate of 8 kHz was used. The remaining four poles were fixed at values that give a natural overall average spectral shape.

The set of plots in the left panel of Fig. 9.13 illustrates how the formants were estimated and tracked in the short-time log spectrum. Note that the smoothed log spectra contain numerous local maxima (peaks). However, from a detailed study of the acoustics of speech production and from extensive analysis of the short-time spectrum of speech, it is well established that the formant frequencies are confined to certain ranges of frequency simply by physical constraints [9.8]. Furthermore, the low-frequency peak in the spectra of Fig. 9.13 can be attributed to the glottal pulse spectrum. Thus, by locating the peaks in the liftered log spectrum that lie in the range of approximately 200–3000 Hz, it is possible to identify and track from frame to frame the first three formant frequencies for voiced speech segments, as illustrated in Fig. 9.13. For unvoiced speech, a much simpler model was proposed consisting of one complex-conjugate pair of zeros and one complex-conjugate pair of poles. The pole and zero locations were estimated from the liftered log magnitude spectra so as to match the overall general shape of the unvoiced spectrum [9.8].

Although this representation of speech is quite simplified and therefore bound to suffer in terms of quality of the synthetic speech that can be reconstructed, its virtue is that the information rate for transmitting the pitch period, gain, and three formant frequencies can be very low. With rather straightforward coding techniques, it was found that the information rate could be as low as 600 bits/s without degradation of the quality that could be achieved when the parameters were unquantized. For many talkers, the synthetic speech was highly intelligible and of acceptable quality for communications. However, the simplicity and the many fixed parameters of the synthesis model limit the quality that can be achieved over a wide range of voices. This is also true of both the LPC vocoder and the homomorphic vocoder of Sect. 9.7.1 with simplified voiced/unvoiced



Amazon EX1010 Page 203

excitation modeling. Such approaches typically achieve a level of speech quality reproduction that is acceptable for some applications at bit rates around 2400 bps, but not nearly transparent to modeling distortions. Such representations are degraded by coarser quantization of the parameters, but quality cannot be improved by simply representing the parameters with greater and greater accuracy.

### 9.7.3 Analysis-by-Synthesis Vocoder

As increased computational power became readily available in decade of the 1980s, speech coding researchers turned their attention toward more-sophisticated techniques for modeling the excitation in LPC vocoders. Such techniques as multipulse [9.15], code excitation (CELP) [9.16], and self-excitation [9.17] led to greatly improved output quality at somewhat higher bit rates than the classical {V/UV, pitch, gain} representation used in the LPC and homomorphic vocoders. All these approaches were based on the idea of *analysis by synthesis*. Although most of the innovations in analysis-by-synthesis methods were originally created in the context of LPC modeling of the vocal-tract filter, *Chung* [9.18, 19] demonstrated that the benefits of the improved excitation modeling also applied to the homomorphic vocoder.

Figure 9.15 shows a general block diagram for an analysis-by-synthesis homomorphic vocoder. In LPC-based coders of this type, the vocal-tract filter is recursive. The homomorphic analysis-by-synthesis coder uses an finite impulse response (FIR) filter derived from the cepstrum as discussed above. As in other coders of this type, analysis is done frame-by-frame. The analysis-by-synthesis excitation modeling is based on finding the excitation sequence $u[n]$ that minimizes the perceptually weighted error between the speech signal samples $x[n]$ and the reconstructed samples $s[n] = h[n] * u[n]$. Over a frame of length $L$ samples, the perceptually weighted error is

$$\begin{aligned} d_{\mathrm{pw}}[n] &= (x[n] - h[n] * u[n]) * g_{\mathrm{pw}}[n] \\ &= x_{\mathrm{pw}}[n] - h_{\mathrm{pw}}[n] * u[n] \end{aligned} \tag{9.42}$$

where $g_{\mathrm{pw}}[n]$ is the impulse response of the perceptual weighting filter and $h_{\mathrm{pw}}[n] = h[n] * g_{\mathrm{pw}}[n]$ is the perceptually-weighted vocal-tract impulse response. The computation of the perceptual weighting filter and weighted impulse response for the homomorphic analysis-by-synthesis coder will be discussed below.

Irrespective of how the vocal-tract impulse response is derived, the basic excitation analysis proceeds by assuming that the excitation signal is composed of a set of $N$ (one or more) scaled component signals in the form

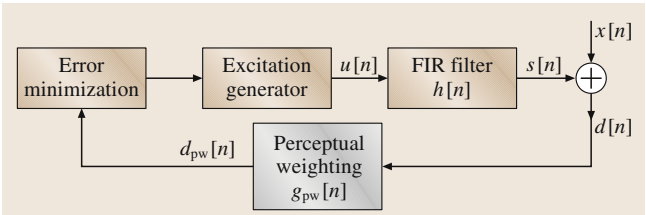



**Fig. 9.15** Analysis-by-synthesis homomorphic vocoder

$$u[n] = \sum_{k=1}^{N} \beta_k u_{\gamma_k}[n] \,. \tag{9.43}$$

The set of excitation components $u_{\gamma_k}[n]$ is predetermined. In the case of multipulse [9.15], the components are shifted unit samples $\delta[n - \gamma_k]$; in CELP [9.16] the components are drawn from a library or codebook of white-noise sequences; and in the self-excited paradigm [9.17], the components are segments drawn from a memory of the past excitation sequence. In fact, the composite excitation in (9.43) can consist of a combination of such components [9.17].

The algorithm for determining $u[n]$ for a given frame begins by computing the output of the vocal-tract filter due to the excitation of previous frames. This is subtracted from the new frame of perceptually weighted speech to initialize the error. Then the analysis proceeds by a process something like the following to determine the $k$-th component:

1. Choose a particular component $u_\gamma[n]$ from the available set, and find the value of $\beta$ that minimizes the total squared residual error over the frame.
2. Repeat step 1 for all possible input components. Choose the component (indexed by $\gamma$) that gave the lowest error, and subtract the output due to that component from the error.

These steps are repeated $N$ times to obtain the indexes $\gamma_k$ for $k = 1, 2, \ldots, N$. New values of the $\beta_k$ can be determined after all the component sequences $u_{\gamma_k}[n]$ have been selected.

The main difference between LPC and homomorphic analysis-by-synthesis coders is the way in which the impulse responses are derived. In the case of the homomorphic vocoder, $h[n]$ is the quantized minimum-phase FIR vocal-tract impulse response obtained by low-pass liftering the cepstrum. The perceptual weighting filter $g_{\mathrm{pw}}[n]$ is derived directly from the cep-



Amazon EX1010 Page 204

strum according to the principles set out by *Atal* and *Schroeder* [9.20], who argued that the approximation errors in the regions of the formant peaks would be aurally masked, so the weighting should be designed to emphasize the regions between the formant peaks. In the homomorphic vocoder system, the perceptually weighted vocal tract impulse response is obtained directly from the cepstrum by [9.18]

$$\hat{h}_{\mathrm{pw}}[n] = \ell_{\mathrm{pw}}[n]c[n]\,, \tag{9.44a}$$

where

$$\ell_{\mathrm{pw}}[n] = \begin{cases} 1-\alpha\sin\left(\frac{\pi n}{2n_{\mathrm{co}}}\right) & 0 \le n \le n_{\mathrm{co}} \\ 0 & \text{otherwise} \end{cases}\,. \tag{9.44b}$$

This low-pass liftering of the cepstrum creates a perceptually weighted vocal-tract filter where the formant resonances are flattened out as suggested by *Atal* and *Schroeder* [9.20]. To obtain the perceptual weighting filter needed to prefilter the input in (9.42), we simply observe that, since $h_{\mathrm{pw}}[n] = h[n] * g_{\mathrm{pw}}[n]$, it follows that the complex cepstrum of $g_{\mathrm{pw}}[n]$ is

$$\begin{aligned}\hat{g}_{\mathrm{pw}}[n] &= \hat{h}_{\mathrm{pw}}[n] - \hat{h}[n] \\ &= (\ell_{\mathrm{pw}}[n]-1)\hat{h}[n]\,. \end{aligned} \tag{9.45}$$

Figure 9.16 illustrates the perceptual weighting filter for the segment of speech used to create the earlier examples. The light-gray solid curve is the smoothed short-time log spectrum corresponding to the vocal-tract filter impulse response $h[n]$. The dark solid curve is the perceptually weighted vocal-tract filter corresponding to $h_{\mathrm{pw}}[n]$, and the dashed curve corresponds to the perceptual weighting filter $g_{\mathrm{pw}}[n]$. Note how the perceptual weighting filter tends to de-emphasize the spectral regions around the formant frequencies.

*Chung* [9.19] performed subjective tests of the homomorphic analysis-by-synthesis coder, and compared it to LPC-based coders. In a comparison where the same analysis structure was used with both LPC and homomorphic vocal-tract filters, he found no significant difference. However, in comparison to the state-of-the-art Department of Defense standard CELP coder available at the time, both his LPC and homomorphic systems were significantly inferior. He attributed this to enhancements such as interpolation of vocal-tract filters and interpolated codebook search that were implemented in the the DoD standard system and not in his LPC and homomorphic coders. While the results of these subjective tests were inconclusive, it was demonstrated that the homomorphic vocoder could be significantly improved by use of analysis-by-synthesis excitation analysis, and that homomorphic processing afforded new flexibilities in implementing perceptual weighting of the coding error.

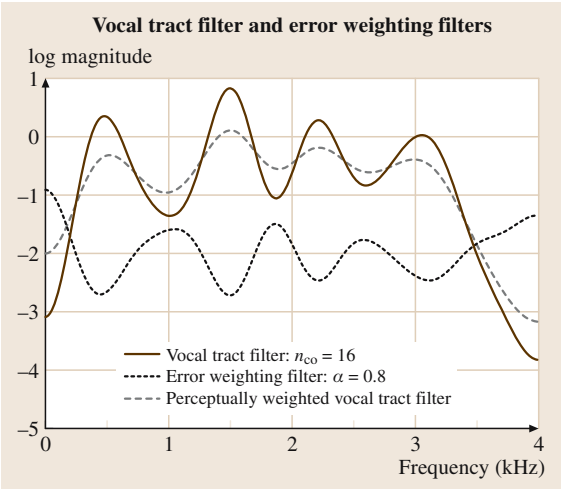



**Fig. 9.16** Illustration of perceptual weighting filter derived from the cepstrum

## 9.8 Applications to Speech Pattern Recognition

Perhaps the most pervasive application of the cepstrum in speech processing is its use in pattern recognition problems such as vector quantization (VQ) and automatic speech recognition (ASR). In such applications, a speech signal is represented on a frame-by-frame basis by a sequence of short-time cepstra. In later discussions in this section, it will be useful to use somewhat more-complicated notation. Specifically, we denote the cepstrum of the $m$-th frame of a signal $x[n]$ as $c_m^{(x)}[n]$, where $n$ denotes the quefrency index of the cepstrum. In cases where it is not necessary to distinguish between signals or frames, these additional designations will be omitted as we have done up to this point.

As we have shown, cepstra can be computed either by LPC analysis or by DFT implementation of the characteristic system. In either case, we can assume that the cepstrum vector corresponds to a gain-normalized ($c[0] = 0$) minimum-phase vocal-tract impulse response



Amazon EX1010 Page 205

that is defined by the complex cepstrum

$$\hat{h}[n] = \begin{cases} 2c[n] & 1 \le n \le n_{co} \\ 0 & n < 0 \end{cases} , \tag{9.46}$$

where we have suppressed for now the notation $\overset{(x)}{m}$.

In problems such as VQ or ASR, a test pattern $c[n]$ (a vector of cepstrum values $n = 1, 2, \ldots, n_{co}$) is compared against a comparable reference pattern $\bar{c}[n]$. Such comparisons require a distortion measure. For example, the Euclidean distance applied to the cepstrum would give

$$D = \sum_{n=1}^{n_{co}} \left| c[n] - \bar{c}[n] \right|^2 . \tag{9.47a}$$

Equivalently in the frequency domain,

$$D = \frac{1}{4\pi} \int_{-\pi}^{\pi} \left| \log |H(e^{i\omega})| - \log |\bar{H}(e^{i\omega})| \right|^2 d\omega , \tag{9.47b}$$

where $\log |H(e^{i\omega})|$ is the log magnitude of the DTFT of $h[n]$ corresponding to the complex cepstrum in (9.46) or the real part of the DTFT of $\hat{h}[n]$ in (9.46). Thus, cepstrum-based comparisons are strongly related to comparisons of smoothed short-time spectra.

The cepstrum offers an effective and flexible representation of speech for pattern recognition problems as we will discuss below.

### 9.8.1 Compensation for Linear Filtering

Suppose that we have only a linearly filtered version of the speech signal, $y[n] = h_d[n] * x[n]$, instead of $x[n]$. If the analysis window is long compared to the length of $h_d[n]$, the short-time cepstrum of one frame of the filtered speech signal $y[n]$ will be approximately

$$c_m^{(y)}[n] = c_m^{(x)}[n] + c^{(h_d)}[n] , \tag{9.48}$$

where $c^{(h_d)}[n]$ will appear more or less the same in each frame. Therefore, if we can estimate $c^{(h_d)}[n]$, which we assume is non-time-varying, we can obtain $c_m^{(x)}[n]$ at each frame from $c_y[n]$ by subtraction; i. e., $c_m^{(x)}[n] = c_m^{(y)}[n] - c^{(h_d)}[n]$. (*Stockham* [9.21] showed how $c^{(h_d)}[n]$ for such linear distortions can be estimated from the signal $y[n]$ by time averaging the log of the short-time Fourier transform.) This property is extremely attractive in situations where the set of reference patterns $\bar{c}[n]$ has been obtained under different recording or transmission conditions from those used to acquire the test vectors. In these circumstances, the test vectors can be compensated for the effects of the linear filtering prior to computing the distance measures used for comparison of patterns.

Another approach to removing the effects of linear distortions is to observe that the cepstrum component due to the distortion is the same in each frame. Therefore it can be removed by a simple first difference operation of the form

$$\Delta c_m^{(y)}[n] = c_m^{(y)}[n] - c_{m-1}^{(y)}[n] . \tag{9.49}$$

It is clear that, if $c_m^{(y)}[n] = c_m^{(x)}[n] + c^{(h_d)}[n]$ with $c^{(h_d)}[n]$ being independent of $m$, then $\Delta c_m^{(y)}[n] = \Delta c_m^{(x)}[n]$; i. e., the linear distortion effects are removed.

*Furui* [9.22] first noted that, in addition to making the cepstrum less susceptible to linear distortions, the sequence of cepstrum values has temporal information that could be of value for a speaker verification system. He used polynomial fits to cepstrum sequences to extract simple representations of the temporal variation. The *delta cepstrum* as defined in (9.49) is simply the slope of a first-order polynomial fit to the cepstrum time evolution.

### 9.8.2 Weighted Distance Measures

In using LPC analysis to obtain cepstrum feature vectors for pattern recognition problems, it was observed that there is significant statistical variability due to a variety of factors, including short-time analysis window position, bias toward harmonic peaks, and noise [9.23, 24]. A solution to this problem is to use weighted distance measures of the form

$$D = \sum_{n=1}^{n_{co}} \ell^2[n] \left| c[n] - \bar{c}[n] \right|^2 , \tag{9.50a}$$

which can be written as the Euclidean distance of liftered cepstra

$$D = \sum_{n=1}^{n_{co}} \left| \ell[n] c[n] - \ell[n] \bar{c}[n] \right|^2 . \tag{9.50b}$$

*Tohkura* [9.23] found, for example, that when averaged over many frames of speech and speakers, cepstrum values $c[n]$ have zero means and variances on the order of $1/n^2$. This suggests that $\ell[n] = n$ for $n = 1, 2, \ldots, n_{co}$ could be used to equalize the contributions for each term to the cepstrum distance.



Amazon EX1010 Page 206

*Juang* et al. [9.24] observed that the variability due to the vagaries of LPC analysis could be lessened by using a high-pass lifter of the form

$$\ell[n] = 1 + 0.5 n_{\mathrm{co}} \sin(\pi n / n_{\mathrm{co}}) \,, \; n = 1, 2, \ldots, n_{\mathrm{co}} \tag{9.51}$$

Tests of weighted distance measures have shown consistent improvements in automatic speech recognition tasks [9.24].

### 9.8.3 Group Delay Spectrum

The weighted cepstral distance measures of the previous subsection were given a new interpretation by *Itakura* and *Umezaki* [9.25], who invoked the following basic property of the DTFT:

$$n\hat{h}[n] \Longleftrightarrow \mathrm{i}\frac{\mathrm{d}\hat{H}(\mathrm{e}^{\mathrm{i}\omega})}{\mathrm{d}\omega} \,, \tag{9.52}$$

where $\Longleftrightarrow$ denotes the unique relationship between a sequence and its DTFT. An interesting result can be obtained if we represent the complex cepstrum as

$$\hat{h}[n] = c[n] + g[n] \,, \tag{9.53}$$

where $c[n] = \mathrm{Ev}\{\hat{h}[n]\}$ is the even part and $g[n] = \mathrm{Odd}\{\hat{h}[n]\}$ is the odd part of the complex cepstrum. Recalling that the DTFT of the complex cepstrum is by definition $\hat{H}(\mathrm{e}^{\mathrm{i}\omega}) = \log|H(\mathrm{e}^{\mathrm{i}\omega})| + \mathrm{i}\arg\{H(\mathrm{e}^{\mathrm{i}\omega})\}$, it can be shown that the following DTFT relations hold:

$$nc[n] \Longleftrightarrow \mathrm{i}\frac{\mathrm{d}\log|H(\mathrm{e}^{\mathrm{i}\omega})|}{\mathrm{d}\omega} \tag{9.54a}$$

and

$$ng[n] \Longleftrightarrow -\frac{\mathrm{d}\arg\{H(\mathrm{e}^{\mathrm{i}\omega})\}}{\mathrm{d}\omega} \,. \tag{9.54b}$$

The DTFT expression on the right in (9.54b) is the group delay for the DTFT $H(\mathrm{e}^{\mathrm{i}\omega})$; i. e.,

$$\mathrm{grd}\{H(\mathrm{e}^{\mathrm{i}\omega})\} = -\frac{\mathrm{d}\arg\{H(\mathrm{e}^{\mathrm{i}\omega})\}}{\mathrm{d}\omega} \,. \tag{9.55}$$

Now if $h[n]$ is obtained by LPC analysis as discussed in Sect. 9.5, the complex cepstrum satisfies $\hat{h}[n] = 0$ for $n < 0$. This means that $\hat{h}[n] = 2c[n] = 2g[n]$ for $n > 0$. If we define $\ell[n] = n$, then

$$D = \sum_{n=-\infty}^{\infty} \left|\ell[n]g[n] - \ell[n]\bar{g}[n]\right| \tag{9.56a}$$

is equivalent to

$$D = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|\mathrm{grd}\{H(\mathrm{e}^{\mathrm{i}\omega})\} - \mathrm{grd}\{\bar{H}(\mathrm{e}^{\mathrm{i}\omega})\}\right| \mathrm{d}\omega \tag{9.56b}$$

or, alternatively,

$$D = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|\frac{\mathrm{d}\log|\bar{H}(\mathrm{e}^{\mathrm{i}\omega})|}{\mathrm{d}\omega} - \frac{\mathrm{d}\log|H(\mathrm{e}^{\mathrm{i}\omega})|}{\mathrm{d}\omega}\right| \mathrm{d}\omega \,. \tag{9.56c}$$

The result of (9.56c) was also given by *Tohkura* [9.23]. Instead of $\ell[n] = n$ for all $n$, or the lifter of (9.51), Itakura proposed the lifter

$$\ell[n] = n^s \mathrm{e}^{-n^2/2\tau^2} \,. \tag{9.57}$$

This lifter has great flexibility. For example, if $s = 0$ we have simply low-pass liftering of the cepstrum. If $s = 1$ and $\tau$ is large, we have essentially $\ell[n] = n$ for small $n$ with high-quefrency tapering. The effect of liftering with (9.57) is illustrated in Fig. 9.17, which shows in (a) the short-time Fourier transform of a segment of voiced speech along with an LPC spectrum with $p = 12$. Figure 9.17b shows the liftered group delay spectrum for $s = 1$ and $\tau$ ranging from 5 to 35 in steps of 10. Observe that, as $\tau$ increases, the formant frequencies are increasingly emphasized. If larger values of $s$ are used, even greater enhancement of the resonance structure is observed.

*Itakura* and *Umezaki* [9.25] tested the group delay spectrum distance measure in an automatic speech recognition system. They found that, for clean test utterances, the difference in recognition rate was small for

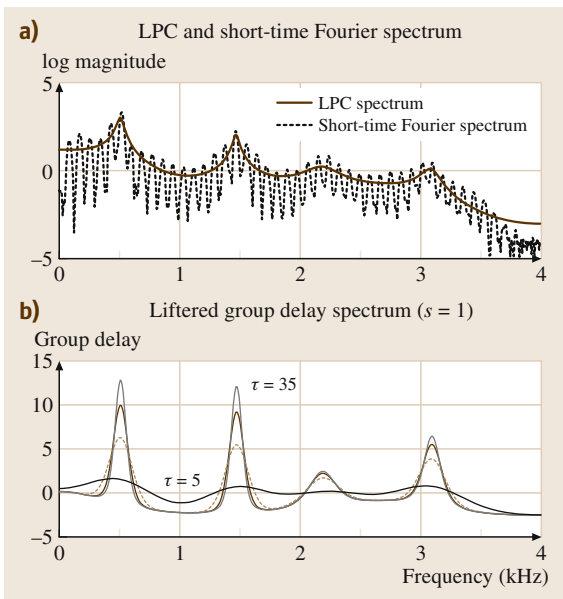



**Fig. 9.17a,b** Illustration of Itakura's liftered group delay spectrum (after [9.25])



Amazon EX1010 Page 207

different values of $s$ when $\tau \approx 5$, although performance suffered with increasing $s$ for larger values of $\tau$. This was attributed to the fact that, for larger $s$, the group delay spectrum becomes very sharply peaked and thus more sensitive to small differences in formant locations. However, in test conditions with additive white noise and also with linear filtering distortions, recognition rates improved significantly with $\tau = 5$ and increasing values of the parameter $s$.

### 9.8.4 Mel-Frequency Cepstrum Coefficients (MFCC)

As we have seen, weighted cepstrum distance measures have a directly equivalent interpretation in terms of distance in the frequency domain. This is significant in light of models for the human perception of sound, which are based on a frequency analysis performed in the inner ear. With this in mind, *Davis* and *Mermelstein* [9.26] were motivated to formulate a new type of cepstrum representation that has come to be widely used and known as the mel-frequency cepstrum coefficients (MFCC).

The basic idea is to compute a frequency analysis based upon a filter bank with approximately critical band spacing of the filters and bandwidths. For 4 kHz bandwidth, approximately 20 filters are used. In most implementations, a short-time Fourier analysis is done first, resulting in a DFT $X_m[k]$ for the $m$-th frame. Then the DFT values are grouped together in critical bands and weighted by triangular weighting functions as depicted in Fig. 9.18. Note that the bandwidths in Fig. 9.18 are constant for center frequencies below 1 kHz and then increase exponentially up to half the sampling rate of 4 kHz, resulting in 24 *filters*. The mel-spectrum of the $m$-th frame is defined for $r = 1, 2, \ldots, R$ as

$$\mathrm{MF}_m[r] = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k] X_m[k]|^2 \qquad (9.58a)$$

where $V_r[k]$ is the weighting function for the $r$-th filter ranging from DFT index $L_r$ to $U_r$, and

$$A_r = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k]|^2 \qquad (9.58b)$$

is a normalizing factor for the $r$-th mel-filter. This normalization is built into the weighting functions in Fig. 9.18. It is needed so that a perfectly flat input Fourier spectrum will produce a flat mel-spectrum. For each frame, a discrete cosine (DCT) transform of the log of the magnitude of the mel-filter outputs is computed to form the function mfcc$[n]$ as in

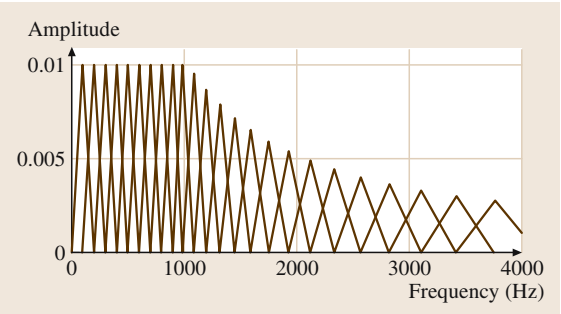



**Fig. 9.18** Weighting functions for mel-scale filtering

$$\mathrm{mfcc}[n] = \frac{1}{R} \sum_{r=1}^{R} \log\left(\mathrm{MF}_m[r]\right) \cos\left[\frac{2\pi}{R}\left(r + \frac{1}{2}\right) n\right] \qquad (9.59)$$

Typically, mfcc$[n]$ is evaluated for a number of coefficients $N_{\mathrm{mfcc}}$ that is less than the number of mel-filters, e.g., $N_{\mathrm{mfcc}} = 13$ and $R = 24$. Figure 9.12 shows the result of MFCC analysis of a frame of voiced speech compared with the short-time spectrum, LPC spectrum, and two homomorphically smoothed spectra. The large dots are the values of $\log\left(\mathrm{MF}_m[r]\right)$ and the curve interpolated between them is a spectrum reconstructed at the original DFT frequencies. Note that all these spectra are different, but they have in common that they have peaks at the formant resonances. At higher frequencies, the reconstructed mel-spectrum of course has more smoothing due to the structure of the filter bank.

Note that the delta cepstrum idea expressed by (9.49) can be applied to MFCC to remove the effects of linear filtering as long as the frequency response of the distorting linear filter does not vary much across each of the mel-frequency bands.

The MFCC have become firmly established as the basic feature vector for many speech and acoustic pattern recognition problems. For this reason, new and efficient ways of computing MFCC are of interest. An intriguing proposal is to use floating gate electronic technology to implement the filter bank and the DCT computation with only microwatts of power [9.27].



Amazon EX1010 Page 208

## 9.9 Summary

This chapter has attempted to summarize the many ways in which the cepstrum has been used in speech processing. We started with definitions and showed many of the analytical results that flow from those definitions. These results have found application in many areas of speech and audio processing. With the advent of new approaches to cepstrum computation, new applications are likely to continue to emerge.

### References


9.1 B.P. Bogert, M.J.R. Healy, J.W. Tukey: The quefrency alanysis of times series for echos: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking, Proc. of the Symposium on Time Series Analysis, ed. by M. Rosenblatt (Wiley, New York 1963)

9.2 R.W. Schafer: *Echo removal by discrete generalized linear filtering* (MIT, Cambridge 1968), Ph.D. dissertation

9.3 A.V. Oppenheim, R.W. Schafer, T.G. Stockham Jr.: Nonlinear filtering of multiplied and convolved signals, Proc. IEEE **56**(8), 1264–1291 (1968)

9.4 A.V. Oppenheim, R.W. Schafer, J.R. Buck: *Discrete-Time Signal Processing* (Upper Saddle River, Prentice-Hall 1999)

9.5 A.V. Oppenheim: *Superposition in a Class of Nonlinear Systems* (MIT, Cambridge 1964), Ph.D. dissertation, Also: MIT Research Lab. of Electronics, Cambridge, Massachusetts, Technical Report 432

9.6 J.M. Tribolet: A new phase unwrapping algorithm, IEEE Trans. Acoust. Speech **ASSP-25**(2), 170–177 (1977)

9.7 G.A. Sitton, C.S. Burrus, J.W. Fox, S. Treitel: Factoring very-high-degree polynomials, IEEE Signal Proc. Mag. **20**(6), 27–42 (2003)

9.8 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice-Hall, Englewood Cliffs 1978)

9.9 A.V. Oppenheim, R.W. Schafer: Homomorphic analysis of speech, IEEE Trans. Audio Electroacoust. **AU-16**, 221–228 (1968)

9.10 G.E. Kopec, A.V. Oppenheim, J.M. Tribolet: Speech analysis by homomorphic prediction, IEEE Trans. Acoust. Speech **ASSP-25**(1), 40–49 (1977)

9.11 A.M. Noll: Cepstrum pitch determination, J. Acoust. Soc. Am. **41**(2), 293–309 (1967)

9.12 B.S. Atal, S.L. Hanauer: Speech analysis and synthesis by linear prediction of the speech wave, J. Acoust. Soc. Am. **50**, 561–580 (1971)

9.13 A.V. Oppenheim: A speech analysis-synthesis system based on homomorphic filtering, J. Acoust. Soc. Am. **45**(2), 293–309 (1969)

9.14 R.W. Schafer, L.R. Rabiner: System for automatic formant analysis of voiced speech, J. Acoust. Soc. Am. **47**(2), 458–465 (1970)

9.15 B.S. Atal, J. Remde: A new model of LPC exitation for producing natural-sounding speech at low bit rates, Proc. IEEE ICASSP (1982), 614–617

9.16 M.R. Schroeder, B.S. Atal: Code-excited linear prediction (CELP): high-quality speech at very low bit rates, Proc. IEEE ICASSP (1985), 937–940

9.17 R.C. Rose, T.P. Barnwell III: The self excited vocoder – an alternate approach to toll quality at 4800 bps, Proc. IEEE ICASSP **11**, 453–456 (1986)

9.18 J.H. Chung, R.W. Schafer: Excitation modeling in a homomorphic vocoder, Proc. IEEE ICASSP **1**, 25–28 (1990)

9.19 J.H. Chung, R.W. Schafer: Performance evaluation of analysis-by-synthesis homomorphic vocoders, Proc. IEEE ICASSP **2**, 117–120 (1992)

9.20 B.S. Atal, M.R. Schroeder: Predictive coding of speech signals and subjective error criterion, IEEE Trans. Acoust. Speech **ASSP-27**, 247–254 (1079)

9.21 T.G. Stockham Jr., T.M. Cannon, R.B. Ingebretsen: Blind deconvolution through digital signal processing, Proc. IEEE **63**, 678–692 (1975)

9.22 S. Furui: Cepstral analysis technique for automatic speaker verification, IEEE Trans. Acoust. Speech **ASSP-29**(2), 254–272 (1981)

9.23 Y. Tohkura: A weighted cepstral distance measure for speech recognition, IEEE Trans. Acoust. Speech **ASSP-35**(10), 1414–1422 (1987)

9.24 B.-H. Juang, L.R. Rabiner, J.G. Wilpon: On the use of bandpass liftering in speech recognition, IEEE Trans. Acoust. Speech **ASSP-35**(7), 947–954 (1987)

9.25 F. Itakura, T. Umezaki: Distance measure for speech recognition based on the smoothed group delay spectrum, Proc. IEEE ICASSP **12**, 1257–1260 (1987)

9.26 S.B. Davis, P. Mermelstein: Comparison of parametric representations for monosyllabic word recognition in continously spoken sentences, IEEE Trans. Acoust. Speech **ASSP-28**(4), 357–366 (1980)

9.27 P.D. Smith, M. Kucic, R. Ellis, P. Hasler, D.V. Anderson: Mel-frequency cepstrum encoding in analog floating-gate circuitry, Proc. ISCAS **2002**(4), 671–674 (2002)




Amazon EX1010 Page 209

# 10. Pitch and Voicing Determination of Speech with an Extension Toward Music Signals

W. J. Hess

This chapter reviews selected methods for pitch determination of speech and music signals. As both these signals are time variant we first define what is subsumed under the term *pitch*. Then we subdivide pitch determination algorithms (PDAs) into short-term analysis algorithms, which apply some spectral transform and derive pitch from a frequency or lag domain representation, and time-domain algorithms, which analyze the signal directly and apply structural analysis or determine individual periods from the first partial or compute the instant of glottal closure in speech. In the 1970s, when many of these algorithms were developed, the main application in speech technology was the vocoder, whereas nowadays prosody recognition in speech understanding systems and high-accuracy pitch period determination for speech synthesis corpora are emphasized. In musical acoustics, pitch determination is applied in melody recognition or automatic musical transcription, where we also have the problem that several pitches can exist simultaneously.

10.1 **Pitch in Time-Variant Quasiperiodic Acoustic Signals** ..... 182
10.1.1 Basic Definitions ..... 182
10.1.2 Why is the Problem Difficult? ..... 184
10.1.3 Categorizing the Methods ..... 185

10.2 **Short-Term Analysis PDAs** ..... 185
10.2.1 Correlation and Distance Function .. 185
10.2.2 Cepstrum and Other Double-Transform Methods ..... 187
10.2.3 Frequency-Domain Methods: Harmonic Analysis ..... 188
10.2.4 Active Modeling ..... 190
10.2.5 Least Squares and Other Statistical Methods ..... 191
10.2.6 Concluding Remarks ..... 192

10.3 **Selected Time-Domain Methods** ..... 192
10.3.1 Temporal Structure Investigation .... 192
10.3.2 Fundamental Harmonic Processing. 193
10.3.3 Temporal Structure Simplification ... 193
10.3.4 Cascaded Solutions ..... 195

10.4 **A Short Look into Voicing Determination** . 195
10.4.1 Simultaneous Pitch and Voicing Determination ..... 196
10.4.2 Pattern-Recognition VDAs ..... 197

10.5 **Evaluation and Postprocessing** ..... 197
10.5.1 Developing Reference PDAs with Instrumental Help ..... 197
10.5.2 Error Analysis ..... 198
10.5.3 Evaluation of PDAs and VDAs– Some Results ..... 200
10.5.4 Postprocessing and Pitch Tracking .. 201

10.6 **Applications in Speech and Music** ..... 201

10.7 **Some New Challenges and Developments** 203
10.7.1 Detecting the Instant of Glottal Closure ..... 203
10.7.2 Multiple Pitch Determination ..... 204
10.7.3 Instantaneousness Versus Reliability ..... 206

10.8 **Concluding Remarks** ..... 207

**References** ..... 208

Pitch and voicing determination of speech signals are the two subproblems of voice source analysis. In voiced speech, the vocal cords vibrate in a quasiperiodic way. Speech segments with voiceless excitation are generated by turbulent air flow at a constriction or by the release of a closure in the vocal tract. The parameters we have to determine are the manner of excitation, i. e., the presence of a voiced excitation and the presence of a voiceless excitation, a problem we will refer to as *voicing determination* and, for the segments of the speech signal in which a voiced excitation is present, the rate of vocal cord vibration, which is usually referred to as *pitch determination* or *fundamental frequency determination* in the literature.



Amazon EX1010 Page 210

Unlike the analysis of vocal-tract parameters, where a number of independent and equivalent representations are possible, there is no alternative to the parameters pitch and voicing, and the quality of a synthesized signal critically depends on their reliable and accurate determination. This chapter presents a selection of the methods applied in pitch and voicing determination. The emphasis, however, is on pitch determination.

Over the last two decades, the task of fundamental frequency determination has become increasingly popular in musical acoustics as well. In the beginning the methodology was largely imported from the speech community, but then the musical acoustics community developed algorithms and applications of their own, which in turn became increasingly interesting to the speech communication area. Hence it appears justified to include the aspect of fundamental frequency determination of music signals and to present some of the methods and specific problems of this area. One specific problem is multipitch determination from polyphonic signals, a problem that might also occur in speech when we have to separate two or more simultaneously speaking voices.

Pitch determination has a rather long history which goes back even beyond the times of vocoding. Literally hundreds of pitch determination algorithms (PDAs) have been developed. The most important developments leading to today's state of the art were made in the 1960s and 1970s; most of the methods that are briefly reviewed in this chapter were extensively discussed during this period [10.1]. Since then, least-squares and other statistical methods, particularly in connection with sinusoidal models [10.2], entered the domain. A number of known methods were improved and refined, whereas other solutions that required an amount of computational effort that appeared prohibitive at the time the algorithm was first developed were revived. With the widespread use of databases containing many labeled and processed speech data, it has nowadays also become possible to thoroughly evaluate the performance of the algorithms.

The bibliography in [10.1], dating from 1983, includes about 2000 entries. To give a complete overview of the more-recent developments, at least another 1000 bibliographic entries would have to be added. It goes without saying that this is not possible here given the space limitations. So we will necessarily have to present a selection, and many important contributions cannot be described.

The remainder of this chapter is organized as follows. In Sect. 10.1 the problems of pitch and voicing determination are described, definitions of what is subsumed under the term *pitch* are given, and the various PDAs are grossly categorized. Sections 10.2 and 10.3 give a more-detailed description of selected PDAs. Section 10.4 shortly reviews a selection of voicing determination algorithms (VDAs); Sect. 10.5 deals with questions of error analysis and evaluation. Selected applications are discussed in Sect. 10.6, and Sect. 10.7 finally presents a couple of new developments, such as determining the instant of glottal closure or processing signals that contain more than one pitch, such as polyphonic music.



## 10.1 Pitch in Time-Variant Quasiperiodic Acoustic Signals

### 10.1.1 Basic Definitions

Pitch, i. e., the fundamental frequency $F_0$ and fundamental period $T_0$ of a (quasi)periodic signal, can be measured in many ways. If a signal is completely stationary and periodic, all these strategies – provided they operate correctly – lead to identical results. Since both speech and musical signals, however, are nonstationary and time variant, aspects of each strategy such as the starting point of the measurement, the length of the measuring interval, the way of averaging (if any), or the operating domain (time, frequency, lag etc.) start to influence the results and may lead to estimates that differ from algorithm to algorithm even if all these results are *correct* and *accurate*. Before entering a discussion on individual methods and applications, we must therefore have a look at the parameter pitch and provide a clear definition of what should be measured and what is actually measured.

A word on terminology first. There are three points of view for looking at such a problem of acoustic signal processing [10.3]: the *production*, the *signal-processing*, and the *perception* points of view. For pitch determination of speech, the production point of view is obviously oriented toward phonation in the human larynx; we will thus have to start from a time-domain representation of the waveform as a train of laryngeal pulses. If a pitch determination algorithm (PDA) works in a speech-production oriented way, it measures individual *laryngeal excitation cycles* or, if some averaging is performed, the *rate of vocal-fold vibration*. The signal-processing point of view, which can be applied

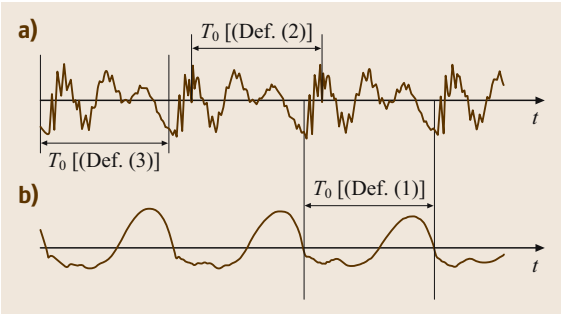



**Fig. 10.1a,b** Time-domain definitions of $T_0$. **(a)** Speech signal (a couple of periods), **(b)** glottal waveform (reconstructed). For further detail, see the text

to any acoustic signal, means that (quasi)periodicity or at least cyclic behavior is observed, and that the task is to extract those features that best represent this periodicity. The pertinent terms are *fundamental frequency* and *fundamental period*. If individual cycles are determined, we may (somewhat inconsistently) speak of *pitch periods* or simply of *periods*. The perception point of view leads to a frequency-domain representation since pitch sensation primarily corresponds to a frequency [10.4, 5] even if a time-domain mechanism is involved [10.6]. This point of view is associated with the original meaning of the term *pitch*. Yet the term *pitch* has consistently been used as some kind of common denominator and a general name for all those terms, at least in the technical literature on speech [10.7]. In the following, we will therefore use the term *pitch* in this wider sense, even for musical signals.

When we proceed from production to perception, we arrive at five basic definitions of pitch that apply to speech signals and could read as follows ([10.1, 8, 9]; Fig. 10.1):

1. *$T_0$ is defined as the elapsed time between two successive laryngeal pulses. Measurement starts at a well-specified point within the glottal cycle, preferably at the instant of glottal closure.*
   PDAs that obey this definition will be able to locate the points of glottal closure and to delimit individual laryngeal cycles. This goes beyond the scope of ordinary pitch determination in speech; if only the signal is available for the analysis, it must be totally undistorted if reliable results are to be expected. For music signals we can apply this definition if we analyze a human voice or an instrument that operates in a way similar to the human voice.
2. *$T_0$ is defined as the elapsed time between two successive laryngeal pulses. Measurement starts at an arbitrary point within an excitation cycle. The choice of this point depends on the individual method, but for a given PDA it is always located at the same position within the cycle.*
   Time-domain PDAs usually follow this definition. The reference point can be a significant extreme, a certain zero crossing, etc. The signal is tracked period by period in a synchronous way yielding *individual pitch periods*. This principle can be applied to both speech and music signals. In speech it may even be possible to derive the point of glottal closure from the reference point when the signal is undistorted.
3. *$T_0$ is defined as the elapsed time between two successive excitation cycles. Measurement starts at an arbitrary instant which is fixed according to external conditions, and ends when a complete cycle has elapsed.*
   This is an *incremental* definition. $T_0$ still equals the length of an individual period, but no longer from the production point of view, since the definition has nothing to do with an individual excitation cycle. The synchronous method of processing is maintained, but the phase relations between the laryngeal waveform and the markers, i. e., the pitch period delimiters at the output of the algorithm, are lost. Once a reference point in time has been established, it is kept as long as the measurement is correct and the signal remains cyclic, for instance as long as voicing continues. If this synchronization is interrupted, the reference point is lost, and the next reference point may be completely different with respect to its position within an excitation cycle.

4a. *$T_0$ is defined as the average length of several periods. The way in which averaging is performed, and how many periods are involved, is a matter of the individual algorithm.*
   This is the standard definition of $T_0$ for any PDA that applies stationary short-term analysis, including the implementations of frequency-domain PDAs. Well-known methods, such as cepstrum [10.10] or autocorrelation [10.11] approaches follow this definition. The pertinent frequency-domain definition reads as follows.

4b. *$F_0$ is defined as the fundamental frequency of an (approximately) harmonic pattern in the (short-term) spectral representation of the signal. It depends on the particular method in which way $F_0$ is calculated from this pattern.*



Amazon EX1010 Page 212

The perception point of view of the problem leads to a different definition of pitch [10.5]:

5. *$F_0$ is defined as the frequency of the sinusoid that evokes the same perceived pitch (residue pitch, virtual pitch, etc.) as the complex sound that represents the input speech signal.*
   Above all, this definition is a long-term definition [10.12]. Pitch perception theories were first developed for stationary complex sounds with constant $F_0$. The question of the behavior of the human ear with respect to short-term perception of time-variant pitch is not yet fully understood. The difference limen for $F_0$ changes, for instance, goes up by at least an order of magnitude when time-variant stimuli are involved [10.13, 14]. In practice even such PDAs that claim to be perception oriented [10.15, 16] enter the frequency domain in a similar way as in definition 4b, i. e., by some discrete Fourier transform (DFT) with previous time-domain signal windowing.

Since the results of individual algorithms differ according to the definition they follow, and since these five definitions are partly given in the time (or lag) domain and partly in the frequency domain, it is necessary to reestablish the relation between the time- and frequency-domain representations of pitch,

$$F_0 = 1/T_0 \tag{10.1}$$

in such a way that, whenever a measurement is carried out in one of these domains, however $T_0$ or $F_0$ is defined there, the representation in the other domain will always be established by this relation.

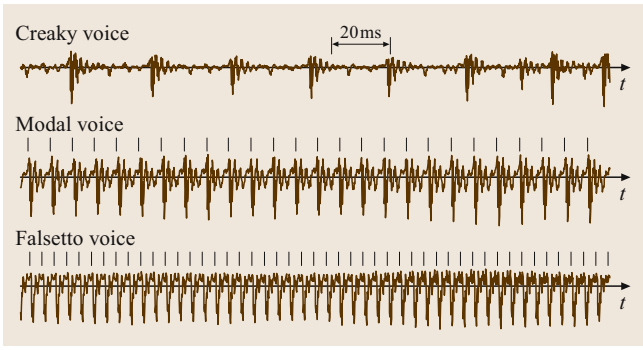



**Fig. 10.2** Speech signal excitation with different voice registers (male speaker, vowel [ɛ])

## 10.1.2 Why is the Problem Difficult?

Literally hundreds of methods for pitch determination have been developed. None of them has been reported to be error free for any signal, application, or environmental condition.

At first glance the task appears simple: one just has to detect the fundamental frequency of a quasiperiodic signal. When dealing with speech signals, however, the assumption of (quasi)periodicity is often far from reality. For a number of reasons, the task of pitch determination must be counted among the most difficult problems in speech analysis.

- In principle, speech is a nonstationary process; the momentary position of the vocal tract may change abruptly at any time. This leads to drastic variations in the temporal structure of the signal, even between subsequent pitch periods.
- In fluent speech there are voiced segments that last only a few pitch periods [10.17].
- Due to the flexibility of articulatory gestures and the wide variety of voices, there exist a multitude of possible temporal structures. Narrowband formants at low harmonics (especially at the second or third harmonic) are a particular source of trouble.
- For an arbitrary speech signal uttered by an unknown speaker, the fundamental frequency can vary over a range of almost four octaves (50–800 Hz). Especially for female voices, $F_0$ thus often coincides with the first formant (the latter being about 200–1400 Hz). This causes problems when inverse-filtering techniques are applied.
- The excitation signal itself is not always regular. Even under normal conditions, i. e., when the voice is neither hoarse nor pathologic, the glottal waveform exhibits occasional irregularities. In addition, the voice may temporarily fall into vocal fry or creak ([10.18, 19]; Fig. 10.2).
- Additional problems arise in speech communication systems, where the signal is often distorted or band limited (for instance, in telephone or even mobile-phone channels).

For music signals, the situation is comparable. The range of $F_0$ can be even wider than for speech. However, structural changes of the signal usually occur more slowly for music. The maximum speed at which a musical instrument can be played is about 10 notes per second so that a single note usually lasts at least 100 ms. For speech, on the other hand, 100 ms is already a lot of time which



Amazon EX1010 Page 213

can consist of three or more segments. An additional problem in music is that we may have to analyze polyphonic signals with several pitches present at the same time.

### 10.1.3 Categorizing the Methods

A PDA is defined as consisting of three processing steps: (a) the preprocessor, (b) the basic extractor, and (c) the postprocessor [10.1, 20]. The basic extractor performs the main task of converting the input signal into a series of pitch estimates. The task of the preprocessor is data reduction and enhancement in order to facilitate the operation of the basic extractor. The postprocessor (Sect. 10.5.4) is more application oriented. Typical tasks are error correction, pitch tracking, and contour smoothing, or visualization.

The existing PDA principles can be split into two gross categories when the input signal of the basic extractor is taken as a criterion. If this signal has the same time base as the original input signal, the PDA operates in the time domain. It will thus measure $T_0$ according to one of the definitions 1–3 above. In all other cases, somewhere in the preprocessor the time domain is left. Since the input signal is time variant, this is done by a short-term transform; and we will usually determine $T_0$ or $F_0$ according to definitions 4a,b or 5; in some cases definition 3 may apply as well. Accordingly, we have the two categories: time-domain PDAs, and short-term analysis PDAs. These will be discussed in the next two sections.

## 10.2 Short-Term Analysis PDAs

In any short-term analysis PDA a short-term (or short-time) transformation is performed in the preprocessor. The speech signal is split into a series of frames; an individual frame is obtained by taking a limited number of consecutive samples of the signal $s(n)$ from a starting point, $n = q$, to the ending point, $n = q + K$. The frame length $K$ (or $K + 1$) is chosen short enough so that the parameter(s) to be measured can be assumed approximately constant within the frame. On the other hand, $K$ must be large enough to guarantee that the parameter remains measurable. For most short-term analysis PDAs a frame thus requires two or three complete periods at least. In extreme cases, when $F_0$ changes abruptly, or when the signal is irregular, these two conditions are in conflict with each other and may become a source of error [10.21]. The frame interval $Q$, i. e., the distance between consecutive frames (or its reciprocal, the frame rate), is determined in such a way that any significant parameter change is documented in the measurements. 100 frames/s, i. e., $Q = 10$ ms, is a usual value.

The short-term transform can be thought of as behaving like a concave mirror that focuses all the information on pitch scattered across the frame into one single peak in the spectral domain. This peak is then determined by a peak detector (the usual implementation of the basic extractor in this type of PDAs). Hence this algorithm yields a sequence of average pitch estimates. The short-term transform causes the phase relations between the spectral domain and the original signal to be lost. At the same time, however, the algorithm loses much of its sensitivity to phase distortions and signal degradation.

Not all the known spectral transforms show the desired focusing effect. Those that do are in some way related to the (power) spectrum: correlation techniques, frequency-domain analysis, active modeling, and statistical approaches (Fig. 10.3). These methods will be discussed in more detail in the following.

### 10.2.1 Correlation and Distance Function

Among the correlation techniques we find the well-known short-term autocorrelation function (ACF)

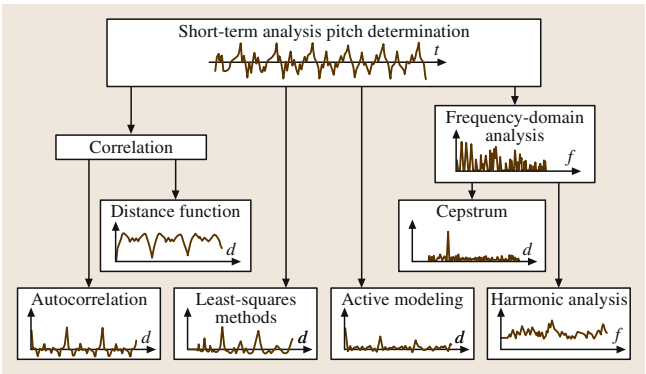



**Fig. 10.3** Methods of short-term analysis (short-time analysis) pitch determination. (Time $t$ and lag $d$ scales are identical; the frequency $f$ scale in the box 'Harmonic analysis' has been magnified)

usually given by

$$r(d,q) = \sum_{n=q}^{q+K-d} s(n)s(n+d)\,. \tag{10.2}$$

The autocorrelation function of a periodic signal exhibits a strong peak when the lag $d$ equals the period $T_0/T$ of the signal, $T$ being the time-domain sampling interval of the signal.

The autocorrelation PDA is among the oldest principles for short-term analysis PDAs. However, it tends to fail when the signal has a strong formant at the second or third harmonic. Therefore this technique became successful in pitch determination of band-limited speech signals when it was combined with time-domain linear or nonlinear preprocessing, such as center clipping or inverse filtering [10.22, 23].

The counterpart to autocorrelation is given by applying a distance function, for instance the average magnitude difference function (AMDF) [10.24, 25]:

$$\mathrm{AMDF}(d,q) = \sum_{n=q}^{q+K} |s(n) - s(n+d)|\,. \tag{10.3}$$

If the signal were strictly periodic, the distance function would vanish at the lag (delay time) $d = T_0/T$. For quasiperiodic signals there will be at least a strong minimum at this value of $d$. So, in contrast to other short-term PDAs where the estimate of $T_0$ or $F_0$ is indicated by a maximum whose position and value have to be determined, the minimum has an ideal target value of zero so that we only need to determine its position. For this reason, distance functions do not require (quasi)-stationarity within the measuring interval; they can cope with very short frames of one pitch period or even less. This principle is thus able to follow definition 3.

*Shimamura* and *Kobayashi* [10.26] combine ACF and AMDF in that they weight the short-term ACF with the reciprocal of the AMDF, thus enhancing the principal peak of the ACF at $d = T_0/T$. For the PDA they named YIN, *De Cheveigné* and *Kawahara* [10.27] start from a squared distance function,

$$D(d,q) = \sum_{n=q}^{q+K} [s(n) - s(n+d)]^2 \tag{10.4}$$

and normalize it to increase its values at low lags,

$$D'(d,q) = \frac{D(d,q)}{\frac{1}{d}\sum_{\delta=1}^{d} D(\delta,q)}\;;\; d>0 \tag{10.5}$$

with $D'(0) = 1$. In doing so, the authors were able to drop the high-frequency limit of the measuring range and to apply their PDA to high-pitched music signals as well. The normalized distance function is locally interpolated around its minima to increase the accuracy of the value of $D'$ at the minima and the pitch estimate at the same time.

Knowing that many errors arise from a mismatch during short-term analysis (which results in too few or too many pitch periods within a given frame), *Fujisaki* et al. [10.21] investigated the influence of the relations between the error rate, the frame length, and the actual value of $T_0$ for an autocorrelation PDA that operates on the linear prediction residual. The optimum occurs when the frame contains about three pitch periods. Since this value is different for every individual voice, a fixed-frame PDA runs nonoptimally for most situations. For an exponential window, however, this optimum converges to a time constant of about 10 ms for all voices. For a number of PDAs, especially for the autocorrelation PDA, such a window permits recursive updating of the autocorrelation function, i. e., sample-by-sample pitch estimation without excessive computational effort.

*Hirose* et al. [10.28] and *Talkin* [10.17] showed that the autocorrelation function can also be computed in a nonstationary way using a suitable normalization,

$$r(d,q) = \frac{\sum_{n=q}^{q+K} s(n)s(n+d)}{\sqrt{\left[\sum_{n=q}^{q+K} s^2(n)\right]\left[\sum_{n=q}^{q+K} s^2(n+d)\right]}}\,. \tag{10.6}$$

In Talkin's PDA a 7.5 ms frame is used; the effective frame length is of course 7.5 ms plus the longest pitch period in the measuring range.

*Terez* [10.29] applies a multidimensional embedding function and a scatter plot procedure derived from chaos theory. The underlying idea, however, is quite straightforward and leads to a distance function in a multidimensional state space. The problem is how to convert the one-dimensional speech signal into a multidimensional representation. In Terez's algorithm a vector is formed from several equally spaced samples of the signal,

$$\boldsymbol{s}(n) = \left[s(n)\; s(n+d)\; \cdots\; s(n+Nd)\right]^{\mathrm{T}}\,, \tag{10.7}$$

(where the frame reference point $q$ has been omitted here and in the following for sake of simplicity) whose components create an $N$-dimensional space, the state space. In Terez's algorithm, $N = 3$ and $d = 12$ samples gave the best results. If the signal is voiced, i. e., cyclic, the vector $\boldsymbol{s}$ will describe a closed curve in the state space as time proceeds, and after one pitch period it is



Amazon EX1010 Page 215

expected to come back near the starting point. We can thus expect the (Euclidian) distance

$$D(n,\,p) = \|\boldsymbol{s}(n) - \boldsymbol{s}(n+p)\| \tag{10.8}$$

generally to become a minimum when the trial period $p$ equals the true period $T_0/T$. If we compute $D(n,\,p)$ for all samples $s(n)$ within the frame and all values of $p$ within the measuring range and count the number of events, depending on $p$, where $D$ lies below a predetermined threshold, we arrive at a periodicity histogram that shows a sharp maximum at $p = T_0/T$.

As it develops the distance function $D$ for all samples of a frame, this PDA follows the short-term analysis principle. Yet one can think of running it with a comparatively short window, thus following definition 3.

The idea of using a multidimensional representation of the signal for a PDA (and VDA) dates back to the 1950s [10.1]. In 1964 *Rader* [10.30] published the *vector* PDA where he used the output signals from a filterbank (cf. *Yaggi* [10.31], Sect. 10.3.3) and their Hilbert transforms to form a multidimensional vector $\boldsymbol{s}(n, q)$. Rader then used the Euclidian distance between the vector at the starting point $n = q$ of the measurement and the points $q + p$ to set up a distance function which shows a strong minimum when p equals the true period $T_0/T$. This PDA follows definition 3 as well.

*Medan* et al. [10.32] present a PDA (they called the super-resolution PDA) that explicitly addresses the problem of granularity due to signal sampling and applies a short-term window whose length depends on the trial pitch period $p$ in that it takes on a length of exactly $2p$. A similarity function is formulated that expresses the relation between the two periods in the window,

$$\begin{aligned} \boldsymbol{s}(n, q) &= a \cdot \boldsymbol{s}(n+p, q) + e(n, q)\,; \\ & n = q, \cdots, q+p-1\,. \end{aligned} \tag{10.9}$$

Here, $a$ is a positive amplitude factor that takes into account possible intensity changes between adjacent periods. Equation (10.9) is optimized with respect to $a$ and the unknown period $p$ applying a least-squares criterion to minimize the error $e$,

$$\hat{p} = \operatorname{argmin}_{a,e} \left( \frac{\sum_{n=q}^{q+p-1} [s(n) - a s(n+p)]^2}{\sum_{n=q}^{q+p-1} s^2(n)} \right)\,. \tag{10.10}$$

This optimization finally results in maximization of the correlation term

$$\hat{p} = \operatorname{argmax} \left( \frac{\left[ \sum_{n=q}^{q+p-1} s(n) s(n+p) \right]^2}{\left[ \sum_{n=q}^{q+p-1} s^2(n) \right] \left[ \sum_{n=q}^{q+p-1} s^2(n+p) \right]} \right)\,. \tag{10.11}$$

This resembles Talkin's ACF approach [10.17] except that here the trial period $p$ determines the length of the window as well.

From (10.11) a pitch period estimate can only be derived as an integer number of samples. In a second pass, this estimate is refined (to yield the super-resolution) by expanding (10.11) for a fraction of a sample using linear interpolation.

### 10.2.2 Cepstrum and Other Double-Transform Methods

The sensitivity against strong first formants, especially when they coincide with the second or third harmonic, is one of the big problems in pitch determination. This problem is suitably met by some procedure for spectral flattening.

Spectral flattening can be achieved in several ways. One of them is time-domain nonlinear distortion, such as center clipping ([10.11, 22]; see previous section). A second way is linear spectral distortion by inverse filtering (e.g., [10.23]). A third way is frequency-domain amplitude compression by nonlinear distortion of the spectrum. This algorithm operates as follows: (1) short-term analysis and transformation into the frequency domain via a suitable discrete Fourier transform (DFT), (2) nonlinear distortion in the frequency domain, and (3) inverse DFT back into the time domain (which we will call the lag domain to avoid ambiguity).

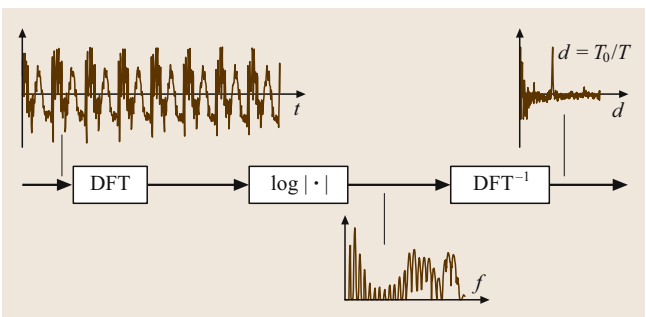



**Fig. 10.4** Cepstrum pitch determination. Signal: vowel [i], 48 ms



Amazon EX1010 Page 216

If we take the logarithm of the power spectrum as the frequency-domain nonlinear distortion, we arrive at the well-known cepstrum PDA ([10.10]; Fig. 10.4). Instead of the logarithmic spectrum, however, one can also compute the amplitude spectrum or its square root and transform it back [10.33, 34]. The inverse Fourier transform of the power spectrum gives the autocorrelation function. All these so-called double-transform techniques [10.33] lead to a lag-domain representation that exhibits a strong maximum at the lag $d = T_0/T$. The independent variable, lag (or *quefrency*, as it is called with respect to the cepstrum [10.10]), has the physical dimension of time, but as long as the phase relations are discarded by the nonlinear distortion, all values of $d$ refer to a virtual point $d = 0$ where we will always find a pitch pulse, and then the next one necessarily shows up at $d = T_0/T$.

Two members of this group were already mentioned: the autocorrelation PDA [10.11] and the cepstrum PDA [10.10]. It is well known that the autocorrelation function can be computed as the inverse Fourier transform of the power spectrum. Here, the distortion consists of taking the squared magnitude of the complex spectrum. The cepstrum, on the other hand, uses the logarithm of the spectrum. The two methods therefore differ only in the characteristics of the respective nonlinear distortions applied in the spectral domain. The cepstrum PDA is known to be rather insensitive to strong formants at higher harmonics but to develop a certain sensitivity to additive noise. The autocorrelation PDA, on the other hand, is insensitive to noise but rather sensitive to strong formants. Regarding the slope of the distortion characteristic, we observe the dynamic range of the spectrum being expanded by squaring the spectrum for the autocorrelation PDA, whereas the spectrum is substantially flattened by taking the logarithm. The two requirements – robustness against strong formants and robustness against additive (white) noise – are in some way contradictory. Expanding the dynamic range of the spectrum emphasizes strong individual components, such as formants, and suppresses wideband noise, whereas spectral flattening equalizes strong components and at the same time raises the level of low-energy regions in the spectrum, thus raising the level of the noise as well. Thus it is worth looking for other characteristics that perform spectral amplitude compression. *Sreenivas* [10.36] proposed the fourth root of the power spectrum instead of the logarithm. For larger amplitudes this characteristic behaves very much like the logarithm; for small amplitudes, however, it has the advantage of going to zero and not to $-\infty$. *Weiss* et al. [10.33] used the amplitude spectrum, i. e., the magnitude of the complex spectrum.

### 10.2.3 Frequency-Domain Methods: Harmonic Analysis

Direct determination of $F_0$ as the location of the lowest peak in the power or amplitude spectrum is unreliable and inaccurate; it is preferable to investigate the harmonic structure of the signal so that all harmonics contribute to the estimate of $F_0$. One way to do this is spectral compression combined with harmonic pattern matching, which computes the fundamental frequency as the greatest common divider of all harmonics. The power spectrum is compressed along the frequency axis by a factor of two, three etc. and then added to the original power spectrum. This operation gives a peak at $F_0$ resulting from the coherent additive contribution of the higher harmonics [10.35, 37]. Some of these PDAs stem from theories and functional models of pitch perception in the human ear [10.12, 15, 16].

The PDA described by *Martin* [10.35] (Fig. 10.5) modifies the harmonic pattern-matching principle in such a way that the computational effort for the spectral transform is minimized. The signal is first decimated to 4 kHz and then Fourier transformed by a 128 point fast Fourier transform (FFT). This yields a spectral resolution of about 30 Hz, which is sufficient to represent

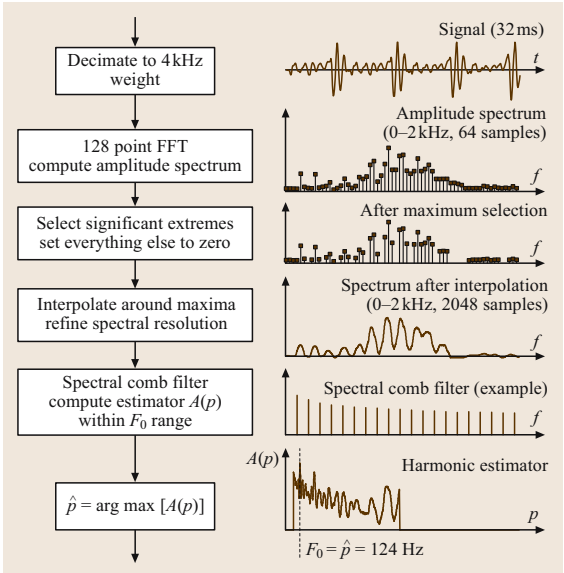



**Fig. 10.5** Frequency-domain PDA by harmonic compression and pattern matching (after *Martin* [10.35])



Amazon EX1010 Page 217

values of $F_0$ down to 60 Hz. The algorithm then enhances any spectral information that may pertain to a harmonic structure by preserving only those spectral samples that represent local maxima or their immediate neighbors and setting everything else to zero. To measure $F_0$ with sufficient accuracy, the spectral resolution is then increased to 1 Hz. A spectral comb filter, which is applied over the whole range of $F_0$, yields the harmonic estimation function $A(p)$; the value of $p$ where this function reaches its maximum is taken as the estimate for $F_0$. In a more-recent version of this PDA, *Martin* [10.38, 39] applies a logarithmic frequency scale for the computation of $A(p)$, which results in another substantial reduction of the computational effort and has the additional advantage that the relative accuracy of the PDA is now constant over the whole range of $F_0$.

Similar to *Martin*'s [10.38] PDA for speech, *Brown* [10.41] developed a frequency-domain PDA for music which uses a logarithmic frequency scale. In Brown's PDA the spacing on the frequency axis equals a quarter tone (about 3%), i. e., 1/24 of an octave. In such a scale that corresponds to a musical interval scale, a harmonic structure, if the timbre is unchanged, always takes on the same pattern regardless of the value of its fundamental frequency (Fig. 10.6). Consequently, a pattern-recognition algorithm is applied to detect such patterns in the spectrum and to locate their starting point corresponding to $F_0$. The patterns themselves depend on the kind of instrument analyzed and can be adjusted according to the respective instruments.

Special attention is given to the frequency resolution of the PDA. To apply a pattern-recognition method, patterns are expected to align with the semitone scale. This requires the frequency scale spacing of a quarter tone. To make sure that a harmonic shows up in one and only one frequency bin, we need a window length of 34 fundamental periods to satisfy the sampling theorem in the frequency domain. For $F_0 = 100$ Hz this would give a window of 340 ms, which is far beyond reality for speech and even excessive for music. However, if we adapt the window-length requirement to the fundamental frequency to be determined, we would need 34 periods at any $F_0$ to be measured, which results in much shorter windows for higher-frequency bins. For the DFT, this leads to a window whose length is inversely proportional to the bin's frequency. If the spectral values are computed individually, both an individual time-domain window for each frequency bin and unequal spacing of the frequency bins are possible. *Brown* and *Puckette* [10.40] showed that a fast Fourier transform can be applied if its kernel is slightly modified. The PDA by *Medan* et al. [10.32] is a time-domain counterpart to this approach.

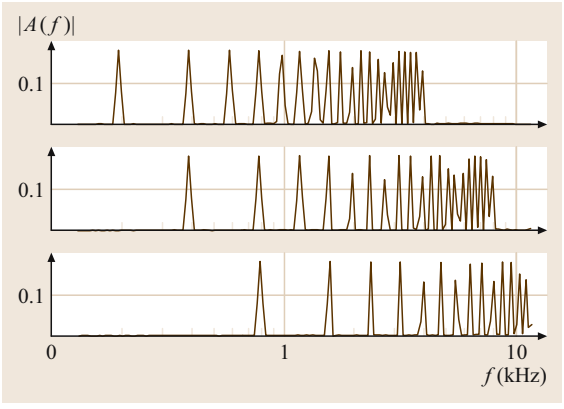



**Fig. 10.6** Harmonic patterns in log frequency scale (after *Brown* and *Puckette* [10.40])

As the accuracy of this PDA was not sufficient to determine $F_0$ for instruments that can vary their frequencies continuously (such as string or wind instruments or a human voice), and as the required window length was excessive even for music, a PDA with a 25 ms window was developed [10.40] whose frequency resolution was refined using a phase-change technique. This technique is based on the instantaneous-frequency approach by *Charpentier* [10.42] (see below) who used the short-term phase spectrum at two successive samples in time to determine the frequencies of the individual harmonics without needing spectral interpolation. When a Hann window is used to weight the time signal, the time shift of one sample can be recursively computed in the frequency domain without needing another DFT.

*Lahat* et al. [10.43] transfer the autocorrelation method into the frequency domain. The amplitude spectrum is passed through a bank of 16 spectral filters (*lifters*), which cover the measuring range of $F_0$. At the output of each lifter a frequency domain autocorrelation function is calculated covering the respective range of each lifter. The estimate for $F_0$ is then determined as the location of the maximum of that function and refined by interpolation.

For harmonic analysis it is often convenient to estimate the number of harmonics, i. e., the order of a harmonic model, simultaneously with the fundamental frequency. For instance, *Doval* and *Rodet* [10.44] apply such a procedure for a PDA with an extremely wide measuring range (40–4000 Hz) for music signals. The algorithm is based on a harmonic-matching procedure using a maximum-likelihood criterion. To obtain



Amazon EX1010 Page 218

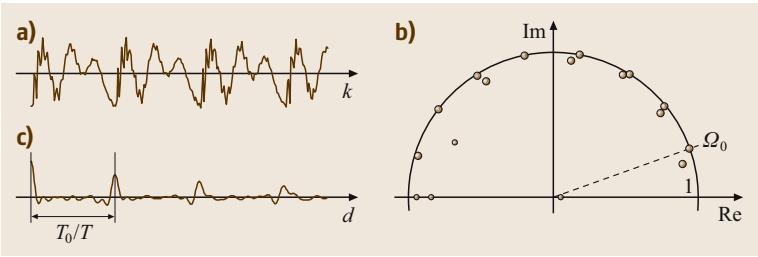



**Fig. 10.7a–c** PDA with active modeling. **(a)** Signal: 32 ms, vowel [e], male voice; **(b)** zeros of the 41-st-order LP polynomial in the $z$ plane (upper half; sampling frequency reduced to 2 kHz); **(c)** reconstructed impulse response with zero phase and equal amplitude of all partials

a rapid initial estimate, the measuring range is split into subranges that have an equal number of partials in the frequency range for which the spectrum is analyzed. The initial estimate is obtained by selecting the subrange that is most likely to contain the optimal estimate for $F_0$. For the final step of refining the estimate, only this subrange is further processed.

The principle of instantaneous frequency (IF) was introduced into pitch determination by *Charpentier* [10.42]. Instantaneous frequency is defined as the time derivative of the instantaneous phase,

$$\dot{\varphi}(t) := \frac{\mathrm{d}\varphi}{\mathrm{d}t} \quad \text{for } s(t) = a(t)\exp[\mathrm{i}\omega(t)t]\,, \tag{10.12}$$

where $a(t)$ is the instantaneous amplitude. The short-term Fourier transform can be viewed as a set of bandpass filters as follows,

$$S(f,t) = \int_{-\infty}^{\infty} s(\tau)w(t-\tau)\mathrm{e}^{-2\pi\mathrm{i}f\tau}\,\mathrm{d}\tau \tag{10.13}$$

$$= \mathrm{e}^{-2\pi\mathrm{i}ft}\int_{-\infty}^{\infty} s(\tau)w(t-\tau)\mathrm{e}^{-2\pi\mathrm{i}f(t-\tau)}\,\mathrm{d}\tau$$

$$F(f,t) = \mathrm{e}^{2\pi\mathrm{i}ft}S(f,t)\,. \tag{10.14}$$

Here, the signal $F$ is the output of the *bandpass* centered around the frequency $f$. The IF for this signal becomes

$$\dot{\varphi}(f,t) = \frac{\partial}{\partial t}\arg[F(f,t)]\,. \tag{10.15}$$

There are different ways to effectively compute the IF from the discrete short-term spectrum [10.42,45,46]. The bandpass filters have a certain bandwidth that depends on the time-domain window applied [10.42] and extends over more than one DFT coefficient. If we now compute the IF for each frequency bin of the DFT spectrum of a voiced speech signal, the IFs of bins adjacent to a strong harmonic tend to cluster around the true frequency of this harmonic, and so it is possible to enhance the harmonics in the spectrum if the bins are regrouped according to their respective IFs, thus forming the so-called IF amplitude spectrum. *Abe* et al. [10.45] transform the IF amplitude spectrum back into the time domain, thus obtaining a representation similar to that of a double-spectral-transform PDA. *Nakatani* and *Irino* [10.46] define a spectral dominance function that suppresses insignificant information in the IF amplitude spectrum and derive $F_0$ by harmonic matching of this dominance function.

### 10.2.4 Active Modeling

Linear prediction (LP) is usually applied to estimating the transfer function of the vocal tract. If a high-order LP filter is applied to a voiced speech signal, however, its poles will match the individual harmonics of the signal rather than the resonances of the vocal tract. A PDA based on this principle was designed by *Atal* (unpublished; see [10.47], or [10.1]). The algorithm operates as follows (Fig. 10.7):

- After decimation to 2 kHz, the signal is analyzed with a 41-st-order LP filter using the covariance method of linear prediction. The high order guarantees that even at the low end of the $F_0$ range, i. e., at $F_0 = 50$ Hz, two complex poles are available for each harmonic. Each complex pole pair represents an exponentially decaying (or rising) oscillation.
- To eliminate phase information, all residues at the pole locations in the $z$ plane are set to unity. The pertinent computation can be avoided when the locations of the poles are explicitly determined.
- The impulse response of the filter now supplies a waveform for the estimation of $T_0$ (Fig. 10.7c). When the poles are explicitly available, it is sufficient to determine and to sum up the impulse responses of the individual second-order partial filters. This method has the advantage that the sampling frequency of the impulse response – and with it the measurement accuracy – can easily be increased. In addition, poles that are situated outside or far inside the unit circle can be modified or excluded from further processing.



Amazon EX1010 Page 219

*Arévalo* [10.48] showed that this PDA is extremely robust to noise and that one can also use it with short frame lengths so that it matches definition 3.

### 10.2.5 Least Squares and Other Statistical Methods

The first statistical approach in pitch determination is based on a least-squares principle. Originally this approach was based on a mathematical procedure to separate a periodic signal of unknown period $T_0$ from Gaussian noise within a finite signal window [10.49]. It computes the energy of the periodic component at a trial period $\tau$ and varies $\tau$ over the whole range of $T_0$. The value of $\tau$ that maximizes the energy of the periodic component for the given signal is then taken as the estimate of $T_0$. *Friedman* [10.50] showed that this PDA has a trivial maximum when $\tau$ equals the window length $K$, and developed a work-around.

With respect to robustness, the least-squares PDA behaves like the autocorrelation principle: it is extremely robust against noise but somewhat sensitive to strong formants. However, there is no algorithmic shortcut so that an order of $K^2$ operations are needed to compute the estimate for a frame. So this PDA was slower than its counterparts that can make use of the FFT; hence this principle was not further pursued until more powerful computers became available.

The method was revived with the upcoming of the sinusoidal model of speech [10.2]. The continuous speech signal $s(n)$ is modeled as a sum of sinusoids with time-varying amplitudes, frequencies, and phases. Within a short-term frame, these parameters can be assumed constant,

$$s(n) = \sum_{m=1}^{M} S_m \ \exp\left(\mathrm{i}\Omega_m n + \varphi_m\right) \ . \tag{10.16}$$

The parameters of this model are estimated from the peaks within the short-term Fourier spectrum of the frame. This can be converted into a PDA [10.51] when the sinusoidal representation in (10.16), whose frequencies are generally *not* harmonics of a fundamental, is matched against a harmonic model,

$$u(n) = \sum_{k=1}^{K} U_k \ \exp\left(\mathrm{i}k\Omega_0 n + \psi_k\right) \ . \tag{10.17}$$

Starting from the difference between $s(n)$ and $u(n)$, the match is done using a modified least-squares criterion, which finally results in maximizing the expression with respect to the trial (angular) fundamental frequency $p$,

$$\varrho(p) = \sum_{k=1}^{K(p)} U(kp) \times \left\{ \max_{\Omega_m \in L(kp)} \left[ S_m D(\Omega_m - kp) \right] - \frac{1}{2} U(kp) \right\} \ . \tag{10.18}$$

Like the model of virtual-pitch perception [10.4], this criterion takes into account near-coincidences between a harmonic $kp$ and the (angular) frequency $\Omega_m$ of the respective component of the sinusoidal model, and it defines a lobe $L$ of width $p$ around each harmonic and the corresponding weighting function

$$D(\Omega - kp) = \frac{\sin[2\pi(\Omega - kp)/p\,]}{(\Omega - kp)/p} \tag{10.19}$$

within the lobe, and zero outside. The lobe becomes narrow for low values of $p$ and broader for higher values. If there are several components $\Omega_m$ within a lobe, only the largest is taken. The amplitude estimates $U(kp)$ are derived from a smoothed Fourier or LP spectrum of the frame. The measurement may be confined to a subband of the signal, e.g., to 2 kHz.

Both the sinusoidal model and the PDA have been applied to speech and music signals.

For pitch detection in noisy speech, third-order statistics are occasionally applied. One such PDA was developed by *Moreno* and *Fonollosa* [10.52]. Their PDA applies a special third-order cumulant,

$$C(0, d) := \sum_{n} s(n)\, s(n)\, s(n + d) \ , \tag{10.20}$$

which tends to vanish for noises with symmetrical distribution, such as Gaussian noise. It also tends to vanish for voiceless fricatives, as *Wells* [10.53] discovered for his VDA. If the signal is periodic, the cumulant $C$ is also periodic, but one cannot expect a maximum to occur at $d = 0$. This PDA thus treats $C(0, d)$ like an ordinary signal, takes the autocorrelation function, and derives $T_0$ therefrom in the same way as it is done in an ordinary autocorrelation PDA. The algorithm was tested with speech in various additive noises at various signal-to-noise ratios (SNRs) against an autocorrelation PDA (without any pre- or postprocessing) and found superior, especially when noise levels were high.

The PDA by *Tabrikian* et al. [10.54] determines the parameters of a harmonic-plus-noise model by maximizing a log-likelihood measure, i. e., the unknown fundamental frequency, the spectral amplitudes of the



Amazon EX1010 Page 220

harmonics, and the variance of the noise, which is modeled to be Gaussian. It then performs pitch tracking over consecutive frames using a method based on the maximum a posteriori probability. The authors tested their PDA under extreme noise conditions and found that it worked even at SNRs of −5 dB and worse. A similar algorithm was developed by *Godsill* and *Davy* [10.55] for music signals.

### 10.2.6 Concluding Remarks

Short-term analysis PDAs provide a sequence of average pitch estimates rather than a measurement of individual periods. They are not very sensitive to phase distortions or to the absence of the first partial. They collect information about pitch from many features and (mostly) from several periods of the signal. They are thus robust against additive noise. Some of them still work at SNRs of 0 dB or worse. On the other hand, they are sensitive when the signal does not fulfil their basic requirement, i. e., periodicity. Rapid within-frame changes of $F_0$ of irregularly excited signals (e.g., laryngealizations) lead to failure.

One advantage of this principle that is not always explicitly mentioned is the ability to give rather accurate estimates and to overcome measurement granularity due to signal sampling. To decrease computational complexity, many of these PDAs perform some moderate low-pass filtering and/or decrease the sampling frequency in the first step and thus increase the granularity. Once a crude estimate is available, it can be refined via a local interpolation routine, which is frequently implemented. This is most evident in active modeling (Sect. 10.2.4) where the impulse response of the model filter can be generated with an arbitrarily high sampling frequency independently from the sampling frequency of the signal. However, any other representation from which pitch is derived – ACF, AMDF, cepstrum, etc. – can be treated like a signal and can be locally upsampled, e.g., via a standard multirate finite impulse response (FIR) filter, to increase measurement accuracy. The evaluation by *McGonegal* et al. [10.56] showed that an increased accuracy is honored by the human ear when listening to synthetic speech generated with such a pitch contour.

## 10.3 Selected Time-Domain Methods

This category of PDAs is less homogenous than that of the short-term analysis methods. One possibility to group them is according to how the data reduction is distributed between the preprocessor and the basic extractor, and we find most of these PDAs between two extremes.

- Data reduction is done in the preprocessor. In the extreme case, only the waveform of the first harmonic is offered to the basic extractor. The basic extractor processes this harmonic and derives pitch from it.
- Data reduction is done in the basic extractor, which then has to cope with the whole complexity of the temporal signal structure. In the extreme case, the preprocessor is totally omitted. The basic extractor investigates the temporal structure of the signal, extracts some key features, and derives the information on pitch therefrom.

A third principle is situated somewhere in the middle of these extremes. Temporal structure simplification performs a moderate data reduction in the preprocessor but preserves the harmonic structure of the signal.

Time-domain PDAs are principally able to track the signal period by period. At the output of the basic extractor we usually obtain a sequence of period boundaries (pitch *markers*). Since the local information on pitch is taken from each period individually, time-domain PDAs are sensitive to local signal degradations and are thus less reliable than most of their short-term analysis counterparts. On the other hand, time-domain PDAs may still operate correctly even when the signal itself is aperiodic (but still cyclic), in speech for instance due to temporary voice perturbation or laryngealization.

Most time-domain PDAs, especially those which follow definitions 2 and 3, were developed before the 1990s. With the introduction of time-domain pitch modification methods [10.57], research in this area concentrated on high-precision algorithms for determination of the instant of glottal closure. This issue will be discussed in Sect. 10.7.1.

### 10.3.1 Temporal Structure Investigation

A pitch period in speech is the truncated response of the vocal tract to an individual glottal impulse. Since the vocal tract behaves like a lossy linear system, its impulse response consists of a sum of exponentially damped oscillations. It is therefore to be expected that



Amazon EX1010 Page 221

the magnitude of the significant peaks in the signal is greater at the beginning of the period than versus the end. Appropriate investigation of the signal peaks (maxima and/or minima) leads to an indication of periodicity.

There are some problems with this approach, however. First, the frequencies of the dominant damped waveforms are determined by the local formant pattern and may change abruptly. Second, damping of the formants, particularly of a low first formant, is often quite weak and can be hidden by temporary changes of the signal level due to articulation. Third, if the signal is phase distorted, different formants may be excited at different points in time. These problems are solvable but lead to complex algorithmic solutions investigating a great variety of temporal structures.

The usual way to carry out the analysis is the following [10.58].

- Do a moderate low-pass filtering to remove the influence of higher formants.
- Determine all the local maxima and minima in the signal.
- Exclude those extremes that are found to be insignificant until one significant point per period is left; this point will become the local pitch marker.
- Reject obviously wrong markers by local correction.

Many individual (and heuristic) solutions have been developed, but they cannot be reviewed here for lack of space. For more details, the reader is referred to the literature [10.1].

### 10.3.2 Fundamental Harmonic Processing

$F_0$ can be detected in the signal via the waveform of the fundamental harmonic. If present in the signal, this harmonic is extracted from the signal by extensive low-pass filtering in the preprocessor. The basic extractor can then be relatively simple. Figure 10.8 shows the principle of three basic extractors: zero-crossings analysis as the simplest one, nonzero threshold analysis, and finally threshold analysis with hysteresis. The zero-crossings analysis basic extractor sets a marker whenever the zero axis is crossed with a defined polarity. This requires that the input waveform has two and only two zero crossings per period. The threshold analysis basic extractor sets a marker whenever a given nonzero threshold is exceeded. When operating with hysteresis, the marker is only set when a second (lower) threshold is crossed in the opposite direction. This more-elaborate device requires less low-pass filtering in the preprocessor.

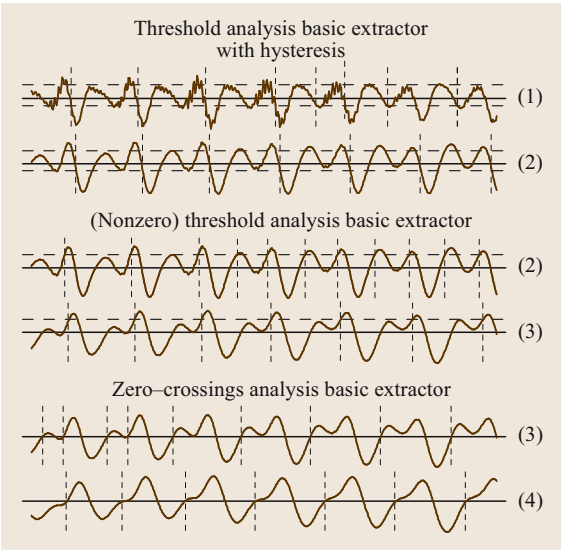



**Fig. 10.8** Example of the performance of basic extractors for fundamental harmonic extraction in speech. Signals: (1) original (vowel [i], 32 ms), (2) low-pass filtered at 6 dB/oct, (3) low-pass filtered at 12 dB/oct, and (4) low-pass filtered at 18 dB/oct. The signal is such that success and failure are displayed at the same time

The requirement for extensive low-pass filtering is a severe weak point of this otherwise fast and simple principle when applied to speech signals. In a number of applications, however, such as voice quality measurement or the preparation of reference elements for time-domain speech synthesis, where the signals are expected to be clean, the use of a PDA applying first-partial processing may be advantageous. *Dologlou* and *Carayannis* [10.59] proposed a PDA that overcomes a great deal of the problems associated with the low-pass filter. An adaptive linear-phase low-pass filter that consists of a variable-length cascade of second-order filters with a double zero in the $z$ plane at $z = -1$ is applied. These filters are consecutively applied to the input signal; after each iteration the algorithm tests whether the higher harmonics are sufficiently attenuated; if they are, the filter stops. $T_0$ is then derived from the remaining first partial by a simple maximum detector. Very low-frequency noise is tolerable since it barely influences the positions of the maxima.

### 10.3.3 Temporal Structure Simplification

Algorithms of this type take some intermediate position between the principles of structural analysis and



Amazon EX1010 Page 222

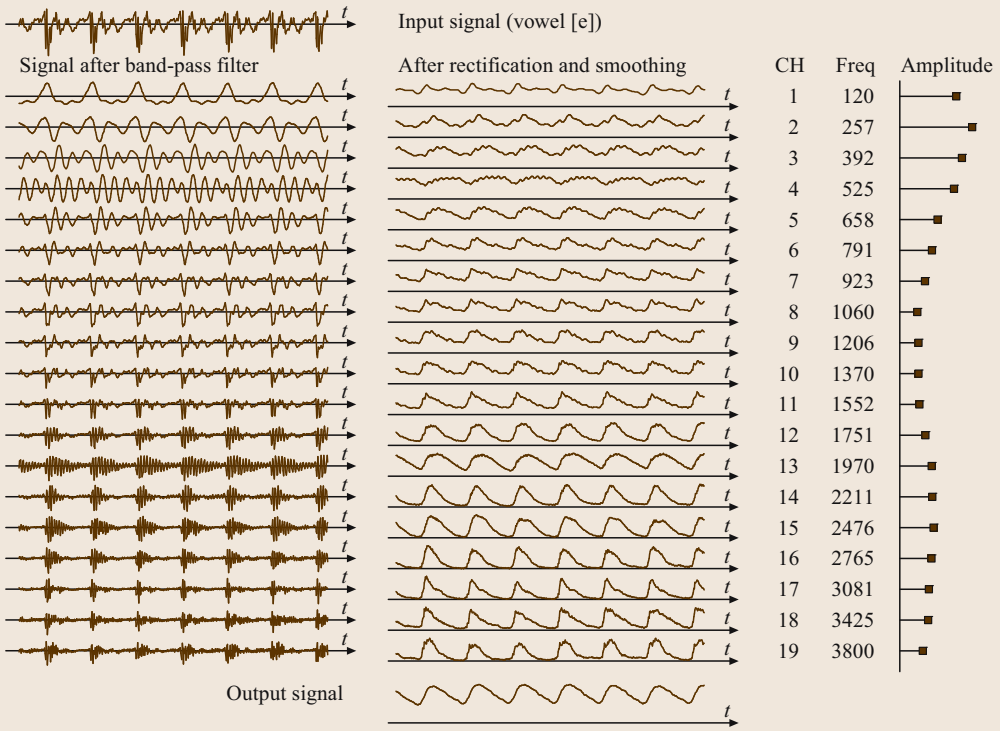



**Fig. 10.9** Example PDA by *Yaggi* (1962 [10.31]). The signal is split into 19 subbands. In each channel (CH) the filtered signal is rectified and smoothed; the weighted outputs of the channels are added, and pitch markers are derived from the resulting signal via maximum detection



fundamental harmonic extraction. The majority of these algorithms follow one of two principles: (1) inverse filtering, and (2) epoch detection. Both of these principles deal with the fact that the laryngeal excitation function has a temporal structure that is much simpler and more regular than the temporal structure of the speech signal itself, and both methods when they work properly are able to follow definition 1 if the signal is not grossly phase distorted.

The inverse filter approach cancels the transfer function of the vocal tract and thus reconstructs the laryngeal excitation function. If one is interested in pitch only and not in the excitation function itself, a crude approximation of the inverse filter is sufficient. For instance, we can confine the analysis to the first formant [10.60].

The second principle, epoch extraction [10.61], is based on the fact that at the beginning of each laryngeal pulse there is a discontinuity in the second derivative of the excitation function. Usually this discontinuity cannot be reliably detected in the speech signal because of phase distortions that occur when the waveform passes the vocal tract. The signal is thus first phase shifted by 90° (by applying a Hilbert transform). The squares of the original and the phase-shifted signals are then added to yield a new signal that shows a distinct peak at the time when the discontinuity in the excitation function occurs. In principle this yields the instantaneous amplitude of the complex analytic signal constructed from the original signal as its real part and the phase-shifted signal as its imaginary part.

The original method [10.61] works best when the spectrum of the investigated signal is flat to some extent. To enforce spectral flatness, the analyzed signal can be band limited to high frequencies well above the narrowband lower formants. Another way is to analyze the LP residual or to filter the signal into subbands.

One prototype of these algorithms, which never became widely known, was developed by *Yaggi* [10.31]. It splits the signal into 19 subbands and subsequently rectifies and smoothes the signal in each channel so that the envelope is extracted. The individual channels are then added, and the individual periods are derived from

the resulting signal (Fig. 10.9). Another prototype is the PDA by *Dolanský* [10.62] that models the envelope of the pitch period by a decaying exponential function (in analog technology) and sets a marker whenever the signal exceeds the modeled envelope, resetting the envelope at the same time.

Both the inverse filter approach and the epoch detection principle have one weak point, which frequently arises with female voices. When $F_0$ is high, it may coincide with the first formant $F1$. In this case the signal becomes nearly sinusoidal since we have something like a double pole *glottal formant* and $F1$ at the same frequency) in the overall transfer function. If an inverse filter is not blocked in this case, it removes the fundamental harmonic from the signal and brings the PDA to failure. For epoch detection, we know that the envelope of a sinusoid is a constant ($\cos^2 x + \sin^2 x = 1$) and does not show any discontinuity. Hence these algorithms need a work-around for low values of $F1$.

This drawback was overcome by the finding that the global statistical properties of the waveform change with glottal opening and closing as well. We will come back to this issue in Sect. 10.7.1.

Structural analysis of the signal itself or of some simplified representation, especially when many possible structures have to be reviewed, is a good candidate for self-organizing, nonlinear pattern-recognition methods, i. e., for artificial neural networks. Such a PDA for speech was introduced by *Howard* and *Walliker* [10.63]. The speech signal is divided into nine subbands with a subsequent half-wave rectification and second-order linear smoothing in each channel. The underlying idea is to obtain a representation similar to that in a wide-band spectrogram. The basic extractor consists of a four-layer perceptron structure, the input layer consisting of 41 successive samples in each subband. Two hidden layers with 10 units each and a fully connected network are followed by a one-unit output layer, which is intended to yield an impulse when the network encounters a signal structure associated to the instant of glottal closure. The network is trained using (differentiated) output signals of a laryngograph as reference data. Such a structure has the advantage that it can be based upon several features occurring at different instants during a pitch period.

### 10.3.4 Cascaded Solutions

Among the many possibilities of such solutions, one is of particular interest here: the cascade of a robust short-term PDA and an accurate but less-robust time-domain PDA. Such an algorithm is further described in Sect. 10.7.1.



## 10.4 A Short Look into Voicing Determination

The task of voicing determination of speech signals may be split up into two subtasks: (1) a decision of whether or not a voiced excitation is present and (2) a decision of whether or not a voiceless excitation is present. If neither of these excitations is active, the current segment represents pause or silence; if both excitations are simultaneously active, we speak of mixed excitation. The two features *voiced* and *voiceless* are binary unless they occur simultaneously. In segments with mixed excitation the degree of voicing – for instance, the energy ratio of the voiced and voiceless excitations – may play a role, although this feature is rarely exploited.

Most voicing determination algorithms (VDAs) thus apply decisions. VDAs exploit almost any elementary speech signal parameter that may be computed independently of the type of input signal: energy, amplitude, short-term autocorrelation coefficients, zero-crossings count, ratio of signal amplitudes in different subbands or after different types of filtering, linear prediction error, or the salience of a pitch estimate. According to the method applied, VDAs can be grouped into three major categories: (1) simple threshold analysis algorithms, which exploit only a few basic parameters [10.64]; (2) more-complex algorithms based on pattern recognition methods; and (3) integrated algorithms for both voicing and pitch determination.

In this section, we distinguish between *voiceless* and *unvoiced*. Unvoiced means that a frame can be anything but voiced, i. e., it can be voiceless or silence. Voiceless means that voiceless excitation is present so that the frame is neither voiced nor silence. We will not review such algorithms that distinguish between speech and nonspeech – many of such algorithms have been developed for other applications, such as voice over Internet protocol (IP) or bandwidth reduction in mobile telephony (see, for instance, *Davis* et al. [10.65], for a survey). The basic task of the VDA in the context of pitch determination is to decide whether a frame or signal segment is voiced (and thus subject to pitch determination) or unvoiced.

Amazon EX1010 Page 224

### 10.4.1 Simultaneous Pitch and Voicing Determination

A number of PDAs – usually pertaining to the short-term analysis category – permit estimation the salience of their results without having to know the actual value of the pitch estimate. This is always possible when the amplitude of the significant maximum or minimum at $T_0$ or $F_0$ in the basic extractor of the PDA can be compared to a reference value. As an example, the ratio of the values of the autocorrelation function at $d = T_0$ and at $d = 0$ (the latter equalling the signal energy) gives a direct measure of the degree of periodicity of the signal. It is dangerous, however, to rely on this feature alone. Of course, it is correct (and almost trivial) to state that pitch can exist only when the signal is voiced. However, this statement cannot be simply reversed; i. e., we cannot say that a segment is unvoiced because pitch is not existent (or not measurable). The corresponding PDA may momentarily fail, or the signal may be voiced but irregular ([10.47, 66]; see also Fig. 10.2 or Sect. 10.1.2). Amplitude changes and especially small intraframe changes of $F_0$ severely degrade the *salience* of the pitch estimate [10.36]. It is thus at least necessary to check adjacent frames before making a decision [10.10]. In this respect, such a VDA behaves very much like a median smoother in pitch determination.

In principle, these VDAs do not make a voiced–unvoiced discrimination; rather they check for the presence of a (sufficient but not absolutely necessary) condition for a voiced signal. An improvement is to be expected when such criteria are only used for declaring a frame as voiced, and when the decision to declare it as unvoiced is based on additional criteria [10.67].

The VDA by *Lobanov* [10.68] avoids this problem, although it is based on a similar principle. A voiceless segment of speech represents a stochastic signal which is continuously excited. In contrast, the excitation of a voiced signal is confined to a few instants per period; major parts of the pitch period are composed of exponentially decaying oscillations, and adjacent samples of the signal are highly correlated. This contrast of a highly stochastic versus a highly deterministic signal is preserved even when a voiced signal becomes irregular or aperiodic. Lobanov's VDA exploits this feature by computing the Hilbert transform of the speech signal, combining the original signal and its Hilbert transform to yield the complex analytic signal, and plotting the momentary amplitude and phase of the analytic signal in the so-called phase plane. For voiced frames the analytic signal will describe a closed curve. During unvoiced segments, where the signal and its Hilbert transform are much less correlated, the curve will touch almost any point in the phase plane within a short interval. In Lobanov's algorithm the phase plane is crudely quantized, and the algorithm simply counts the number of points which have been touched within a given frame.

*Talkin's* PDA [10.17] integrates the VDA into the postprocessor that applies dynamic programming. Among the various estimates for $T_0$ to be tracked, there is always a candidate *unvoiced*, which is selected when it lies on the optimal path (Sect. 10.5.4).

*Ahmadi* and *Spanias* [10.67] present an improved VDA module within an implementation of the cepstrum PDA [10.10] for telephone-bandwidth speech. An utterance is processed in two passes. The first pass, covering the whole utterance, is to derive gross initial thresholds for a rough voiced–unvoiced decision. Distributions are taken for the relative amplitude of the main cepstral peak, the relative zero-crossings rate, and normalized signal energy. The medians of these distributions serve as initial thresholds for the decisions to be made in the second pass. A frame is roughly declared unvoiced if its energy and cepstral peak amplitudes are below and its zero crossings rate is above the respective threshold. Frames are declared voiced according to their cepstral peak amplitudes and a continuity criterion. The algorithm was evaluated on data from the TIMIT corpus; reference values were obtained using the PDA by *McAulay* and *Quatieri* [10.51] with visual inspection of uncertain frames. For clean speech, voiced-to-unvoiced and unvoiced-to-voiced errors together were about 1.5%.

*McAuley* and *Quatieri* [10.51] use their harmonic-model PDA (Sect. 10.2.5) to incorporate a VDA. It is based on the energy ratio between the harmonic energy and the energy of the nonharmonic part of the signal (the *noise*) which consists of everything not captured by the harmonic structure. Frames for which this ratio is above 10 dB are certainly voiced, while those for which the ratio is below 4 dB are certainly unvoiced.

*Fisher* et al. [10.69] start from a generalized log likelihood measure that is separately and independently evaluated for the two hypotheses that the frame is (1) voiced, or that it is (2) unvoiced. The measure for the frame being voiced is based on the aforementioned ratio between harmonic and nonharmonic energy, whereas the measure for unvoiced is based on a model of colored Gaussian noise. The hypothesis with the higher likelihood value wins for each frame; a dynamic-programming postprocessor (Sect. 10.5.4) integrates the VDA into the PDA which is also based on the harmonic-plus-noise model.



Amazon EX1010 Page 225

### 10.4.2 Pattern-Recognition VDAs

One of the motivations for applying a pattern-recognition algorithm in a VDA was the wish to get away from the conjunction of voicing determination and pitch determination [10.70]. The VDA by *Atal* and *Rabiner* [10.70] (the first of a series of VDAs developed at Bell Laboratories in the late 1970s) uses a statistical classifier and is based on five parameters: the signal energy, the zero-crossing rate, the autocorrelation coefficient at a delay of one sample, the first predictor coefficient of a 12 pole LP analysis, and the energy of the normalized prediction error. For a given environmental condition the algorithm works well, but it is rather sensitive to environmental changes, e.g., from high-quality speech to a telephone channel [10.47].

The usual classification problems in speech recognition, where we have to cope with a large number of different classes, require that the input parameters form specific clusters in the parameter space, which are then separated by the classifier. In contrast, the voicing determination problem has at most four categories (silence, voiced, voiceless, and mixed) and the distribution of the patterns in the parameter space is rather diffuse. It is thus appropriate to concentrate the VDA on patterns that are situated at or near the boundaries between the different categories in the parameter space. Such a VDA was developed by *Siegel* and *Bessey* [10.66]. For some applications, such as high-quality analysis–synthesis systems, incorporation of a mixed excitation source is desirable: (1) for voiced fricatives, and (2) for some high vowels, which tend to become partly devoiced in connected speech [10.71]. Siegel and Bessey further found that for the voiced–voiceless–mixed classification, the number of features used for a voiced-unvoiced classifier is insufficient. Their VDA is realized in two steps using a binary decision tree structure. The first step is a classifier which separates frames that are predominantly voiced from those that are predominantly unvoiced. It uses a minimum-distance statistical classifier exploiting seven features: normalized autocorrelation coefficient at unit sample delay, minimum normalized LP error, zero-crossings rate, signal energy, overall degree of periodicity (via AMDF), and degree of periodicity measured via the cepstrum in two subbands. In both categories the mixed frames are split off during the second step. The voiced–mixed decision uses another six features, mostly cepstral and LP measures, whereas the voiceless–mixed decision is based on two features alone. The VDA is reported to work with 94% overall accuracy and 77% correct identification of the mixed frames.

## 10.5 Evaluation and Postprocessing

To evaluate the performance of a measuring device, one should have another instrument with at least the same accuracy. If this is not available, at least objective criteria – or data – are required to check and adjust the behavior of the new device. In pitch and voicing determination both these bases of comparison are tedious to generate. There is no VDA or PDA that operates without errors [10.47]. There is no reference algorithm, even with instrumental support, that operates completely without manual inspection or control [10.8, 72]. Yet nowadays speech databases with reference pitch contours and voicing information have become widely available so that at least reliable reference data are there and are being used for evaluation.

In this section, we first deal with the question of how to generate reference data (Sect. 10.5.1). Then we consider the question of error analysis (Sect. 10.5.2) and present the results of some comparative evaluations (Sect. 10.5.3). Finally, we describe the problem of pitch tracking (Sect. 10.5.4), which is the foremost task of the postprocessor.

### 10.5.1 Developing Reference PDAs with Instrumental Help

A number of evaluations compared the algorithm(s) to be tested to the results of a well-known algorithm such as the cepstrum PDA, whose performance was known to be good. *Rabiner* et al. [10.47] used an interactive PDA to generate reference data. This procedure proved reliable and accurate but required a great deal of human work. Other evaluations [10.1] used the output signal of a vocoder for which the pitch contour was exactly known or the output signal of a mechanic accelerometer which derives the information on pitch from the vibrations of the neck tissue at the larynx. The latter device [10.73] was used by *Viswanathan* and *Russell* [10.74] for their evaluation of five PDAs. *Indefrey* et al. [10.34] used a laryngograph to obtain the signal for generating a reference contour.

Among the algorithms used for determining a reference pitch contour, methods that make use of an instrument (such as a mechanic accelerometer or a laryn-



Amazon EX1010 Page 226

gograph) that derives pitch directly from the laryngeal waveform have been shown to be most effective. This type of algorithm avoids most errors pertinent to the problem of pitch determination from the speech signal, and permits using natural speech for the evaluation of the performance of PDAs. Among the many instruments available, the laryngograph [10.72,75] is especially well suited for this kind of application. It is robust and reliable, does not prevent the speaker from natural articulation, and gives a good estimate for the instant of glottal closure. A number of PDAs have been designed for this device [10.8,72]. In addition, *Childers* et al. [10.76] propose a four-category VDA that exploits the speech signal and the laryngogram.

The principle of the laryngograph [10.75] is well known. A small high-frequency electric current is passed through the larynx by a pair of electrodes that are pressed against the neck at the position of the larynx from both sides. The opening and closing of the glottis during each pitch period cause the laryngeal conductance to vary; thus the high-frequency current is amplitude modulated. In the receiver the current is demodulated and amplified. Finally, the resulting signal is high-pass filtered to remove unwanted low-frequency components due to vertical movement of the larynx.

Figure 10.10 shows an example of the laryngogram (the output signal of the laryngograph) together with the corresponding speech signal. In contrast to the speech signal, the laryngogram is barely affected by the instantaneous configuration of the vocal tract, and the changes in shape or amplitude are comparatively small. Since every glottal cycle is represented by a single pulse, the use of the laryngograph reliably suppresses gross period-determination errors. In addition, it supplies the basis for a good voiced–unvoiced discrimination since the laryngogram is almost zero during unvoiced segments, when the glottis is always open. Nonetheless, the laryngograph is not free from problems: it may fail temporarily or permanently for some individual speakers, or it may miss the beginning or end of a voiced segment by a short interval – for instance, when the vocal folds, during the silent phase of a plosive, continue to oscillate without producing a signal, or when voicing is resumed after a plosive and the glottis does not completely close during the first periods [10.72]. For such reasons, visual inspection of the reference contour is necessary even with this configuration; these checks, however, can be confined to limited segments of the signal.

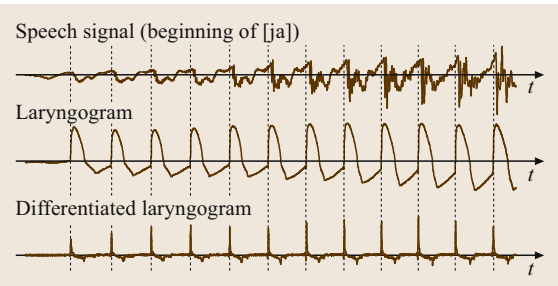



**Fig. 10.10** Speech signal, laryngogram, differentiated laryngogram, and instants of glottal closure

The instant of glottal closure is the point of maximum vocal-tract excitation, and it is justifiable to define this instant as the beginning of a pitch period. In the laryngogram this feature is well documented. As long as the glottis is open, the conductance of the larynx is at a minimum and the laryngogram is low and almost flat. When the glottis closes, the laryngeal conductance goes up and the laryngogram shows a steep upward slope. The point of inflection during the steep rise of the laryngogram, i. e., the instant of maximum change of the laryngeal conductance, was found best suited to serve as the reference point for this event.

### 10.5.2 Error Analysis

According to the classic study by *Rabiner* et al. [10.47], which established the guidelines for the performance evaluation of these algorithms for speech, PDAs and VDAs commit four types of errors:

1. Gross pitch-determination errors
2. Fine pitch-determination errors, i. e., measurement inaccuracies
3. Voiced-to-unvoiced errors
4. Unvoiced-to-voiced errors

The latter two types represent errors of voicing determination, whereas the former two refer to pitch determination.

*Gross pitch-determination errors* are *drastic failures of a particular method or algorithm to determine pitch* [10.47]. Usually an error is considered to be gross when the deviation between the correct value of $T_0$ or $F_0$ and the estimate of the PDA exceeds the maximum rate of change a voice can produce without becoming irregular (*Rabiner* et al. [10.47]: 1 ms; *Hess* and *Indefrey* [10.8], *Mousset* et al. [10.77]: 10%; *Krubsack* and *Niederjohn* [10.78]: 0.25 octave). Typical gross errors are confusions between $F_0$ and the formant $F1$, which usually falls into the measuring range. Other typical errors are the so-called octave errors, i. e., taking $F_0/2$ or $2F_0$ as the pitch estimate.



Amazon EX1010 Page 227

On the other hand, error situations may also arise from *drastic failures of the voice to produce a regular excitation pattern*, which is not very frequent in well-behaved speech but is nearly always the case when the voice temporarily falls into creak or vocal fry. *Hedelin* and *Huber* [10.18] distinguish between four main types of irregularity of phonation that occur frequently in running speech: (1) laryngealization (a temporal near-closure of the glottis resulting in single, irregular glottal pulses); (2) creak or vocal fry as a temporal voice register (Fig. 10.2); (3) creaky voice, which is even less structured than creak; and (4) diplophonic excitation, which shows an irregular pattern between adjacent periods but a more-regular one between every second pitch period. A further problem, which may sometimes become severe, is the rate of change of fundamental frequency. *Xu* and *Sun* [10.79], also referring to earlier studies, give data for the maximum rate of change of $F_0$ that a human voice is able to produce without becoming irregular. They found that a human voice can change its $F_0$ at a speed up to 100 semitones per second, and that this limit is frequently reached during running speech. One hundred semitones per second means one semitone (6%) per 10 ms or two semitones (12%) per 20 ms. According to *Sreenivas* [10.36], a 4% within-frame $F_0$ change already affects the salience of the estimate in the same way as additive noise with 15 dB SNR. As we see from these data, a 10% change of $F_0$ within a frame can easily occur. If we interpret these data with respect to individual pitch periods, we see that deep male voices with long periods (10 ms and more) are more strongly affected than female voices. Nonetheless, a deviation of 10% for $F_0$ estimates between adjacent frames seems reasonable as a lower bound for gross errors because a larger change is beyond the capabilities of a human voice.

Hence, gross errors arise mainly from three standard situations.

- Adverse signal conditions: strong first formants, rapid change of the vocal-tract position, band-limited or noisy recordings. Good algorithms reduce these errors to a great extent but cannot completely avoid them [10.47].
- Insufficient algorithm performance, e.g., mismatch of $F_0$ and frame length [10.21]; temporary absence of the key feature in some algorithms.
- Errors that arise from irregular excitation of voiced signals. Since most algorithms perform some averaging or regularity check, they can do nothing but fail when the source becomes irregular.

When a PDA is equipped with an error-detecting routine (the majority of cases, even if no postprocessor is used), and when it detects that an individual estimate may be wrong, it is usually not able to decide reliably whether this situation is a true measurement error, which should be corrected or at least indicated, or a signal irregularity, where the estimate may be correct and should be preserved as it is. This inability of most PDAs to distinguish between the different sources of error situations is one of the great unsolved problems in pitch determination.

In the study by *Rabiner* et al. [10.47] gross errors are simply counted, and the percentage of frames with gross errors compared to the total number of (correctly recognized) voiced frames is given as the gross error rate. However, the perceptual importance of gross errors depends on the deviation between the estimate and the correct value as well as the energy of the frame [10.74, 80], from which a weighted gross error measure was derived [10.67],

$$\mathrm{GPE} = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{E(k)}{E_{\max}}\right)^2 \left|\frac{p(k)-F_0(k)}{F_0(k)}\right| , \quad (10.21)$$

where $p(k)$ is the incorrect estimate, $E(k)$ is the energy of the frame, and $E_{\max}$ is the maximum energy in the utterance. It appears useful to include both these measures in an evaluation. The gross error count evaluates the PDA performance from a signal-processing point of view, whereas GPE says something about their perceptual relevance.

*Measurement inaccuracies* cause noisiness of the obtained $T_0$ or $F_0$ contour. They are small deviations from the correct value but can nevertheless be annoying to the listener. Again there are three main causes.

- Inaccurate determination of the key feature. This applies especially to algorithms that exploit the temporal structure of the signal, for instance, when the key feature is a principal maximum whose position within a pitch period depends on the formant $F1$.
- Intrinsic measurement inaccuracies, such as the ones introduced by sampling in digital systems.
- Errors from small fluctuations (*jitter*) of the voice, which contribute to the perception of *naturalness* and should thus be preserved.

*Voicing errors* are misclassifications of the VDA. We have to distinguish between voiced-to-unvoiced errors, in which a frame is classified as unvoiced although it is in fact voiced, and unvoiced-to-voiced errors, with the opposite misclassification. This scheme, as established



Amazon EX1010 Page 228

in [10.47], does not take into account mixed excitation. Voiced-to-unvoiced and unvoiced-to-voiced errors must be regarded separately because they are perceptually inequivalent [10.74], and the reasons for such errors in an actual implementation may be different and even contradictory. A number of VDAs even allow a tradeoff between these two errors by adjusting their parameters.

### 10.5.3 Evaluation of PDAs and VDAs– Some Results

Due to the absence of reliable criteria and systematic guidelines, few publications on early PDAs included a quantitative evaluation. As this situation has thoroughly changed, publications presenting new PDAs increasingly also evaluate them. Mostly the newly developed PDA is evaluated against some well-known PDA(s) to show that the new approach is in some way or for some kind of signals and conditions better or at least equivalent to the known algorithms [10.46,52]. The *Keele* database [10.81] has played a major role in this respect. We will not discuss these evaluations here due to lack of space; we rather deal with a couple of studies that did not aim at developing a new PDA but were done to establish guidelines and show the state of the art.

The classic studies by *Rabiner* et al. [10.47] and *McGonegal* et al. [10.56] investigated seven PDAs (two time domain, five short-term analysis) with respect to pitch and voicing determination. The main results were:

- None of the PDAs investigated were error free, even under good recording conditions. Each PDA had its own favorite error; nevertheless, all error conditions occurred for all of the PDAs.
- Almost any gross error was perceptible; in addition, unnatural noisiness of a pitch contour was well perceived.
- The subjective evaluation did not match the preference of the objective evaluation. In fact, none of the objective criteria (number of gross errors, noisiness of the pitch contour, or voicing errors) correlated well with the subjective scale of preference.

Hence the question of which errors in pitch and voicing determination are really annoying for the human ear remained open. This issue was further pursued by *Viswanathan* and *Russell* [10.74], who developed objective evaluation methods that are more closely correlated to the subjective judgments. The individual error categories are weighted according to the consistency of the error, i. e., the number of consecutive erroneous frames, the momentary signal energy, the magnitude of the error, and the special context.

*Indefrey* et al. [10.34], concentrating on the evaluation of pitch determination errors only, investigated several short-term PDAs in various configurations. They showed that in many situations different short-term analysis PDAs behave in a complementary way so that combining them in a multichannel PDA could lead to better overall performance.

*Indefrey* et al. [10.34] also investigated the performance of double-transform PDAs (cf. Sect. 10.2.2) with additive Gaussian noise. Under this condition these algorithms tend to break down at SNRs between 0 and −6 dB. It does not make a big difference whether the SNR is defined globally (i. e., with a constant noise level over a whole utterance) or segmentally (i. e., with the same SNR for each frame), except that the slope of the error curve at the breakdown point is larger for segmental SNR. These results were confirmed in a number of other studies [10.26, 46, 52]. *Moreno* and *Fonollosa* [10.52] evaluated several autocorrelation PDAs (among them their own) with several kinds of noise signals and found that for the low-frequency-biased car noise the breakdown starts at an SNR of about 6 dB. The same holds for babble noise [10.46].

*De Cheveigné* and *Kawahara* [10.27] investigated eight PDAs whose software was available via the Internet together with two of their own developments ([10.27], cf. Sect. 10.2.1; [10.82], based on an IF principle). Only gross errors were considered. The evaluation was based upon an extensive database (almost two hours of speech) with samples from three languages (English, French, and Japanese), including falsetto speech from male speakers, and laryngograms as reference signals. Obviously aperiodic voiced signals were excluded. Postprocessors and VDAs were disabled in the algorithms as far as possible. The evaluation showed great differences between algorithms and partly rather bad performance (more than 10% gross errors for some of them); the best one produced about 0.6% on average. The evaluation also showed considerable dependency of the error rate on the data so that the authors claim the need for large databases when performing such evaluations.

All these evaluations show that there is still no PDA that works without errors, although they work better now than 20 years ago. A gross error count of 0.6% is regarded as excellent; nonetheless we must not forget that, with the usual frame rate of 100 frames per second, such an algorithm still produces a grossly wrong estimate every two seconds of speech on average.



Amazon EX1010 Page 229

### 10.5.4 Postprocessing and Pitch Tracking

A standard procedure for the reduction of pitch determination errors is smoothing. Smoothing is possible when a pitch contour is given as a sequence of $T_0$ or $F_0$ estimates and not as delimiters of individual periods. The two standard smoothing methods are linear smoothing using some kind of low-pass filter and (nonlinear) median smoothing [10.83]. Linear smoothing reduces measurement inaccuracies but is unable to cope with the effect of gross pitch determination errors, which are reduced in size and distributed over a larger amount of time but are not really removed. Median smoothing, on the other hand, replaces each pitch estimate with the middle value of an ordered sequence of three, five, or seven adjacent estimates; gross outliers are removed, but measurement inaccuracies remain unchanged. *Rabiner* et al. [10.83] combine these methods and propose a two-step smoothing procedure with median smoothing coming first, followed by a linear smoother. Linear smoothing, however, can be dangerous since it may replace a gross error that has been left in the median-smoothed contour by some estimates lying between the correct value and the error and so cause an inflection in the contour that is not due to the signal.

Applying such a smoothing algorithm was shown to substantially improve the (objective and subjective) performance of any PDA to which it was applied [10.47, 56]. *Specker* [10.84] showed that postprocessing is able to reduce the number of gross errors in a time-domain PDA by almost an order of magnitude.

*Secrest* and *Doddington* [10.80] used dynamic programming methods to find an optimal path through a list of pitch estimate candidates with the smoothness of the contour as the major criterion. They showed that this technique performed better than any linear, nonlinear, or median smoothing. This approach was further developed by *Talkin* [10.17]. Dynamic programming is well suited to pitch tracking since it allows the basic extractor to give several pitch candidates so that we can deal with more than only the best choice in each frame. Each candidate is accompanied by a salience measure (usually the relative amplitude of the corresponding peak in the representation from where the estimate is derived, with respect to the reference point, e.g., the value of the ACF at zero lag). In addition, Talkin's PDA supplies one candidate *unvoiced* per frame. Pitch tracking is done by searching for an optimal path through the candidates from consecutive frames by minimizing a global cost function. This global cost function is formed as the sum of weighted local per-frame cost functions of two types: (1) candidate costs, and (2) transition costs between consecutive frames.

Candidate costs distinguish between pitch candidates and the unvoiced candidate. The cost of a pitch candidate equals one minus the salience measure of this candidate. The cost of the unvoiced candidate is a constant penalty plus the maximum salience measure within the current frame.

The transition cost between consecutive frames also depends on voicing. Between two unvoiced candidates it is zero. Between two pitch candidates it depends on the difference in frequency between the two estimates, and special attention is given to octave jumps, which are made costly but not totally impossible. The costs for voiced-to-unvoiced transitions and vice versa include a term with the reciprocal *Itakura* distance [10.85], an energy term, and an extra penalty for this transition. The rationale is that (1) these transitions are not too frequent, (2) there are usually large spectral changes between a voiced and an unvoiced frame, and (3) energy usually decreases at the end of a voiced part and increases at its beginning.

There is no latency limit for the algorithm to find the optimal path; in principle the search can extend over a whole utterance. *Talkin* [10.17], however, reports that it rarely takes more than 100 ms for all possible paths to converge to a single point. The algorithm is part of the well-known ESPS software package for speech analysis. A comparable postprocessor operating on a probabilistic approach is described in [10.86].



## 10.6 Applications in Speech and Music

Applications for PDAs in speech can be grouped into four areas [10.1]:

1. Speech technology
2. Basic linguistic research, e.g., investigation of intonation
3. Education, such as teaching intonation to foreign-language learners or to hard-of-hearing persons

4. Clinical applications, such as the investigation of voice quality

Some of these application areas have changed greatly over the last two decades. The vocoder, which was the main application for earlier speech technology, has almost disappeared. Instead, investigating prosodic events such as intonation, particularly in spontaneous speech, has become an important issue in speech understanding systems [10.87], and many of these systems now contain a prosody module. As it is a long-term goal in speech technology to make such devices operable from almost anywhere, a PDA may even have to cope with signals from mobile phones, which can be extremely bad and inconsistent. Another new application area is data-driven speech synthesis. Algorithms for time-domain pitch modification, such as the well-known *pitch-synchronous overlap add* (PSOLA) algorithm [10.57], require precise pitch period determination to work properly, and with the recent technology of nonuniform unit selection synthesis large speech corpora have to be analyzed, yet usually with excellent-quality signals free from phase distortions.

What are the implications of this application shift for the development of algorithms? PDAs for precise pitch period determination of good-quality speech signals have been known for a long time; nonetheless the main problem is exact synchronization with laryngeal cycles, such as the instant of glottal closure. Such algorithms, which originally come from clinical applications where they were applied to isolated vowels, have been extended to work for running speech as well.

In prosody recognition, intonation research and speech technology now go together to a certain extent. Prosody recognition needs intonation contours, not individual periods, and a certain lag between the running time of the signal and the time of release of an estimate is tolerable, in contrast to a vocoder where the result must be available without delay. On the other hand, prosody recognition must rely on automatic estimates and cope with adverse conditions; above all, this requires robustness.

The number of devices available for computer-aided intonation teaching has been small [10.88]. However, with the increased use of high-quality PDAs for intonation research, this will change. In the clinical area, digital hearing prostheses have created a new application area [10.63, 89]. We cannot discuss these applications here for reasons of space; the reader is referred to [10.88, 89] for surveys.

Pitch determination of musical signals has three main application areas, two of which are closely related:

1. Automatic notation of melodies and tunes, and automatic scoring
2. Information retrieval from audio data
3. Real-time capturing of tone frequencies of musical instruments for musical instrument digital interface (MIDI) applications

Automatic notation of melodies is a long-standing problem. As early as 1979 *Askenfelt* [10.90] reported on a project to automatically transcribe folk melodies that had been collected in a large corpus. In this case most of the melodies were one-voiced so that PDAs developed for speech signals could be used.

This is not always the case for music, however, and we must therefore count as a particular problem in music that more than one note can be played at the same time, and that PDAs for this application may have to cope with multiple pitches and have to determine them all, if possible.

Information retrieval from audio data (audio data mining) also involves pitch determination of musical signals. This application area is closely related to the preceding one and also involves the problem of multiple pitches. An evolving application is the so-called query by humming. Consider a person who listens to an unknown tune in the radio, likes it, calls a call center with a musical database, hums or sings the melody, and wants the title of the song retrieved. This is a difficult task with pitch determination being a part thereof.

Transcription of a musical melody into musical notes is much more than mere pitch determination. One has to recover the rhythm from the timing of the melody, and one has to take care of the timing, i. e., when one note ends and another one starts. These questions, which are partly still open, go beyond the scope of pitch determination and will not be further discussed here.

Real-time capturing for MIDI is closely related to the old vocoder application in speech. A PDA is always desirable if a MIDI synthesizer is to be driven from an instrument other than a keyboard, e.g., from an electric guitar. Here the main problem is that the response must be almost instantaneous; a delay of even 50 ms could be detrimental for a live performance.



Amazon EX1010 Page 231

## 10.7 Some New Challenges and Developments

This section will concentrate on three problems:

1. Detecting the instant of glottal closure for applications in speech synthesis
2. Multiple pitch detection, particularly in music [10.93]
3. Instantaneousness versus reliability

### 10.7.1 Detecting the Instant of Glottal Closure

If a PDA in speech is required to be very accurate, it is optimal to synchronize it with the instants of glottal closure. A cascaded PDA for this purpose was developed by *Hess* [10.92] based on a previous algorithm by *Cheng* and *O'Shaughnessy* [10.91]. It is intended for work in time-domain speech synthesis and determines the glottal closure instant (GCI) from undistorted signals. The first part of the cascade is a short-term PDA applying a double-spectral-transform principle [10.34]. The second part uses the estimate of the first PDA to restrict its momentary $F_0$ range to an octave around this value.

According to *Mousset* et al. [10.77] a GCI detector consists of four steps:

1. Acoustic speech signal pre-emphasis (optional)
2. A transformation aiming to produce peaks at GCIs
3. Postprocessing aiming to increase contrast in the resulting signal (optional)
4. A peak picking operation

The algorithm described here follows this scheme.

In the source-filter approach the speech signal is the response of the supraglottal system to the pulse train generated by the source, and a pitch period can be regarded as the beginning of the impulse response of the vocal tract. If we now compute the correlation function $c(n)$ between the speech signal $s(n)$ and the beginning of the pertinent impulse response $h(n)$ of the vocal tract,

$$c(n) = \sum_k s(n+k)h(k) \qquad (10.22)$$

there will be maxima at those instants $n$ that correspond to an GCI. The impulse response $h(n)$ is estimated via linear prediction. The main peaks of $c(n)$ synchronize well with the individual GCIs. Strong formants, however, also show up in $c(n)$, and further processing is required. Cheng and O'Shaughnessy suggested that one should calculate the envelope of $c(n)$, send it through a high-pass filter and a half-way rectifier to enhance the leading amplitudes at the beginning of each pitch period, and multiply $c(n)$ by the resulting waveform $d(n)$. The envelope

$$e(n) = \sqrt{[c(n)]^2 + [c_H(n)]^2}\,, \qquad (10.23)$$

where $c_H(n)$ is the Hilbert transform of $c(n)$, is calculated using a digital Hilbert filter. Figure 10.11 shows an example.

This approach has the problem known from all PDAs that apply some sort of inverse filtering (cf. Sect. 10.3.3) that it fails systematically when the formant $F1$ coincides with $F_0$ and the signal becomes almost sinusoidal. Then $e(n)$ becomes a constant, and $d(n)$ is either zero or fluctuates at random around zero instead of displaying the envelope of a damped formant oscillation. A way out of this problem is to partly bridge the high-pass filter so that the constant component of the envelope is not completely removed. If the short-term analysis enters a reliable estimate of $T_0$, rather stringent correction routines can remove the unwanted peaks in $c(n)$ and at the same time preserve correct processing. The exact position of the GCIs is derived from $c(n)$ anyway.

Group delay methods have also been proposed for GCI determination [10.94, 95]. Group delay is defined as the derivative of the phase spectrum with respect to




**Fig. 10.11a–e** Determining the instant of glottal closure via a maximum-likelihood criterion [10.91, 92]: example of performance. **(a)** Signal frame (45 ms); **(b)** impulse response $h(n)$ of the vocal tract, estimated by linear prediction; **(c)** correlation function $c(n)$; **(d)** envelope $e(n)$ (dotted line) and high-pass filtered envelope; **(e)** product of $c(n)$ and the filtered envelope



Amazon EX1010 Page 232

frequency and can be calculated [10.94] via the DFT:

$$\tau_{\mathrm{G}}(m) = \mathrm{Re}\left(\frac{\tilde{S}(m)}{S(m)}\right) ; \qquad (10.24)$$

$$S(m) = \mathrm{DFT}\{s(n)\} ; \quad \tilde{S}(m) = \mathrm{DFT}\{ns(n)\}$$

If a frame contains a single impulse at position $k = n_0$, this will produce a constant group delay of $n_0$ for all frequency indexes $m$. If there is additional noise in the frame, one can expect that at least the average group delay (averaged over all frequencies) equals $n_0$. If we now compute the average group delay for each sample of the time signal $s(n)$, it will show a negative-going zero crossing (with a slope of $-1$) when the impulse is at the starting position of the frame. GCIs mark nonstationarities in the signal and show up as impulse-like structures, for instance, in an LP residual. This makes the method useful for GCI detection, and there are algorithmic shortcuts that allow the average group delay to be computed without needing two DFTs per sample. *Brookes* et al. [10.95] investigate several weighted measures for the average group delay and find that these measures are quite robust against additive noise; however, they critically depend on the length of the time-domain window used for group delay measurement. If there is no impulse in the frame, the average group delay will vary around zero; if there is more than one impulse, the results are unpredictable. In a cascaded implementation using an auxiliary short-term PDA, when an estimate for $T_0$ is available, it is straightforward to adjust the window length according to this estimate.

Another method, rather simple to implement, is based on singular value decomposition. *Ma* et al. [10.96] show that the Frobenius norm $||\mathbf{S}||_{\mathrm{F}}$ of an $N \times (K+1)$ matrix $\mathbf{S}$, being

$$||\mathbf{S}||_{\mathrm{F}} = \sqrt{\sum_{n=1}^{N}\sum_{k=1}^{K+1} s_{nk}^2} \quad \text{with} \qquad (10.25)$$

$$\mathbf{S} = (s_{nk};\ 1 \le n \le N;\ 1 \le k \le K+1)$$

equals the product of the squared singular values of the singular value decomposition (SVD) of $\mathbf{S}$,

$$||\mathbf{S}||_{\mathrm{F}} = \sqrt{\sum_{k=1}^{K+1} \sigma_k^2} \, . \qquad (10.26)$$

SVD is known to model linear dependencies of the rows and columns of the associated matrices. GCIs provide new information and cannot be covered by linear modeling. Hence, singular values tend to be large when the pertinent speech data matrix representing a signal frame,

$$\mathbf{S} = \begin{pmatrix} s(K) & s(K-1) & \cdots & s(0) \\ s(K+1) & s(K) & \cdots & s(1) \\ \vdots & \vdots & \ddots & \vdots \\ s(K+N-1) & s(K+N-2) & \cdots & s(N-1) \end{pmatrix}$$

contains a glottal impulse. (We assume $N > K$ and full column rank of the matrix $\mathbf{S}$ [10.96].) The singular values become largest when the glottal impulse is found in the first row of $\mathbf{S}$. So the Frobenius norm gives an algorithmic shortcut to the costly SVD of $\mathbf{S}$ which would have to be performed otherwise. This algorithm had a couple of forerunners that are well described in [10.96].

Other methods to determine the GCI involve neural networks with appropriate training [10.63], wavelet functions [10.97], simplified inverse filtering [10.77], nonlinear filtering [10.98] or statistical evaluation of the nonstationarity of the signal at the GCI [10.99]. *Mousset* et al. [10.77] evaluated several of these methods using the Keele database for PDA evaluation [10.81]. They used the analysis scheme proposed by *Rabiner* et al. [10.47] and extended it by two error classes suitable for any time-domain PDA: (1) insertion of a GCI marker where no GCI is present, and (2) miss (deletion) of a GCI that should have been detected. Their results show that the methods evaluated are about equivalent but show some sensitivity to the length of the respective time windows and to the speakers.

### 10.7.2 Multiple Pitch Determination

Simultaneous tones with different frequencies require some frequency-domain processing since, in this domain, peaks resulting from different tones show up at their respective frequencies. The same holds for the lag domain of a double-spectral-transform PDA when the nonlinear distortion in the frequency is of local nature, such as in computing a cepstrum or an autocorrelation function. Hence PDAs that explicitly or implicitly involve a Fourier transform will have this property, as was shown in [10.33] using simultaneous speech from two talkers as well as musical signals. Although not explicitly stated in the literature, active modeling would also allow multipitch tracking.

Such PDAs have been used in several configurations for speaker separation. The usual procedure is that the PDA determines pitch for one of the speakers; then a speech enhancement procedure (e.g., spectral subtraction) is applied to remove this speaker from the signal. The PDA then determines the pitch of the sec-



Amazon EX1010 Page 233

ond speaker. The problem with this configuration is the correct assignment of the signal to the two speakers, particular when the signal from a speaker is unvoiced. *De Cheveigné* [10.100] describes a PDA designed for this purpose that estimates two pitches simultaneously using a cascaded comb filter and a two-dimensional AMDF estimation procedure.

Multiple pitch determination, particularly for music, is a wide field that would justify a chapter of its own. In this section we can give only a small selection of examples. For more-comprehensive surveys of the activities in this field, the reader is referred to *De Cheveigné* [10.100] or *Klapuri* [10.101].

*Goto*'s algorithm [10.102] for music signals focuses on extracting *leading voices* from a polyphonic signal, in this case a melody line and a bass line which occupy different and non-overlapping $F_0$ ranges (32–260 Hz versus 260–4100 Hz). The PDA works in the frequency domain. It basically estimates the fundamental frequency of the most predominant harmonic structure corresponding to the melody or bass line. It simultaneously takes into account all possibilities for $F_0$ and treats the input spectrum as if it contains all possible harmonic structures with different weights (amplitudes). It regards a probability density function (PDF) of the input frequency components as a weighted mixture of harmonic-structure tone models (represented by PDFs) of all possible pitches and simultaneously estimates both their weights corresponding to the relative dominance of every possible harmonic structure and the shape of the tone models by maximum a-posteriori probability (MAP) estimation regarding their prior distribution. It then considers the maximum-weight model as the most predominant harmonic structure and obtains its fundamental frequency. A multiple-agent architecture evaluates the temporal continuity of the estimates.

The PDA uses a logarithmic frequency scale corresponding to the musical notation unit *cent*,

$$\frac{f}{\text{cent}} = 1200 \log_2 \frac{f/\text{Hz}}{440 \cdot 2^{\frac{3}{12}-5}} , \qquad (10.27)$$

where 1 cent equals 1/100 of a tempered semitone so that an octave consists of 1200 cents. Each tone model corresponds to a trial fundamental frequency $p$ and provides a harmonic structure; the individual harmonics are modeled as weighted one-dimensional Gaussian distribution. The weights of all models are estimated simultaneously, where the same frequency can be shared by different harmonics of different tone models. The tone model with maximum weight yields the estimate for the predominant $F_0$. A multiple-agent architecture performs a temporal track across frames and gives the most stable trajectory as the final result. It consists of a salience detector that picks promising $F_0$ candidates, and a number of agents that interact to allocate the salient peaks among themselves according to peak closeness. Each agent has its own penalty record, and it gets a penalty when no suitable peak can momentarily be allocated to it. If a penalty threshold is exceeded, the agent is terminated. If a peak cannot be assigned to a running agent, a new agent is created. The final output is determined on the basis of which agent has the highest reliability and greatest total power along the trajectory of the peak it is tracking. Detection rates for melody and bass are reported as 80–90% and depend on the respective tune being processed.

The PDA by *Tolonen* and *Karjalainen* [10.103] uses a perception-oriented double-transform PDA based on the unitary model of pitch perception [10.6]. The original model in [10.6] employs a large number of critical-band filters (spaced at much less than a critical band). In each channel the signal is half-wave rectified and low-pass filtered, and its short-term ACF is determined. The ACFs of all channels are then added to give an estimate of pitch. In Tolonen and Karjalainen's PDA the signal is first filtered by an optional pre-whitening filter and then split into only two subbands with a cutoff frequency of 1000 Hz. The high-frequency channel is then rectified and smoothed. The lag-domain representations obtained by the double-spectral-transform principle for each subband are added up to the so-called summary ACF. This function is then enhanced. It is first clipped from under so that only positive values are retained. Then it is time stretched by a factor of 2, 3, etc. and subtracted from the original summary ACF, and again only positive values are retained. From this enhanced summary ACF significant maxima are sought. For musical sound analysis, relatively long windows (up to 180 ms) are employed. The algorithm works well when the simultaneous tones in the input signal do not differ too much in amplitude. The method is limited to fundamental frequencies below 1000 Hz [10.101].

*Klapuri* [10.101], besides giving a comprehensive survey of PDAs for multipitch determination, developed two PDAs for this task. One of these is based on the unitary model of pitch perception [10.6], with some modification that makes the PDA more reliable and computationally less costly. The other is based on an iterative frequency-domain technique. The predominant pitch is detected, then the corresponding harmonic structure is removed from the spectrum, and the procedure is repeated for the next predominant pitch. The



Amazon EX1010 Page 234

iteration is stopped when the energy of the harmonic structure drops below a given threshold.

Determination of the predominant harmonic structure follows a perception-oriented approach that resembles Terhardt's virtual pitch model [10.4] in that it is tolerant toward inharmonicity. The spectrum is subdivided into 18 subbands with overlapping triangular transfer functions which add to unity for adjacent subbands. In each subband salience estimates are made for each trial fundamental frequency $p$ in the measuring range,

$$L(p,k) = \max_m \left( c(m,i) \sum_i^{(\text{in subband } k)} S(m+ip) \right) , \tag{10.28}$$

where $m$ stands for a possible (small) offset due to inharmonicity, and $c$ specifies a weighting function that depends on $m$ and the harmonic number $i$. The salience estimates are then added over all subbands with the additional possibility of taking into account shift of higher harmonics toward higher frequencies, as it sometimes occurs with musical instruments. One of the trial frequencies $p$ emerges as predominant, and the pertinent harmonic amplitude spectrum is smoothed. To remove the tone, the smoothed harmonic structure is subtracted from the overall spectrum.

Special attention is given to the problem of several tones with harmonic frequency ratios. The smoothing procedure helps to solve this problem. Think of two tones with a frequency ratio of 1:3. All the harmonics of the higher tone coincide with harmonics of the lower one, but we can expect that every third harmonic has an outstanding amplitude. It is likely that the lower tone will be detected first. The smoothing procedure equalizes these amplitude fluctuations so that the higher tone will be preserved when the lower one is removed.

The PDA by *Kameoka* et al. [10.104] performs simultaneous multipitch extraction in the frequency domain based on a statistical model given by

$$\{\Theta\} = \{\mu(k), w(k), \sigma | k = 1, \ldots, K\} . \tag{10.29}$$

The model consists of a set of $K$ harmonic structures. Each of these is described by a tied Gaussian mixture (which corresponds to *Goto*'s PDA [10.102], see above). Its vector $\boldsymbol{\mu}$ of mean values stands for the frequencies of the partials and has only one degree of freedom, i. e., its fundamental frequency. The weight $w(k)$ stands for the predominance of the $k$-th harmonic structure; the variance is kept constant for the sake of simplicity. The model is initialized, and the parameters are iteratively improved using a log likelihood criterion and the expectation maximization algorithm [10.105]. As the true number of simultaneous tones in the signal is unknown, the model order $K$ is initialized too high and becomes part of the optimization. Akaike's information criterion (AIC), which specifies a tradeoff between the order of a model and its accuracy,

$$\begin{aligned} \text{AIC} = {} & 2 \cdot (\text{number of free parameters}) \\ & - 2 \cdot (\text{maximum of log likelihood}) \end{aligned} \tag{10.30}$$

is used to determine the optimal value of $K$. A harmonic structure is abandoned when its weight becomes low or when it is placed between two other structures that move toward each other and have higher weights. If a harmonic structure is abandoned, the number of free parameters decreases by two (frequency and weight), and the likelihood gets worse. Whenever a harmonic structure is to be abandoned, AIC is computed. The iteration is stopped when AIC has reached a minimum.

The role of gross pitch determination errors in this context differs from that in speech. In speech we have the special problem of octave errors (cf. Sect. 10.5.2) which are to be strictly avoided. In polyphonic music, on the other hand, two instruments frequently play an octave apart from each other, and should then both be detected. *Klapuri*'s smoothing procedure [10.101] explicitly takes into account two tones at any harmonic interval. Nevertheless it is reported [10.101, 104] that the PDAs have problems when notes are played at certain musical intervals, among them the octave.

### 10.7.3 Instantaneousness Versus Reliability

It is always desirable to get the estimate from a PDA instantaneously. Processing time depends on two factors: (1) computational complexity of the PDA and speed of the device running it, and (2) latency, i. e., the amount of signal required to get a reliable estimate plus computing time. A PDA runs in real time if the processing time required is less than the elapsed time, say, between two successive frames. As today's computers are able to run even complex PDAs in real time, we can put aside the first issue. With the amount of signal required, however, there may be a hidden problem. An ordinary short-term PDA needs (at least) two complete pitch periods to detect periodicity in an orderly manner. If the spectral transform is done in one step for the whole range of $F_0$, latency will be twice the longest possible period in the window. For speech, with $F_0$ ranging from 50 Hz for a deep male voice up to, say, 1000 Hz for a child in spontaneous speech, this means a lag of at least 40 ms from



Amazon EX1010 Page 235

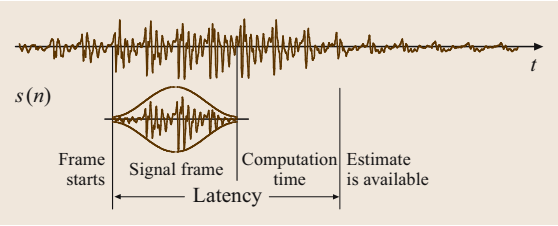



**Fig. 10.12** The latency problem. Latency is defined as the elapsed time from the beginning of the frame until the estimate is available

the beginning of the window until we obtain an estimate, not including processing time. In vocoder telephony this may just be tolerable, but this will be a problem for online capturing of music in a MIDI application. Of course the latency will go down when the lower end of the frequency range gets higher. One period of the lowest tone of an ordinary guitar lasts about 12 ms, while that of a violin lasts about 5 ms. However, if the PDA applies postprocessing, for instance, pitch contour tracking by dynamic programming methods [10.17], latencies will go up drastically. It goes without saying that such methods cannot be applied when tough time constraints are given.

How can this problem be solved? If reliability requires a short-term analysis PDA, we must speed it up. If we can assume that the signal is nearly sinusoidal, we can apply time-domain methods, which may be more instantaneous.

Several attempts have been made to speed up short-term analysis PDAs. It has been known for a long time that distance functions, such as the AMDF, can work with short frames. Since no Fourier transform is involved, the function can be computed synchronously as time runs, and so we may be able to obtain estimates of $T_0$ in little more than the duration of the period to be measured. Estimates of short periods will thus be available sooner than those of long ones. Such nonstationary approaches do not only exist for distance functions. For instance, *Talkin*'s PDA ([10.17]; Sect. 10.2.1) uses a nonstationary autocorrelation function. *Yoo* and *Fujinaga* [10.106] performed some practical experiments with several hardware and software PDAs for MIDI capture on a violin and found latencies of 15–90 ms. For a MIDI application this may already be unacceptable.

If the signal is near-sinusoidal, time-domain pitch determination can be very fast. The lag between two consecutive zero crossings or between a maximum and the consecutive minimum of a sinusoid equals half a period, and only a quarter of a period elapses between an extreme (maximum or minimum) and the adjacent zero crossing. Even faster methods needing fewer signal samples can be conceived, leading to a kind of *anytime* algorithm that can yield a rather imprecise estimate almost instantaneously but is able to refine this estimate when given more time. (Remember, however, that in MIDI applications a tone command can be given only once; if it is in error, the wrong tone will come out.) In this case the problem is: how can we obtain a signal close to a sinusoid? In speech this is unrealistic due to the transfer function of the vocal tract, where lip radiation introduces a zero at $f = 0$ that strongly attenuates the first harmonic. In music, however, such an approach can be of interest when an instrument yields a signal that is almost sinusoidal, or when the capturing microphone or sensor is placed on the instrument in such a way that it performs as a low-pass filter.

## 10.8 Concluding Remarks

The problem of pitch determination and fundamental frequency tracking is long-standing, known, and yet unsolved in a general sense. However, for most applications, algorithms that yield good and acceptable solutions have been developed. Applications in speech and music have moved away from the vocoder toward prosody recognition, automatic melody detection, acoustic data retrieval, computational auditory scene analysis, and high-precision analysis of speech synthesis corpora. New challenges for the development of PDAs, among others, include high-precision pitch period determination, processing of signals with multiple pitches, and PDAs with very short latencies.



Amazon EX1010 Page 236

## References


10.1 W.J. Hess: *Pitch Determination of Speech Signals – Algorithms and Devices* (Springer, Berlin, Heidelberg 1983)

10.2 R.J. McAulay, T.F. Quatieri: Speech analysis/synthesis based on a sinusoidal representation, IEEE Trans. Acoust. Speech Signal Process. **34**, 744–754 (1986)

10.3 E. Zwicker, W.J. Hess, E. Terhardt: Erkennung gesprochener Zahlworte mit Funktionsmodell und Rechenanlage, Kybernetik **3**, 267–272 (1967), (in German)

10.4 E. Terhardt: Calculating virtual pitch, Hearing Res. **1**, 155–182 (1979)

10.5 R. Plomp: *Aspects of Tone Sensation* (Academic, London 1976)

10.6 R. Meddis, L. O'Mard: A unitary model for pitch perception, J. Acoust. Soc. Am. **102**, 1811–1820 (1997)

10.7 K.J. Kohler: 25 Years of Phonetica: Preface to the special issue on pitch analysis, Phonetica **39**, 185–187 (1992)

10.8 W.J. Hess, H. Indefrey: Accurate time-domain pitch determination of speech signals by means of a laryngograph, Speech Commun. **6**, 55–68 (1987)

10.9 W.J. Hess: Pitch and voicing determination. In: *Advances in Speech Signal Processing*, ed. by M.M. Sondhi, S. Furui (Dekker, New York 1992), p.3–48

10.10 A.M. Noll: Cepstrum pitch determination, J. Acoust. Soc. Am. **41**, 293–309 (1967)

10.11 L.R. Rabiner: On the use of autocorrelation analysis for pitch detection, IEEE Trans. Acoust. Speech Signal Process. **25**, 24–33 (1977)

10.12 E. Terhardt, G. Stoll, M. Seewann: Algorithm for extraction of pitch and pitch salience from complex tonal signals, J. Acoust. Soc. Am. **71**, 679–688 (1982)

10.13 M.S. Harris, N. Umeda: Difference limens for fundamental frequency contours in sentences, J. Acoust. Soc. Am. **81**, 1139–1145 (1987)

10.14 J. 't Hart: Differential sensitivity to pitch distance, particularly in speech, J. Acoust. Soc. Am. **69**, 811–822 (1981)

10.15 H. Duifhuis, L.F. Willems, R.J. Sluyter: Measurement of pitch in speech: an implementation of Goldstein's theory of pitch perception, J. Acoust. Soc. Am. **71**, 1568–1580 (1982)

10.16 D.J. Hermes: Measurement of pitch by subharmonic summation, J. Acoust. Soc. Am. **83**, 257–264 (1988)

10.17 D. Talkin: A robust algorithm for pitch tracking (RAPT). In: *Speech Coding and Synthesis*, ed. by B. Kleijn, K. Paliwal (Elsevier, Amsterdam 1995), p.–495–518

10.18 P. Hedelin, D. Huber: Pitch period determination of aperiodic speech signals, Proc. IEEE ICASSP (1990) pp. 361–364

10.19 H. Hollien: On vocal registers, J. Phonet. **2**, 225–243 (1974)

10.20 N.P. McKinney: *Laryngeal Frequency Analysis for Linguistic Research* (Univ. Michigan, Ann Arbor 1965), Res. Rept. No. 14

10.21 H. Fujisaki, K. Hirose, K. Shimizu: A new system for reliable pitch extraction of speech, Proc. IEEE ICASSP (1986), paper 34.16

10.22 M.M. Sondhi: New methods of pitch extraction, IEEE Trans. Acoust. Speech Signal Process. **26**, 262–266 (1968)

10.23 J.D. Markel: The SIFT algorithm for fundamental frequency estimation, IEEE Trans. Acoust. Speech Signal Process. **20**, 149–153 (1972)

10.24 V.N. Sobolev, S.P. Baronin: Investigation of the shift method for pitch determination, Elektrosvyaz **12**, 30–36 (1968), in Russian

10.25 J.A. Moorer: The optimum comb method of pitch period analysis of continuous digitized speech, IEEE Trans. Acoust. Speech Signal Process. **22**, 330–338 (1974)

10.26 T. Shimamura, H. Kobayashi: Weighted autocorrelation for pitch extraction of noisy speech, IEEE Trans. Speech Audio Process. **9**, 727–730 (2001)

10.27 A. de Cheveigné, H. Kawahara: YIN, a fundamental frequency estimator for speech and music, J. Acoust. Soc. Am. **111**, 1917–1930 (2002)

10.28 K. Hirose, H. Fujisaki, S. Seto: A scheme for pitch extraction of speech using autocorrelation function with frame length proportional to the time lag, Proc. IEEE ICASSP (1992) pp. 149–152

10.29 D.E. Terez: Robust pitch determination using nonlinear state-space embedding, Proc. IEEE ICASSP (2002)

10.30 C.M. Rader: Vector pitch detection, J. Acoust Soc. Am. **36**(C), 1463 (1964)

10.31 L.A. Yaggi: *Full Duplex Digital Vocoder* (Texas Instruments, Dallas 1962), Scientific Report No.1, SP14-A62; DDC-AD-282986

10.32 Y. Medan, E. Yair, D. Chazan: Super resolution pitch determination of speech signals, IEEE Trans. Signal Process. **39**, 40–48 (1991)

10.33 M.R. Weiss, R.P. Vogel, C.M. Harris: Implementation of a pitch-extractor of the double spectrum analysis type, J. Acoust. Soc. Am. **40**, 657–662 (1966)

10.34 H. Indefrey, W.J. Hess, G. Seeser: Design and evaluation of double-transform pitch determination algorithms with nonlinear distortion in the frequency domain, Proc. IEEE ICASSP, Vol. 2 (1985), paper 11.12

10.35 P. Martin: Comparison of pitch detection by cepstrum and spectral comb analysis, Proc. IEEE ICASSP (1982) pp. 180–183




Amazon EX1010 Page 237

10.36 V.T. Sreenivas: *Pitch estimation of aperiodic and noisy speech signals* (Indian Institute of Technology, Bombay 1982), Diss., Department of Electrical Engineering, Indian Institute of Technology

10.37 M.R. Schroeder: Period histogram and product spectrum: new methods for fundamental-frequency measurement, J. Acoust. Soc. Am. **43**, 819–834 (1968)

10.38 P. Martin: A logarithmic spectral comb method for fundamental frequency analysis, Proc. 11th Int. Congr. on Phonetic Sciences Tallinn (1987), paper 59.2

10.39 P. Martin: WinPitchPro – a tool for text to speech alignment and prosodic analysis, Proc. Speech Prosody 2004 (2004) pp. 545–548, http://www.isca-speech.org/archive/sp2004 and http://www.winpitch.com

10.40 J.C. Brown, M. Puckette: A high-resolution fundamental frequency determination based on phase changes of the Fourier transform, J. Acoust. Soc. Am. **94**, 662–667 (1993)

10.41 J.C. Brown: Musical fundamental frequency tracking using a pattern recognition method, J. Acoust. Soc. Am. **92**, 1394–1402 (1992)

10.42 F. Charpentier: Pitch detection using the short-term phase spectrum, Proc. IEEE ICASSP (1986) pp. 113–116

10.43 M. Lahat, R.J. Niederjohn, D.A. Krubsack: A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech, IEEE Trans. Acoust. Speech Signal Process. **35**, 741–750 (1987)

10.44 B. Doval, X. Rodet: Estimation of fundamental frequency of musical sound signals, Proc. IEEE ICASSP (1991) pp. 3657–3660

10.45 T. Abe, K. Kobayashi, S. Imai: Robust pitch estimation with harmonics enhancement in noisy environments based on instantaneous frequency, Proc. ICSLP'96 (1996) pp. 1277–1280, http://www.isca-speech.org/archive/icslp_1996

10.46 T. Nakatani, T. Irino: Robust and accurate fundamental frequency estimation based on dominant harmonic components, J. Acoust. Soc. Am. **116**, 3690–3700 (2004)

10.47 L.R. Rabiner, M.J. Cheng, A.E. Rosenberg, C.A. McGonegal: A comparative study of several pitch detection algorithms, IEEE Trans. Acoust. Speech **24**, 399–423 (1976)

10.48 L. Arévalo: *Beiträge zur Schätzung der Frequenz gestörter Schwingungen kurzer Dauer und eine Anwendung auf die Analyse von Sprachsignalen* (Ruhr-Universität, Bochum 1991), Diss. in German

10.49 A.M. Noll, A. Michael: Pitch determination of human speech by the harmonic product spectrum the harmonic sum spectrum and a maximum likelihood estimate, Symp. Comput. Process. Commun. **19**, 779–797 (1970), ed. by the Microwave Inst., New York: Univ. of Brooklyn Press

10.50 D.H. Friedman: Pseudo-maximum-likelihood speech pitch extraction, IEEE Trans. Acoust. Speech Signal Process. **25**, 213–221 (1977)

10.51 R.J. McAulay, T.F. Quatieri: Pitch estimation and voicing detection based on a sinusoidal speech model, Proc. IEEE ICASSP (1990) pp. 249–252

10.52 A. Moreno, J.A.R. Fonollosa: Pitch determination of noisy speech using higher order statistics, Proc. IEEE ICASSP (1992) pp. 133–136

10.53 B.B. Wells: Voiced/Unvoiced decision based on the bispectrum, Proc. IEEE ICASSP (1985) pp. 1589–1592

10.54 J. Tabrikian, S. Dubnov, Y. Dickalov: Speech enhancement by harmonic modeling via MAP pitch tracking, Proc. IEEE ICASSP (2002) pp. 3316–3319

10.55 S. Godsill, M. Davy: Bayesian harmonic models for musical pitch estimation and analysis, Proc. IEEE ICASSP (2002) pp. 1769–1772

10.56 C.A. McGonegal, L.R. Rabiner, A.E. Rosenberg: A subjective evaluation of pitch detection methods using LPC synthesized speech, IEEE Trans. Acoust. Speech Signal Process. **25**, 221–229 (1977)

10.57 C. Hamon, E. Moulines, F. Charpentier: A diphone synthesis system based on time-domain prosodic modifications of speech, Proc. IEEE ICASSP (1989) pp. 238–241

10.58 D.M. Howard: Peak-picking fundamental period estimation for hearing prostheses, J. Acoust. Soc. Am. **86**, 902–910 (1989)

10.59 I. Dologlou, G. Carayannis: Pitch detection based on zero-phase filtering, Speech Commun. **8**, 309–318 (1989)

10.60 W.J. Hess: An algorithm for digital time-domain pitch period determination of speech signals and its application to detect F0 dynamics in VCV utterances, Proc. IEEE ICASSP (1976) pp. 322–325

10.61 T.V. Ananthapadmanabha, B. Yegnanarayana: Epoch extraction of voiced speech, IEEE Trans. Acoust. Speech Signal Process. **23**, 562–569 (1975)

10.62 L.O. Dolanský: An instantaneous pitch-period indicator, J. Acoust. Soc. Am. **27**, 67–72 (1955)

10.63 I.S. Howard, J.R. Walliker: The implementation of a portable real-time multilayer-perceptron speech fundamental period estimator, Proc. EUROSPEECH-89 (1989) pp. 206–209, http://www.isca-speech.org/archive/eurospeech_1989

10.64 W.J. Hess: A pitch-synchronous digital feature extraction system for phonemic recognition of speech, IEEE Trans. Acoust. Speech Signal Process. **24**, 14–25 (1976)

10.65 A. Davis, S. Nordholm, R. Togneri: Statistical voice activity detection using low-variance spectrum estimation and an adaptive threshold, IEEE Trans. Audio Speech Lang. Process. **14**, 412–424 (2006)




Amazon EX1010 Page 238

10.66 L.J. Siegel, A.C. Bessey: Voiced/unvoiced/mixed excitation classification of speech, IEEE Trans. Acoust. Speech Signal Process. **30**, 451–461 (1982)

10.67 S. Ahmadi, A.S. Spanias: Cepstrum-based pitch detection using a new statistical V/UV classification algorithm, IEEE Trans. Speech Audio Process. **7**, 333–338 (1999)

10.68 B.M. Lobanov, M. Boris: Automatic discrimination of noisy and quasi periodic speech sounds by the phase plane method, Soviet Physics – Acoustics 16, 353–356 (1970) Original (in Russian) in Akusticheskiy Zhurnal **16**, 425–428 (1970)

10.69 E. Fisher, J. Tabrikian, S. Dubnov: Generalized likelihood ratio test for voiced-unvoiced decision in noisy speech using the harmonic model, IEEE Trans. Audio Speech Lang. Process. **14**, 502–510 (2006)

10.70 B.S. Atal, L.R. Rabiner: A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition, IEEE Trans. Acoust. Speech Signal Process. **24**, 201–212 (1976)

10.71 O. Fujimura: An approximation to voice aperiodicity, IEEE Trans. Acoust. Speech Signal Process. **16**, 68–72 (1968)

10.72 A.K. Krishnamurthy, D.G. Childers: Two-channel speech analysis, IEEE Trans. Acoust Speech Signal Process. **34**, 730–743 (1986)

10.73 K.N. Stevens, D.N. Kalikow, T.R. Willemain: A miniature accelerometer for detecting glottal waveforms and nasalization, J. Speech Hear. Res. **18**, 594–599 (1975)

10.74 V.R. Viswanathan, W.H. Russell: *Subjective and objective evaluation of pitch extractors for LPC and harmonic-deviations vocoders* (Bolt Beranek and Newman, Cambridge 1984), MA: Report No. 5726

10.75 A.J. Fourcin, E. Abberton: First applications of a new laryngograph, Med Biol Illust **21**, 172–182 (1971)

10.76 D.G. Childers, M. Hahn, J.N. Larar: Silent and voiced/Unvoiced/Mixed excitation (four-way) classification of speech, IEEE Trans. Acoust. Speech Signal Process. **37**, 1771–1774 (1989)

10.77 E. Mousset, W.A. Ainsworth, J.A.R. Fonollosa: A comparison of several recent methods of fundamental frequency and voicing decision estimation, Proc. ICSLP'96 (1996) pp. 1273–1276, http://www.isca-speech.org/archive/icslp_1996

10.78 D.A. Krubsack, R.J. Niederjohn: An autocorrelation pitch detector and voicing decision with confidence measures developed for noise-corrupted speech, IEEE Trans. Signal Process. **39**, 319–329 (1991)

10.79 Y. Xu, X. Sun: Maximum speed of pitch change and how it may relate to speech, J. Acoust. Soc. Am. **111**, 1399–1413 (2002)

10.80 B.G. Secrest, G.R. Doddington: Postprocessing techniques for voice pitch trackers, Proc. IEEE ICASSP (1982) pp. 172–175

10.81 F. Plante, G.F. Meyer, W.A. Ainsworth: A pitch extraction reference database, Proc. EUROSPEECH'95 (1995) pp. 837–840, http://www.isca-speech.org/archive/eurospeech_1995

10.82 H. Kawahara, H. Katayose, A. de Cheveigné, R.D. Patterson: Fixed point analysis of frequency to instantaneous frequency mapping for accurate estimation of F0 and periodicity, Proc. EUROSPEECH'99 (1999) pp. 2781–2784, http://www.isca-speech.org/archive/eurospeech_1999

10.83 L.R. Rabiner, M.R. Sambur, C.E. Schmidt: Applications of nonlinear smoothing algorithm to speech processing, IEEE Trans. Acoust. Speech Signal Process. **23**, 552–557 (1975)

10.84 P. Specker: A powerful postprocessing algorithm for time-domain pitch trackers, Proc. IEEE ICASSP (1984), paper 28B.2

10.85 F. Itakura: Minimum prediction residual applied to speech recognition, IEEE Trans. Acoust. Speech Signal Process. **23**, 67–72 (1975)

10.86 Y.R. Wang, I.J. Wong, T.C. Tsao: A statistical pitch detection algorithm, Proc. IEEE ICASSP (2002) pp. 357–360

10.87 Y. Sagisaka, N. Campbell, N. Higuchi (eds.): *Computing prosody. Computational models for processing spontaneous speech* (Springer, New York 1996)

10.88 P. Bagshaw: *Automatic prosodic analysis for computer aided pronunciation teaching* (Univ. of Edinburgh, Edinburgh 1993), PhD Thesis http://www.cstr.ed.ac.uk/projects/fda/Bagshaw_PhDThesis.pdf

10.89 R.J. Baken: *Clinical Measurement of Speech and Voice* (Taylor Francis, London 1987)

10.90 A. Askenfelt: Automatic notation of played music: The Visa project, Fontes Artis Musicae **26**, 109–120 (1979)

10.91 Y.M. Cheng, D. O'Shaughnessy: Automatic and reliable estimation of glottal closure instant and period, IEEE Trans. Acoust. Speech Signal Process. **37**, 1805–1815 (1989)

10.92 W.J. Hess: Determination of glottal excitation cycles in running speech, Phonetica **52**, 196–204 (1995)

10.93 W.J. Hess: Pitch determination of acoustic signals – an old problem and new challenges, Proc. 18th Intern. Congress on Acoustics, Kyoto (2004), paper Tu2.H.1

10.94 B. Yegnanarayana, R. Smits: A robust method for determining instants of major excitations in voiced speech, Proc. IEEE ICASSP (1995) pp. 776–779

10.95 M. Brookes, P.A. Naylor, J. Gudnason: A quantitative assessment of group delay methods for identifying glottal closures in voiced speech, IEEE Trans. Audio Speech Language Process. **14**, 456–466 (2006)

10.96 C.X. Ma, Y. Kamp, L.F. Willems: A Frobenius norm approach to glottal closure detection from the speech signal, IEEE Trans. Speech Audio Process. **2**, 258–265 (1994)




Amazon EX1010 Page 239

10.97 L. Du, Z. Hou: Determination of the instants of glottal closure from speech wave using wavelet transform, http://www.icspat.com/papers/329mfi.pdf
10.98 K.E. Barner: Colored L-$\ell$ filters and their application in speech pitch detection, IEEE Trans. Signal Process. **48**, 2601–2606 (2000)
10.99 J.L. Navarro-Mesa, I. Esquerra-Llucià: A time-frequency approach to epoch detection, Proc. EUROSPEECH'95 (1995) pp. 405–408, http://www.isca-speech.org/archive/eurospeech_1995
10.100 A. de Cheveigné: Separation of concurrent harmonic sounds: Fundamental frequency estimation and a time-domain cancellation model of auditory processing, J. Acoust. Soc. Am. **93**, 3279–3290 (1993)
10.101 A.P. Klapuri: *Signal processing methods for the automatic transcription of music* (Tampere Univ. Technol., Tampere 2004), Ph.D. diss. http://www.cs.tut.fi/sgn/arg/klap/klap_phd.pdf
10.102 M. Goto: A predominant F0-estimation method for polyphonic musical audio signals, Proc. 18th Intern. Congress on Acoustics Kyoto (2004), paper Tu2.H.4
10.103 T. Tolonen, M. Karjalainen: A computationally efficient multipitch analysis model, IEEE Trans. Speech Audio Process. **8**, 708–716 (2000)
10.104 H. Kameoka, T. Nishimoto, S. Sagayama: Separation of harmonic structures based on tied Gaussian mixture model and information criterion for concurrent sounds, Proc. IEEE ICASSP (2004), paper AE-P5.9
10.105 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc. B **39**, 1–38 (1977)
10.106 L. Yoo, I. Fujinaga: A comparative latency study of hardware and software pitch-trackers, Proc. 1999 Int. Computer Music Conf. (1999 ) pp. 36–40



Amazon EX1010 Page 240

# 11. Formant Estimation and Tracking

D. O'Shaughnessy

This chapter deals with the estimation and tracking of the movements of the spectral resonances of human vocal tracts, also known as formants. The representation or modeling of speech in terms of formants is useful in several areas of speech processing: coding, recognition, synthesis, and enhancement, as formants efficiently describe essential aspects of speech using a very limited set of parameters. However, estimating formants is more difficult than simply searching for peaks in an amplitude spectrum, as the spectral peaks of vocal-tract output depend upon a variety for factors in complicated ways: vocal-tract shape, excitation, and periodicity. We describe in detail the formal task of formant tracking, and explore its successes and difficulties, as well as giving reasons for the various approaches.

11.1 **Historical** ..... 213
11.2 **Vocal Tract Resonances** ..... 215
11.3 **Speech Production** ..... 216
11.4 **Acoustics of the Vocal Tract** ..... 218
11.4.1 Two-Tube Models for Vowels ..... 218
11.4.2 Three-Tube Models for Nasals and Fricatives ..... 219
11.4.3 Obstruents ..... 220
11.4.4 Coarticulation ..... 220
11.5 **Short-Time Speech Analysis** ..... 221
11.5.1 Vowels ..... 221
11.5.2 Nasals ..... 221
11.5.3 Fricatives and Stops ..... 222
11.6 **Formant Estimation** ..... 223
11.6.1 Continuity Constraints ..... 223
11.6.2 Use of Phase Shift ..... 224
11.6.3 Smoothing ..... 225
11.7 **Summary** ..... 226
**References** ..... 226

## 11.1 Historical

In this chapter, we deal with a focused problem of speech analysis – trying to identify some very specific aspects of speech that have been found to be of great use in a wide variety of speech applications. These parameters of speech are called formants. They are generally viewed to be the resonances of the vocal tract (VT), which often appear in spectral displays (such as spectrograms) as regions of high energy, slowly varying in time as the vocal tract moves (Fig. 11.1).

Formants are useful in the coding, recognition, synthesis, and enhancement of speech, as they efficiently describe essential aspects of speech using a very limited set of parameters. For coding, if speech can be reduced to formant parameters to represent the VT shape (and a few other parameters to represent VT excitation), then very efficient coding is possible [e.g., 2.4 kbps linear predictive coding (LPC)]. Standard coders (e.g., in cellphone technology) use LPC with 10 or so coefficients to characterize the VT; it is feasible instead to represent the VT with even fewer parameters if formants are properly employed.

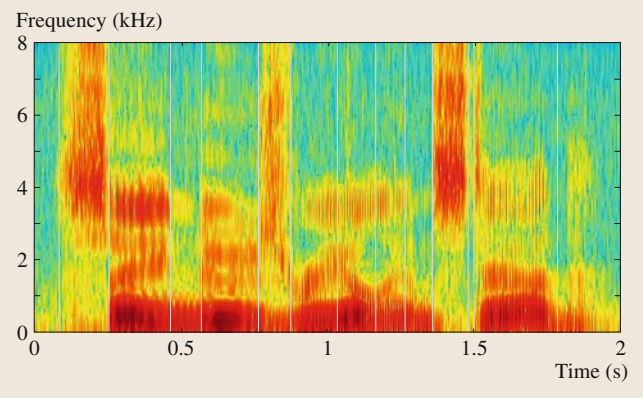



**Fig. 11.1** Wideband spectrogram of an adult male speaker saying 'Say newsreel instead'. *Light vertical lines* (added manually) denote phoneme boundaries



Amazon EX1010 Page 241

In the 1970s, formants were the primary focus of automatic speech recognizers (ASRs), as it is necessary in ASR (as in coding) to greatly reduce the information present in a speech signal (e.g., 64 kbps for toll-quality speech in the basic telephone network), without sacrificing useful information about VT shape [11.1]. As many studies of human speech production and perception have identified formants as prime candidates to represent the VT spectrum efficiently, they were popular parameters to try to estimate from speech signals. It was found, however, that tracking formants reliably was not an easy task. Since the mid-1980s, ASR has primarily relied on the mel-frequency cepstral coefficients (MFCCs) instead of formant-based parameters to represent VT information. The advantage of the MFCC approach has been an automatic way to reduce the amount of information in a Fourier transform (FT) of a frame of speech (which is always assumed to reasonably capture the essential information about VT shape at any specific point in time; a frame is a short section of speech, e.g., 20 ms) to a small set of parameters, e.g., 10–16. The data reduction factor is about the same as for LPC, except that the MFCC is able to utilize some auditory factors in warping frequency scales to model the human ear better than LPC can (i. e., a mel or bark scale: linear up to 1 kHz, and logarithmic thereafter). Nonetheless, there still remains interest in formants for ASR purposes, as MFCC and LPC tend to suffer significantly when increasing amounts of noise are present in the received speech signal. Both MFCC and LPC take global approaches to speech analysis, which makes it difficult to separate noise from speech in corrupted signals.

The MFCC and LPC have been so popular for speech recognition and coding, respectively, because they are parameters that are obtained by simple mathematical rules (algorithmic transformations not subject to discrete, and hence nonlinear, decisions). Such transformations reduce the size of speech representations. A Fourier display does little data reduction as it occupies about as much data space as the untransformed speech. The MFCC and LPC are more compressed, but still leave room for further compression, as is possible with formants. *Features* such as formants must be estimated using error-prone methods, as they can only be obtained by applying (possibly faulty) decisions in the data reduction process. They achieve greater data reduction, at the cost of making errors. Given the widespread use of MFCC for ASR, some formant trackers employ the MFCC as the spectral input to their algorithms [11.2, 3], even though the MFCC tend to smooth spectra in ways that may obscure complicated formant structure. (Because ASRs using MFCCs rarely track formants, any formant tracking errors that might result from such an approach would not be pertinent for ASR, but it is nonetheless useful to see how well one can track formants using such a popular spectral estimation method.)

If instead we return to an original (possibly noisy) spectrum and properly examine the short-time Fourier transform (STFT) to find formants amid any interfering noise, there is still potential to carry out better ASR with formants [11.4]. Recently, formants have seen increasing use in text-to-speech (TTS) synthesis applications [11.5], as the trend has been to employ very large databases of small speech units (extracted from the speech of a single professional training speaker). In TTS, given a text to pronounce, a phonetic sequence is automatically determined, allowing access to these units. A major issue in recent research has been the efficient determination of which units to use, which requires searching a large space of alternatives (typically, through a Viterbi search) making many cost estimations about the spectral similarity between units. If these units are characterized via formants, these costs are often easier to compute than if other parameters are used.

Finally, aids for people with speech difficulties are often designed around the essential aspects or features of speech such as formants. In some speech aids, information additional to the normal speech input may be available to assist in the formant tracking task. Attaching a laryngeal sensor to a speaker's throat can provide detailed information about the VT excitation in a relatively unobtrusive fashion, and appears to assist in some formant tracking methods [11.6].

Certainly, a visual display capturing the mouth of the speaker corresponding to the input speech can help both ASR and formant tracking, as such imagery yields important information about (at least) mouth opening. As most speech applications do not have access to imagery, but only to the speech itself, we will not assume image assistance in this section. There are several formant trackers in commercial applications; a common one freeware package is Wavesurfer [11.7].



Amazon EX1010 Page 242

## 11.2 Vocal Tract Resonances

While formants are commonly viewed as peaks in the speech spectrum, we must be more rigorous in defining them, as spectral displays of speech vary greatly in the number and types of peaks seen (Fig. 11.2). Formants are usually understood to be broad spectral peaks in an STFT of speech, corresponding to the underlying vocal tract resonances (VTRs). Such basic resonances can often be calculated from VT area functions, if available, e.g., via X-rays or electromyography. Few speech applications have access to such data, however, and thus resonance estimations must usually be based on analysis of the speech coming from the mouth. However, such resonances are only present in output speech energy to the extent that the VT system is excited with sufficient energy at those frequencies. We must recall that speech output is the convolution of the VT excitation waveform and the impulse response of the VT (or, equivalently, the speech spectrum is the product of these two spectral representations), and thus the nature of the VT excitation is a major factor in whether VTRs form visible peaks in the output speech STFT.

As we are primarily interested in formant estimation during strong sonorants, we first look at VT excitation for vowels, where the excitation consists of glottal puffs of air modulated by the vocal folds with a low-pass nature (about −12 dB/octave fall-off). As a result, the first formant ($F_1$) normally has the highest intensity, and the amplitude decreases at about −6 dB/octave for formants at higher frequencies (−6 is the net result, after including the +6 dB/octave radiation effect at the lips). Spectrograms often show clear bands of formant energy for the first 3–5 formants in vowels, but this situation varies greatly depending on the spectrum of each vowel [e.g., /u/ has a greater fall-off, with little energy above $F_2$ (second formant), as its $F_1$ and $F_2$ are quite low in frequency, whereas /i/, with a much higher frequency $F_2$, has much more energy in the $F_3$–$F_4$ region]. The style of speech also has effects: breathy voice has a greater rate of fall-off, and hence weaker formants at high frequencies, than a shouting voice. Few speech applications have shown interest in the estimation of formants above $F_4$, as higher formants are quite weak (especially relative to any background noise) and have much less perceptual relevance than $F_1$–$F_4$. Indeed, estimations of $F_3$ and $F_4$ are often difficult owing to the low energy in their frequency ranges (e.g., for back vowels such as /u/).

Issues of weak formants are not limited to high frequencies. Nonvowel phonemes, such as nasals and obstruents, partly owing to the presence of zeros in their spectra, have varied spectral displays in terms of which resonances are visible. In the case of obstruents, a major additional factor is the different nature and location of the noise excitation, which is much higher in the VT, hence exciting only higher resonances, as will be

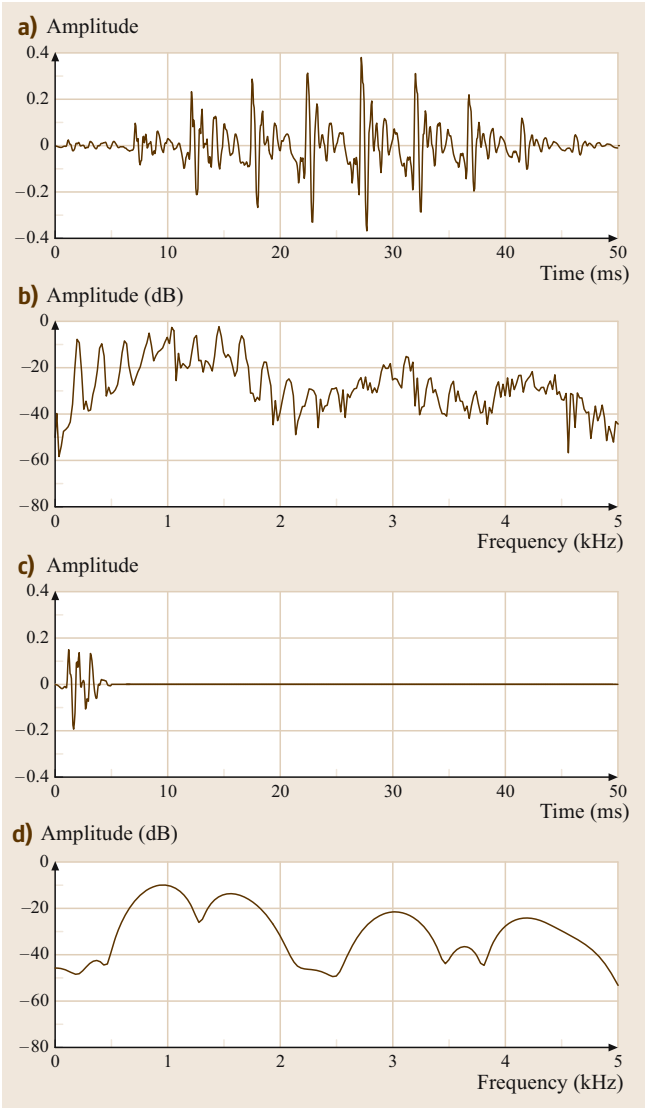



**Fig. 11.2a–d** Time signals and spectra of a vowel. **(a)** Speech signal weighted by a 50 ms Hamming window, **(b)** the corresponding spectrum, **(c)** speech signal weighted by a 5 ms Hamming window, and **(d)** the corresponding spectrum



Amazon EX1010 Page 243

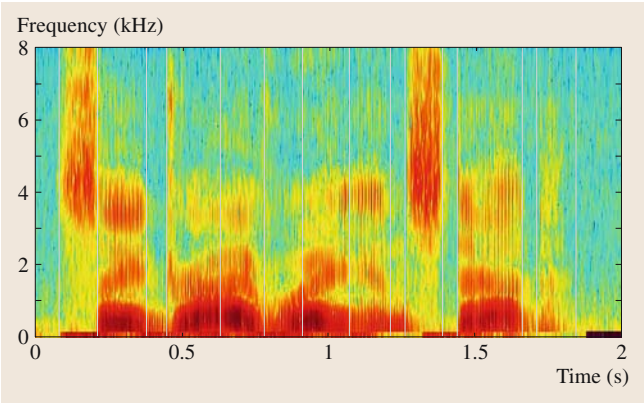



**Fig. 11.3** Wideband spectrogram of the same speaker saying 'Say driveway instead'

discussed later. Thus, the varying display of bands of energy that we wish to associate with VTRs is a major factor in the difficulty of formant tracking, and not just for a minority of speech sounds.

Part of the difficulty of formant estimation and tracking has been not properly exploiting much of what is known about human speech production and perception. This failing has occurred too often in the speech research field (e.g., for many years, ASR ignored the fact that speech had an underlying message component; it was only in the 1980s that language models were commonly used for ASR). A major difficulty of tracking formants is their tendency in spectral displays to weaken, disappear, and/or merge (Figs. 11.1–11.3), as the VT undergoes changes in shape or as the VT excitation changes. As speech is inherently dynamic (e.g., 12 phonemes/second on average), the ability to track temporal changes in any parametric representation (e.g., formants) is essential. While such changes (e.g., abrupt appearance or disappearance of formants, as well as formants approaching each other so closely as to appear as one band of energy in time displays) are readily apparent, the underlying resonances of the VT are much more well behaved, albeit harder to observe. VTRs derive directly from VT shape, and are the poles of the VT transfer function. As the VT moves smoothly from one position to another, the VTRs smoothly change values [11.8]. Even when occlusions are made or released (as in stop consonants), the positions of the VTRs change smoothly. As such, VTRs would be much easier to track than formants (whose appearance on spectrograms is more varied). However, lacking visual displays of the VT (e.g., a camera focused on the speaker's mouth, or a radiographic image of the inside of the mouth), which is normally the case in almost all speech applications, we must rely on formant rather than VTR tracking. We can, nonetheless, exploit presumed knowledge about VTRs when estimating formants.

## 11.3 Speech Production

To better understand the foundation for formants, let us examine some aspects of human speech production. In any estimation task, it is often useful to examine important properties of the data source, rather than attack a problem with few assumptions. Too often in the past, research on formant estimation simply viewed STFTs, looked for bands of energy, and applied basic peak-picking approaches (with some constraints on continuity), without further regard for the nature or origin of the speech signal. It is true that the essential information about formants (for most speech applications) is readily available in such displays, but such an approach is like doing ASR without a language model. An essential step of any pattern recognition task, whether complete ASR or simpler formant tracking, is to reduce the input data while minimizing loss of useful information; this *preprocessing* step not only makes the process more efficient (as subsequent recognition steps treat smaller amounts of data), but also allows more-accurate estimation decisions, by focussing more closely on aspects of the data that are crucial (for the task at hand). In many cases of pattern recognition, an underlying source undergoes several (often nonlinear) mappings before a data acquisition device can capture a readily observable signal. A better understanding of these mappings can aid the parameter estimation process. In our case, the basic underlying source is a message in a speaker's brain. ASR attempts to recover that message (e.g., a text), given the observed speech signal. More specifically here we are concerned with identifying certain intermediate parameters – the formants – in this process. In any event, better knowledge about where formants originate is useful.

In human speech production, the lungs (as a pressure source) push air past the vocal folds, which often modulate the airflow to create the excitation sound for

the VT, which in turn acts as a filter to amplify certain frequencies while attenuating others. The VT is the most important component in speech production, as it controls the formants and its variation leads directly to the perception (by listeners) of the individual phonemes (sounds) of speech. A tubular passageway composed of muscular and bony tissues, the VT, via its shape, modifies the spectral distribution of energy in glottal sound waves. Different sounds of speech are primarily distinguished by their periodicity (voiced or unvoiced), spectral shape (which frequencies have the most energy), and duration. The vocal folds specify the voicing feature, but the major partitioning of speech into sounds is accomplished by the VT via spectral filtering. The VT is often modeled as an acoustic tube with resonances called formants, but it can have antiresonances as well. In $z$-transform terminology, each resonance corresponds to a complex conjugate pair of poles, whose angular position corresponds to the center frequency of the resonance (usually, we refer to this as the formant value) and whose radial position corresponds to the resonance bandwidth. (In formant tracking, the amplitudes of the resonance peaks are usually of considerably less interest than the center frequencies and bandwidths, as amplitudes are greatly influenced by the spectral tilt of the glottal excitation, as well as by the proximity of nearby formants; listeners seem to take account of this by utilizing rather little relative peak amplitude information in speech perception.) The formants are often abbreviated as $F_i$, e.g., $F_1$ is the formant with the lowest frequency. ($F_0$ is associated with the fundamental frequency of the vocal folds during voiced speech, and is not a resonance. $F_0$ is quite visible in narrowband spectrograms, as speech consists of $F_0$ and its many harmonics, modulated by the VT transfer function, i. e., the spectral envelope.)

Figure 11.2 shows displays of the amplitude of an STFT for a typical vowel. If, as is usual, we use a time window (i. e., the length of the STFT) of about 20–25 ms, perhaps shaped by a Hamming window to reduce edge effects [11.1], the display will show a large amount of detail, corresponding to many aspects of the windowed speech signal. (Typical FFT lengths are 256, 512, or 1024 samples, which correspond to common frame durations at sampling rates around 16 kHz; as a result, sample points in the STFT are about 30 Hz apart, which means that each harmonic has a few values, and the STFT is highly variable in frequency.) Most notable are the harmonics, peaks spaced every $F_0$ Hz (Fig. 11.2b), which correspond to multiples of the vocal fold vibration rate. (These equally spaced peaks have finite width, corresponding to the inverse of the window duration; if one were to theoretically reiterate a vowel's pitch period indefinitely, we would see an actual line spectrum.) Superimposed on this base periodic set of spectral lines is the VT transfer function or spectral envelope, as the STFT is the product of the excitation spectrum ($F_0$ lines) and the VT filter response. It is this spectral envelope, and more particularly its formant peaks, that interest us the most in this Section. Note, however, that the formants are not immediately accessible in the FT, as the spectral envelope is essentially only specified at multiples of $F_0$. Typically, one may smooth the STFT to render the formants more evident.

Another way to suppress the spectral effects of the (interfering) harmonics is to use an STFT with a shorter time window (i. e., one shorter than the pitch period) (Fig. 11.2d). A disadvantage of this approach is that the shorter window needs to be placed synchronously with the pitch period for optimal formant estimation, and any corrupting noise in the speech signal has greater negative effects on formant estimation, as a much shorter set of speech data is used (owing to the shorter window).

Antiresonances correspond to zeros in the output speech, and occur owing to aspects of glottal excitation or to the existence of side-branches in the airflow path in the VT. This section examines formants, which correspond to the resonances. As such, we will not explicitly try to track the zeros, which are considerably more difficult to locate reliably (very recently, particle filtering has been used to track the poles and zeros of the VT [11.9]). Luckily, zeros are much less relevant for most speech applications.

Figures 11.1 and 11.3 show typical wideband spectrograms. Many speech applications make reference to such spectrograms, although few actually include these three-dimensional displays explicitly in their speech analysis (e.g., LPC and the MFCC are much more commonly used in coding and in ASR, respectively). They clearly show the formant bands of energy (where darkness indicates intensity) as a function of both time and frequency. In wideband spectrograms, use of a short time window yields much better time resolution (e.g., about 3 ms, making the vocal fold closure excitations readily visible as increases in energy every pitch period). As a result, frequency is typically smoothed automatically over a range of 300 Hz (roughly the inverse of 3 ms, and chosen to exceed most people's $F_0$, and hence include two or more harmonics within the window's low-pass filter range of smoothing). In such a dis-



Amazon EX1010 Page 245

play, the time signal is covered roughly continuously, thus obviating the need for pitch-synchronous analysis; a frame-based analysis method, as in ASR applications, would, on the other hand, need to address that issue.

## 11.4 Acoustics of the Vocal Tract

Speech is produced when air passes through the VT, which can be modeled as an acoustic tube of variable cross-sectional area $A(x, t)$, approximately closed at the glottal end and roughly open at the lips [$A$ varies in space ($x = 0$ at the glottis and $x = L$ at the lips; $L$ will be assumed to be 17 cm) and in time $t$]. VT length varies greatly among speakers (about 13 cm for women, and less for children); we choose 17 cm as the nominal value here as this is an average value for men, and yields simple values for the average positions of the formants: 500 Hz, 1500 Hz, 2500 Hz, 3500 Hz, ... etc. (as we will see later). Glottal area is small relative to typical $A$ values, although the glottal end of the VT is truly closed only during glottal stops and during the closed phases of voicing. Lip rounding or closure often narrows the acoustic tube at the lips. A tube (e.g., the VT) closed at one end and open at the other resembles an organ pipe and is called a quarter-wavelength resonator, as the frequencies at which the tube resonates are those where sound waves traveling up and down the tube reflect and coincide at the ends of the tube. VTRs can be heuristically computed using only the boundary conditions of the VT, along with the phase relationship between the pressure and volume velocity in the traveling sound waves. Formant frequencies match the boundary conditions for pressure $P$ (relative to atmospheric pressure) and volume velocity $U$: a closed end of the tube makes $U = 0$, whereas $P \approx 0$ at an open end. $P$ is 90° out of phase with $U$, owing to the inductance and capacitance of the VT. Resonances occur at frequencies $F_i$, $i = 1, 2, 3, \ldots$, where $|U|$ is maximum at the open end of the VT and $|P|$ is maximum at the closed end. Such frequencies have wavelengths where the VT length $l$ is an odd multiple of a quarter-wavelength; hence, at 500 Hz, 1500 Hz, 2500 Hz, ... etc..

A uniform VT is only a good model for a schwa vowel /ə/. With other sounds, $A(x)$ is a complicated function of space along the VT, and as a result, the formants move to other frequencies. Normally, the deviations are within a range of a few (or several) hundred Hz. Thus, for a 17 cm VT, $F_1$ is usually in the range 300–800 Hz; $F_1$ tends to be low when the VT is relatively closed (e.g., for most consonants, and for vowels with a raised tongue – high vowels), and high for low vowels. $F_2$ is usually in the range 700–2200 Hz; $F_2$ tends to be high when the tongue is relatively forward, and low when the tongue is more to the rear. In general, the range of $F_2$ is greater than for the other formants. $F_3$ is usually in the range 1800–2800 Hz; $F_3$ tends to be high when the tongue is relatively forward and high, and low when the tongue is retroflexed (as in /r/). Higher formants are usually progressively weaker in intensity, and less relevant for most applications. (Synthesizers that use formants often fix $F_4$ near 3500 Hz and $F_5$ near 4500 Hz; variations in these numbers appear to have little useful perceptual effect.) Thus, formant estimators focus on tracking $F_1$–$F_3$.

Almost all languages employ the *cardinal* vowels of /i/, /a/, /u/, and these three often represent VT shapes that are far from a neutral schwa (uniform VT) shape, i. e., other vowels (if present for a given language) have more intermediate shapes, and therefore formant frequencies that are closer to the neutral values (500, 1500, 2500 Hz). Tracking formants is typically more difficult for more extreme VT shapes, as some formants often appear to merge in such cases. (Examples are seen in Figs. 11.1 and 11.3, where formants change intensity suddenly and make apparent jumps, usually at phoneme boundaries, but certainly not at all phoneme boundaries.) Thus, let us examine further how VT shape relates to such vowels. The vowel /a/ can be roughly modeled by a (lower) narrow tube (representing the pharynx) opening relatively abruptly into a wide (upper) tube (the oral cavity). Assuming a 17 cm VT, for simplicity suppose that each tube has a length of 8.5 cm; then, each tube would produce the same set of resonances, at odd multiples of 1 kHz (1000, 3000, 5000 Hz, ... ). Each tube is a quarter-wavelength resonator, since its back end is relatively closed and its front end is relatively open (i. e., the two tubes are half-versions of the full schwa VT, and thus have formants at twice the original values). As with all 17 cm models of the VT, this two-tube model has the same number (one) of formants per kHz, on average, but each one is moved by 500 Hz.

### 11.4.1 Two-Tube Models for Vowels

At any boundary between sections of the VT, whenever the change in areas is sufficiently abrupt (as with /a/, near the velum), the acoustic coupling between cavities



Amazon EX1010 Page 246

is small and the interaction between cavity resonances will be slight; each section then controls its own number of the overall set of formants. Due to some acoustic coupling, formants never approach each other by less than about 200 Hz; thus, e.g., $F_1$ and $F_2$ for /a/ are not both at 1000 Hz, but rather near these values: $F_1 = 900$ Hz, $F_2 = 1100$ Hz, $F_3 = 2900$ Hz, and $F_4 = 3100$ Hz. The reverse holds for /i/: a wide pharyngeal tube narrowing abruptly into the oral cavity tube. Theory would have $F_1 = 100$ Hz, $F_2 = 1900$ Hz, $F_3 = 2100$ Hz, and $F_4 = 3900$ Hz, but in practice, $F_1$ is closer to 280 Hz, and $F_4$ often approaches $F_3$ (making a group of 2–3 formants around 2 kHz). Actual observed values for real versions of /a/ and /i/ will show deviations from these model numbers due to modeling inaccuracies; nonetheless, these simple models give reasonably accurate results and are easy to interpret physically.

The third cardinal vowel /u/ has a more-complicated analysis: anytime that the lips are rounded (as in /u/), all formants lower in frequency. This can be roughly seen by considering a VT mostly closed at both ends; the VT then becomes a half-wavelength resonator, as its boundary conditions are the opposite of the normal open-mouth model (e.g., the volume velocity is minimized at both (closed) ends of the VT, whereas it is maximized at the mouth in the open-mouth version). The theoretical locations for such a VT model are $F_1 = 0$, $F_2 = 1000$ Hz, $F_3 = 2000$ Hz, etc. Again, in practice, $F_1$ rarely goes so low, and approaches 200 Hz as the lips are almost closed. Examining such an analysis, one may be tempted to conclude that formants do not deviate more than about 400 Hz from the nominal neutral values of 500, 1500, 2500 Hz. Empirical evidence, however, shows that $F_2$, in particular, has a rather wider range (and $F_3$ goes as low as 1900 Hz). Again, all values given here assume a typical man's VT; for people of other sizes, one generally scales frequency values by the average length of the VT, relative to 17 cm; as VTs are not linear versions of each other, there may be significant deviations from such simple modeling.

### 11.4.2 Three-Tube Models for Nasals and Fricatives

We cannot easily extend the simple results of the two-tube VT model to more-complicated VT shapes for other phonemes. Nonetheless, some approximate extensions are feasible, and help understanding of the task of formant estimation. Nasal consonants require analysis of a three-tube VT model, as the nasal cavity is involved when the velum is lowered for such consonants. The entire system can be modeled as three tubes (pharyngeal, nasal, and oral) joined at one point (the velum), with acoustic circuits for the three in parallel. The poles of such a model are specified by [11.1]: $1/Z_\mathrm{p} + 1/Z_\mathrm{m} + 1/Z_\mathrm{n} = 0$, where $Z_\mathrm{p} = -\mathrm{i}Z_{0\mathrm{p}}\cot(\beta l_\mathrm{p})$, $Z_\mathrm{m} = -\mathrm{i}Z_{0\mathrm{m}}\cot(\beta l_\mathrm{m})$, and $Z_\mathrm{n} = \mathrm{i}Z_{0\mathrm{n}}\tan(\beta l_\mathrm{n})$. $Z_\mathrm{p}$, $Z_\mathrm{m}$, and $Z_\mathrm{n}$ refer to the acoustic impedance seen in the velar region of the VT, in the direction of the pharynx, nasal passages, and mouth, respectively, while $Z_{0\mathrm{p}}$, $Z_{0\mathrm{m}}$, and $Z_{0\mathrm{n}}$ are the characteristic impedances of these respective cavities. The mouth and pharyngeal tubes have closed acoustic terminations, while the nasal tube is open (at the nostrils). Dimensional similarity of the pharyngeal and nasal tubes allows a simplification: if $l_\mathrm{p} = l_\mathrm{n} = 10.5$ cm, each of $1/Z_\mathrm{p}$ and $1/Z_\mathrm{n}$ has periods of about 1.6 kHz and the function $1/Z_\mathrm{p} + 1/Z_\mathrm{n}$ has infinite values about every 800 Hz.

The mouth tube for nasal consonants is often significantly shorter than the other tubes (e.g., 3–7 cm). As a result, nasal consonants are characterized by: (a) formants every 800 Hz (due to the longer pharynx+nasal tube than the normal pharynx+mouth tube), (b) wider formant bandwidths, and (c) zeros in the spectrum. When airflow from the lungs reaches the velum junction, it divides according to the impedances of the mouth and nasal tubes. Spectral zeros occur at frequencies where $Z_\mathrm{m} = 0$, which results in no airflow into the nasal tube and thus no nasal speech output. Solving $Z_\mathrm{m} = -\mathrm{i}Z_{0\mathrm{m}}\cot(2\pi F_i l_\mathrm{m}/c) = 0$ for $F_i$ yields zeros at odd multiples of $c/4l_\mathrm{m}$. The mouth tube for /m/ is about 7 cm, which gives zeros at 1.2, 3.6, 6.0 kHz, ... Shorter tubes for /n/ and /ŋ/, about 5 and 3 cm, respectively, mean fewer zeros below 5 kHz: only one each at 1.7 and 2.8 kHz, respectively. Besides the poles due to the pharynx and nasal cavities, which occur every 800 Hz, nasal spectra have *pole–zero pairs* due to the mouth cavity; i. e., each zero is close in frequency to a mouth cavity pole. In spectra, nasal consonants appear as sounds with relatively steady formants, weaker than for vowels and with less coarticulation, as movements within the VT are muffled by oral tract closure.

Unlike the sonorants, all obstruents (except the glottal /h/) have an excitation source in the (upper) oral cavity. Air passing through a narrow constriction there creates frication noise just in front of the opening, which excites the remainder of the oral cavity in front of the constriction. This is usually modeled as random (Gaussian) noise with an approximately flat spectrum. For voiced obstruents (e.g., /v/ and /z/), the noise is modulated to be pulse-like, as the vocal folds vibrate; in spectrograms, this adds a *voice bar* at very low fre-



Amazon EX1010 Page 247

quencies (0–150 Hz). This bar is weak enough not to be confused with formants, as well as being outside the range of $F_1$.

### 11.4.3 Obstruents

As noted above, formants are dynamic, varying greatly in time and intensity. They often come close together, and at other times fade in or out. Such changes are due to VT movements, and to the nature of VT excitation. Voiced excitation is glottal and usually relatively strong, but its intensity decreases with increasing frequency, owing to the low-pass nature of glottal puffs of air. As a result, a voiced speech spectrum falls off with frequency at about -6 dB/octave, which leads to weaker high-frequency formants. For most obstruent sounds (i. e., stops and fricatives) the VT excitation occurs much higher in the VT than at the glottis (except for the fricative /h/), and as a result, a much shorter VT is excited. This (often much) shorter VT leads to much higher resonances appearing in speech spectra, or more precisely, to the lower-frequency resonances being canceled by antiresonances of the back cavity of the VT. Many researchers do not refer to details of obstruent spectra as *formants* (i. e., they reserve the term formants for spectra of sonorant phonemes); to describe obstruents, they instead note general details such as the approximate cutoff frequency below which little energy occurs. The justification for this is that listeners pay much less attention to detailed aspects of the spectra in obstruents, and speakers appear to exercise less control over the positioning of resonances there. Hence, we focus our attention on formant estimation for sonorants.

### 11.4.4 Coarticulation

The motion of the VT during speech causes very dynamic patterns in formants, i. e., we often see rapid movements of formants. This phenomenon is called coarticulation, because the articulatory configurations of neighboring phonemes affect the articulation of each phoneme. Even though text is written with discrete letters, and phoneticians note that speech consists of a sequence of individual phonemes, the actual speech signal is without obvious phoneme boundaries, owing to coarticulation. Speakers smoothly move their VT from positions appropriate for each phoneme in turn, spending as much time on each phoneme as is deemed appropriate. As a result, it is often hard to segment speech into discrete units for analysis, as would be very useful in ASR. In some cases, such a division is easy, as when excitation changes abruptly with the start or end of vocal fold vibration (i. e., a voiced–unvoiced transition), or when the oral portion of the VT closes or opens (e.g., lip closure). More often, the transition between phonemes is more subtle, as the tongue and lips move between positions appropriate for successive phonemes.

Each phoneme has a nominal or *target* VT shape that a speaker would more or less assume if the sound were to be produced in isolation. In context, however, the speaker thinks ahead, and is continually moving various parts of the VT to accomplish the dynamic sequence of phonemes. Thus, coarticulation effects can extend over several phonemes; e.g., in the word 'strew' (/stru/, phonetically), the lips round in anticipation of the /u/ during the earlier /s/.

In many applications, it is useful to divide speech into segments of linguistic relevance, e.g., words, syllables, or phonemes. For example, for speech synthesis-by-rule (TTS), continuously spoken speech from a training speaker must be segmented into such units for storage (for later concatenation, as needed for a specified input text). As manual segmentation is tedious, forced alignment is often imposed on such training speech, given the assumed corresponding text. As a result, we have applications where formant tracking is simplified by having a priori knowledge of what phoneme sequence was actually spoken [11.10]. Such studies report quite high accuracy of forced alignment to phonemic boundaries in speech of a known text, to within about 40 ms [11.11]. In more-general ASR applications, of course, we do not know beforehand what was said, and formant estimation must rely on general principles.

During slow speech, the VT shape and type of excitation may not alter for periods of up to 200 ms. In these sections of speech, formant tracking is usually much easier, as the formants vary little in either position or amplitude. Most of the time, however, the VT changes more rapidly, as phonemes last, on average, about 80 ms. Coarticulation and changing $F_0$ can render each pitch period different from its neighbor. Nonetheless, a basic assumption of speech analysis is that the signal properties change relatively slowly with time. This allows examination of a short time window of speech (e.g., multiplying the speech signal by a Hamming window) to extract parameters presumed to remain fixed for the duration of the window. Most techniques thus yield parameters averaged over the course of the time window. To model dynamic parameters, we divide the signal into successive windows (ana-



Amazon EX1010 Page 248

lysis frames). Slowly changing formants in long vowels could allow windows as large as 100 ms without obscuring the desired parameters via averaging, but rapid events (e.g., stop releases) need short windows of about 5–10 ms to avoid averaging spectral transitions with the steadier spectra of adjacent sounds.

## 11.5 Short-Time Speech Analysis

A basic tool for spectral analysis is the spectrogram, which converts a two-dimensional speech waveform (amplitude versus time) into a three-dimensional pattern (amplitude/frequency/time). With time and frequency on the horizontal and vertical axes, respectively, amplitude is noted by the darkness of the display (Fig. 11.1). Peaks in the spectrum appear as dark horizontal bands. The center frequencies of these bands are generally considered to be the formant frequencies (subject to the discussion below of how to handle merged formants, i. e., single, wider bands that display two or more resonances that have come close for certain periods of time; e.g., in the middle of Fig. 11.3, $F_1$ and $F_2$ combine at very low frequencies in /w/). Voiced sounds cause vertical marks in the spectrogram due to an increase in the speech amplitude each time the vocal folds close. The noise in unvoiced sounds causes rectangular dark patterns, randomly punctuated with light spots due to instantaneous variations in energy. Spectrograms portray only spectral amplitude, ignoring phase information, on the assumption that phase is less important for most speech applications. We will thus ignore phase in our discussion of formants.

In the spectrogram, the amplitude of the STFT $|S_n(e^{i\omega})|$ is plotted with time $n$ on the horizontal axis, frequency $\omega$ (from 0 to $\pi$) on the vertical axis (i. e., 0 to $F_s/2$ in Hz, $F_s$ being the sampling frequency), and with magnitude indicated as darkness, typically on a logarithmic scale (e.g., decibels). Two different display styles are typical: wideband and narrowband, with wideband displays used mostly for formant tracking.

Wideband spectrograms display individual pitch periods as vertical striations corresponding to the large speech amplitude each time the vocal cords close. Voicing is readily seen in the presence of these periodically spaced striations. Fine time resolution here permits accurate temporal location of spectral changes corresponding to VT movements. A wide filter bandwidth smooths the harmonic amplitudes under each formant across a range of (typically) 300 Hz, displaying a band of darkness (of width proportional to the formant's bandwidth) for each formant. The center of each band is a good estimate of formant frequency. Formant detectors generally prefer spectral representations that smooth the fine structure of the harmonics while preserving formant structure. Traditional wideband spectrograms use a window of about 3 ms, which corresponds to a bandwidth of 300 Hz and smooths harmonic structure (unless $F_0 > 300$ Hz, which occurs with children's voices). Narrowband spectrograms, on the other hand, generally use a window with a bandwidth of approximately 45 Hz and thus a duration of about 20 ms, which allows resolution of individual harmonics (since $F_0 > 45$ Hz) but smooths speech in time over a few pitch periods.

### 11.5.1 Vowels

Vowels are voiced and have the greatest intensity of all phonemes. They normally range in duration from 50 to 400 ms [11.1]. Vowel energy is mostly concentrated below 1 kHz and falls off at about −6 dB/oct with frequency. Spectral displays thus often use pre-emphasis to boost higher frequencies to facilitate formant tracking; e.g., LPC treats all frequencies the same, so LPC analysis without pre-emphasis would tend to have poorer spectral estimates at higher frequencies. Other formant methods that rely on peak-picking of spectra would likely have similar difficulties without pre-emphasis. As such boosting also raises the level of any background noise, it should be noted that formant estimation for high frequencies where noise dominates will necessarily be less accurate. Because of the −6 dB/oct fall-off, few formants above $F_4$ are reliable in many formant estimation methods. Pre-emphasizing the speech is usually done by differencing in discrete time:

$$y(n) = s(n) - as(n-1)\,,$$

where $a$ is typically 0.9–1.0. While this may greatly attenuate frequencies below 200 Hz, such low frequencies are rarely of interest in most speech applications. Vowels are distinguished primarily by the locations of their first three formant frequencies ($F_1$, $F_2$, and $F_3$).

### 11.5.2 Nasals

In nasal consonants, $F_1$ near 250 Hz dominates the spectrum, $F_2$ is usually very weak, and $F_3$ near 2200 Hz



Amazon EX1010 Page 249

has the second-highest formant peak. A spectral zero, whose frequency is inversely proportional to the length of the oral cavity behind the constriction, occurs near 1 kHz for /m/, near 2 kHz for /n/, and above 3 kHz for the velar nasal. Spectral jumps in both formant amplitudes and frequencies coincide with the occlusion and opening of the oral tract for nasals. These abrupt changes cause difficulties for the continuity constraints for formant trackers, as the usual trend toward smooth formant movements is invalid at nasal boundaries. VTRs change abruptly when the oral cavity opens or closes. The lowering of the velum is not the principal factor in the spectral change here; it often lowers during a vowel preceding a nasal consonant, which causes nasalization of the vowel, widening the formants and introducing zeros into the spectrum. Vowel nasalization primarily affects spectra in the $F_1$ region.

### 11.5.3 Fricatives and Stops

As obstruents, fricatives and stops are very different from sonorants: aperiodic, much less intense, and often with most energy at high frequencies. Obstruents may be either voiced or unvoiced. Unvoiced fricatives have a high-pass spectrum, with a cutoff frequency approximately inversely proportional to the length of the front cavity of the VT. Thus the palatal fricatives are most intense, with energy above about 2.5 kHz; they have a large front cavity. The alveolar fricatives (e.g., /s/) lack significant energy below about 3.2 kHz and are thus less intense. The labial and dental fricatives are very weak, with little energy below 8 kHz, due to a very small front cavity. The glottal fricative /h/, despite exciting the full VT, also has relatively low intensity as its noise source at the glottis (effectively a whisper) is usually weaker than noise from oral tract constrictions.

It is not obvious how to handle formant tracking for obstruents. Traditionally, formants are well defined only for sonorant sounds, where the general rules of strong resonances, spaced roughly every 1000 Hz, apply. Except for /h/, obstruents have little energy in the low-frequency range of 0–2 kHz, where strong $F_1$ and $F_2$ (for sonorants) have most energy. Depending on the length of the VT in front of the constriction noise source, there may be little energy in most of the useful auditory range. As noted above, VTRs are always present, no matter where the excitation is; however, low-frequency VTRs are not excited in most obstruents, and thus are not accessible in speech analysis.

In a transition from a sonorant to an obstruent (the observations that follow here also apply, in reverse, for a transition from an obstruent to a sonorant), the visible formants usually show movements from spectral positions pertinent for the sonorant toward targets for the ensuing obstruent (e.g., a decrease in $F_1$ as the VT closes; $F_2$ falling for a labial obstruent or rising for an alveolar; $F_3$ rising for an alveolar and falling for a velar). In some cases, for a given formant, a smooth formant transition (obeying continuity) is clearly seen; when this happens, it is usually for $F_3$ or $F_4$, as these tend to be in the frequency range of overlap between sonorants and obstruents. Thus continuity constraints should vary with context, and not be applied across all frequencies equally.

Tracking formants at stop transitions is particularly interesting and difficult. It is important because crucial phonetic information is present during these brief periods, which are major factors in informing listeners of the articulation point of the stop. Other major phonetic cues are much more prominent in the speech signal, e.g., over longer durations and with greater intensity. Stops, on the other hand, do not cue their place during most of their duration, as the VT closure at that time means the only audible energy is the voice bar (if voiced).

The release of the VT occlusion at the end of a stop creates a brief (few ms) explosion of noise, which tends to excite all frequencies. Then, turbulent noise (frication) continues as the constriction opens for 10–40 ms, exciting the front cavities (usually $F_2$–$F_4$), as the VT moves toward the position for an ensuing sonorant. A velar constriction provides a long front cavity, with a low resonance near 2 kHz ($F_2$ or $F_3$). Velar resonances are higher due to a shorter front cavity. The spectrum of a labial burst is relatively flat and weak since there is essentially no front cavity to excite.

Most formant estimators either do poorly during obstruents, or simply claim that such regions do not need formant estimation. A statement such as the latter is relatively true, as many speech applications have a primary need for spectral estimation during the strong sections of speech; weaker sounds may often be modeled more simply, and their perception by listeners may be less critical to the communication task at hand. Nonetheless, knowing some details about what happens during the weaker sounds is of interest. Identifying the articulation point is mostly done via the formant transitions in adjacent sonorants, and the formant behavior at the stop release (normally considered to be part of the obstruent, and not part of the ensuing sonorant) is relevant. Furthermore, weak fricatives such as /v/ require attention.



Amazon EX1010 Page 250

## 11.6 Formant Estimation

Typical methods to estimate formants involve searching for peaks in spectral representations, usually from an STFT or LPC analysis [11.12, 13]. As LPC imposes an assumed simplified structure on the speech spectrum, it appears to have been employed most often in recent methods. In most LPC applications, one uses two poles per kilohertz of bandwidth (plus 2–4 additional poles to model other factors, such as the spectral tilt, which is due to glottal effects), on the assumption that speech contains one such formant in that range. A spectrum derived from an LPC model of $N$=10–16 poles is much more limited in variation (across frequency) than an STFT, which thus simplifies peak-picking. A disadvantage of using LPC is that its all-pole modeling is not perfect [11.14]; it chooses its pole positions to minimize mean-square error (MSE) for a fit of the speech to an $N$-pole spectral envelope. If the number of poles is not well chosen (e.g., to match the number of resonances clearly present in the speech), then the model spectrum is not as accurate as desired. For example, if there are too few poles in the model, the poles (selected via automatic analysis) have to place themselves in compromise locations between actual formants, and significant errors in formant tracking will result. Thus, it is rare that too few poles are used in LP analysis. Use of too many poles is a more likely risk; it is standard to follow the rule of thumb given above (2 poles/kHz), yet a given speech spectrum will often display fewer resonances than other phonemes (e.g., /u/ vowels and nasals often only show 2–3 formants, even in wideband applications, as higher formants tend to be very weak and thus poorly modeled via LPC, which focuses on peak energy). In this case, the extra poles (i. e., those not needed to directly model the actual resonances) locate themselves at non-formant frequencies to reduce the MSE further. Sometimes, if there are only, say, two extra poles, the additional poles will be real or have very wide bandwidth (to model the speech better via some broad spectral tilt effect). As LPC formant trackers usually examine the bandwidths of the LPC poles and reject (as potential formants) poles with wide bandwidth, such an effect is not a major problem. More-serious errors can arise when the additional poles model individual harmonics, e.g., in the strong $F_1$ region, as may happen in cases where individual formants have few harmonics; if $F_1$ has two dominant harmonics (e.g., the $F_1$ center frequency is located between two harmonics and the $F_1$ bandwidth is similar to $F_0$), then LPC may well assign four poles to model the two harmonics of $F_1$; this would lead to two formant candidates in the $F_1$ region, and require postprocessing to decide whether these candidates need to be merged into a single formant value.

Use of an STFT instead of an LPC spectrum avoids this latter problem, as it does not impose a specific (e.g., all-pole) model on the speech spectrum. Nonetheless, the issue of data reduction remains when using the STFT, as a typical STFT has 256–512 samples, corresponding to common FFT duration choices, for windows approximately 20–25 ms in length. One normally needs to smooth the STFT, and then do peak-picking. One can *pad with zeros*, i. e., select a much shorter range of speech samples than the FFT length (i. e., fewer samples than an estimated pitch period), to eliminate the harmonics from the spectral display, which effectively smooths the spectrum. In such a case, one normally needs to do pitch-synchronous analysis to choose at least the initial strong samples of each pitch period. Alternatively, one can smooth the FT of a full window with a low-pass filter operating in the frequency domain.

### 11.6.1 Continuity Constraints

It has generally been found that peak-picking methods need to be subject to continuity constraints so as to select from among multiple formant candidates, as there are often more candidates (i. e., spectral peaks) than formants [11.15]. One normally prunes away any candidate whose bandwidth is beyond the range of formants, i.e., a candidate whose bandwidth is less than 50 Hz, which is likely to be an interfering tone or an individual harmonic rather than a formant, or more than, say, 300 Hz, as formant bandwidths increase with center frequency, e.g., a roughly constant $Q$ of around 5–6, so this upper threshold should increase with frequency. In sections of speech that appear to be sonorants, i. e., strong, voiced speech, a tracker aims to assign one formant to each possible range, i. e., roughly one formant per 1000 Hz. We need to allow for many individual cases where there are two formants below 1000 Hz (e.g., /o/ and /u/) and $F_2$ above 2 kHz (e.g., /i/), but within, say, the typical range of 0–4 kHz, there should be four formants (assuming a man's voice). If the signal has passed along a telephone line, then we should not expect to see more than three formants, as $F_4$ is often lost to the upper frequency cutoff of the phone lines (which preserve only the range of approximately 300–3200 Hz). Similarly, if the dynamic range of the spectral ampli-



Amazon EX1010 Page 251

tude obtained is limited, then $F_4$ may in general be too weak to be clearly observed, especially if the background noise level is high enough to obscure the weaker formants, which often include $F_4$.

The most difficult area for designing good continuity constraints is in the temporal dynamics. Automatic tracking of formants is difficult mostly owing to rapid changes in the formant patterns when a VT closure occurs or when the VT excitation changes state (between voiced and unvoiced); such changes often occur several times per second in speech. From frame to frame, in most speech, the formants generally change slowly. Other than the abrupt formant changes (which are due to major VT changes), the most rapid formant changes are normally seen in $F_2$ when the tongue moves quickly in lateral motion (e.g., /ai/ and /oi/) or when the lips round/unround. The maximum rate of change for a formant is approximately 20 Hz/ms (e.g., a change of 1200 Hz over 60 ms); so any proposed formant changes exceeding this threshold should be limited to major phoneme boundaries, where all the formants change and the overall intensity level also changes abruptly.

The apparent merging of formants has been noted as a major problem for formant trackers, i. e., when two or more formants are sufficiently close to each other to present an almost solid band of energy in a given spectral display. Many sounds have two formants close enough that they may potentially appear as one spectral peak (e.g., $F_1$–$F_2$ in /a/, /o/, and /u/, and $F_2$–$F_3$ in /i/ and /r/). One should not normally rely on small rises and falls (across frequency) in a spectrum to delineate individual formants, as such small changes can easily be due to window or harmonic effects. (An LPC spectrum, of course, normally does not have such rapid, small changes, but close formants also cause difficulties with LPC as well [11.14].) Thus, a spectral peak normally requires a significant rise in spectral amplitude over a frequency range on the order of a typical formant bandwidth, in order to be declared a good formant candidate.

It is generally by applying continuity constraints that one can resolve formant merges. While formants may be quite close during the steady state of a vowel, coarticulation with adjacent phonemes generally causes sufficient formant motion of all formants that nearby frames of speech clearly show separating tracks of spectral peaks. When a formant tracker is in doubt about a wide band of speech energy in a given section of speech, it can scan left and right (i. e., before and after) to look for a possible peak that may be moving away from the main band of energy. Often, this is a case of a weaker, rising $F_3$ or $F_4$ that was temporarily *merged* with $F_2$ or $F_3$, respectively; in recent papers on formant tracking, one often sees examples of the tracker incorrectly choosing a stronger and higher-frequency spectral peak as a formant, while ignoring a weaker track (a true formant) moving away from a wide band of speech energy (in which case all of these are formants).

A number of formant trackers focus on precise estimates of the center frequencies of resonances within frequency bands that are assumed to contain a single formant [11.16]. For each frame, they divide up the spectrum into such estimated bands in an initial analysis step, then use simpler estimation methods within each band. If indeed each band has a single resonance, the precise location of its center frequency and bandwidth is simpler through the use of adaptive bandpass filters that seek to isolate individual formants, thus allowing a more-precise focus on details of the (presumed single) resonance within each chosen band of frequency [11.16]. At first glance, this might seem to beg the question of formant tracking, as we have already identified the issue of separating formants that may closely approach each other as potentially difficult. Yet reasonable results appear possible with appropriate filtering, even in noisy conditions [11.17, 18]. One approach uses a parallel formant synthesizer, as was common in older TTS systems to generate a hypothesized synthetic speech spectrum (and thus with specific, known formants), and compares that spectrum to the actual speech spectrum, minimizing a spectral distance measure (often a quadratic cost function relating the two speech spectra, and also aiming for temporal continuity) via dynamic programming (DP, often via hidden Markov models) [11.19]. While DP is popular in formant trackers [11.11], some recent versions of such an approach seem not to need DP [11.20].

### 11.6.2 Use of Phase Shift

A common way to track formants is to estimate speech $S(z)$ in terms of a ratio of $z$-polynomials (e.g., an all-pole LPC spectrum), solve directly for the roots of the denominator, and identify each root as a formant if it has a narrow bandwidth at a reasonable frequency location [11.14]). Other approaches use related phase information [from $S(z)$] to decide whether a spectral peak is a formant. When one evaluates $S(z)$ along the unit circle $z = \exp(i\omega)$, a large negative phase shift occurs when $\omega$ passes a pole close to the unit circle. As formants correspond to complex-conjugate pairs of poles with relatively narrow bandwidths (i. e., near the



Amazon EX1010 Page 252

unit circle), each spectral peak having such a phase shift is normally a formant. The phase shift approaches −180° for small formant bandwidths.

In cases where two formants appear as one broad spectral peak, a modified discrete Fourier transform DFT can resolve the ambiguity. The chirp $z$-transform (CZT) calculates the $z$-transform of the windowed speech on a contour inside the unit circle. Whereas the DFT samples $S(z)$ at uniform intervals on the unit circle, the CZT may take a spiral contour anywhere in the $z$-plane. It may be located near poles corresponding to a spectral peak of interest and thus need to be evaluated only for a small range of frequency samples in pertinent cases. As a contour can be much closer to the formant poles than for the DFT, the CZT can resolve two poles (for two closely spaced formants) into two spectral peaks. Since formant bandwidths tend to increase with frequency, the spiral contour often starts near $z = \alpha$, just inside the unit circle (e.g., $\alpha = 0.9$), and gradually spirals inward with increasing frequency $\omega_k = 2\pi k/N$ ($z_k = \alpha\beta^k \exp(\mathrm{i}\omega_k)$, with $\beta$ just less than 1). This contour would thus follow the expected path of the formant poles and can eliminate problems of merged peaks in DFT displays.

Other recent formant tracking methods are also based on phase in related ways [11.21]. Formant trackers experience increased difficulty when $F_0$ exceeds formant bandwidths, e.g., $F_0 > 250$ Hz, as in children's voices. Harmonics in such speech are so widely separated that only one or two appear in each formant. As a result, spectral analyzers have a tendency to label the prominent harmonics as formants, which is generally wrong, as the center frequency of a formant is rarely an exact multiple of $F_0$.

### 11.6.3 Smoothing

Many pattern estimation algorithms, including formant estimators, need to do postprocessing on the raw data that comes out of the main estimation processing step. Usually, such schemes produce estimated values once per frame (e.g., every 5–10 ms in many speech applications); this applies to ASR and to speech coders, as well as to $F_0$ trackers. In ASR, estimated values for VT representations are modeled by probability distributions, and decisions are made on global probabilities combining hundreds (or more) of computations; a small deviation in any one parameter has little effect, so smoothing of individual parameters in a frame is rarely needed for ASR. Here, we assume that estimated formants may be used for a variety of applications, and thus we judge performance on how well each output value matches the actual speech, and thus are less tolerant of even small errors.

While formant decisions involve continuity constraints, these constraints normally do not greatly affect localized estimations; they are instead mostly used to avoid large errors (such as missing a formant entirely). The decision in each frame is usually based on the immediate speech window, whose placement and duration are rarely synchronized for optimal estimation results. As a result, estimated formant values are often noisy, in the sense that they err slightly above or below their actual values, owing to suboptimal calculations. The estimations are usually within perceivable ranges (i. e., if we were to synthesize speech with the estimated formant values, and compare this with synthesis based on the true values, listeners would hear no difference), but it is usually preferred to smooth such noisy formant signals to present a final formant contour that is both accurate and tidy.

One initial idea here would be simply to pass any noisy formant contour, as a time signal, through a low-pass filter, choosing the cutoff frequency of the filter so as to smooth the small random deviations and not the useful formant movements related to VT movements. However, as when smoothing the output of $F_0$ detectors, formant estimations sometimes change rapidly between individual frames, and applying a linear low-pass filter would produce a poor result when formants do actually change abruptly; instead of a true abrupt formant jump for a nasal, the smoothed pattern would show a gradual rise or fall.

Another difficulty with linear filtering is its behavior when mistakes occur in parameter extraction. Formant and $F_0$ estimators sometimes produce erroneous isolated estimates (i. e., for individual frame outputs) called *outliers*, which deviate greatly from the rest of the parameter contour. Such mistakes must be corrected in postprocessing. As linear filters give equal weight to all signal samples, they would propagate the effect of such a mistake into adjacent sections of the smoothed output formant contour.

Thus, a common alternative to linear filtering is median smoothing [11.1], which preserves sharp signal discontinuities while eliminating fine errors and gross outliers. Most smoothers (linear and nonlinear) operate on a finite time window of the input signal, but linear smoothers combine the windowed samples linearly to produce the smoothed output sample, whereas median smoothing chooses a single value from among



Amazon EX1010 Page 253

the window samples. In each data window, the samples are ordered in amplitude with no regard for timing within the window. The output sample is the median, i. e., the $[(N+1)/2]$th of $N$ ordered samples (for odd $N$). Sudden discontinuities are preserved because no averaging occurs. Up to $(N-1)/2$ outlier samples, above or below the main contour, do not affect the output.

Median smoothers do well in eliminating outliers and in global smoothing, but do not provide very smooth outputs when dealing with noisy signals. Thus they are often combined with linear smoothers to yield a compromise smoothed output, whose sharp transitions are better preserved than with only linear filtering but with a smoother output signal than would be possible using only median smoothing.

## 11.7 Summary

This chapter has presented an introduction to formant tracking methods for speech. We have concentrated on the major approaches to formant tracking, generally using Fourier or LPC displays, and either peak-picking or solving for the roots in the LPC all-pole polynomial. There are also other proposed methods, such as a bank of inverse filters [11.22].

### References


11.1 D. O'Shaughnessy: *Speech Communication: Human and Machine*, 2nd edn. (IEEE, Piscataway 2000)

11.2 J. Darch, B. Milner, S. Vaseghi: MAP prediction of formant frequencies and voicing class from MFCC vectors in noise, Speech Commun. **11**, 1556–1572 (2006)

11.3 R. Togneri, L. Deng: A state-space model with neural-network prediction for recovering vocal tract resonances in fluent speech from Mel-cepstral coefficients, Speech Commun. **48**(8), 971–988 (2006)

11.4 K. Weber, S. Ikbal, S. Bengio, H. Bourlard: Robust speech recognition and feature extraction using HMM2, Comput. Speech Lang. **17**(2–3), 195–211 (2003)

11.5 W. Ding, N. Campbell: Optimizing unit selection with voice source and formants in the CHATR speech synthesis system, Proc. Eurospeech (1997) pp. 537–540

11.6 J. Malkin, X. Li, J. Bilmes: A graphical model for formant tracking, Proc. IEEE ICASSP, Vol. 1 (2005) pp. 913–916

11.7 K. Sjlander, J. Beskow: WAVESURFER – an open source speech tool, Proc. ICSLP (2000)

11.8 L. Deng, L.J. Lee, H. Attias, A. Acero: A structured speech model with continuous hidden dynamics and prediction-residual training for tracking vocal tract resonances, Proc. IEEE ICASSP, Vol. 1 (2004) pp. 557–560

11.9 Y. Zheng, M. Hasegawa-Johnson: Formant tracking by mixture state particle filter, Proc. IEEE ICASSP, Vol. 1 (2004) pp. 565–568

11.10 D.T. Toledano, J.G. Villardebo, L.H. Gomez: Initialization, training, and context-cependency in HMM-based formant tracking, IEEE Trans. Audio Speech **14**(2), 511–523 (2006)

11.11 M. Lee, J. van Santen, B. Mobius, J. Olive: Formant tracking using context-dependent phonemic information, IEEE Trans. Speech Audio Process. **13**(5), 741–750 (2005), Part 2

11.12 S. McCandless: An algorithm for automatic formant extraction using linear prediction spectra, Proc. IEEE ICASSP **22**(2), 135–141 (1974)

11.13 G. Kopec: Formant tracking using hidden Markov models and vector quantization, Proc. IEEE ICASSP **34**(4), 709–729 (1986)

11.14 G.K. Vallabha, B. Tuller: Systematic errors in the formant analysis of steady-state vowels, Speech Commun. **38**(1–2), 141–160 (2002)

11.15 Y. Laprie, M.-O. Berger: Cooperation of regularization and speech heuristics to control automatic formant tracking, Speech Commun. **19**(4), 255–269 (1996)

11.16 K. Mustafa, I.C. Bruce: Robust formant tracking for continuous speech with speaker variability, IEEE Trans. Audio Speech **14**(2), 435–444 (2006)

11.17 A. Rao, R. Kumaresan: On decomposing speech into modulated components, IEEE Trans. Speech Audio Process. **8**(3), 240–254 (2000)

11.18 I.C. Bruce, N.V. Karkhanis, E.D. Young, M.B. Sachs: Robust formant tracking in noise, Proc. IEEE ICASSP, Vol. 1 (2002) pp. 281–284

11.19 L. Welling, H. Ney: Formant estimation for speech recognition, IEEE Trans. Speech Audio Process. **6**(1), 36–48 (1998)

11.20 B. Chen, P.C. Loizou: Formant frequency estimation in noise, Proc. IEEE ICASSP, Vol. 1 (2004) pp. 581–584




Amazon EX1010 Page 254

11.21 D.J. Nelson: Cross-spectral based formant estimation and alignment, Proc. IEEE ICASSP, Vol. 2 (2004) pp. 621–624

11.22 A. Watanabe: Formant estimation method using inverse-filter control, IEEE Trans. Speech Audio Process. **9**(4), 317–326 (2001)



Amazon EX1010 Page 255

# 12. The STFT, Sinusoidal Models, and Speech Modification

M. M. Goodwin

Frequency-domain signal representations are used for a wide variety of applications in speech processing. In this Chapter, we first consider the short-time Fourier transform (STFT), presenting a number of interpretations of the analysis–synthesis process in a consistent mathematical framework. We then develop the sinusoidal model as a parametric extension of the STFT wherein the data in the STFT is compacted, sacrificing perfect reconstruction at the benefit of achieving a sparser and essentially more meaningful representation. We discuss several methods for sinusoidal parameter estimation and signal reconstruction, and present a detailed treatment of a matching pursuit algorithm for sinusoidal modeling. The final part of the Chapter addresses speech modifications such as filtering, enhancement, and time-scaling, for which both the STFT and the sinusoidal model are effective tools.

12.1 **The Short-Time Fourier Transform** ..... 230
12.1.1 The STFT as a Sliding-Window Transform ..... 230
12.1.2 The STFT as a Modulated Filter Bank 233
12.1.3 Original Formulation of the STFT ..... 234
12.1.4 The Time Reference of the STFT ..... 234
12.1.5 The STFT as a Heterodyne Filter Bank ..... 235
12.1.6 Reconstruction Methods and Signal Models ..... 235
12.1.7 Examples ..... 236
12.1.8 Limitations of the STFT ..... 240

12.2 **Sinusoidal Models** ..... 242
12.2.1 Parametric Extension of the STFT ..... 242
12.2.2 The Sinusoidal Signal Model ..... 244
12.2.3 Sinusoidal Analysis and Synthesis .. 244
12.2.4 Signal Modeling by Matching Pursuit ..... 245
12.2.5 Sinusoidal Matching Pursuits ..... 246
12.2.6 Sinusoidal Analysis ..... 247
12.2.7 Sinusoidal Synthesis ..... 250

12.3 **Speech Modification** ..... 253
12.3.1 Comparing the STFT and Sinusoidal Models ..... 253
12.3.2 Linear Filtering ..... 253
12.3.3 Enhancement ..... 254
12.3.4 Time-Scale Modification ..... 254
12.3.5 Pitch Modification ..... 255
12.3.6 Cross-Synthesis and Other Modifications ..... 255
12.3.7 Audio Coding with Decode-Side Modification ..... 256

**References** ..... 256

Frequency-domain representations have a rich history in applied mathematics and signal processing; their uses range from solving differential equations to compressing music. The cornerstone of frequency-domain approaches is the continuous-frequency Fourier transform of infinite-duration continuous-time signals; in the field of signal processing, this is of course an essential tool for the analysis of linear systems. For real-world digital signal processing applications, however, it is necessary to modify the Fourier transform to handle discrete-time signals of arbitrary duration in a segment-by-segment fashion, creating a *local* discrete-frequency Fourier representation for each segment. This idea of a short-time Fourier transform (STFT) was developed largely in the speech processing research community based on early work in signal processing and communications theory [12.1], for instance the key contributions by *Gabor* on time–frequency representations based on using modulated and time-shifted Gaussian windows to analyze the signal [12.2, 3]. Given this historical context, the STFT is sometimes referred to as a Gabor transform [12.4], although in some cases a distinction is drawn.

The short-time Fourier transform has proven useful for many speech processing and speech communication applications, including time scaling, pitch shifting, noise



Amazon EX1010 Page 256

reduction, and echo cancelation. Furthermore, development and understanding of the STFT led to advances in the theory and implementation of other types of transforms and filter banks, such as those that are now central to widely used image and audio compression schemes [12.4, 5]. In addition, a progression of ideas can be traced from the STFT and early vocoders to the sinusoidal model, which has been shown to be effective for low-rate speech and audio coding as well as for signal modification [12.6]. Of course, it should be noted that many of these frequency-domain processing schemes rely on the fast Fourier transform (FFT) and other similar algorithms, without which many of the implementations would not be computationally practical for real-world deployment [12.7, 8].

This chapter is organized as follows. Section 12.1 develops the short-time Fourier transform and interprets it in three different ways: as a sliding-window transform, as a modulated filter bank, and as a heterodyne filter bank. Section 12.2 then introduces the sinusoidal model as a parametric extension of the STFT. Methods for sinusoidal analysis and synthesis are discussed, with a focus on matching pursuit adapted for sinusoidal modeling. Section 12.3 provides an overview of the use of the STFT and sinusoidal models to realize desirable speech modifications such as filtering and time scaling.

## 12.1 The Short-Time Fourier Transform

The short-time Fourier transform (STFT) is a commonly used tool for analysis, modification, and synthesis of speech and audio signals. In this Section, the STFT is developed as a sliding-window transform, and conditions for perfectly reconstructing the input signal with an inverse transform and overlap-add are derived. It is then shown that the sliding-window transform corresponds to a modulated filter bank, and that the conditions for perfect reconstruction using a synthesis filter bank are equivalent to those for the sliding-window case. These formulations are related to the original definition of the STFT, which is shown to correspond to a heterodyne filter bank. The properties of the STFT are illustrated with examples, and some limitations of the STFT for processing real-world signals are described.

### 12.1.1 The STFT as a Sliding-Window Transform

In practical applications, the short-time Fourier transform is typically implemented in a sliding-window fashion. Given an input signal $x[t]$ of arbitrary duration, data segments are extracted at regular intervals using a time-limited window $w[n]$; these signal segments or frames can be expressed as

$$x_l[n] = w[n]x[n+lL], \quad 0 \le n \le N-1\,, \tag{12.1}$$

where $N$ is the window length, $l$ is a frame index, and $L$ is the hop size, i. e., the spacing in samples between consecutive applications of the sliding extraction window; the index $n$ is a local time index, i. e., an index relative to the start of the sliding window. For each signal frame, a discrete Fourier transform (DFT) is carried out, yielding

$$\begin{aligned} X[k,l] &= \sum_{n=0}^{N-1} x_l[n]\mathrm{e}^{-\mathrm{i}2\pi nk/K} \\ &= \sum_{n=0}^{N-1} w[n]x[n+lL]\mathrm{e}^{-\mathrm{i}2\pi nk/K}\,, \end{aligned} \tag{12.2}$$

where $K$ is the DFT size and $k$ is a frequency index or *bin* index. The STFT $X[k,l]$ then characterizes the local time–frequency behavior of the signal around time $lL$ and bin $k$; for a sampling rate $F_\mathrm{s}$, these discrete indices correspond to continuous time $lL/F_\mathrm{s}$ and frequency $kF_\mathrm{s}/K$. To simplify the notation, a radial frequency

$$\omega_k = \frac{2\pi k}{K} \tag{12.3}$$

is often incorporated, such that the STFT expression becomes

$$X[k,l] = \sum_{n=0}^{N-1} w[n]x[n+lL]\mathrm{e}^{-\mathrm{i}\omega_k n}\,. \tag{12.4}$$

Note that for a hop size of $L = 1$, the STFT can be written directly as a function of the original time variable $t$:

$$X[k,t] = \sum_{n=0}^{N-1} w[n]x[n+t]\mathrm{e}^{-\mathrm{i}\omega_k n}\,. \tag{12.5}$$

Equations (12.4) and (12.5) are somewhat different from the formulation of the STFT in early references [12.9–12], as will be explored in Sect. 12.1.3.

In the sliding-window framework, the STFT is thought of as a spectral representation of a time slice



Amazon EX1010 Page 257

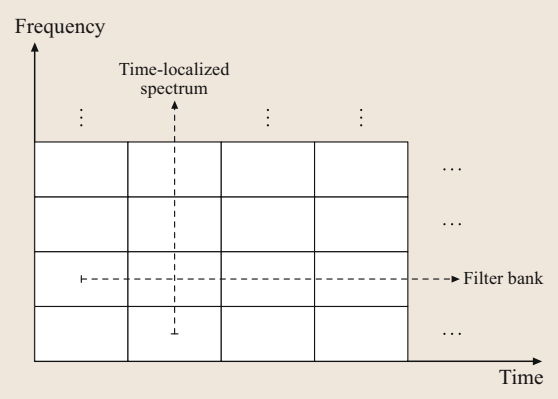



**Fig. 12.1** Interpretations of the short-time Fourier transform as a series of time-localized spectra (*vertical*) and as a bank of bandpass filters (*horizontal*)

of the input signal; interpreting $X[k,l]$ as a function of the frequency $k$ for each value of the time index $l$, the STFT corresponds to a series of time-localized spectra. Alternatively, the STFT can be viewed as a function of time for each frequency; interpreting $X[k,l]$ as a time series that is a function of $l$ for each bin $k$, the STFT then corresponds to a filter bank which decomposes the input signal into frequency channels or subbands. These two interpretations of the STFT are depicted with respect to the time–frequency plane in Fig. 12.1. Such time–frequency *tilings* have become common in the modern signal processing literature as a way to depict the time and frequency resolution of signal representations [12.4]. In the remainder of this Section, the *time-localized spectra* interpretation of the tiling is considered; the *filter bank* interpretation is treated further in Sects. 12.1.2 and 12.1.5.

The STFT is an *analysis* operation; it provides a representation of the signal that exposes some fundamental information about the signal. As will be discussed in Sect. 12.3, the STFT domain furthermore enables useful modifications of the signal, some of which are guided by this information captured in the STFT representation. In such scenarios, the STFT of the input time-domain signal is modified so as to create a desired effect in a new time-domain signal. To generate this modified time-domain signal, though, an appropriate *synthesis* operation is needed. Ideally, such a synthesis operation would reconstruct the original signal perfectly in the event that no STFT-domain modification is carried out. In the following, a synthesis procedure based on this *perfect reconstruction* property is developed.

For the sliding-window case where the STFT is interpreted as a series of time-localized spectra, it is straightforward to envision a reconstruction framework that is basically the opposite of the analysis: first, an inverse DFT (IDFT) of each local spectrum is carried out; then, the resulting signal frames are aggregated to synthesize the signal. If the DFT size is sufficiently large ($K \geq N$), the IDFT simply returns the windowed signal segment:

$$\begin{aligned}\hat{x}_l[n] &= \mathrm{IDFT}\{X[k,l]\} \\ &= w[n]x[n+lL], \quad 0 \leq n \leq N-1 \end{aligned} \tag{12.6}$$

or, with respect to a global time $t = n + lL$ instead of the in-frame index $n$,

$$\hat{x}_l[t-lL] = w[t-lL]x[t], \quad lL \leq t \leq lL+N-1\,, \tag{12.7}$$

where the latter expression is introduced to simplify the upcoming formulation. Regarding the size of the DFT, when $K > N$ the DFT is *oversampled*; this frequency-domain oversampling results in time-limited interpolation of the spectrum, which is analogous to the bandlimited interpolation that is characteristic of time-domain oversampling [12.13]. In the *undersampled* case $K < N$, time-domain aliasing is introduced. To avoid this difficulty, the condition $K \geq N$ is imposed at this point.

If the DFT is large enough that no aliasing occurs, reconstruction can be simply carried out by an *overlap-add* (OLA) process, possibly with a synthesis window [12.10, 14, 15]. Denoting this synthesis window by $v[n]$, the OLA reconstruction is given by

$$\begin{aligned}\hat{x}[t] &= \sum_l v[t-lL]\hat{x}_l[t-lL] \\ &= \sum_l v[t-lL]w[t-lL]x[t]\,. \end{aligned} \tag{12.8}$$

To construct the full output signal $\hat{x}[t]$, each signal frame generated by the IDFT is weighted by the synthesis window and added to any neighboring windows which overlap in time. Since $x(t)$ is not a function of $l$, (12.8) can be rewritten as

$$\hat{x}[t] = x[t]\left(\sum_l w[t-lL]v[t-lL]\right), \tag{12.9}$$

so perfect reconstruction is achieved if the analysis and synthesis windows satisfy the constraint

$$\sum_l w[t-lL]v[t-lL] = 1\,. \tag{12.10}$$



Amazon EX1010 Page 258

This constraint is similar to but somewhat more general than the perfect reconstruction constraints given in [12.1, 10–12, 14, 15]. Note that throughout this section the analysis and synthesis windows, as well as the signal, will be assumed to be real-valued.

In cases where $v[n]$ is not explicitly specified, the equivalent synthesis window is a rectangular window covering the same time span as $w[n]$. Then, the constraint in (12.10) becomes

$$\sum_l w[t-lL] = 1 \,. \tag{12.11}$$

The construction of windows with this property has been explored in the literature; a variety of such *perfect reconstruction windows* have been proposed, for example rectangular and triangular windows and the Blackman–Harris family, which includes the familiar Hanning and Hamming windows [12.16, 17]. These are also referred to as windows with the *overlap-add property* and will be denoted by $w_{\mathrm{PR}}[n]$ in the following derivations. Several issues are worth noting: *any* window function satisfies the condition in (12.11) when $L = 1$; for $L = N$, the only window that has the overlap-add property is a rectangular window of length $N$; for $L > N$, the overlap-add property cannot be satisfied as there are time gaps between successive frames.

There are a number of methods to design analysis and synthesis windows that satisfy (12.10). The methods discussed here rely on the use of familiar windows that satisfy (12.11) to jointly construct analysis and synthesis windows that satisfy (12.10); various analysis–synthesis window pairs designed in this way exhibit computational and modeling advantages. In one design approach, complementary powers of a perfect reconstruction window provide the analysis and synthesis windows:

$$\sum_l \left(w_{\mathrm{PR}}[t-lL]\right)^c \left(w_{\mathrm{PR}}[t-lL]\right)^{1-c} = 1$$

$$\Longrightarrow \begin{cases} \text{Analysis} & w[n] = \left(w_{\mathrm{PR}}[n]\right)^c \\ \text{Synthesis} & v[n] = \left(w_{\mathrm{PR}}[n]\right)^{1-c} \end{cases} . \tag{12.12}$$

The case $c = 1/2$, where the analysis and synthesis windows are equivalent, is of some interest because of its symmetry. A second approach is as follows; given a perfect reconstruction window $w_{\mathrm{PR}}[n]$ and an arbitrary window $b[n]$ that is strictly nonzero over the time support of $w_{\mathrm{PR}}[n]$, the overlap-add property can be rephrased as:

$$\sum_l w_{\mathrm{PR}}[t-lL]\left(\frac{b[t-lL]}{b[t-lL]}\right) = 1$$

$$\sum_l b[t-lL]\left(\frac{w_{\mathrm{PR}}[t-lL]}{b[t-lL]}\right) = 1$$

$$\Longrightarrow \begin{cases} \text{Analysis} & w[n] = b[n] \\ \text{Synthesis} & v[n] = \frac{w_{\mathrm{PR}}[n]}{b[n]} \end{cases} . \tag{12.13}$$

Windows of this form are used in the sinusoidal synthesis scheme described in [12.18, 19].

In practice, the STFT analysis window is generally chosen based on its frequency resolution and sidelobe behavior. Most commonly, the synthesis window is selected to provide perfect reconstruction given the analysis window; in cases where modifications of the STFT are carried out prior to synthesis, the synthesis window may be designed to minimize reconstruction artifacts [12.20].

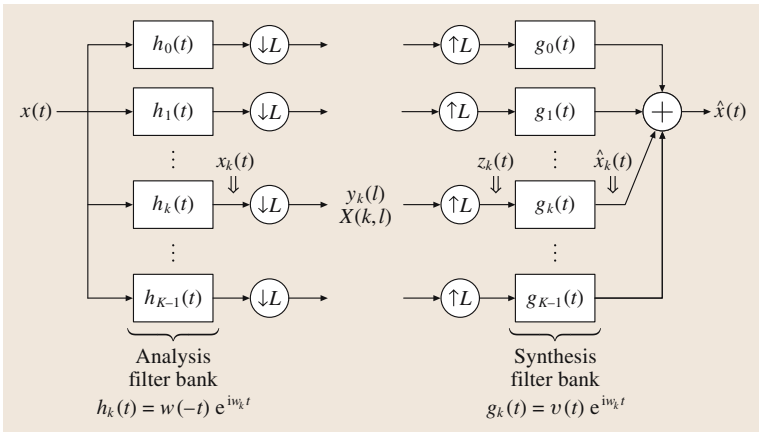



**Fig. 12.2** Formulation of the short-time Fourier transform as a modulated filter bank



Amazon EX1010 Page 259

### 12.1.2 The STFT as a Modulated Filter Bank

Modulated filter banks, in which the constituent filters are modulated versions of a prototype low-pass filter, have received considerable attention in the signal processing literature, in large part due to the relative ease of designing such filter banks and their effectiveness for signal compression [12.4, 5, 21]. In this Section, we demonstrate that the STFT can be implemented with a modulated filter bank structure. Note that this equivalence of sliding-window sinusoidal transforms and modulated filter banks is indeed one of the basic concepts in the filter bank and subband coding literature [12.4].

Consider the general modulated filter bank shown in Fig. 12.2, with analysis filters $h_k[t]$ and synthesis filters $g_k[t]$. If the filters are defined as

$$h_k[t] = w[-t]\,\mathrm{e}^{\mathrm{i}\omega_k t}\,, \tag{12.14}$$

$$g_k[t] = v[t]\,\mathrm{e}^{\mathrm{i}\omega_k t}\,, \tag{12.15}$$

then the filter bank implements an STFT analysis–synthesis. The derivation of this correspondence is straightforward; using the notation in Fig. 12.2, where the subscript is now a subband index instead of a time index as earlier, the outputs of the analysis filter bank can be written

$$\begin{aligned} x_k[t] &= \sum_n h_k[n]x[t-n] \\ &= \sum_n w[-n]x[t-n]\,\mathrm{e}^{\mathrm{i}\omega_k n} \\ &= \sum_n w[n]x[n+t]\,\mathrm{e}^{-\mathrm{i}\omega_k n} \\ &= X[k,t]\,. \end{aligned} \tag{12.16}$$

Subsampling by $L$ then yields the subband signals

$$\begin{aligned} y_k[l] &= x_k[lL] \\ &= \sum_n w[n]x[n+lL]\,\mathrm{e}^{-\mathrm{i}\omega_k n} \\ &= X[k,l]\,, \end{aligned} \tag{12.17}$$

which shows that the modulated analysis filter bank does indeed carry out the STFT analysis operation defined in (12.4). Note that subsampling by $L$ in the filter bank subbands corresponds to using a hop size of $L$ in the sliding-window framework.

In the synthesis filter bank, the subband signals $y_k[l]$ are first upsampled by $L$; the resulting signals can be expressed as

$$z_k[t] = x_k[t]\sum_l \delta[t-lL]\,, \tag{12.18}$$

which describes the effect of successive downsampling and upsampling on the signal $x_k[t]$. The outputs of the subband synthesis filters $g_k[t]$ can then be written

$$\begin{aligned} \hat{x}_k[t] &= z_k[t] * g_k[t] = \sum_n g_k[n]z_k[t-n] \\ &= \sum_n v[n]\,\mathrm{e}^{\mathrm{i}\omega_k n}x_k[t-n]\sum_l \delta[t-n-lL]\,. \end{aligned} \tag{12.19}$$

Using (12.16) then yields

$$\begin{aligned} \hat{x}_k[t] = \sum_n \sum_l \sum_m & v[n]w[m]x[t-n+m] \\ & \times \mathrm{e}^{\mathrm{i}\omega_k(n-m)}\delta[t-n-lL]\,. \end{aligned} \tag{12.20}$$

Perfect reconstruction is achieved if the outputs of the synthesis filters sum to the original signal:

$$\hat{x}[t] = \sum_k \hat{x}_k[t] = x[t]\,. \tag{12.21}$$

If the expression in (12.20) is substituted for $\hat{x}_k[t]$, the summation can be rearranged such that the sum over the frequency $k$ is the innermost sum; since $\omega_k = 2\pi k/K$, this sum can be written as

$$\sum_{k=0}^{K-1} \mathrm{e}^{\mathrm{i}\omega_k(n-m)} = K\sum_r \delta[n-m+rK]\,. \tag{12.22}$$

If $|n-m| < K$ for all possible combinations of $n$ and $m$, then the only relevant term in the right-hand sum is for $r = 0$, in which case the equation simplifies to

$$\sum_{k=0}^{K-1} \mathrm{e}^{\mathrm{i}\omega_k(n-m)} = K\delta[n-m]\,. \tag{12.23}$$

This restriction on the values of $n$ and $m$ corresponds to the $K \geq N$ constraint discussed in Sect. 12.1.1 for the sliding-window analysis–synthesis; namely, time-domain aliasing is introduced if $n$ and $m$ do not meet this criterion. It is thus assumed here, as in the earlier discussion, that $K \geq N$.

Using the result in (12.23), the output of the synthesis filter bank can be written

$$\begin{aligned} \hat{x}[t] = K\sum_n \sum_l & v[n]\delta[t-n-lL] \\ & \times \sum_m w[m]x[t-n+m]\delta[n-m]\,. \end{aligned} \tag{12.24}$$

The innermost sum is nonzero only for $n = m$, so the expression for $\hat{x}[t]$ can be simplified to

$$\hat{x}[t] = Kx[t]\sum_n \sum_l w[n]v[n]\delta[t-n-lL]\,. \tag{12.25}$$



Amazon EX1010 Page 260

Then, since $\delta[t-n-lL]$ is nonzero only for $n=t-lL$, this can be further simplified to

$$\hat{x}[t] = Kx[t]\sum_l w[t-lL]v[t-lL]\,. \tag{12.26}$$

Perfect reconstruction for the analysis–synthesis filter bank is thus achieved if

$$K\sum_l w[t-lL]v[t-lL] = 1\,. \tag{12.27}$$

With the exception of the gain term, this is the same condition derived for the sliding-window framework. Because of this functional equivalence, the analysis–synthesis window pairs described in Sect. 12.1.1 can be used as prototype functions for perfect reconstruction modulated filter banks.

Note that if $L>1$, the synthesis filter bank interpolates the subband signals. In the non-subsampled case ($L=1$), when no interpolation is needed, perfect reconstruction can be achieved with any analysis–synthesis window pair for which $\sum_n w[n]v[n]\neq 0$. For example, the synthesis can be performed with the trivial filter bank $g_k[t]=\delta[t]$ if the analysis window satisfies the constraint

$$\sum_i w[n-i] = 1\,, \tag{12.28}$$

which indeed holds for *any* window, within a gain term. The generality of this constraint is an example of the design flexibility that results from using oversampled or overcomplete approaches [12.4, 22–24].

### 12.1.3 Original Formulation of the STFT

The definition of the STFT given in (12.4) differs from that in the early literature [12.9–12, 14, 15], where the transform is given as

$$\tilde{X}[k,t] = \sum_{m=-\infty}^{\infty} \tilde{w}[t-m]x[m]\mathrm{e}^{-\mathrm{i}\omega_k m} \tag{12.29}$$

$$= \sum_{m=t}^{t+N-1} \tilde{w}[t-m]x[m]\mathrm{e}^{-\mathrm{i}\omega_k m}\,, \tag{12.30}$$

or in subsampled form as

$$\tilde{X}[k,l] = \sum_{m=lL}^{lL+N-1} \tilde{w}[lL-m]x[m]\mathrm{e}^{-\mathrm{i}\omega_k m}\,, \tag{12.31}$$

where $\tilde{w}[n]$ is again a time-localized window, as $w[n]$ in Sects. 12.1.1 and 12.1.2. The index $m$ is a global time index, as opposed to the local time index $n$ used in the earlier STFT formulation; the range of $m$ in the sum, and hence the support of the window $\tilde{w}[n]$, is defined here in such a way that this alternate transform and $X[k,l]$ in (12.4) refer to the same $N$-point segment of the signal and can thus be compared. In some treatments, the STFT is expressed as in (12.31) but without the time reversal of the window [12.5]. This reversal of the time index affects the interpretation of the transform as a filter bank. More importantly, however, the filter bank interpretation is affected by the time reference of the expansion functions, as discussed below.

### 12.1.4 The Time Reference of the STFT

In the STFT formulation in (12.29–12.31), the signal is analyzed with respect to sinusoids whose time reference is absolute rather than relative to the windowed segments; for different segments, these functions have the same time reference, namely the time origin of the input signal $x[t]$. On the other hand, in (12.4), the time origin of the STFT sinusoids is the starting point of the signal segment in question; the phases of the STFT for a particular segment then refer to the time start of that segment. Note that the STFT can also be formulated such that the phase is referenced to the center of the time window, which is desirable in some cases [12.19, 25]; this extension plays a role in sinusoidal modeling, but such phase-centering will not be considered in the treatment of the STFT because of the slight complications it introduces.

The two formulations of the STFT have different ramifications for the interpretation of the STFT and the signal representation it yields; the differences can be seen by relating the two definitions. A change of index ($n=m-t$) in (12.30) yields

$$\begin{aligned}\tilde{X}[k,t] &= \sum_{n=0}^{N-1} \tilde{w}[-n]x[t+n]\mathrm{e}^{-\mathrm{i}\omega_k(t+n)} \\ &= \mathrm{e}^{-\mathrm{i}\omega_k t}\sum_{n=0}^{N-1} \tilde{w}[-n]x[t+n]\mathrm{e}^{-\mathrm{i}\omega_k n}\,.\end{aligned} \tag{12.32}$$

Letting $\tilde{w}[n]=w[-n]$,

$$\tilde{X}[k,t] = \mathrm{e}^{-\mathrm{i}\omega_k t}\sum_{n=0}^{N-1} w[n]x[t+n]\mathrm{e}^{-\mathrm{i}\omega_k n}. \tag{12.33}$$

Comparing this to (12.5), two simple relationships are clear:



Amazon EX1010 Page 261

$$X[k,t] = \tilde{X}[k,t]\,e^{i\omega_k t}\,, \tag{12.34}$$

$$|X[k,t]| = |\tilde{X}[k,t]|\,. \tag{12.35}$$

Note that these also hold in the subsampled case. The first relationship affects the interpretation of the STFT as a filter bank; the time signal $X[k,t]$ is a modulated version of the baseband envelope $\tilde{X}[k,t]$, so the corresponding filter banks for the two cases have different structures. The second expression affects the interpretation of the STFT as a series of time-localized spectra. The short-time magnitude spectra are equivalent in the two formulations. With respect to phase, however, the approaches differ in that $X[k,t]$ provides a *local* phase. An estimate of the local phase of each partial is important for building a sinusoidal signal model, so $X[k,t]$ is more useful for sinusoidal modeling than $\tilde{X}[k,t]$. This advantage will become more apparent in the discussion of sinusoidal modeling in Sect. 12.2.

### 12.1.5 The STFT as a Heterodyne Filter Bank

In [12.9–12, 14, 15], where the STFT is defined as in (12.31) with an absolute time reference, the transform can be interpreted as a filter bank with a heterodyne structure. Making the substitution

$$x_k[m] = x[m]\,e^{-i\omega_k m} \tag{12.36}$$

in (12.31) yields an expression that is immediately recognizable as a convolution

$$\tilde{X}[k,t] = \sum_{m=t}^{t+N-1} \tilde{w}[t-m]x_k[m]. \tag{12.37}$$

The filter $\tilde{w}[n]$ is typically low pass; it thus extracts the baseband spectrum of $x_k[m]$. According to the modulation relationship in (12.36), $x_k[m]$ is a version of $x[m]$ that has been modulated down by $\omega_k$; thus, the baseband spectrum of $x_k[m]$ corresponds to the spectrum of $x[m]$ in the neighborhood of frequency $\omega_k$. The $k$-th branch of the STFT filter bank thus extracts information about the signal in a frequency band around $\omega_k = 2\pi k/K$.

Figure 12.3 depicts one branch of a heterodyne STFT filter bank and provides an equivalent structure based on modulated filters [12.5]. Mathematically, the equivalence is straightforward:

$$\begin{aligned}\tilde{X}_1[k,t] &= \sum_m \tilde{w}[t-m](x[m]\,e^{-i\omega_k m}) \\ &= e^{-i\omega_k t}\sum_m (\tilde{w}[t-m]\,e^{i\omega_k(t-m)})x[m] \\ &= \tilde{X}_2[k,t]\,.\end{aligned} \tag{12.38}$$

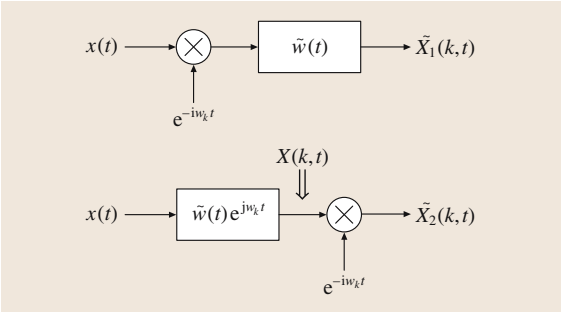



**Fig. 12.3** One channel of a heterodyne filter bank for computing the STFT $\tilde{X}[k,t]$ defined in (12.29). The two structures are equivalent, as indicated in (12.38). The STFT $X[k,t]$ as defined in (12.5) is an intermediate signal in the second structure

Given the relationship in (12.34), namely that $X[k,t] = \tilde{X}[k,t]\,e^{i\omega_k t}$, it is clear that $X[k,t]$ is the immediate output of the modulated filter $\tilde{w}[n]\,e^{i\omega_k n}$ without the ensuing modulation to the baseband, as indicated in Fig. 12.3. This relationship sheds light on the impact of the time-base difference of the STFT variations in (12.4) and (12.31). Using an absolute time reference for the analysis leads to a heterodyne filter bank, while using a relative time reference results in a modulated filter bank; as shown above, the difference between the filter bank structures is a modulation of the subband signals to baseband.

### 12.1.6 Reconstruction Methods and Signal Models

In [12.1, 12] and other early treatments of the short-time Fourier transform, the two interpretations of the STFT lead to different reconstruction algorithms. For the sliding-window framework, where the STFT is viewed as a series of time-localized spectra, an OLA synthesis is formulated; for the filter bank case, the signal reconstruction approach is referred to as *filter bank summation* (FBS). In these early frameworks, the different reconstruction algorithms lead to different perfect reconstruction conditions that are related by a duality [12.12]. Here, with the STFT redefined to use a local time reference, the perfect reconstruction conditions for the two approaches are equivalent as derived in Sects. 12.1.1 and 12.1.2; that these conditions are ultimately equivalent is not particularly surprising since the representation of the STFT as a time–frequency tiling as in Fig. 12.1 suggests that a distinction between the two interpretations is actually artificial.



Amazon EX1010 Page 262

The reconstruction formula in an analysis–synthesis system corresponds to a *signal model*; with this in mind, it is worthwhile to consider the filter bank summation approach as it highlights the signal modeling inherent in the STFT. In the non-subsampled case, the FBS reconstruction formula for the original STFT of (12.29) is given by

$$\hat{x}[t] = \sum_k \tilde{X}[k,t]\,\mathrm{e}^{\mathrm{i}\omega_k t}\,. \tag{12.39}$$

For a fixed $k$, the STFT $\tilde{X}[k,t]$ can be interpreted as the amplitude envelope of a sinusoid with frequency $\omega_k$; reconstruction is achieved by modulating each of the envelopes to the appropriate frequency and summing the resulting signals. The original signal is thus being modeled as a sum of amplitude-modulated sinusoids. For the phase-localized formulation of the STFT in (12.5), the FBS reconstruction formula is

$$\hat{x}[t] = \sum_k X[k,t]\,. \tag{12.40}$$

Here, $X[k,t]$ corresponds directly to a sinusoid at frequency $\omega_k$ rather than its amplitude envelope; the appropriate modulation is essentially built into the STFT representation.

### 12.1.7 Examples

Here the short-time Fourier transforms of several example signals are considered. This discussion should serve to clarify the behavior of the STFT and underscore some of its strengths and weaknesses as an analysis–synthesis tool.

Consider a signal consisting of a single sinusoid of amplitude $A$ and frequency $\omega_0 = 2\pi f_0/F_\mathrm{s}$:

$$\begin{aligned} x_0[t] &= A\cos(\omega_0 t) \\ &= \frac{A}{2}\mathrm{e}^{\mathrm{i}\omega_0 t} + \frac{A}{2}\mathrm{e}^{-\mathrm{i}\omega_0 t}\,. \end{aligned} \tag{12.41}$$

Using (12.2), the STFT of $x_0[t]$ is given by

$$\begin{aligned} X_0[k,l] = {} & \frac{A}{2}\mathrm{e}^{\mathrm{i}\omega_0 lL}\sum_{n=0}^{N-1} w[n]\,\mathrm{e}^{-\mathrm{i}(\omega_k-\omega_0)n} \\ & + \frac{A}{2}\mathrm{e}^{-\mathrm{i}\omega_0 lL}\sum_{n=0}^{N-1} w[n]\,\mathrm{e}^{-\mathrm{i}(\omega_k+\omega_0)n}\,. \end{aligned} \tag{12.42}$$

Denoting the discrete-time Fourier transform (DTFT) of $w[n]$ by $W(\omega)$, the STFT can be rewritten as

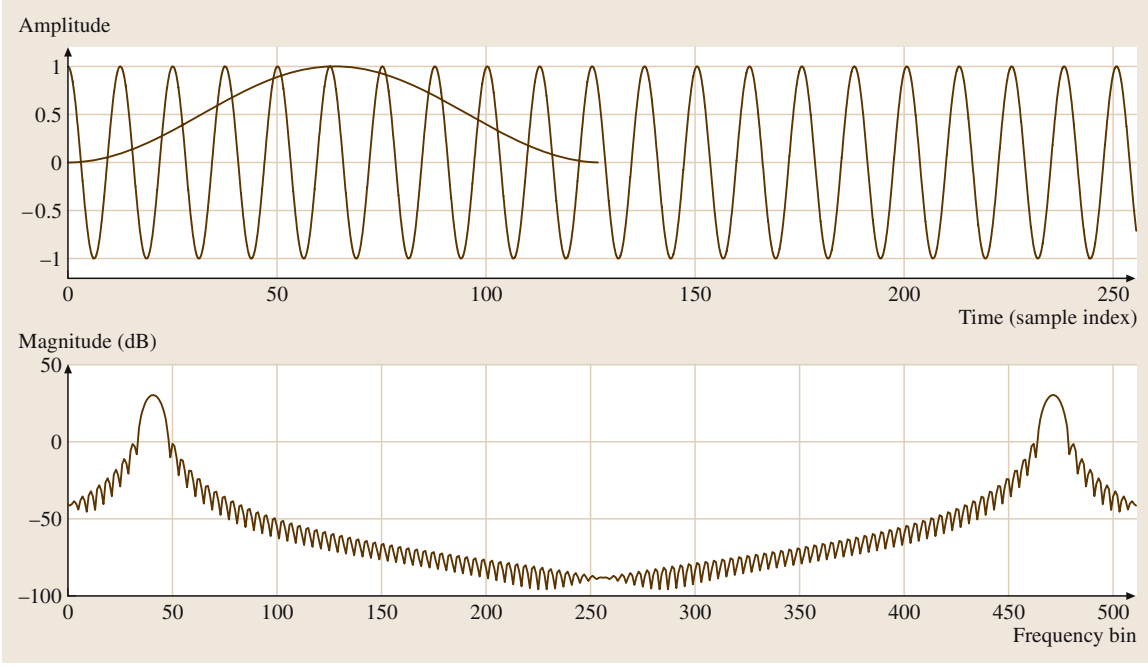



**Fig. 12.4** A sinusoid and its STFT magnitude for one time window. The spectrum exhibits two main lobes as indicated by (12.43)



Amazon EX1010 Page 263

$$X_0[k,l] = \frac{A}{2} e^{i\omega_0 lL} W(\omega_k - \omega_0) + \frac{A}{2} e^{-i\omega_0 lL} W(\omega_k + \omega_0) . \quad (12.43)$$

The STFT of a single real sinusoid is thus a sum of two modulated versions of the window spectrum $W(\omega)$, each with a different phase based on the frequency $\omega_0$, the time index $l$, and the hop size $L$. In a given frame of $X_0[k,l]$, then, two peaks should be apparent in the magnitude of the time-localized spectrum, one at $+\omega_0$ and one at $-\omega_0$; this is illustrated in Fig. 12.4 for the case $\omega_0 = 0.5$, $N = 128$, and $K = 512$. Note that *negative* frequencies are shown in the plots as bin indices $K/2 < k < K$. The two modulated copies of $W(\omega)$ interfere with each other such that the peaks in $|X_0[k,l]|$ are not necessarily at $\pm\omega_0$. If the frequency $\omega_0$ is small enough that the peaks in the modulated copies actually overlap, the resulting deviation in the peak location can be substantial; for sufficiently high $\omega_0$ such that the interference is limited to low-level sidelobes, as in Fig. 12.4, the error may be negligible. As will be seen in Sect. 12.2, the spectral peaks in the STFT play a key role in the analysis methods for sinusoidal modeling, as does the interference between multiple peaks.

For the constant-frequency sinusoid $x_0[t]$, the STFT varies only slightly from frame to frame, as indicated in (12.43). In contrast, consider the chirp signal

$$x_1[t] = A\cos(\omega_0 t + \alpha_0 t^2) . \quad (12.44)$$

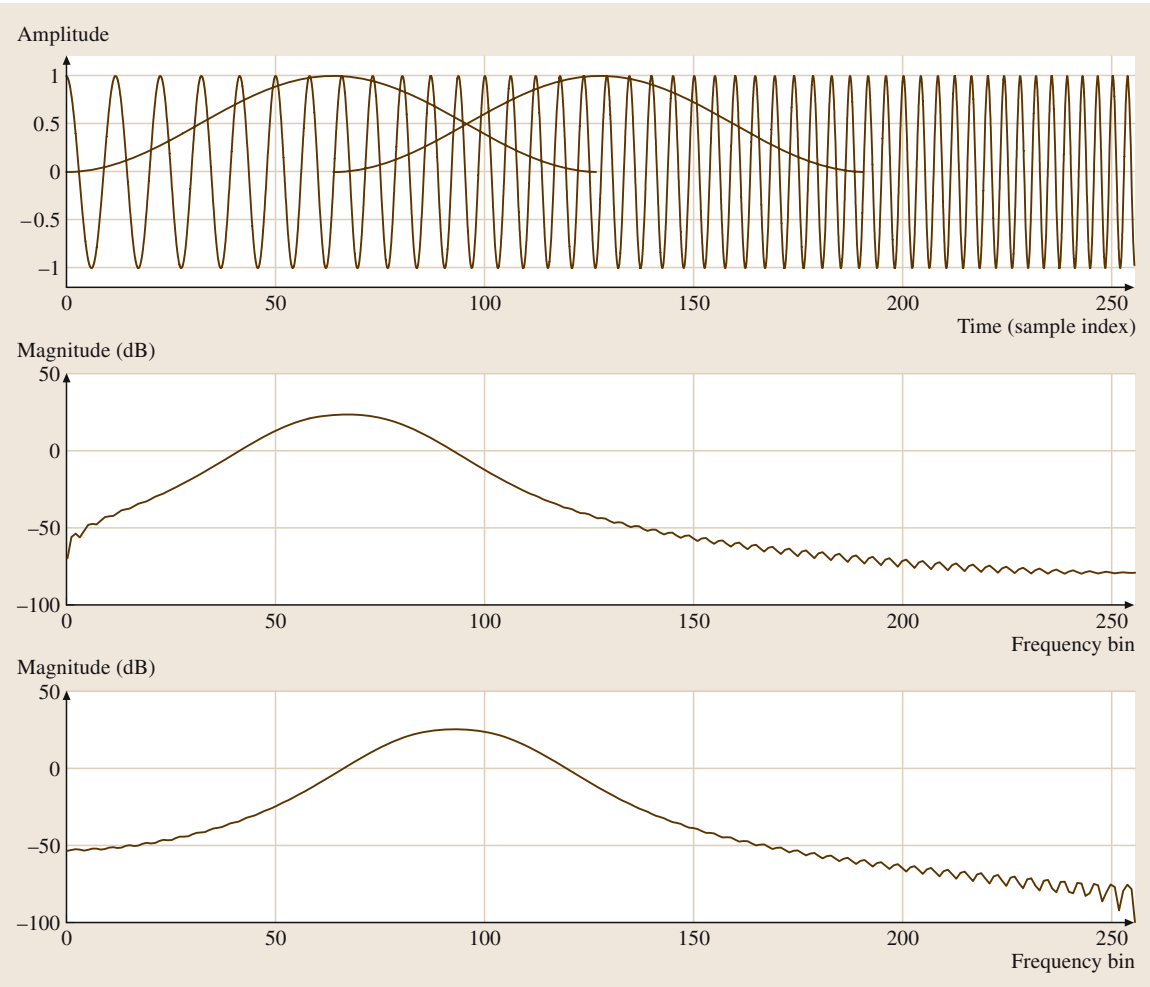



**Fig. 12.5** A chirp and the magnitude of its STFT for the two time windows indicated in the top plot



Amazon EX1010 Page 264

When the signal characteristics vary from frame to frame as for $x_1[t]$, the STFT in turn varies from frame to frame; from a filter bank perspective, the chirp moves across the subbands as time progresses. The frame-to-frame variation of the STFT is shown in Fig. 12.5 for $\omega_0 = 0.5$ and $\alpha_0 = 0.0025$; note that only the positive-frequency bins $0 \le k \le \frac{K}{2}$ are included in the plot. The progression of the chirp across the subbands is illustrated in Fig. 12.6, which depicts the (non-subsampled) STFT filter bank subband signals for several subbands.

As a final synthetic example, consider a sum of weighted sinusoids

$$x_2[t] = \sum_q A_q \cos(\omega_q t + \phi_q) \,, \tag{12.45}$$

where an arbitrary phase $\phi_q$ has been added to each component for the sake of generality. By linearity, the STFT of $x_2[t]$ can easily be derived from (12.43) as

$$X_2[k,l] = \sum_q \frac{A_q}{2} \mathrm{e}^{\mathrm{i}(\omega_q lL + \phi_q)} W(\omega_k - \omega_q) + \frac{A_q}{2} \mathrm{e}^{-\mathrm{i}(\omega_q lL + \phi_q)} W(\omega_k + \omega_q) \,. \tag{12.46}$$

In this compound signal, there is interference not only between positive- and negative-frequency components (as for $x_0[t]$), but also between components of different frequencies. A multi-sinusoid signal and its STFT magnitude are depicted in Fig. 12.7; note that the magnitude differs somewhat from window to window due to the time-dependent interaction of the various components.

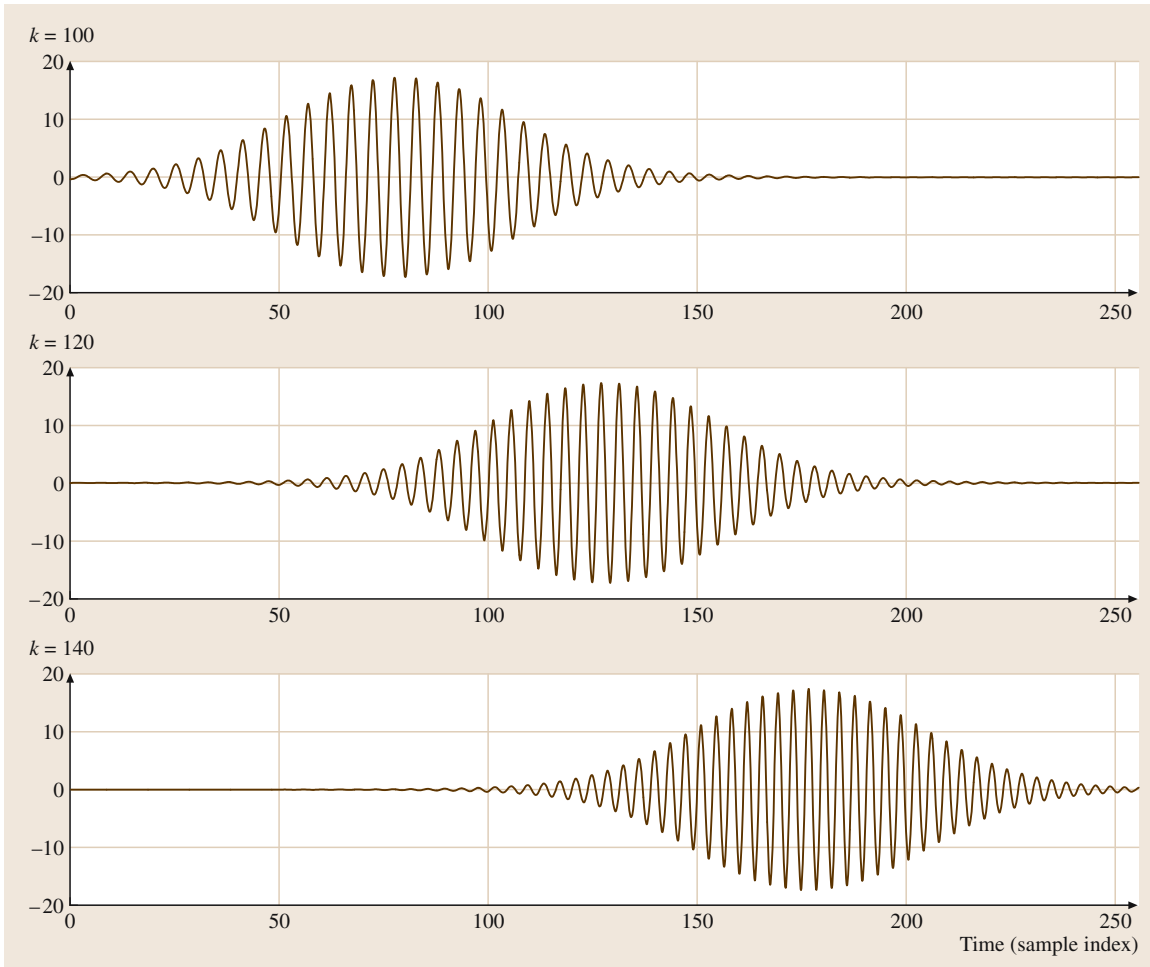



**Fig. 12.6** Non-subsampled STFT subband signals (real part) for the chirp signal of Fig. 12.5



Amazon EX1010 Page 265

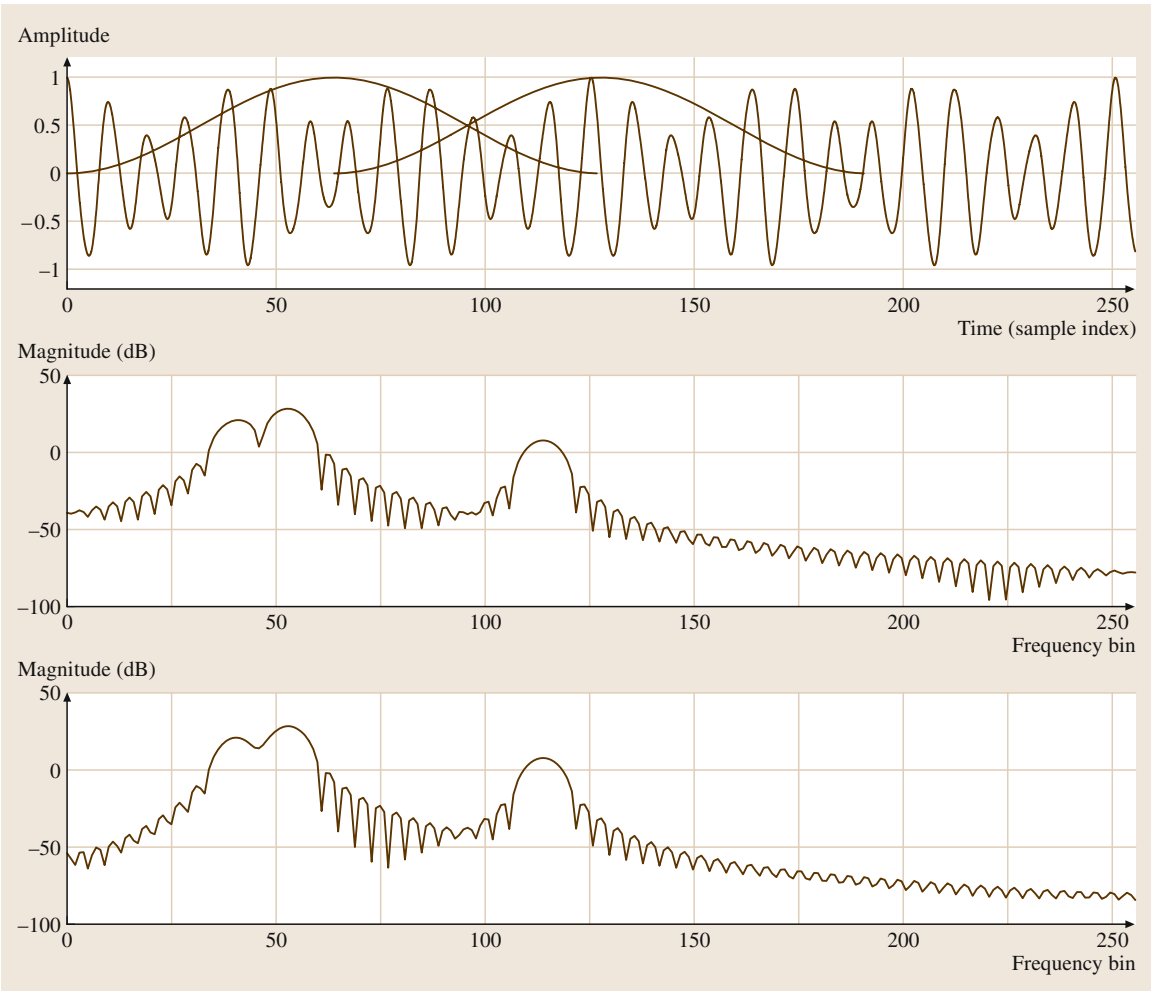



**Fig. 12.7** A sum-of-sinusoids signal and the magnitude of its STFT for the two indicated time windows; the spectrum exhibits lobes centered at the frequencies of the constituent sinusoids, which may overlap for closely spaced frequencies

These example signals are intended as a starting point for understanding the practical behavior of the STFT. Signals of interest for speech processing applications are of course typically more complex than sinusoids or chirps. However, for many speech signals, such as the example in Fig. 12.8, a sum of weighted sinusoids is a reasonable model. In a given frame, the STFT magnitude exhibits spectral peaks that can be interpreted as local sinusoids. Also, it is clear that the STFT magnitude changes from frame to frame, but some of the spectral peaks persist; these can be interpreted as persistent sinusoids in the speech signal. These observations suggest that the signal could be approximated as a sum of short-term or potentially long-term sinusoids, as will be explored in Sect. 12.2. Figure 12.9 depicts the (non-subsampled) STFT filter bank subband signals for several subbands for the speech signal of Fig. 12.8; the subband signals correspond to amplitude-modulated sinusoids at the subband center frequencies. Recall from (12.40) that the original speech signal can be reconstructed directly as the sum of the non-subsampled STFT subband signals; this example sheds light on the interpretation of this FBS reconstruction as a high-order sinusoidal model of the signal (with one sinusoid per STFT subband). This interpretation serves as another motivation for the sinusoidal modeling approaches to be described in Sect. 12.2 in that the sinusoidal model attempts to extract a representa-



Amazon EX1010 Page 266

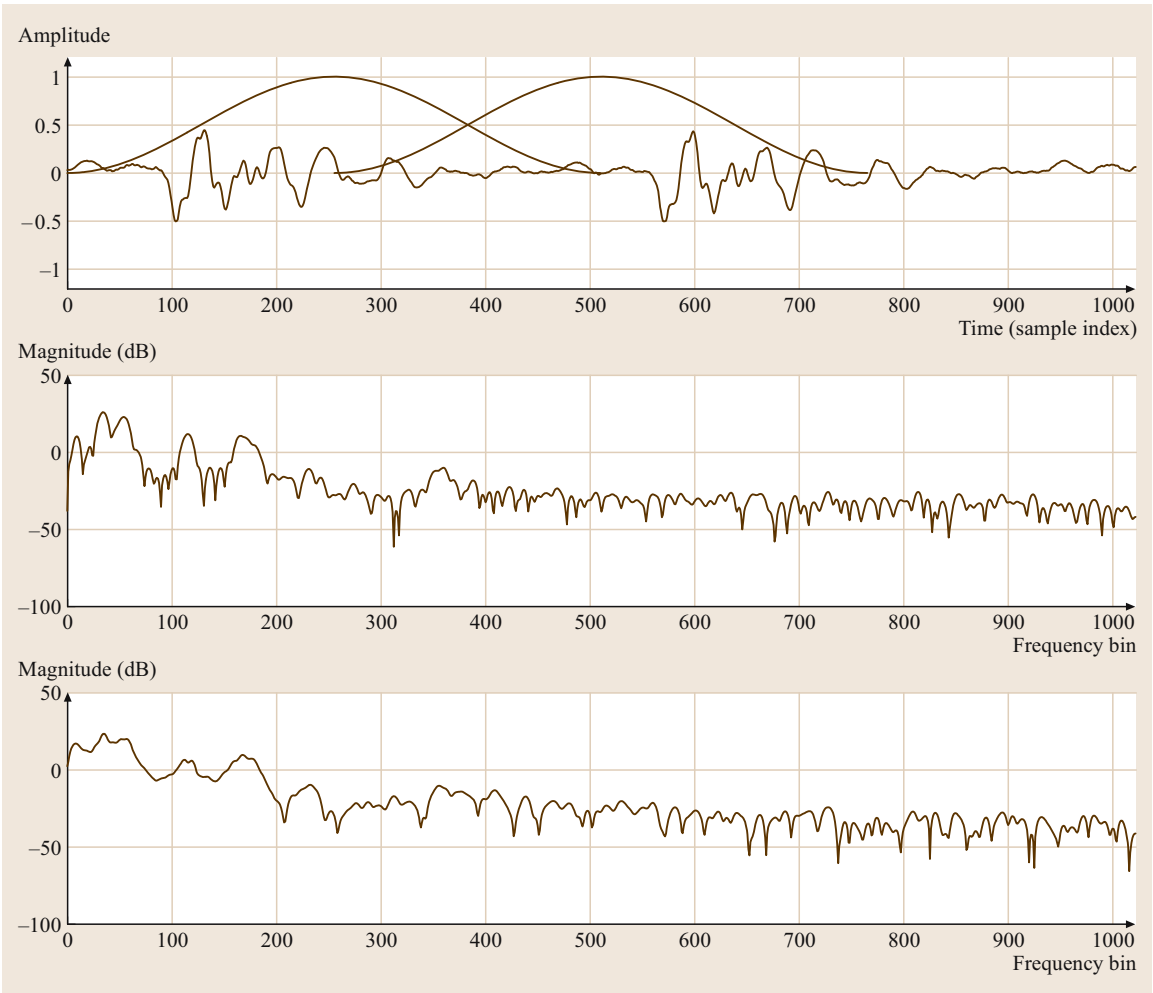



**Fig. 12.8** A speech signal and the magnitude of its STFT for the two indicated time windows; the STFT clearly changes from frame to frame, but some spectral peaks persist

tion that consists of fewer sinusoids and is therefore more amenable to some processing applications (such as coding).

### 12.1.8 Limitations of the STFT

The short-time Fourier transform has proven effective for many applications in speech signal processing, some of which will be discussed later in Sect. 12.3. In this section, several limitations of the STFT are described; these undermine the applicability of the STFT for some problems, and indeed served as the motivation for further advances in filter bank and signal modeling methods.

#### Signal Coding

Filter banks and sinusoidal transforms are widely used for signal coding since they can be designed to expose the salient signal structure and thereby enable robust perceptually motivated quantization schemes [12.26]. In the signal coding scenario, three design conditions arise for the signal transform or representation: it should provide perfect (or at least near-perfect) reconstruction in the event that no quantization is imposed on the transform coefficients; it should be critically sampled, i. e., it should not introduce additional data; and, the synthesis algorithm should use overlap-add to reduce blocking artifacts in the reconstruction. It was shown in Sect. 12.1.1 that the STFT can provide perfect recon-



Amazon EX1010 Page 267

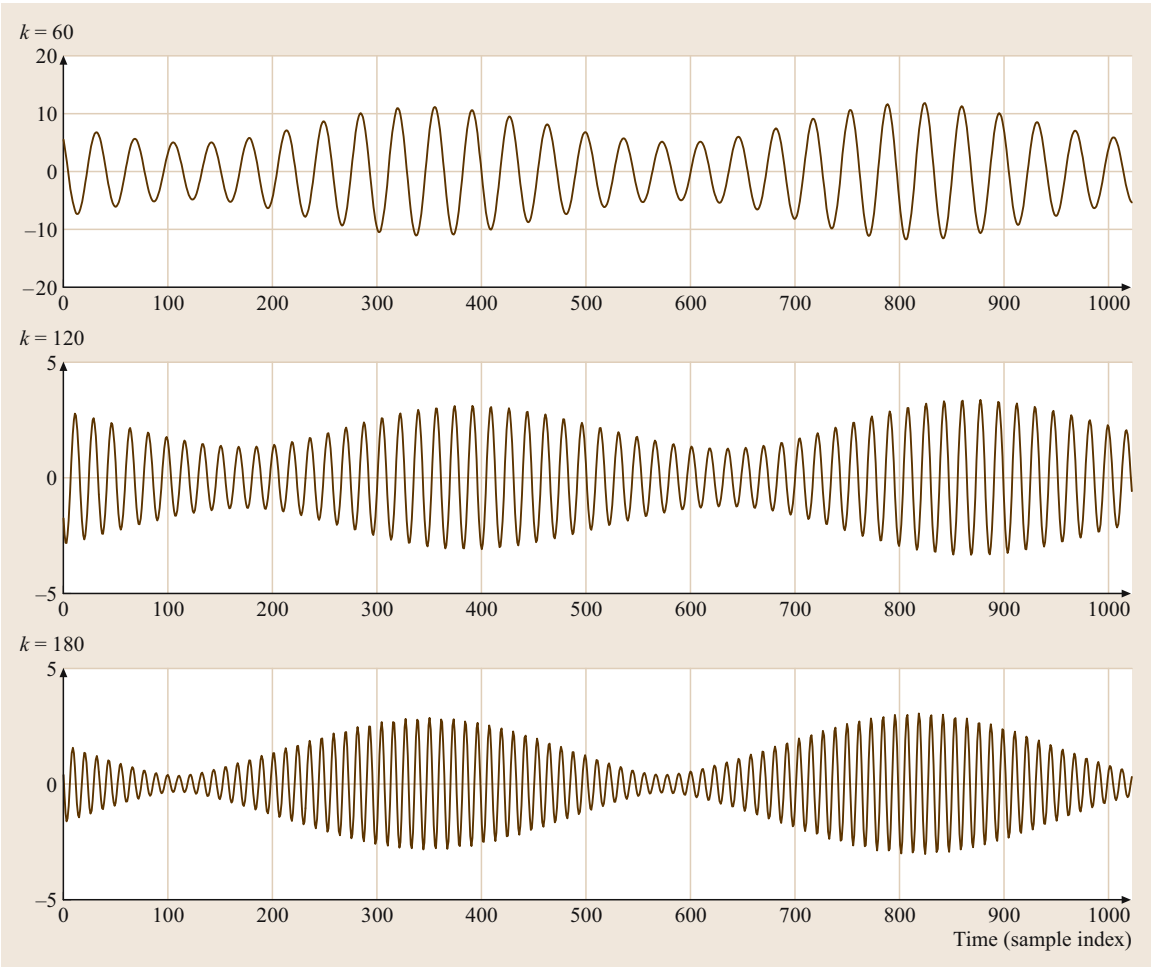



**Fig. 12.9** Non-subsampled STFT subband signals (real part) for the speech signal of Fig. 12.8

struction with overlap-add synthesis; however, the rate of the STFT representation exceeds that of the original signal. This data increase is straightforward to quantify using the sliding-window framework. For a real-valued signal, each time-localized STFT spectrum comprises $K$ unique values: one real value at $k = 0$, one real value at $k = K/2$, and $K/2 - 1$ complex values for $0 < k < K/2$. To achieve perfect reconstruction of a frame, $K \geq N$ is required to avoid time-domain aliasing. The data rate is then $K/L$ since $K$ values are needed to represent each $L$ samples perfectly. This can only take on a value of one, i. e., achieve critical sampling, when $K = L = N$, meaning that perfect reconstruction with critical sampling cannot be achieved with an STFT whose frames overlap.

To overcome this limitation of the STFT, sliding-window transforms that admit cancelable time-domain aliasing have been developed; these achieve perfect reconstruction at critical sampling in an overlap-add framework [12.21, 27, 28]. The STFT cannot be structured so as to incorporate such time-domain aliasing cancelation [12.4, 19]. Further discussion of this issue is beyond the scope of this chapter; the key point is that the STFT is not directly practical for signal coding due to an inherent data increase in the transform representation. As will be discussed in Sect. 12.2, this problem is addressed in the sinusoidal model by applying a parametric representation to the STFT to achieve compaction.

### Time–Frequency Localization

As discussed in the previous section, the design of STFT filter banks is very limited in the critically sampled case. The only real-valued prototype windows that lead to



Amazon EX1010 Page 268

perfect reconstruction filter banks are rectangular windows; this result is a discrete-time equivalent of the Balian–Low theorem, which essentially states that there are no continuous-time orthogonal short-time Fourier transform bases that are well localized in both time and frequency [12.4]. Further discussion is beyond the scope of this chapter; the interested reader is referred to the literature on cosine-modulated filter banks [12.4,5], which can achieve good localization in time and frequency and which can also satisfy the design conditions given earlier for signal coding.

#### Signal Adaptivity

Perhaps the most obvious (or at least most easily demonstrable) limitation of the short-time Fourier transform results from its fixed structure. Since the STFT analysis–synthesis does not adapt to the signal behavior, it yields a *noncompact* model for nonstationary signals [12.19]. Consider again the chirp example from Fig. 12.6, which illustrated that a sinusoid with time-varying frequency moves across the subbands as the frequency evolves. The subbands of the STFT filter bank do not characterize the chirp signal as a single evolving sinusoid but instead as a conglomeration of short-lived components, which is obviously not a compact way to represent a simple chirp. A similarly dispersed representation is evident in Fig. 12.9 for a pseudoperiodic speech signal. This lack of compactness suggests a shortcoming of the STFT for low-rate signal coding; furthermore, it indicates that the STFT does not directly provide a high-level understanding of the signal behavior. These issues motivate the use of a signal-dependent sparse parametric model of the STFT to capture the most relevant information about the signal, be it for coding or analysis. Such parametrization is the key idea in the sinusoidal models to be considered in Sect. 12.2.

## 12.2 Sinusoidal Models

Sinusoidal modeling can be viewed as an evolution of short-time Fourier transform and phase vocoder techniques. In this treatment, the shortcomings of the STFT serve to motivate the general sinusoidal model, which is developed as a parametric extension of the STFT. Two sinusoidal modeling approaches are described, one in which the prominent magnitude peaks of the STFT are tracked from frame to frame and one in which each time-localized spectrum of the STFT is modeled as a sum of sinusoids. These are loosely analogous to the interpretations of the STFT discussed in Sect. 12.1, but the parametric analysis–synthesis ultimately extends the flexibility of the representation beyond that afforded by sliding-window transforms or filter banks. The latter method of modeling the time-localized spectra is treated in detail in Sects. 12.2.4 and 12.2.5, where a sinusoidal analysis–synthesis algorithm based on *matching pursuit* is presented.

### 12.2.1 Parametric Extension of the STFT

The term *vocoder*, a contraction of *voice* and *coder*, was coined by early speech analysis–synthesis researchers [12.29], but has now become a general designation for a range of algorithms which are by no means limited to voice coding applications. Of particular interest as a starting point for considering the parametric extension of the STFT is the *channel vocoder*, which originated as a voice coder that represented a speech signal based on the characteristics of filter bank channels or subbands. In the STFT context, the speech signal is filtered into a large number of channels using an STFT analysis filter bank. Each of the subbands is then modeled in terms of its short-time energy; with respect to the $k$-th channel, this provides an amplitude envelope $A_k[t]$ that is used to modulate a sinusoidal oscillator at the channel center frequency $\omega_k$. The outputs of these oscillators are then accumulated to reconstruct the signal.

The channel vocoder parameterizes the subband signal in terms of its energy or amplitude only; the *phase vocoder* is an extension that also includes the phase behavior in the model parameterization [12.30]. In the literature, the term *phase vocoder* is sometimes used as a synonym for the short-time Fourier transform [12.31], but in some applications the approach involves interpreting the STFT analysis data with respect to a structure like the one shown in Fig. 12.10, where the subband signals are parameterized in terms of magnitude envelopes and functions that describe the frequency and phase evolution. These functions serve as inputs to a bank of oscillators that reconstruct the signal from the parametric model [12.10, 30, 32, 33]. If the analysis filter bank is subsampled, the sample-rate oscillator control functions are derived from the subsampled frame-rate STFT representation. This phase vocoder structure has



Amazon EX1010 Page 269

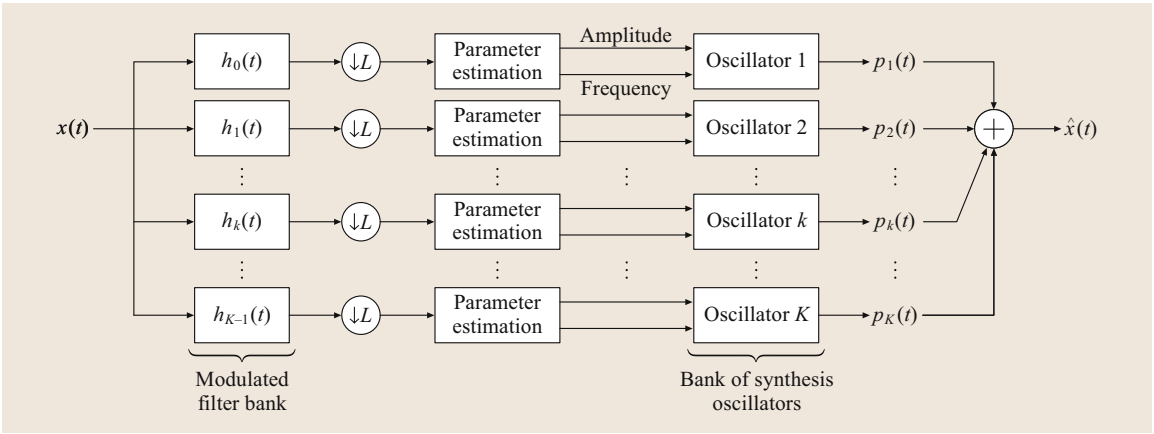



**Fig. 12.10** Block diagram of the phase vocoder. The amplitude and frequency (total phase) control functions for the $K$ oscillators are derived from the filter bank subband signals by the parameter estimation blocks

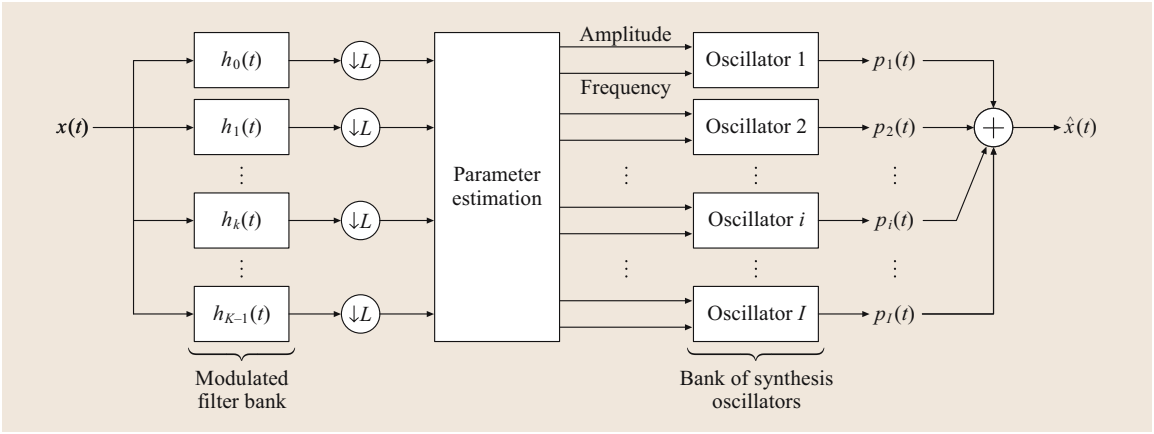



**Fig. 12.11** Block diagram of the partial-tracking sinusoidal model. The amplitude and frequency (total phase) control functions are derived from the filter bank outputs by tracking spectral peaks in time as they move from band to band for an evolving signal. The parameter estimation block detects and tracks spectral peaks; unless $I$ is externally constrained, the number of peaks detected dictates the number of oscillators used for synthesis

been widely applied to modification of speech and audio signals [12.31, 34–36].

The phase vocoder as depicted in Fig. 12.10 does not address the shortcoming of the STFT, i. e., lack of signal adaptivity. While its parametric nature does facilitate some modifications, the structure is still of limited use for compact modeling of evolving signals. A further generalization leads to a sinusoidal model in which partials, or sinusoidal components, are tracked in time from frame to frame, i. e., a *partial-tracking* sinusoidal model. The basic observation underpinning this model is that, if the signal consists of one nonstationary sinusoid such as a chirp, synthesis can be achieved with one oscillator. There is no need to implement an oscillator for every branch of the analysis filter bank. Instead, the outputs of the analysis bank can be examined across frequency for peaks, which are interpreted as sinusoids in the signal in the corresponding time frame. These spectral peaks can then be tracked from frame to frame as the signal evolves, and only one oscillator per tracked peak is required for synthesis. This structure is depicted in Fig. 12.11. Note that, because the synthesis for this case is based on a bank of sinusoidal oscillators, the prototype window for the analysis filter bank does not have to satisfy an overlap-add property.



Amazon EX1010 Page 270

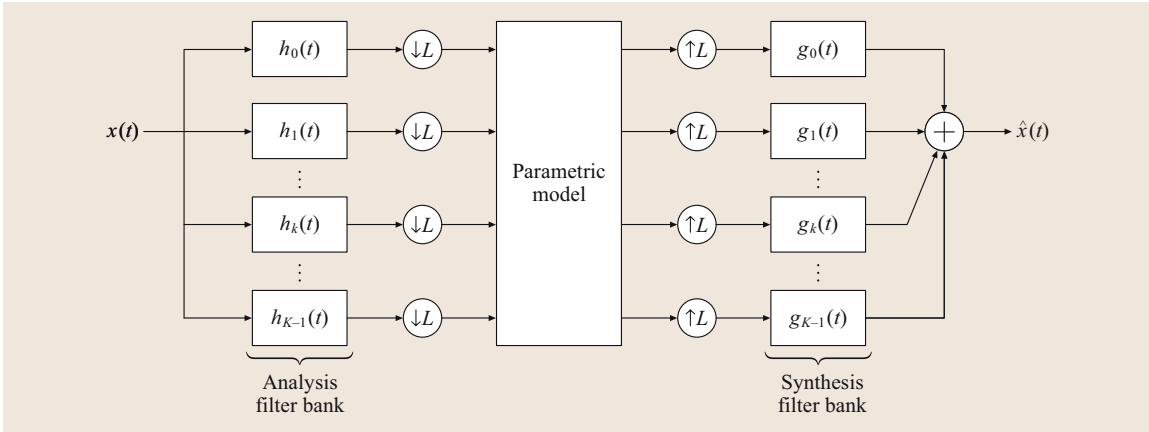



**Fig. 12.12** Block diagram of frequency-domain synthesis for sinusoidal modeling in an overlap-add framework. The parametric model includes the sinusoidal analysis and the construction of short-time spectra from the analysis data. The IDFT/OLA process in the frequency-domain synthesizer can be interpreted as an STFT synthesis filter bank

As an alternative to the partial-tracking approach, each time-localized spectrum of the STFT can be approximated as a sum of short-term sinusoids. Synthesis is then carried out using the spectra of the sinusoidal approximations; each approximate spectrum is inverse-transformed and the full reconstruction is carried out using an overlap-add process. Such *overlap-add* sinusoidal modeling is depicted in Fig. 12.12, where the synthesis procedure is interpreted as a filter bank. In this approach, the overlap-add property of the analysis and synthesis windows plays an important role.

### 12.2.2 The Sinusoidal Signal Model

The basic idea in sinusoidal modeling is to approximate a signal as a sum of sinusoids whose parameters vary in time:

$$\hat{x}[t] = \sum_{i=1}^{I[t]} a_i[t] \cos \Phi_i[t], \qquad (12.47)$$

where $a_i[t]$ is the amplitude variation of the $i$-th sinusoidal partial; the total phase $\Phi_i[t]$ represents its frequency and phase variation. Note that the number of sinusoidal components $I[t]$ may also vary in time (depending on modeling error criterion and/or bit-rate constraints). The formulation is expressed in terms of the reconstructed signal $\hat{x}[t]$ since this is a *synthesis* model; the goal in the analysis, then, is to extract model parameters from the original signal $x[t]$ so as to create a reasonable approximation via the model-based reconstruction, i. e., such that $\hat{x}[t] \approx x[t]$.

### 12.2.3 Sinusoidal Analysis and Synthesis

Analysis for the sinusoidal model can be carried out in a variety of ways. The fundamental task is that of finding time-varying amplitude, frequency, and phase parameters for a set of sinusoids such that their sum accurately represents the behavior of the original signal. In some methods, the parameters are extracted from the signal simultaneously, i. e., the parameter sets of all $I[t]$ partials are determined concurrently; in other cases, parameters are extracted sequentially, i. e., one component at a time. Numerous approaches have been considered for the estimation of such sinusoidal parameters, either simultaneously or sequentially; this multiplicity of approaches is partly a result of the wide range of applications beyond speech and audio processing. In this chapter, we limit consideration to methods based on the short-time Fourier transform, which was fundamental to early sinusoidal models [12.37, 38]. In this Section, we discuss these methods briefly before considering a sequential analysis scheme in detail in Sects. 12.2.4 and 12.2.5.

In simultaneous sinusoidal analysis based on the STFT, the model parameters for the $l$-th time frame are determined from the short-time magnitude spectrum $|X[k, l]|$ via peak-picking, perhaps with a curve-fitting scheme employed to account for the spectral shape of the analysis window [12.19, 37, 38]. The $I$ most prominent peaks are determined, and the model parameters $\{a_i, \omega_i, \phi_i\}_l$ are then given respectively by the magnitude, frequency, and phase of $X[k, l]$ at these spectral magnitude peaks. As illustrated in the STFT examples



Amazon EX1010 Page 271

of Sect. 12.1.7, such simultaneous estimation can be impacted by interference between multiple sinusoidal components in the signal.

In sequential analysis, the idea is to overcome the component interference problem by first selecting the most dominant component, then subtracting its contribution to the signal model from the analysis STFT, and thereafter iterating the estimation on the residual. This is referred to as analysis by synthesis [12.39]. By subtracting a component, the estimation of subsequent sinusoidal parameters is not affected by interference (spectral overlap) with that component. So, since the more-dominant components are extracted first, the estimation of minor components is not biased by their presence. Note of course that the parameter estimation is biased by the presence of components not yet extracted.

Either simultaneous or sequential analysis based on the STFT yields a set of sinusoidal parameters $\{a_i, \omega_i, \phi_i\}_l$ for each time frame of the STFT. There are two distinct methods to reconstruct an approximation of the original signal based on these parameter sets: parameter interpolation and overlap-add [12.19,37,39]. In parameter interpolation, partials are tracked from frame to frame via parameter matching, e.g., spectral peaks at the same or similar frequencies in successive time frames are assumed to correspond to a longer-term underlying sinusoidal component that spans multiple signal frames. The reconstruction is then achieved by using amplitude and phase interpolation to connect the matched parameter sets. In overlap-add modeling, the original signal is windowed and each windowed segment is modeled as a sum of sinusoids; for synthesis, each windowed segment is reconstructed and OLA is used to generate the full output. This approach overcomes the frame-to-frame tracking difficulties of time-domain interpolation and enables accurate approximation of the signal waveform, which is not readily achievable in interpolative synthesis.

Details on partial tracking for sinusoidal synthesis are available in the literature [12.37,38,40–42]; overlap-add synthesis for the sinusoidal model is discussed in [12.19,39,43,44]. In the remainder of Sect. 12.2, we focus on the overlap-add approach, providing a detailed treatment of overlap-add sinusoidal modeling based on the matching pursuit algorithm. The general pursuit algorithm is discussed in Sect. 12.2.4, and the specific sinusoidal case is developed in Sect. 12.2.5.

### 12.2.4 Signal Modeling by Matching Pursuit

Matching pursuit (MP) is an iterative analysis-by-synthesis algorithm for deriving compact signal approximations of the form

$$x[t] \approx \sum_{i=1}^{I} \alpha_i g_i[t], \tag{12.48}$$

where the expansion functions $g_i[t]$ are chosen from a dictionary [12.45]. MP operates by finding the element in the dictionary that best matches the signal, projecting the signal onto that element, and iterating on the projection residual. For a dictionary $D$, the task at the $i$-th stage of MP is to find the function $g_i[t] \in D$ and the coefficient $\alpha_i$ which minimize the norm of the residual

$$r_{i+1}[t] = r_i[t] - \alpha_i g_i[t], \tag{12.49}$$

where the initial condition is $r_1[t] = x[t]$. The solution is given by orthogonality [12.19,45]; for unit-norm dictionary functions $\boldsymbol{g}_i$,

$$\|\boldsymbol{r}_{i+1}\|^2 = \|\boldsymbol{r}_i\|^2 - |\alpha_i|^2 \tag{12.50}$$

$$\alpha_i = \langle \boldsymbol{g}_i, \boldsymbol{r}_i \rangle = \boldsymbol{g}_i^{\mathrm{H}} \boldsymbol{r}_i \tag{12.51}$$

$$\boldsymbol{g}_i = \arg\max_{\boldsymbol{g} \in D} |\langle \boldsymbol{g}, \boldsymbol{r}_i \rangle| , \tag{12.52}$$

where the signals are treated as column vectors and H denotes the conjugate transpose. The optimal function $\boldsymbol{g}_i$ is the one whose correlation with the signal has the largest magnitude.

Each iteration in MP requires all of the correlations between the dictionary functions and the current residual; these can be derived efficiently using an update formula [12.45]:

$$\langle \boldsymbol{g}, \boldsymbol{r}_{i+1} \rangle = \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle - \alpha_i \langle \boldsymbol{g}, \boldsymbol{g}_i \rangle . \tag{12.53}$$

The dictionary cross-correlation terms $\langle \boldsymbol{g}, \boldsymbol{g}_i \rangle$ can generally be precomputed and stored to reduce the computation required for each iteration. For some dictionaries, the correlation can be carried out efficiently without using this update formulation [12.46].

In basic MP, each iteration searches for a single function for the signal model. An alternative is to consider subspaces at each iteration; the goal in such *subspace pursuit* is to find the matrix $G$ that minimizes the norm of $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - G\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is a coefficient vector and the columns of $G$ are dictionary functions [12.19,46]. The orthogonality constraint $\langle \boldsymbol{r}_i - G\boldsymbol{\alpha}, G \rangle = 0$ yields the solution:

$$||\boldsymbol{r}_{i+1}||^2 = ||\boldsymbol{r}_i||^2 - \boldsymbol{r}_i^{\mathrm{H}} G (G^{\mathrm{H}} G)^{-1} G^{\mathrm{H}} \boldsymbol{r}_i , \tag{12.54}$$

$$\alpha_i = (G^{\mathrm{H}} G)^{-1} G^{\mathrm{H}} \boldsymbol{r}_i , \tag{12.55}$$

$$G_i = \arg\max_{G \in D} \{\boldsymbol{r}_i^{\mathrm{H}} G \alpha_i\} . \tag{12.56}$$



Amazon EX1010 Page 272

The search in (12.56) is computationally intensive unless $G$ consists of orthogonal vectors or has some other special structure [12.19, 47].

In some scenarios, such as sinusoidal modeling, it is useful to carry out pursuit based on the subspace spanned by a function and its complex conjugate. Here, the columns of $G$ are a function $\boldsymbol{g}$ and its conjugate $\boldsymbol{g}^*$. If the signal $\boldsymbol{r}_i$ is real and if $\boldsymbol{g}$ and $\boldsymbol{g}^*$ are linearly independent, the subspace pursuit can be simplified. Using (12.55), the optimal coefficients for a conjugate pair $\{\boldsymbol{g}, \boldsymbol{g}^*\}$ are

$$\begin{pmatrix} \alpha_i \\ \alpha_i^* \end{pmatrix} = \frac{1}{1-|c|^2} \begin{pmatrix} \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle - c \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle^* \\ \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle^* - c^* \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle \end{pmatrix} , \quad (12.57)$$

where $c = \langle \boldsymbol{g}, \boldsymbol{g}^* \rangle$ and where $\alpha_i$ is again a scalar, and the pursuit optimization of (12.56) is given by

$$\boldsymbol{g}_i = \arg \max_{\boldsymbol{g} \in D} (\langle \boldsymbol{g}, \boldsymbol{r}_i \rangle^* \alpha_i + \langle \boldsymbol{g}, \boldsymbol{r}_i \rangle \alpha_i^*) \quad (12.58)$$

$$= \arg \max_{\boldsymbol{g} \in D} (\mathrm{Re}\{\langle \boldsymbol{g}, \boldsymbol{r}_i \rangle^* \alpha_i\}) . \quad (12.59)$$

The algorithm simply searches for the function $g_i[t]$ that minimizes the norm of the residual

$$\begin{aligned} r_{i+1}[t] &= r_i[t] - \alpha_i g_i[t] - \alpha_i^* g_i^*[t] \\ &= r_i[t] - 2\,\mathrm{Re}\{\alpha_i g_i[t]\} . \end{aligned} \quad (12.60)$$

The resulting signal approximation has the form

$$x[t] \approx 2 \sum_{i=1}^{I} \mathrm{Re}\{\alpha_i g_i[t]\} . \quad (12.61)$$

The conjugate-subspace pursuit provides real models of real signals based on a complex dictionary, which enables the use of fast complex algorithms (e.g., the FFT) and simplifies phase estimation [12.19, 46]. Note however that this formulation only holds when $\boldsymbol{g}$ and $\boldsymbol{g}^*$ are linearly independent; other dictionary functions must be evaluated separately using appropriate selection metrics for comparison with the conjugate subspaces.

Since MP is a greedy algorithm, it does not necessarily arrive at the optimal global solution. Equations (12.50) and (12.54) indicate that the error of pursuit-based models decreases as the model order increases; indeed, the model converges to the signal in the mean-square sense [12.45]. However, for some model order $I$, there may be other $I$-term expansions based on the same dictionary that have a smaller modeling error.

Several variations of MP that attempt to find more-accurate solutions have been described [12.19, 48–52]. Such methods generally involve an orthogonalization at each step of the pursuit: in *orthogonal matching pursuit* [12.48], a new atom for the decomposition is selected at each iteration using the same metric as in MP, but after the selection the original signal is projected onto the current model subspace to minimize the modeling error; in *order-recursive matching pursuit* (ORMP) [12.49–51], also developed as *forward-orthogonal matching pursuit* [12.19] and *optimized orthogonal matching pursuit* [12.53], at each iteration the atom is selected so as to minimize the error for the resulting model subspace, which is equivalent to using the MP criterion and then orthogonalizing the dictionary to the selected atom [12.19]. These orthogonalization schemes typically outperform basic MP, but improved performance is not guaranteed; the resulting models may actually have a larger error than the basic MP model, and the additional computation required for these approaches may be a hindrance for real-time modeling.

One strategy for maintaining fast computation and improving the performance of the basic MP model is simply to accept the decomposition provided by MP and project the signal onto the model subspace after the pursuit is complete [12.54, 55]. This post-orthogonalization, or terminal back-projection, never yields a worse model than the basic MP, and entails significantly less computation than OMP or ORMP.

### 12.2.5 Sinusoidal Matching Pursuits

In many cases, e.g., some image coding schemes, the matching pursuit modeling algorithm is applied to the entire signal in question. Such an approach poses problems for natural speech and audio signals in that the pursuit search is computationally unwieldy for long-duration signals; it is clearly inapplicable for online applications such as speech coding for real-time communication. In order to handle general speech and audio, then, a sliding-window framework will be used for the sinusoidal modeling pursuit presented in this section. The idea in the analysis is to use MP to model windows of the signal as sums of windowed sinusoids; synthesis of the signal reconstruction is then carried out via overlap-add of these modeled windows. This approach can be specified mathematically as follows. Let $x_l[n]$ be the $l$-th windowed segment of the signal, namely $x_l[n] = w[n]x[n + lL]$, where $w[n]$ is a window of length $N$ applied to the signal with a stride of $L$. OLA signal reconstruction from the windowed segments is given by

$$\hat{x}[t] = \sum_l x_l[t - lL] = x[t] \sum_l w[t - lL] , \quad (12.62)$$



Amazon EX1010 Page 273

which shows that $w[n]$ must satisfy the overlap-add property of (12.11) to avoid amplitude modulation artifacts in the reconstruction. For MP-based OLA modeling, each segment $x_l[n]$ is modeled using MP. Then, the convergence of the segment models implies a convergence of the OLA reconstruction to the original signal; and, if each segment is modeled exactly, perfect reconstruction is achieved.

The use of matching pursuit for sinusoidal modeling has been explored in various contexts in the literature. In [12.39], which predates the formalization of MP in [12.45], correlation is used as a metric for estimation of the amplitudes, frequencies, and phases of sinusoidal components for overlap-add synthesis. In [12.19,46,55], conjugate-subspace pursuit based on a dictionary of complex sinusoids is used to derive signal models. MP is used for fast parameter estimation in [12.56]; in [12.57], an approach related to MP-based OLA sinusoidal modeling is presented; sinusoidal speech coding based on MP is considered in [12.58]. Psychoacoustic metrics for sinusoidal matching pursuit of audio signals are described in [12.59, 60]. While such metrics have proven effective for low-rate parametric audio coding, we adhere to a mean-squared error metric in this treatment.

### 12.2.6 Sinusoidal Analysis

In this section, we expand on the MP-based OLA sinusoidal model described in [12.55], wherein the object of the pursuit is to find a set of windowed sinusoids to accurately model each windowed segment. The MP dictionary contains the functions

$$g[k,n] = \bar{w}[n]\,\mathrm{e}^{\mathrm{i}\omega_k n} \tag{12.63}$$

where $k$ is a frequency index $(0 \le k \le K-1)$, $\omega_k = 2\pi k/K$, and $\bar{w}[n]$ is a normalized version of the $N$-point analysis window: $\bar{w}[n] = \frac{w[n]}{\|w[n]\|}$. The dictionary has $K$ elements; completeness $(K \ge N)$ is required for convergence of the pursuit, and overcompleteness $(K > N)$ improves the accuracy of low-order models [12.19,45,46]. In the above formulation, a local time index $n$ is used instead of the global index $t$ used in Sect. 12.2.4 to emphasize that the signal is being modeled on a per-window basis instead of globally; note that this is analogous to the local time index used in practical sliding-window STFT implementations (as explained in Sect. 12.1.1). The implications of using a global versus a local time index in the modulation of sinusoidal atoms for MP are discussed in [12.46], especially with regard to increased computational requirements for the pursuit in the global case.

The MP dictionary defined in (12.63) consists of complex exponentials. Since the goal is to model a real-valued audio signal, the conjugate-subspace pursuit algorithm is applicable, and for this dictionary, the formulation can be greatly simplified. To begin with, consider the dictionary cross-correlation $\Gamma[k,m] = \langle g[k,n], g[m,n]\rangle$; this can be computed as follows:

$$\begin{aligned}\Gamma[k,m] &= \sum_{n=0}^{N-1} \bar{w}[n]^2\,\mathrm{e}^{-\mathrm{i}2\pi(k-m)n/K} \\ &= W_K[k-m]\,,\end{aligned} \tag{12.64}$$

where $W_K[k] = \mathcal{F}_K\left\{\bar{w}[n]^2\right\}$. The expression $\mathcal{F}_K\{\}$ denotes the $K$-point DFT; in (12.64), note that the summation terms from $n = N$ to $K-1$ are zero by definition of the window. The conjugate cross-correlation $\langle g[k,n], g[k,n]^*\rangle$ can then be expressed in terms of $W_K[k]$:

$$\begin{aligned}c[k] &= \langle g[k,n], g^*[k,n]\rangle \\ &= \Gamma[k,-k] \\ &= W_K[2k]\,.\end{aligned} \tag{12.65}$$

The correlation term $\langle g[k,n], r_i[n]\rangle$ can also be expressed in the DFT domain:

$$\begin{aligned}R_i[k] &= \langle g[k,n], r_i[n]\rangle \\ &= \sum_{n=0}^{N-1} \bar{w}[n] r_i[n]\,\mathrm{e}^{-\mathrm{i}2\pi nk/K} \\ &= \mathcal{F}_K\left\{\bar{w}[n] r_i[n]\right\}\,.\end{aligned} \tag{12.66}$$

Given the quantities $c[k]$ and $R_i[k]$, the optimal expansion coefficients given in (12.57) for the pair $\{g[k,n], g^*[k,n]\}$ can be specified for the sinusoidal modeling case as:

$$\begin{pmatrix}\alpha_i[k] \\ \alpha_i^*[k]\end{pmatrix} \quad \text{with} \quad \alpha_i[k] = \frac{(R_i[k] - c[k]R_i[k]^*)}{1-|c[k]|^2}\,. \tag{12.67}$$

Note that these coefficients are complex and include both the amplitude and phase of the conjugate-subspace components.

The conjugate-subspace pursuit optimization given in (12.58) and (12.59) can be specified for the sinusoidal pursuit as

$$g_i[k,n] = \arg\max_{g\in D} \Psi_i[k]\,, \tag{12.68}$$



Amazon EX1010 Page 274

where the optimization metric $\Psi_i[k]$ is only a function of $R_i[k]$ and $c[k]$:

$$\Psi_i[k] = 2\,\mathrm{Re}\{R_i[k]^*\alpha_i[k]\}\,. \tag{12.69}$$

Note that when $|c[k]| = |W_K[2k]| = 1$, the denominator in (12.67) is zero. Since $\bar{w}[n]$ has unit norm, $W_K[0] = W_K[K] = 1$ so that $|c[k]| = 1$ at $k = 0$ and $k = \frac{K}{2}$, namely when $g[k, n]$ is purely real. In this case, the conjugate-subspace formulation of the pursuit is not applicable since $g[k, n]$ and its conjugate are linearly dependent (identical, indeed). To account for this, each stage of the pursuit must compare $\Psi[k]$ for $1 \le k \le \frac{K}{2} - 1$ with the metrics $|\langle g[k, n], r_i[n]\rangle|^2$ for $k \in \{0, \frac{K}{2}\}$ to determine if a conjugate pair or a real function should be added to the model at that iteration; since $\Psi_i[k]$ and $|\langle g[k, n], r_i[n]\rangle|^2$ each quantify how much energy will be removed from the residual by the respective dictionary atoms, this is a fair comparison to determine the selection. Note that in practice components at $\frac{K}{2}$ are not considered and the $k$ range of the pursuit is narrowed to focus the modeling to the frequency range of interest, e.g., an 8 kHz bandwidth for modeling speech signals. Note also that the $k$ range for the pursuit covers only *positive* frequencies; unlike the basic pursuit metric, the conjugate-subspace pursuit metric is symmetric in $k$; with respect to search complexity, a sinusoidal dictionary of size $K$ is essentially reduced in size by a factor of two by the conjugate-subspace formulation.

The conjugate-subspace pursuit metric given in (12.69) accounts for the spectral overlap of positive- and negative-frequency components. If magnitude correlations with complex sinusoids are used as the pursuit metric for modeling a real signal, parameter estimates derived from the positive-frequency spectrum can be biased by the negative-frequency components, especially at low frequencies; using such correlations is analogous to DFT peak-picking, where a similar bias occurs. The bias can be avoided by using conjugate-subspace pursuit, or quadrature models, which are functionally equivalent but less computationally efficient [12.39, 57]. An example of the estimation bias is depicted in Fig. 12.13, which shows the conjugate-pursuit metric of (12.69) and the correlation metric $|\langle \boldsymbol{g}, \boldsymbol{r}_i\rangle|^2$ for the first sinusoidal pursuit iteration for modeling a signal consisting of a single real sinusoid. In the simulation, the signal is of length $N = 128$, the dictionary is of size $K = 512$, and $w[n]$ is a Hanning window. The conjugate-pursuit metric yields a correct estimate of the sinusoidal parameters, while the correlation metric yields inaccurate estimates since its peak is at a lower frequency than the actual sinu-

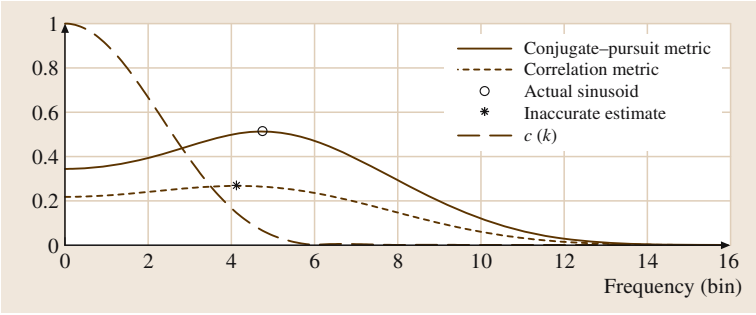



**Fig. 12.13** Metrics for sinusoidal pursuit. For pursuit of a single real sinusoid (*open circle*), the conjugate-subspace pursuit metric exhibits a peak at the correct location and therefore results in a correct estimate of the sinusoid's parameters; the correlation metric of basic MP yields incorrect parameter estimates

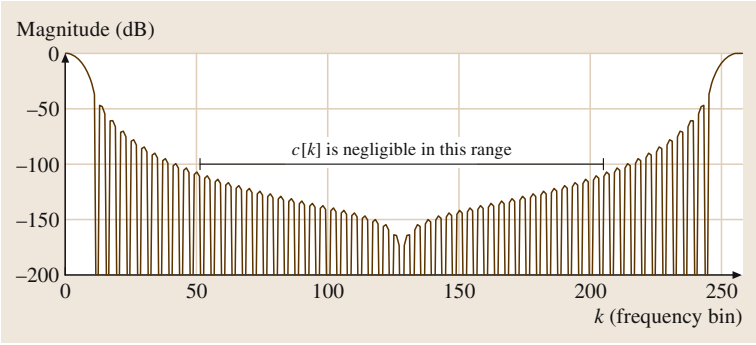



**Fig. 12.14** Conjugate-pair cross-correlations for a Hanning-windowed sinusoidal dictionary with $N = 64$ and $K = 512$. At mid-range frequencies, a complex sinusoid and its conjugate can be reasonably approximated to be orthogonal, such that $c[k]$ can be neglected in (12.67)

soid due to the interaction of the negative-frequency component.

The extent of this interaction for the $N \times K$ pursuit dictionary is described by $c[k]$, the cross-correlation between a complex dictionary sinusoid of frequency $2\pi k/K$ and its conjugate. This conjugate cross-correlation $c[k]$ is also plotted in Fig. 12.13; at the frequency of the sinusoid in the signal, the conjugate components are not orthogonal and $c[k]$ is not negligible.

Despite the estimation bias incurred, some pursuit schemes described in the literature neglect the interaction of positive- and negative-frequency components; in these approaches, complex positive-frequency components are estimated using magnitude correlations between the real signal and a complex sinusoidal dictionary, and the model is subsequently made real by adding a corresponding conjugate at the end of each pursuit iteration [12.59, 60]. This approach is equivalent to assuming that $c[k] \approx 0$, which can be seen in Fig. 12.14 to be a reasonable approximation for mid-range frequencies. Since $W_K[K] = W_K[0] = \sum_n \tilde{w}[n]^2$, $c[k] = W_K[2k]$ takes on significant values in the vicinity of both $k = 0$ and $k = K/2$, so estimation bias is incurred for both low- and high-frequency components if $c[k]$ is altogether neglected, and modeling performance is degraded with respect to the conjugate-subspace approach. An example of mean-square modeling performance is given in Fig. 12.15, which shows the convergence of three pursuit-based sinusoidal models as a function of model order for a speech signal; *basic pursuit* refers to modeling a real signal with complex sinusoids and *basic real pursuit* refers to adding a conjugate component at the end of each iteration. Using the conjugate-subspace pursuit approach tends to improve the rate of model convergence, but this modeling improvement is sacrificed in the cited methods since adhering to the $c[k] \approx 0$ assumption (for all frequencies) facilitates the incorporation of psychoacoustic modeling criterion [12.59, 60].

In addition to the pursuit metric, it is necessary to formulate the correlation update for the sinusoidal pursuit. For the sinusoidal dictionary, the efficient update in (12.53) can be specified as follows. For $g[k_i, n]$ chosen at the $i$-th iteration, the new residual at stage $i+1$ is

$$r_{i+1}[n] = r_i[n] - \alpha_i g[k_i, n] - \alpha_i^* g^*[k_i, n] \,, \quad (12.70)$$

where the shorthand $\alpha_i = \alpha_i[k_i]$ is being used; note that the initial condition for modeling the $l$-th signal block is $r_1[n] = x_l[n] = w[n]x[n + lL]$. Taking the correlation of both sides of (12.70) with a dictionary atom $g[k, n]$, the correlation of the updated residual with the dictionary

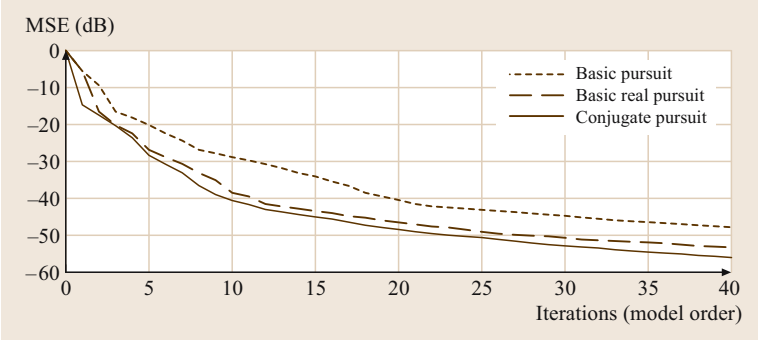



**Fig. 12.15** Comparison of sinusoidal model convergence for basic pursuit, basic pursuit with a conjugate component added at the end of each iteration, and conjugate-subspace pursuit. In the simulation, $N = 256$ and $K = 1024$

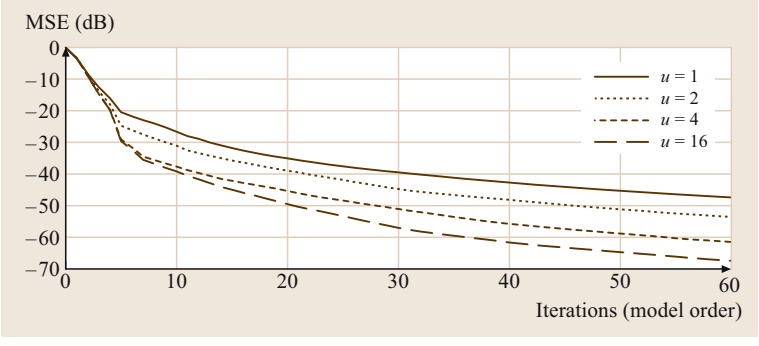



**Fig. 12.16** Sinusoidal pursuit convergence as a function of model order for different frequency-domain oversampling factors. In the simulation, the block size is $N = 1024$ and the DFT size is $K = uN$, where $u$ is indicated in the legend. The rate of model convergence increases as the dictionary size is increased



Amazon EX1010 Page 276

can be written as

$$R_{i+1}[k] = R_i[k] - \alpha_i W_K[k-k_i] - \alpha_i^* W_K[k+k_i] \,. \tag{12.71}$$

With this result, we see that all of the computation needed for the proposed sinusoidal pursuit can be carried out in the $K$-point DFT domain. The choice of $K$, namely the size of the dictionary, dictates the convergence rate; a larger dictionary, i. e., a larger oversampling ratio $K/N$ for the DFT, leads to faster convergence of the signal model. Examples of sinusoidal model convergence for various oversampling ratios are depicted in Fig. 12.16. The modeling improvement achieved by using a larger $K$ comes at the cost of additional model data; more bits are required for indexing into larger dictionaries.

The sinusoidal pursuit algorithm for modeling a signal frame $x_l[n] = w[n]x[n+lL]$ can be summarized as follows, starting with $i = 1$:

1. Compute $R_1[k] = \mathcal{F}_K\{\bar{w}[n]x_l[n]\}$.
2. Compute $\alpha_i[k] = \frac{(R_i[k]-c[k]R_i[k]^*)}{1-|c[k]|^2}$.
3. Compute $\Psi_i[k] = 2\,\mathrm{Re}\{R_i[k]^*\alpha_i[k]\}$.
4. Find $k_i = \arg\max_k \Psi_i[k]$.
5. Add $g[n,k_i]$ and $\alpha_i[k_i]$ to the model.
6. Unless a stopping criterion (such as model error) is met, compute $R_{i+1}[k] = R_i[k] - \alpha_i W_K[k-k_i] - \alpha_i^* W_K[k+k_i]$ and iterate from step 2.

The pursuit iterations are discontinued when the signal model meets some criterion such as an error threshold or a maximum allowed order; in coding applications, a psychoacoustic constraint or a bit budget might be incorporated. Then, the model coefficients can be refined by projecting the signal onto the model subspace in a terminal back-projection step. This post-orthogonalization can be carried out by an explicit projection, which would take the form of (12.55) with $G$ containing the model functions, or by a gradient descent, which is more cost-efficient than an explicit projection.

Note that, if the sinusoidal model does not fit the signal well, a higher model order is required to reach a given modeling error threshold; an obvious example of such a mismatched signal is white noise, since it has no coherent pseudo-sinusoidal structure. To account for such cases, it is useful to include additional model components as in sinusoid-plus-residual models [12.19,38,61–64] and sinusoids-plus-transients-plus-noise representations [12.41]. Such hybrid models have been shown to effectively overcome the problem of model mismatch. Multiresolution analysis–synthesis frameworks have also proven useful for improving the effectiveness of the sinusoidal model for representing arbitrary input signals [12.55,65].

### 12.2.7 Sinusoidal Synthesis

Synthesis in the MP-based sinusoidal model is achieved by overlap-add of the modeled segments: $\hat{x}[t] = \sum_l \hat{x}_l[t-lL]$. For each segment, the signal model derived by the conjugate-subspace sinusoidal pursuit has the following form, assuming only conjugate pairs are selected for the model (no components at $k = 0$ or $k = \frac{K}{2}$ are selected):

$$\begin{aligned}\hat{x}_l[n] &= \sum_{i=1}^{I_l} \alpha_{l,i}\bar{w}[n]\mathrm{e}^{\mathrm{i}2\pi n k_{l,i}/K} \\ &\quad + \sum_{i=1}^{I_l} \alpha_{l,i}^*\bar{w}[n]\mathrm{e}^{-\mathrm{i}2\pi n k_{l,i}/K} \\ &= 2\bar{w}[n]\sum_{i=1}^{I_l} |\alpha_{l,i}|\cos\left(\frac{2\pi n k_{l,i}}{K} + < \alpha_{l,i}\right) \,.\end{aligned} \tag{12.72}$$

Efficient synthesis for this model can be achieved using the inverse FFT. Letting $\bar{W}[k] = \mathcal{F}_K\left\{\bar{w}[n]\right\}$, the $K$-point DFT of $\hat{x}_l[n]$ is

$$\begin{aligned}\hat{X}_{l,K}[k] &= \sum_{i=1}^{I_l} \alpha_{l,i}\bar{W}_K[k-k_{l,i}] \\ &\quad + \alpha_{l,i}^*\bar{W}_K[k+k_{l,i}] \,.\end{aligned} \tag{12.73}$$

For a $K$-point synthesis DFT, synthesis is achieved by accumulating weighted and shifted versions of the normalized window spectrum $\bar{W}[k]$. If integer oversampling $\mu = \frac{K}{N}$ is used in the analysis, i. e., if the analysis DFT is zero-padded with $(\mu-1)N$ zeros, the oversampling redundancy can be removed to reduce the synthesis complexity by directly deriving the $N$-point DFT of $\hat{x}_l[n]$:

$$\begin{aligned}\hat{X}_{l,N}[l] &= \sum_{i=1}^{I_l} \alpha_{l,i}\bar{W}_K[\mu l - k_{l,i}] \\ &\quad + \alpha_{l,i}^*\bar{W}_K[\mu l + k_{l,i}] \,.\end{aligned} \tag{12.74}$$

An $N$-point inverse DFT is then used for synthesis.

The MP-based overlap-add sinusoidal model uses the same window $w[n]$ for analysis and synthesis. It is chosen to have a sparse spectral representation, i. e., $\bar{W}_K[k]$ and $W_K[k]$ both consist of a concentrated main lobe and low-level sidelobes. For $K = N$, general Blackman–Harris or cosine-sum windows are good



Amazon EX1010 Page 277

choices since their spectra have only a few nonzero values, which simplifies the computation of the synthesis spectra. When $K > N$, the window spectra have many low-level nonzero values in the sidelobe regions. To reduce the synthesis complexity, these values can be excluded in the frequency-domain synthesis of (12.74) at the cost of minor errors [12.18]. In the analysis, the sidelobe regions of $W_K[k]$ and $c[k]$ can be excluded to reduce the computational cost, but this counteracts some of the advantages of analysis by synthesis and may lead to degraded parameter estimation and the identification of sidelobes as sinusoidal signal components.

Since the analysis and synthesis for the sinusoidal matching pursuit algorithm can be carried out in the DFT domain, this modeling approach fits into the framework depicted in Fig. 12.12 as anticipated earlier. In the parametric model block of the figure, each time-localized spectrum of the STFT $X[k, l]$ is approximated via the sinusoidal pursuit; the synthesis accumulates the spectral representations of windowed sinusoids to construct $\hat{X}[k, l]$, which is provided to the synthesis filter bank to generate the time-domain output. From this perspective, the synthesis window $g[n]$ is a rectangular $K$-point window, or, for the integer oversampling case described above, $g[n]$ is an $N$-point rectangular window. Alternatively, the synthesis window $g[n]$ can be selected to be the modeling window $\bar{w}[n]$, in which case the parametric model block simply provides a sparse set of inputs to the synthesis filter bank. Only the synthesis filters corresponding to the selected sinusoidal model components are fed nonzero inputs at time $l$, and these inputs are given by the corresponding component weights deter-

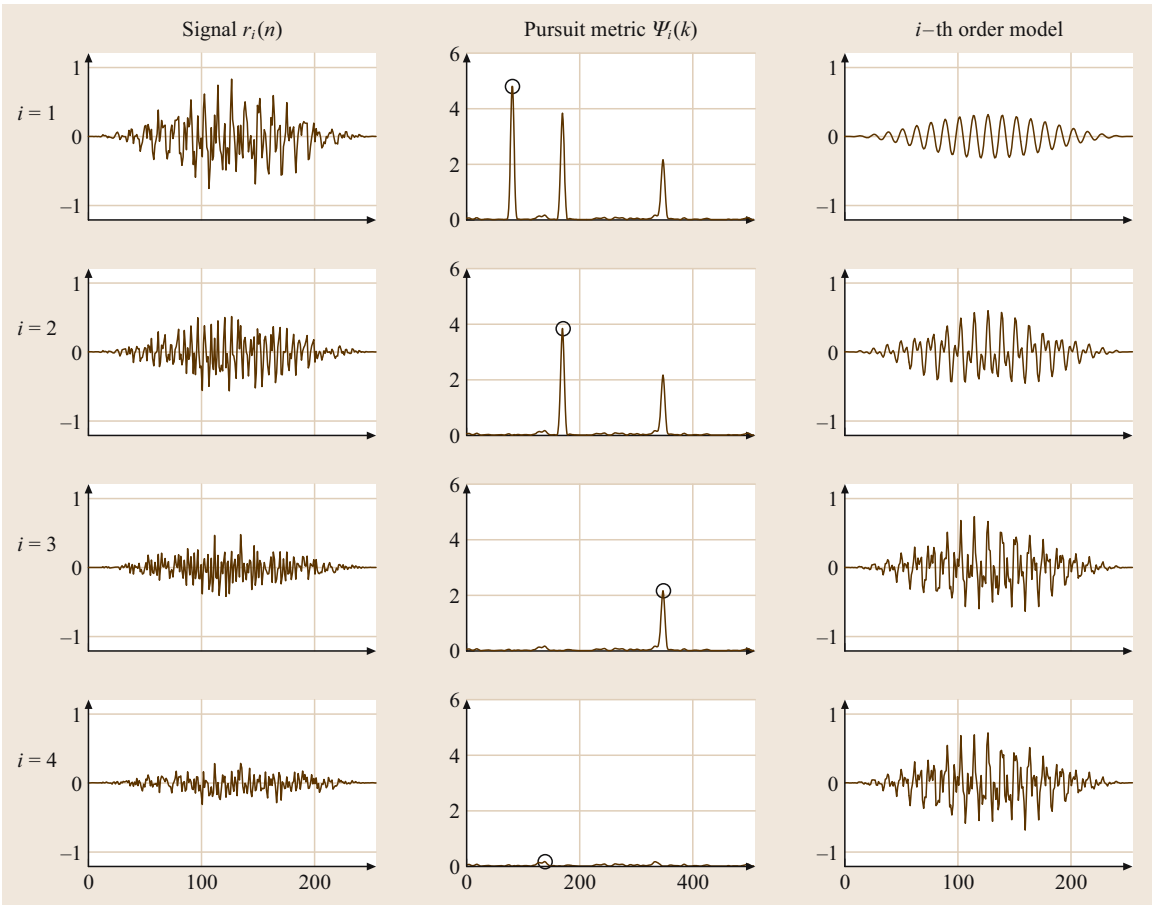



**Fig. 12.17** Iterations of sinusoidal matching pursuit for a synthetic multi-sinusoid signal. The first column shows the residual $r_i[n]$ at the start of the iteration; the second column shows the MP metric $\Psi_i[k]$ from (12.69) and its maximum (*open circles*); the third column shows the $i$-term signal model after the $i$-th iteration



Amazon EX1010 Page 278

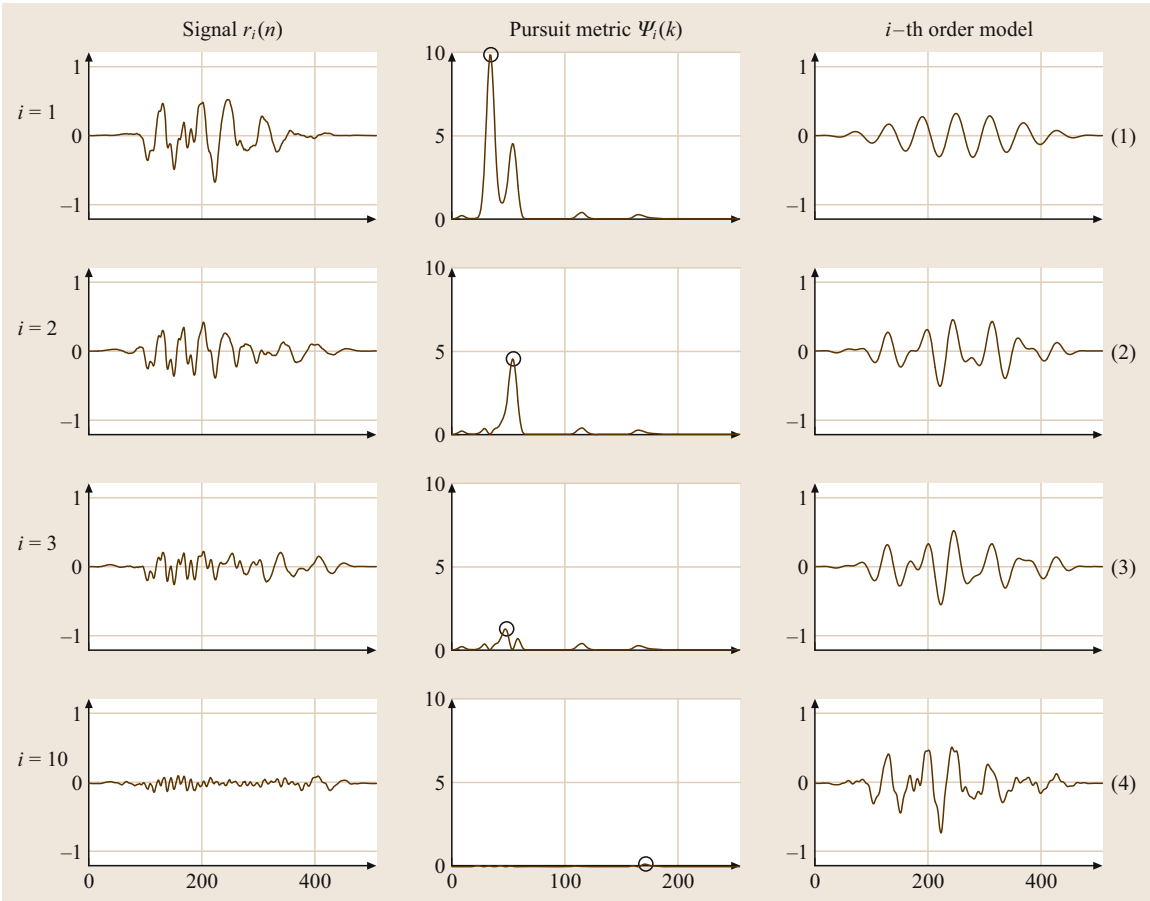



**Fig. 12.18** Iterations of sinusoidal matching pursuit for a speech signal. The first column shows the residual $r_i[n]$ at the start of the iteration; the second column shows the MP metric $\Psi_i[k]$ from (12.69) and its maximum (*open circles*); the third column shows the $i$-term signal model after the iteration. Rows 1–3 show the first three pursuit iterations; the fourth row shows the tenth iteration

mined by the pursuit. In this interpretation, the synthesis filter bank accumulates the weighted windowed sinusoids. In practice, the spectral synthesis approach is used since it leads to more-efficient implementations; the synthesis filter bank is not explicitly realized in the time domain but rather as an inverse DFT and overlap-add procedure.

Figure 12.17 depicts four iterations of the sinusoidal matching pursuit algorithm for a synthetic multi-sinusoid signal. The first column shows $r_i[n]$, the time-domain signal that the pursuit is attempting to match at iteration $i$. Recall from the algorithm description that this time-domain residual signal is not computed explicitly since the required metrics can all be maintained in the frequency domain; it is simply shown here to illustrate the modeling behavior. The second column in the figure shows the conjugate-subspace pursuit metric $\Psi_i[k]$; the circle indicates the peak identified by the metric maximization search. Finally, the third column depicts the signal model reconstruction at the end of the $i$-th iteration. The modeling of a speech signal is depicted in Fig. 12.18; note that the fourth row depicts the tenth iteration, and that the primary structure of the original signal is clearly captured by these early stages of the pursuit. As will be discussed in Sect. 12.3, compactly representing the salient features of the signal in such a way enables efficient coding and modification.



Amazon EX1010 Page 279

## 12.3 Speech Modification

Modification of speech signals is a wide-ranging subject covering applications from improving intelligibility for the sake of communication to changing ones persona for entertainment or virtual reality. At the signal level, speech modification involves such processes as noise removal, dereverberation, equalization, time scaling and pitch shifting. In this section, we discuss the use of the STFT and sinusoidal representations to carry out such signal-level modifications and provide relevant pointers to the literature for additional algorithmic details.

### 12.3.1 Comparing the STFT and Sinusoidal Models

Many signal modification schemes based on the STFT and/or sinusoidal modeling have been described in the literature. In early approaches based directly on the STFT subband signals $X[k, l]$, modifications were restricted to a relatively rigid framework because the signal is being modeled in terms of subbands that interact in complicated ways in the reconstruction process; this restrictive framework is namely that a modification is proposed, e.g., a time-varying spectral multiplication, and then the effect of the modification on the output signal is formulated [12.1, 11, 12, 14]. This approach is effective for understanding the operating principles of the STFT, but is somewhat removed from the desired end of carrying out an intended modification on the original signal.

In later approaches, modifications based only on the STFT magnitude were considered; the magnitude is first modified and then a phase that will minimize synthesis discontinuities is derived [12.20]. This removal of the phase essentially results in a pseudo-parametric representation that is more flexible than the complex subband signals. For some modifications, the phase of the original signal can be reused with the modified magnitude without introducing objectionable artifacts [12.66]; for others, such as time-scale modification, a consistent phase must be generated using the original phase and various continuity and validity constraints [12.19, 36].

The usefulness of the sinusoidal model is underscored by the difficulties of some STFT-based modifications, namely those of formalizing arbitrary modifications in a rigid synthesis framework and of generating an appropriate phase for magnitude-only modifications. Since it is fully parametric, the synthesis of a modified sinusoidal model is not constrained in the same way as an STFT; the model parameters are more malleable than STFT subband data in that there are no consistency or validity concerns or frame-boundary discontinuity issues.

In the following, we describe several classes of modification and highlight the differences between the STFT and sinusoidal approaches.

### 12.3.2 Linear Filtering

Linear filtering is a ubiquitous signal processing operation, and it is well known that the time-domain convolution operation of linear filtering corresponds to multiplication in the frequency domain. In the DFT domain, this principle can be stated as follows:

$$x[n] * f[n] \overset{\text{DFT}}{\Longleftrightarrow} X_K[k]F_K[k] \,, \tag{12.75}$$

where $X_K[k]$ and $F_K[k]$ are the $K$-point DFTs of the input sequence $x[n]$ and the filter impulse response $f[n]$, respectively; the DFT of the convolution is the product of the DFTs of the input signal and the filter impulse response. This property only holds exactly, however, if the DFT length satisfies

$$K \geq N_x + N_f - 1 \,, \tag{12.76}$$

where $N_x$ and $N_f$ are the signal length and the impulse response length, respectively. If $K$ is not large enough, the spectral multiplication introduces time-domain aliasing.

DFT-based filtering is of interest since the use of the FFT leads to implementations that require less computation than direct time-domain convolution. The signal and filter are transformed to the frequency domain via the FFT, multiplied, and then inverse-transformed back to the time domain. Given the DFT size requirement, however, this approach is not practical for arbitrarily long signals; to avoid time-domain aliasing, a correspondingly large FFT would be required. To enable the practical DFT-based filtering of arbitrarily long signals, the input signal is windowed into short segments, and the filtering operation is applied individually to each segment. In the *overlap-add* approach, the window length and DFT size are selected such that (12.76) holds for each segment. Then, each segment is individually filtered via spectral multiplication; each filtered spectrum is subsequently inverse-transformed and the time-domain outputs are combined via overlap-add with a rectangular synthesis window of length $K$. An alternative approach is the *overlap-save* method, where



Amazon EX1010 Page 280

the size constraint is less restrictive than (12.76); some time-domain aliasing is incurred, but corrupted output samples are discarded. More details on filter implementation using FFTs are available in the literature [12.67].

The overlap-add filtering approach fits readily into the modulated filter bank interpretation of the STFT shown in Fig. 12.2; a linear filter can be realized by multiplication of the subband signals $X[k, l]$ with the filter spectrum $F[k]$, and synthesis is carried out with a rectangular $v[n]$ of length $K$. Similarly, such overlap-add filtering can be employed in the parametric modeling scheme of Fig. 12.12. The approximate spectrum $\hat{X}[k, l]$ constructed from the sinusoidal parameters can be multiplied by $F[k]$ to achieve *exact* filtering of the output signal; this is not exact filtering of the input signal since the parametrically modeled spectrum is an approximation. An alternate approach is to carry out the filtering in the parametric domain by multiplying the sinusoidal model component amplitudes by the value of the filter spectrum at the corresponding frequencies; sinusoidal synthesis is then carried out based on these scaled model coefficients. This method yields *approximate* filtering of the output signal with significantly less computation than the former exact filtering [12.68].

Applications of linear filtering in speech processing include equalization of communication channels, noise suppression, and spatialization in binaural rendering, among many others such as speech enhancement, which will be discussed briefly in the following section.

### 12.3.3 Enhancement

In signal processing, the term *enhancement* typically refers to improving the signal-to-noise ratio or the intelligibility of a speech signal. A review of this large field is beyond the scope of this chapter, but it merits mentioning that many enhancement approaches involve adaptive linear filtering. Others are based on suppression of spectral components, either using noise estimates [12.69] or operating on the assumption that small spectral values correspond primarily to unwanted noise. This latter notion of signal denoising by thresholding the coefficients of the signal representation has been explored in detail in the literature [12.70, 71]; the idea is that large model coefficients capture the salient signal features while small model coefficients are essentially more representative of noise and should be discarded. Since a sinusoidal model captures the key features of typical speech signals, this notion suggests that sinusoidal modeling would prove effective for enhancement of degraded speech since the noise would not be captured in a compact sinusoidal representation. In an enhancement experiment, however, the sinusoidal model proved somewhat ineffective in this regard, so further investigation of this concept is needed [12.72].

### 12.3.4 Time-Scale Modification

In some scenarios, it is useful to change the duration of a speech or audio signal without altering its general character; interpolating the time-domain signal samples is insufficient since a spectral shift (e.g., a pitch shift) is incurred (if the same sampling rate is used). Techniques for high-quality time-scale modification are thus of interest; applications range from audio–video synchronization to slowing down a signal for music instruction or transcription or for improving the identifiability of the signal content. A wide range of algorithms have been proposed in the literature, including STFT magnitude modification followed by phase estimation [12.34, 35], sinusoidal modeling methods [12.41, 43, 65, 73, 74], and analyzing the signal for time-domain regions, e.g., pitch periods, which can be spliced out of the signal for time-scale compression or repeated for time-scale expansion [12.19, 75]; many other methods have also been considered [12.31], but a full survey is beyond the scope of this chapter.

In [12.19], the application of compact signal models to signal modification is discussed at length. Fundamentally, the idea is that signal modifications can be realized by modifying the components of a model of the signal. The sinusoidal model is particularly amenable to this approach because modifications of general interest are easy to carry out on sinusoids. For the case of time scaling, it is simple to increase or decrease the duration of a sinusoid – so if a signal is modeled as a sum of sinusoids, it becomes relatively simple to carry out time scaling on the entire signal. One caveat to note is that in some time-scaling scenarios it is important to preserve the rate of variation in the amplitude envelope of the signal, i. e., the signal dynamics, but this can be taken into account [12.74]. One sinusoidal approach described in the literature involves segmenting the signal into transient and sinusoidal regions; high-quality time scaling is achieved by translating the transient segments and then appropriately interpolating the sinusoidal parameters to shorten or lengthen the regions between transients [12.41, 65].

Sinusoidal modeling has been applied successfully to speech coding and postprocessing for packet-switched networks [12.76]. In the event of a lost packet, it is necessary for the decoder to construct the time-domain



Amazon EX1010 Page 281

audio output without the coded data contained in the missing packet, i. e., based only on the information in the adjacent received packets. In a sinusoidal modeling scheme, partials of the speech signal can be matched and interpolated across these adjacent frames to create a consistent time-domain replacement for the missing data. This can be thought of as time scaling the adjacent sinusoidal data to fill in the time-domain gap resulting from the missing packet.

### 12.3.5 Pitch Modification

Changing the pitch of a speech signal is another modification of interest. This is often thought of as the dual of time-scale modification; the goal is to shift the pitch of the signal without changing the duration, whereas in time-scaling the object is to change the duration without altering the pitch.

Signals with a well-defined pitch have a periodic or pseudoperiodic structure in the time domain; in the frequency domain, such signals have spectral peaks at or near integer multiples of a fundamental frequency. One approach to pitch modification, then, is to move the spectral peaks such that they are related by a new fundamental frequency and thus result in a new perceived pitch in the resynthesized signal. STFT-domain methods based on such peak moving have been discussed in the literature, and methods for establishing an appropriate phase for the shifted peaks have been considered [12.35, 36].

Given its parameters, a sinusoidal signal model of course enables modification of the signal's pitch. Most simply, pitch modification can be carried out by scaling the frequencies prior to synthesis. For voice applications, however, this results in unnatural reconstructed speech. Natural pitch transposition can be achieved by interpreting the sinusoidal parametrization as a source-filter model and carrying out formant-corrected pitch shifting as described below.

The sinusoidal model parametrization of a signal inherently includes a description of the signal's spectral envelope, which can be interpreted as a time-varying filter in a source-filter model in which the source is a sum of unweighted sinusoids. In voice applications, the filter corresponds to the vocal tract and the source represents the glottal excitation. Using this analogy, modifications can incorporate the physical underpinning that a pitch shift in speech is produced primarily by a change in the rate of glottal vibration and not by a change in the vocal tract shape or its resonances. To achieve natural pitch shifting of speech or the singing voice using the sinusoidal model, then, the spectral envelope must be preserved in the modification stage so as to preserve the formant structure of the vocal tract. Pitch modification is thus carried out by scaling the frequency parameters of the excitation sinusoids and then deriving new amplitudes for these pitch-scaled sinusoids by sampling the spectral envelope at the new frequencies, using interpolation of the envelope for frequencies that do not fall on the spectral bins. Such formant correction can be used in STFT-based peak-moving schemes to similarly improve the naturalness of the modification.

### 12.3.6 Cross-Synthesis and Other Modifications

A considerable array of signal manipulations beyond time-scale and pitch modification are enabled by STFT and sinusoidal representations [12.19,35]. Many of these are beyond the scope of this section, but some deserve brief mention for speech processing applications. For instance, the source-filter interpretation of the sinusoidal model is useful for a variety of spectral manipulations. In general, any sort of time-varying filtering can be carried out by modifying the amplitudes in the sinusoidal model representation. For instance, the formants in the spectral envelope can be adjusted to yield gender modifications; by moving the formants down in frequency, a female voice can be transformed into a male voice, and vice versa. Also, the amplitude ratios of odd and even harmonics in a pitched signal can be adjusted to achieve interesting signal changes.

So far the modifications we have discussed are operations carried out a single original signal; *cross-synthesis* refers to methods in which a new signal is created via the interactions of two or more original signals. An example of cross-synthesis is a source-filter model combination created by using the source from one signal's model and the filter from another signal's model, for instance exciting the vocal-tract filter estimated from a male voice by the glottal source estimated from a female voice. Such cross-synthesis is useful for voice persona modification.

The sinusoidal model enables compelling cross-synthesis modifications since the parameters directly describe perceptually important signal qualities such as the pitch as well as the shape and evolution of the spectral envelope. One immediate example of a modification is interpolation between the sinusoidal parameters of two sounds; this yields a hybrid signal perceived as a coherent merger of the two original sounds, and not simply a cross-fade or averaging. This type of modification



Amazon EX1010 Page 282

has received considerable attention for image *morphing*, which is carried out by parameterizing the salient features of an original image and a target image (such as edges or prominent regions) and creating a map between these parameterized features that can be traversed to construct a morphed intermediate image [12.77]. A related idea has been applied in the STFT domain for morphing between different audio signals [12.78].

### 12.3.7 Audio Coding with Decode-Side Modification

As described in Sect. 12.1.8, the STFT is ill-suited for audio coding since the subband data must be oversampled to achieve perfect reconstruction with an overlapping synthesis window. There is a data increase in the STFT-domain representation, which counteracts the key goal of signal compression. However, in the event that modification of the coded audio signal is required or desired at the decoder, it may be worthwhile to sacrifice the data rate for the added post-processing flexibility that an STFT representation provides over standard transform-coding representations such as the modified discrete cosine transform (MDCT). In [12.68], it is shown that using a signal representation amenable to modification can provide a significant reduction in the computational complexity of the decoder, especially for multichannel signals.

As for the sinusoidal model, using an efficient and flexible parametric representation for signal coding provides both a substantial data-rate improvement over the STFT as well as a further reduction in the decoder complexity since modifications can be carried out efficiently on the compact signal parameters. This benefit is one of the motivating factors for using sinusoidal models for low-rate coding of speech and audio signals. At very low data rates, where perceptually transparent coding of audio signals is not currently achievable, parametric coders using sinusoidal models can outperform subband/transform coders and have become part of modern audio compression standards [12.56, 79]. At higher rates, subband/transform coders are currently more effective than state-of-the-art parametric methods for near-transparent coding, although the malleability of a sinusoidal model provides an incentive for accepting a data-rate penalty [12.68, 80]; as signal modeling algorithms improve and parametric coding methods approach the maturity of transform coders, it can be expected that this data-rate penalty will become minimal or even negligible [12.41, 55, 57, 60, 64, 81]. As for pure speech coding, the situation is similar: low-rate sinusoidal coders have been proven effective, but not quite as effective as linear predictive coders [12.58, 82, 83]. However, the increased flexibility of the sinusoidal representation for desirable post-processing of the output signal is a strong incentive for adopting a sinusoidal coder in some scenarios.

## References


12.1 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice-Hall, Englewood Cliffs 1978)

12.2 D. Gabor: Theory of communication, J. IEE **93**(III-26), 429–457 (1946)

12.3 D. Gabor: Acoustical quanta and the theory of hearing, Nature **159**(4044), 591–594 (1947)

12.4 M. Vetterli, J. Kovačević: *Wavelets and Subband Coding* (Prentice-Hall, Englewood Cliffs 1995)

12.5 P.P. Vaidyanathan: *Multirate Systems and Filter Banks* (Prentice-Hall, Englewood Cliffs 1993)

12.6 T.F. Quatieri: *Discrete-Time Speech Signal Processing* (Prentice-Hall, Upper Saddle River 2002)

12.7 J. Cooley, J. Tukey: An algorithm for the machine calculation of complex Fourier series, Math. Comput. **19**(90), 297–301 (1965)

12.8 P. Duhamel, M. Vetterli: Fast Fourier transforms: a tutorial review and a state of the art, Signal Process. **4**(19), 259–299 (1990)

12.9 R. Schafer, L. Rabiner: Design and simulation of a speech analysis-synthesis system based on short-time Fourier analysis, IEEE Trans. Audio Electroacoust. **AU-21**(3), 165–174 (1973)

12.10 M.R. Portnoff: Implementation of the digital phase vocoder using the fast Fourier transform, IEEE Trans. Acoust. Speech **24**(3), 243–248 (1976)

12.11 J. Allen: Short term spectral analysis, synthesis, and modification by discrete Fourier transform, IEEE Trans. Acoust. Speech **25**(3), 235–238 (1977)

12.12 J. Allen, L. Rabiner: A unified approach to short-time Fourier analysis and synthesis, Proc. IEEE **65**(11), 1558–1564 (1977)

12.13 J.O. Smith: *Mathematics of the Discrete Fourier Transform (DFT)*, 2nd edn. (Booksurge, Seattle 2007), http://ccrma.stanford.edu/jos/mdft/

12.14 M.R. Portnoff: Time-frequency representation of digital signals and systems based on short-time




Amazon EX1010 Page 283

Fourier analysis, IEEE Trans. Acoust. Speech **28**(1), 55–69 (1980)
12.15 R. Crochiere: A weighted overlap-add method of short-time Fourier analysis/synthesis, IEEE Trans. Acoust. Speech **28**(1), 99–102 (1980)
12.16 F.J. Harris: On the use of windows for harmonic analysis with the discrete Fourier transform, Proc. IEEE **66**(1), 51–83 (1978)
12.17 A.H. Nuttall: Some windows with very good sidelobe behavior, IEEE Trans. Acoust. Speech **29**(1), 84–91 (1981)
12.18 X. Rodet, P. Depalle: Spectral envelopes and inverse FFT synthesis, Proc. 93rd Conv. of the Audio Eng. Soc. (1992), Preprint 3393
12.19 M. Goodwin: *Adaptive Signal Models: Theory, Algorithms, and Audio Applications* (Kluwer Academic, Boston 1998)
12.20 D. Griffin, J. Lim: Signal estimation from modified short-time Fourier transform, IEEE Trans. Acoust. Speech **32**(2), 236–243 (1984)
12.21 H.S. Malvar: *Signal Processing with Lapped Transforms* (Artech House, Boston 1992)
12.22 Z. Czetkovic: *Overcomplete Expansions for Digital Signal Processing* (Univ. California, Berkeley 1995), PhD Dissertation
12.23 Z. Czetkovic, M. Vetterli: Oversampled filter banks, IEEE Trans. Signal Process. **46**(5), 1245–1255 (1998)
12.24 H. Bölcskei, F. Hlawatsch: Oversampled filter banks: Optimal noise shaping, design freedom, and noise analysis, IEEE ICASSP, Vol. 3 (1997) pp. 2453–2456
12.25 F. Léonard: Referencing the phase to the centre of the spectral window. Why?, Mech. Syst. Signal Process. **2**(1), 75–90 (1997)
12.26 M. Bosi, R. Goldberg: *Introduction to Digital Audio Coding and Standards* (Kluwer Academic, Boston 2003)
12.27 J.P. Princen, A.B. Bradley: Analysis/synthesis filter bank design based on time domain aliasing cancellation, IEEE Trans. Acoust. Speech **34**(5), 1153–1161 (1986)
12.28 H.S. Malvar, D.H. Staelin: The LOT: Transform coding without blocking effects, IEEE Trans. Acoust. Speech **37**(4), 553–559 (1989)
12.29 H. Dudley: The vocoder, Bell Lab. Rec. **18**, 122–126 (1939)
12.30 J.L. Flanagan, R.M. Golden: Phase vocoder, Bell Syst. Tech. J. **45**(9), 1493–1509 (1966)
12.31 E. Moulines, J. Laroche: Non-parametric techniques for pitch-scale and time-scale modification of speech, Speech Commun. **16**(2), 175–205 (1995)
12.32 J.A. Moorer: The use of the phase vocoder in computer music applications, J. Audio Eng. Soc. **26**(1/2), 42–45 (1978)
12.33 M. Dolson: The phase vocoder: A tutorial, Comput. Music J. **10**(4), 14–27 (1986)
12.34 J. Laroche, M. Dolson: Improved phase vocoder time-scale modification of audio, IEEE Trans. Speech Audio Process. **7**(3), 323–332 (1999)
12.35 J. Laroche, M. Dolson: New phase-vocoder techniques for real-time pitch shifting, chorusing, harmonizing, and other exotic audio modifications, J. Audio Eng. Soc. **47**(11), 928–936 (1999)
12.36 J. Laroche, M. Dolson: About this phasiness business, Proc. IEEE Workshop on Applications of Signal Process. to Audio and Acoust. (1997)
12.37 R.J. McAulay, T.F. Quatieri: Speech analysis/synthesis based on a sinusoidal representation, IEEE Trans. Acoust. Speech **34**(4), 744–754 (1986)
12.38 X. Serra, J. Smith: Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition, Comput. Music J. **14**(4), 12–24 (1990)
12.39 E.B. George, M.J.T. Smith: Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones, J. Audio Eng. Soc. **40**(6), 497–516 (1992)
12.40 P. Depalle, G. Garcia, X. Rodet: Tracking of partials for additive sound synthesis using hidden Markov models, IEEE ICASSP, Vol. 1 (1993) pp. 225–228
12.41 S. Levine, J.O. Smith: A sines+transients+noise audio representation for data compression and time/pitch scale modifications, 105th Audio Eng. Soc. Conv. (1998), Preprint 4781.
12.42 M. Lagrange, S. Marchand, J.-B. Rault: Using linear prediction to enhance the tracking of partials, IEEE ICASSP, Vol. 4 (2004) pp. 241–244
12.43 E.B. George, M.J.T. Smith: Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model, IEEE Trans. Speech Audio Process. **5**(5), 389–406 (1997)
12.44 R.J. McAulay, T.F. Quatieri: Computationally efficient sine-wave synthesis and its application to sinusoidal transform coding, IEEE ICASSP, Vol. 1 (1988) pp. 370–373
12.45 S. Mallat, Z. Zhang: Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. **41**(12), 3397–3415 (1993)
12.46 M. Goodwin, M. Vetterli: Matching pursuit and atomic signal models based on recursive filter banks, IEEE Trans. Signal Process. **47**(7), 1890–1902 (1999)
12.47 G. Davis: *Adaptive Nonlinear Approximations* (New York University, New York 1994), PhD Dissertation
12.48 Y. Pati, R. Rezaiifar, P. Krishnaprasad: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, Conf. Record of the Twenty-Seventh Asilomar Conf. on Signals, Systems, and Comput., Vol. 1 (1993) pp. 40–44
12.49 S. Chen, J. Wigger: Fast orthogonal least squares algorithm for efficient subset model selection, IEEE Trans. Signal Process. **43**(7), 1713–1715 (1995)



Amazon EX1010 Page 284

12.50 B. Natarajan: Sparse approximate solutions to linear systems, SIAM J. Comput. **24**(2), 227–234 (1995)
12.51 S. Singhal, B. Atal: Amplitude optimization and pitch prediction in multipulse coders, IEEE Trans. Acoust. Speech **37**(3), 317–327 (1989)
12.52 J. Adler, B. Rao, K. Kreutz-Delgado: Comparison of basis selection methods, Conf. Record of the Thirtieth Asilomar Conf. on Signals, Systems, and Comput., Vol. 1 (1996) pp. 252–257
12.53 L. Rebollo-Neira, D. Lowe: Optimized orthogonal matching pursuit approach, IEEE Signal Proc. Let. **9**(4), 137–140 (2002)
12.54 G. Davis, S. Mallat, Z. Zhang: Adaptive time-frequency decompositions with matching pursuit, Opt. Eng. **33**(7), 2183–2191 (1994)
12.55 M. Goodwin: Multiscale overlap-add sinusoidal modeling using matching pursuit and refinements, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (2001) pp. 207–210
12.56 H. Purnhagen, N. Meine, B. Edler: Speeding up HILN – MPEG-4 parametric audio coding with reduced complexity, 109th Audio Eng. Soc. Conv. (2000), Preprint 5177
12.57 K. Vos, R. Vafin, R. Heusdens, W. Kleijn: High-quality consistent analysis-synthesis in sinusoidal coding, 17th Audio Eng. Soc. Int. Conf. (1999) pp. 244–250
12.58 C. Etemoglu, V. Cuperman: Matching pursuits sinusoidal speech coding, IEEE Trans. Speech Audio Process. **11**(5), 413–424 (2003)
12.59 T.S. Verma, T. Meng: Sinusoidal modeling using frame-based perceptually weighted matching pursuits, IEEE ICASSP (1998)
12.60 R. Heusdens, R. Vafin, W.B. Kleijn: Sinusoidal modeling using psychoacoustic-adaptive matching pursuits, IEEE Signal Proc. Lett. **9**(8), 262–265 (2002)
12.61 J. Laroche, Y. Stylianou, E. Moulines: HNM: A simple, efficient harmonic + noise model for speech, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (1993) pp. 169–172
12.62 M. Goodwin: Residual modeling in music analysis-synthesis, IEEE ICASSP, Vol. 2 (1996) pp. 1005–1008
12.63 K. Hamdy, M. Ali, A. Tewfik: Low bit rate high quality audio coding with combined harmonic and wavelet representations, IEEE ICASSP, Vol. 2 (1996) pp. 1045–1048
12.64 A. Oomen, A. Den Brinker: Sinusoids plus noise modeling for audio signals, 17th Audio Eng. Soc. Int. Conf. (1999) pp. 226–232
12.65 S. Levine, T. Verma, J. Smith: Alias-free, multiresolution sinusoidal modeling for polyphonic wideband audio, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (1997)
12.66 M. Goodwin, C. Avendano: Frequency-domain algorithms for audio signal enhancement based on transient modification, J. Audio Eng. Soc. **54**(9), 827–840 (2006)
12.67 A.V. Oppenheim, R.W. Schafer: *Discrete-Time Signal Processing* (Prentice-Hall, Englewood Cliffs 1989)
12.68 M. Goodwin, M. Wolters, R. Sridharan: Post-processing and computation in parametric and transform audio coders, AES 22nd Int. Conf.: Virtual, Synthetic, and Entertainment Audio (2002) pp. 149–158
12.69 S.F. Boll: Suppression of acoustic noise in speech using spectral subtraction, IEEE Trans. Acoust. Speech **27**(2), 113–120 (1979)
12.70 D.L. Donoho: De-noising by soft-thresholding, IEEE Trans. Inform. Theory **41**(3), 613–627 (1995)
12.71 S. Mallat, D. Donoho, A. Willsky: Best basis algorithm for signal enhancement, IEEE ICASSP, Vol. 3 (1995) pp. 1561–1564
12.72 J.M. Kates: Speech enhancement based on a sinusoidal model, J. Speech Hear. Res. **37**(2), 449–464 (1994)
12.73 T.F. Quatieri, R.J. McAulay: Speech transformations based on a sinusoidal representation, IEEE Trans. Acoust. Speech **34**(6), 1449–1464 (1986)
12.74 T.F. Quatieri, R.J. McAulay: Shape invariant time-scale and pitch modification of speech, IEEE Trans. Signal Process. **40**(3), 497–510 (1992)
12.75 D.L. Jones, T.W. Parks: Generation and combination of grains for music synthesis, Comput. Music J. **12**(2), 27–34 (1988)
12.76 C.A. Rodbro, M.G. Christensen, S.H. Jensen, S.V. Andersen: Compressed domain packet loss concealment of sinusoidally coded speech, IEEE ICASSP, Vol. 1 (2003) pp. 104–107
12.77 G. Wolberg: Recent advances in image morphing, Proc. of Comput. Graph. Int. (1996) pp. 64–71
12.78 M. Slaney, M. Covell, B. Lassiter: Automatic audio morphing, IEEE ICASSP, Vol. 2 (1996) pp. 1001–1004
12.79 H. Purnhagen, B. Edler, C. Ferekidis: Object-based analysis/synthesis audio coder for very low bit rates, 104th Audio Eng. Soc. Conv. (1998), Preprint 4747
12.80 M. Goodwin, C. Avendano: Parametric coding and frequency-domain processing for multichannel audio applications, AES 24th Int. Conf.: Multichannel Audio (2003) pp. 280–285
12.81 M.G. Christensen: Estimation and modeling problems in parametric audio coding. Ph.D. Thesis (Aalborg University, Aalborg 2005)
12.82 R.J. McAulay, T.F. Quatieri: Multirate sinusoidal transform coding at rates from 2.4 kbps to 8 kbps, IEEE ICASSP, Vol. 3 (1987) pp. 1645–1648
12.83 S. Ahmadi: New techniques for sinusoidal coding of speech at 2400 bps, Conf. Record of the Thirtieth Asilomar Conf. on Signals, Systems, and Comput., Vol. 1 (1996) pp. 770–774



Amazon EX1010 Page 285

# 13. Adaptive Blind Multichannel Identification

Y. Huang, J. Benesty, J. Chen

Blind multichannel identification was first introduced in the mid 1970s and initially studied in the communication society with the intention of designing more-efficient communication systems by avoiding a training phase. Recently this idea has become increasingly interesting for acoustics and speech processing research, driven by the fact that in most acoustic applications for speech processing and communication very little or nothing is known about the source signals. Since human ears have an extremely wide dynamic range and are much more sensitive to weak tails of the acoustic impulse responses, these impulse responses need to be modeled using fairly long filters. Attempting to identify such a multichannel system blindly with a batch method involves intensive computational complexity. This is not just bad system design, but technically rather implausible, particularly for real-time systems. Therefore, adaptive blind multichannel identification algorithms are favorable and pragmatically useful. This chapter describes some fundamental issues in blind multichannel identification and reviews a number of state-of-the-art adaptive algorithms.

13.1 **Overview** ..... 259
13.2 **Signal Model and Problem Formulation** ..... 260
13.3 **Identifiability and Principle** ..... 261
13.4 **Constrained Time-Domain Multichannel LMS and Newton Algorithms** ..... 262
13.4.1 Unit-Norm Constrained Multichannel LMS Algorithm ..... 262
13.4.2 Unit-Norm Constrained Multichannel Newton Algorithm .... 265
13.5 **Unconstrained Multichannel LMS Algorithm with Optimal Step-Size Control** 266
13.6 **Frequency-Domain Blind Multichannel Identification Algorithms** .. 268
13.6.1 Frequency-Domain Multichannel LMS Algorithm ..... 268
13.6.2 Frequency-Domain Normalized Multichannel LMS Algorithm ..... 273
13.7 **Adaptive Multichannel Exponentiated Gradient Algorithm** ..... 276
13.8 **Summary** ..... 279
**References** ..... 279

## 13.1 Overview

The need for identifying an acoustic multichannel system arises in a variety of speech processing and communication applications. While in some cases a reference source signal is known (acoustic echo cancelation is a typical example), for the majority of the applications (e.g., speech dereverberation, time delay estimation for source localization, source separation, speech enhancement, etc.) a priori knowledge of the source speech signal is either inaccessible or very expensive to acquire. This makes blind multichannel identification methods necessary and desirable.

The innovative idea of blind system identification was first proposed by *Sato* in [13.1]. Early studies of blind channel identification and equalization focused primarily on methods based on higher (than second)-order statistics (HOS) [13.2–4] (see [13.5] for a tutorial on this class of approaches). Because HOS cannot be accurately computed from a small number of observations, slow convergence is the critical drawback of all existing HOS methods. In addition, a cost function based on the HOS is barely convex and an HOS algorithm can be easily misled to a local minimum by corrupting noise in the observations. After it was recognized that the problem can be solved in the light of only second-order statistics (SOS) of system outputs [13.6], the focus of the blind multichannel identification research has shifted to SOS methods.



Amazon EX1010 Page 286

Using SOS to blindly identify a linear system requires that the number of outputs would be greater than the number of inputs. Therefore, only single-input multiple-output (SIMO) and multiple-input multiple-output (MIMO) acoustic systems are of interest in this problem. Blind MIMO identification, as of today, is still an open research problem and, in this handbook, only reviews of supposedly well-understood techniques are included. As a result, the focus will be on acoustic SIMO systems.

There is already a large literature on SOS-based blind SIMO identification algorithms with numerous variants. Celebrated work includes the cross-relation (CR) algorithm [13.7–10], the subspace algorithm [13.11], the linear prediction-based subspace algorithm [13.12], and the two-step maximum-likelihood algorithm [13.13], (see [13.14] for a more-comprehensive survey and a more-systematic classification of SOS-based blind SIMO identification algorithms). These *batch* methods can accurately estimate an identifiable acoustic SIMO system using a finite number of samples when additive noise in the system outputs is weak. However, they are in general computationally intensive and it was believed that they would be difficult to implement in an adaptive mode [13.14]. For blind multichannel identification to be practically useful in real-time speech applications, it is imperative to have some algorithms that are computationally simple and can be adaptively implemented. In this chapter, we will present a review of the state of the art of *adaptive* blind SIMO identification algorithms.

This Chapter is organized as follows. Section 13.2 describes the SIMO model used in this review and formulates the problem of blind multichannel identification. Section 13.3 presents the sufficient and necessary conditions for an acoustic SIMO to be blindly identifiable using only SOS and explains the principle of SOS-based blind SIMO identification techniques. In Sect. 13.4, the time-domain unit-norm constrained multichannel LMS (least mean square) (MCLMS) and Newton algorithms are developed. In Sect. 13.5, an optimal step size is deduced for the unconstrained MCLMS algorithm, which leads to the variable-step-size unconstrained MCLMS algorithm. Section 13.6 discusses the frequency-domain blind multichannel identification algorithms. Section 13.7 explores how to take advantage of spareness in acoustic impulse responses for faster convergence in blind multichannel identification methods. Finally, we provide a brief summary in Sect. 13.8.

## 13.2 Signal Model and Problem Formulation

This chapter considers an acoustic SIMO system in which there is one speech source and $N$ microphones, as illustrated in Fig. 13.1. The $n$-th microphone output is expressed as:

$$x_n(k) = h_n * s(k) + b_n(k), \quad n = 1, 2, \cdots, N, \quad (13.1)$$

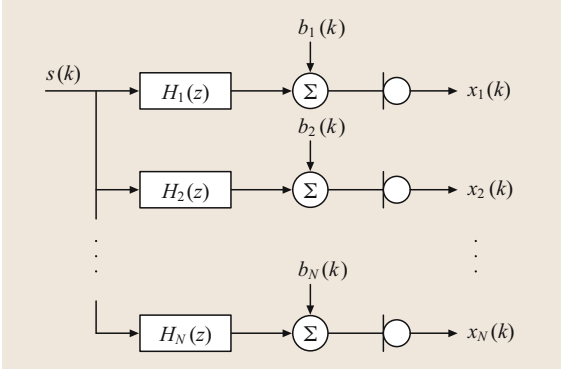



**Fig. 13.1** Illustration of an acoustic SIMO system

where $h_n$ is the channel impulse response from the source to the $n$-th microphone, the symbol $*$ denotes the linear convolution operator, $s(k)$ is the source signal at time $k$, and $b_n(k)$ is the additive noise at the $n$-th microphone output. The channel impulse responses are delineated with finite impulse response (FIR) filters. The additive noise components in different channels are assumed to be uncorrelated with the source signal even though they might be mutually dependent.

In vector/matrix form, the SIMO signal model (13.1) is written as:

$$\boldsymbol{x}_n(k) = \boldsymbol{H}_n \cdot \boldsymbol{s}(k) + \boldsymbol{b}_n(k), \quad n = 1, 2, \cdots, N, \quad (13.2)$$

where

$$\boldsymbol{x}_n(k) = [x_n(k) \;\; x_n(k-1) \;\; \cdots \;\; x_n(k-L+1)]^{\mathrm{T}},$$

$$\boldsymbol{H}_n = \begin{pmatrix} h_{n,0} & \cdots & h_{n,L-1} & 0 & \cdots & 0 \\ 0 & h_{n,0} & \cdots & h_{n,L-1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,0} & \cdots & h_{n,L-1} \end{pmatrix}_{L\times(2L-1)},$$



Amazon EX1010 Page 287

$$\boldsymbol{s}(k) = [s(k)\ \ s(k-1) \cdots s(k-L+1) \cdots s(k-2L+2)]^{\mathrm{T}},$$
$$\boldsymbol{b}_n(k) = [b_n(k)\ \ b_n(k-1) \cdots b_n(k-L+1)]^{\mathrm{T}},$$

$[\cdot]^{\mathrm{T}}$ denotes the transpose of a matrix or a vector, and $L$ is the length of the longest channel impulse response in such a SIMO system.

Therefore, mathematically speaking, the blind multichannel identification problem is to estimate the channel impulse responses

$$\boldsymbol{h}_n = [h_{n,0}\ \ h_{n,1} \cdots h_{n,L-1}]^{\mathrm{T}}, \quad n = 1, 2, \cdots, N,$$

from the observation $x_n(k)$ without knowledge of the source signal $\boldsymbol{s}(k)$.

## 13.3 Identifiability and Principle

An acoustic SIMO system is not always blindly identifiable using only the second-order statistics (SOS) of the microphone outputs. A multichannel FIR system can be blindly identified primarily because of the channel diversity. As an extreme counterexample, if all channels of a SIMO system are identical, the system reduces to a single-input single-output (SISO) system, which one can easily tell is blindly unidentifiable using only SOS. In addition, the source signal needs to have sufficient modes to make the channels fully excited. According to [13.10], two inductive conditions (one on the channel diversity and the other on the input signals) are necessary and sufficient to ensure blind SIMO identifiability, and are shared by all SOS-based blind SIMO identification methods:

1. The polynomials formed from $\boldsymbol{h}_n$ $(n = 1, 2, \cdots, N)$ are co-prime, i. e., the corresponding channel transfer functions $H_n(z) = \sum_{l=0}^{L-1} h_{n,l} z^{-l}$ do not share any common zeros;
2. The autocorrelation matrix of the input signal $\boldsymbol{R}_{ss} = E\{\boldsymbol{s}(k)\boldsymbol{s}^{\mathrm{T}}(k)\}$, where $E\{\cdot\}$ denotes mathematical expectation, is of full rank (such that the SIMO system can be fully excited).

In the rest of this chapter, these two conditions are assumed to hold so that we will be dealing with a blindly identifiable FIR SIMO system.

The easiest way to show how a SIMO system can be blindly identified probably is to use the cross-relation method. From (13.1), we derive the fact that, in the absence of noise,

$$x_i * h_j = s * h_i * h_j = x_j * h_i, \quad i, j = 1, 2, \cdots, N, \ \ i \neq j\,. \tag{13.3}$$

At time $k$, we then have

$$\boldsymbol{x}_i^{\mathrm{T}}(k)\boldsymbol{h}_j = \boldsymbol{x}_j^{\mathrm{T}}(k)\boldsymbol{h}_i\,, \quad i, j = 1, 2, \cdots, N, \ \ i \neq j\,. \tag{13.4}$$

Multiplying (13.4) by $\boldsymbol{x}_i(k)$ from the left side and taking expectation yields,

$$\boldsymbol{R}_{x_i x_i}\boldsymbol{h}_j = \boldsymbol{R}_{x_i x_j}\boldsymbol{h}_i\,, \quad i, j = 1, 2, \cdots, N, \ \ i \neq j\,, \tag{13.5}$$

where

$$\boldsymbol{R}_{x_i x_j} = E\{\boldsymbol{x}_i(k)\boldsymbol{x}_j^{\mathrm{T}}(k)\}\,.$$

Formula (13.5) consists of $N(N-1)$ distinct equations. By summing up the $N-1$ cross-relations associated with one particular channel $h_j$, we obtain

$$\sum_{i=1, i\neq j}^{N} \boldsymbol{R}_{x_i x_i}\boldsymbol{h}_j = \sum_{i=1, i\neq j}^{N} \boldsymbol{R}_{x_i x_j}\boldsymbol{h}_i\,, \quad j = 1, 2, \cdots, N\,. \tag{13.6}$$

Over all channels, we then have a total of $N$ equations. In matrix form, this set of equations is written:

$$\boldsymbol{R}_{x+}\boldsymbol{h} = \boldsymbol{0}\,, \tag{13.7}$$

where

$$\boldsymbol{R}_{x+} = \begin{pmatrix} \sum_{n\neq 1}\boldsymbol{R}_{x_n x_n} & -\boldsymbol{R}_{x_2 x_1} & \cdots & -\boldsymbol{R}_{x_N x_1} \\ -\boldsymbol{R}_{x_1 x_2} & \sum_{n\neq 2}\boldsymbol{R}_{x_n x_n} & \cdots & -\boldsymbol{R}_{x_N x_2} \\ \vdots & \vdots & \ddots & \vdots \\ -\boldsymbol{R}_{x_1 x_N} & -\boldsymbol{R}_{x_2 x_N} & \cdots & \sum_{n\neq N}\boldsymbol{R}_{x_n x_n} \end{pmatrix},$$

$$\boldsymbol{h} = \left[\boldsymbol{h}_1^{\mathrm{T}}\ \ \boldsymbol{h}_2^{\mathrm{T}} \cdots \boldsymbol{h}_N^{\mathrm{T}}\right]^{\mathrm{T}}.$$

If the SIMO system is blindly identifiable, the matrix $\boldsymbol{R}_{x+}$ is rank deficient by 1 (in the absence of noise) and the channel impulse responses $\boldsymbol{h}$ can be uniquely determined up to a scale.

When additive noise is present, the right-hand side of (13.7) is no longer zero and an error vector is produced:

$$\boldsymbol{e} = \boldsymbol{R}_{x+}\boldsymbol{h}\,. \tag{13.8}$$

This error can be used to define a cost function

$$J = \|\boldsymbol{e}\|^2 = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}\,. \tag{13.9}$$



Amazon EX1010 Page 288

We can then determine a vector $\hat{\boldsymbol{h}}$ as the solution by minimizing this cost function in the least-squares sense:

$$\hat{\boldsymbol{h}} = \arg\min_{\boldsymbol{h}} J = \arg\min_{\boldsymbol{h}} \boldsymbol{h}^{\mathrm{T}} \boldsymbol{R}_{x+}^{\mathrm{T}} \boldsymbol{R}_{x+} \boldsymbol{h} \,. \tag{13.10}$$

In this case, $\boldsymbol{R}_{x+}$ is positive definite rather than positive semidefinite and the desired estimate $\hat{\boldsymbol{h}}$ would be the eigenvector of $\boldsymbol{R}_{x+}$ corresponding to its smallest eigenvalue (here we assume that the noise is white, incoherent or uncorrelated, and weaker than the source signal).

Note that the estimated channel impulse response vector is aligned to the true one, but up to a nonzero scale. This inherent scale ambiguity is usually harmless in most, if not all, of acoustic signal processing applications. But in the development of an adaptive algorithm, attention needs to be paid to prevent convergence to a trivial all-zero estimate. This will become clearer after the reader finishes reading this chapter.

## 13.4 Constrained Time-Domain Multichannel LMS and Newton Algorithms

In this section, we intend to develop two time-domain *adaptive* algorithms for blind SIMO identification, namely the multichannel LMS (MCLMS) and multichannel Newton (MCN) algorithms [13.15].

### 13.4.1 Unit-Norm Constrained Multichannel LMS Algorithm

To begin, we use the cross-relations between the $i$-th and $j$-th outputs given by (13.4). When noise is present and/or the estimate of channel impulse responses deviates from the true value, an a priori error signal is produced:

$$e_{ij}(k+1) = \boldsymbol{x}_i^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k) - \boldsymbol{x}_j^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_i(k) \,, \quad i, j = 1, 2, \cdots, N \,, \tag{13.11}$$

where $\hat{\boldsymbol{h}}_i(k)$ is the model filter for the $i$-th channel at time $k$. Assuming that these error signals are equally important, we now define a cost function as follows:

$$\chi(k+1) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} e_{ij}^2(k+1) \,, \tag{13.12}$$

where we exclude the cases of $e_{ii}(k) = 0$ $(i = 1, 2, \cdots, N)$ and count the $e_{ij}(k) = -e_{ij}(k)$ pair only once.

In order to avoid a trivial estimate with all zero elements, a constraint should be imposed on the model filters $\hat{\boldsymbol{h}}(k)$. Two constraints were proposed. One is the easily understood unit-norm constraint, i. e., $\|\hat{\boldsymbol{h}}(k)\| = 1$. The other is the component-normalization constraint, i. e., $\boldsymbol{c}^{\mathrm{T}}\hat{\boldsymbol{h}}(k) = 1$, where $\boldsymbol{c}$ is a constant vector. As an example, if we know that one coefficient, say $h_{n,l}$ $(n = 1, 2, \cdots, N,\ l = 0, 1, \cdots, L-1)$, is equal to $\alpha$, which is not zero, then we may properly specify $\boldsymbol{c} = [0, \cdots, 1/\alpha, \cdots, 0]^{\mathrm{T}}$ with $1/\alpha$ being the $(nL+l)$-th element of $\boldsymbol{c}$. Even though the component-normalization constraint can be more robust to noise than the unit-norm constraint [13.16], knowledge of the location of the constrained component $h_{n,l}$ and its value $\alpha$ may not be available in practice. So the unit-norm constraint was more widely used and will be employed in this chapter.

With the unit-norm constraint enforced on $\hat{\boldsymbol{h}}(k)$, the normalized error signal is obtained:

$$\epsilon_{ij}(k+1) = e_{ij}(k+1)/\|\hat{\boldsymbol{h}}(k)\| \,. \tag{13.13}$$

Accordingly, the cost function is formulated as:

$$J(k+1) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \epsilon_{ij}^2(k+1) = \frac{\chi(k+1)}{\|\hat{\boldsymbol{h}}(k)\|^2} \,. \tag{13.14}$$

The update equation of the unit-norm-constrained MCLMS algorithm is then given by

$$\hat{\boldsymbol{h}}(k+1) = \hat{\boldsymbol{h}}(k) - \mu \nabla J(k+1) \,, \tag{13.15}$$

where $\mu$ is a small positive step size and $\nabla J(k+1)$ denotes the gradient of $J(k+1)$ with respect to $\hat{\boldsymbol{h}}(k)$. The gradient is determined as follows:

$$\begin{aligned} \nabla J(k+1) &= \frac{\partial J(k+1)}{\partial \hat{\boldsymbol{h}}(k)} = \frac{\partial}{\partial \hat{\boldsymbol{h}}(k)} \left( \frac{\chi(k+1)}{\|\hat{\boldsymbol{h}}(k)\|^2} \right) \\ &= \frac{\partial}{\partial \hat{\boldsymbol{h}}(k)} \left( \frac{\chi(k+1)}{\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\hat{\boldsymbol{h}}(k)} \right) \\ &= \frac{1}{\|\hat{\boldsymbol{h}}(k)\|^2} \\ &\quad \times \left( \frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}(k)} - 2J(k+1)\hat{\boldsymbol{h}}(k) \right) \,, \end{aligned} \tag{13.16}$$

where

$$\frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}(k)} = \left[ \left( \frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_1(k)} \right)^{\mathrm{T}} \left( \frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_2(k)} \right)^{\mathrm{T}} \cdots \left( \frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_N(k)} \right)^{\mathrm{T}} \right]^{\mathrm{T}} \,.$$



Amazon EX1010 Page 289

Let us now evaluate the partial derivative of $\chi(k+1)$ with respect to the coefficients of the $n$-th ($n = 1, 2, \cdots, N$) channel impulse response:

$$\begin{aligned}\frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_n(k)} &= \frac{\partial\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} e_{ij}^2(k+1)\right]}{\partial \hat{\boldsymbol{h}}_n(k)} \\ &= \sum_{i=1}^{n-1} 2e_{in}(k+1)\boldsymbol{x}_i(k+1) \\ &\quad + \sum_{j=n+1}^{N} 2e_{nj}(k+1)[-\boldsymbol{x}_j(k+1)] \\ &= \sum_{i=1}^{n-1} 2e_{in}(k+1)\boldsymbol{x}_i(k+1) \\ &\quad + \sum_{j=n+1}^{N} 2e_{jn}(k+1)\boldsymbol{x}_j(k+1) \\ &= \sum_{i=1}^{N} 2e_{in}(k+1)\boldsymbol{x}_i(k+1)\,, \end{aligned} \tag{13.17}$$

where the last step follows from the fact that $e_{nn}(k+1) = 0$. We may express this equation concisely in matrix form as

$$\begin{aligned}\frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_n(k)} &= 2\boldsymbol{X}(k+1)\boldsymbol{e}_n(k+1) \\ &= 2\boldsymbol{X}(k+1)[\boldsymbol{C}_n(k+1) \\ &\quad - \boldsymbol{D}_n(k+1)]\hat{\boldsymbol{h}}(k)\,, \end{aligned} \tag{13.18}$$

where we have defined, for convenience,

$$\begin{aligned}\boldsymbol{X}(k+1) &= (\boldsymbol{x}_1(k+1)\ \boldsymbol{x}_2(k+1) \\ &\quad \cdots\ \boldsymbol{x}_N(k+1))_{L\times N}\,, \\ \boldsymbol{e}_n(k+1) &= [e_{1n}(k+1)\ e_{2n}(k+1) \\ &\quad \cdots e_{Nn}(k+1)]^{\mathrm{T}} \\ &= \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_n(k) - \boldsymbol{x}_n^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_1(k) \\ \boldsymbol{x}_2^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_n(k) - \boldsymbol{x}_n^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_2(k) \\ \vdots \\ \boldsymbol{x}_N^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_n(k) - \boldsymbol{x}_n^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_N(k) \end{pmatrix} \\ &= [\boldsymbol{C}_n(k+1) - \boldsymbol{D}_n(k+1)]\hat{\boldsymbol{h}}(k)\,, \end{aligned}$$

and

$$\begin{aligned}\boldsymbol{C}_n(k+1) &= \begin{pmatrix} \boldsymbol{0}\ \cdots\ \boldsymbol{0}\ \boldsymbol{x}_1^{\mathrm{T}}(k+1)\ \boldsymbol{0}\ \cdots\ \boldsymbol{0} \\ \boldsymbol{0}\ \cdots\ \boldsymbol{0}\ \boldsymbol{x}_2^{\mathrm{T}}(k+1)\ \boldsymbol{0}\ \cdots\ \boldsymbol{0} \\ \vdots\ \cdots\ \vdots\ \ \vdots\ \ \vdots\ \cdots\ \vdots \\ \underbrace{\boldsymbol{0}\ \cdots\ \boldsymbol{0}}_{(n-1)L}\ \boldsymbol{x}_N^{\mathrm{T}}(k+1)\ \underbrace{\boldsymbol{0}\ \cdots\ \boldsymbol{0}}_{(N-n)L} \end{pmatrix}_{N\times NL} \\ &= \left(\boldsymbol{0}_{N\times(n-1)L}\ \ \boldsymbol{X}^{\mathrm{T}}(k+1)\ \ \boldsymbol{0}_{N\times(N-n)L}\right), \\ \boldsymbol{D}_n(k+1) &= \begin{pmatrix} \boldsymbol{x}_n^{\mathrm{T}}(k+1) & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{x}_n^{\mathrm{T}}(k+1) & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{x}_n^{\mathrm{T}}(k+1) \end{pmatrix}_{N\times NL}. \end{aligned}$$

Continuing, we evaluate the two matrix products in (13.18) individually as follows:

$$\begin{aligned}&\boldsymbol{X}(k+1)\boldsymbol{C}_n(k+1) \\ &\quad = \boldsymbol{X}(k+1) \\ &\qquad \cdot \left(\boldsymbol{0}_{N\times(n-1)L}\ \ \boldsymbol{X}^{\mathrm{T}}(k+1)\ \ \boldsymbol{0}_{N\times(N-n)L}\right)_{N\times NL} \\ &\quad = \Bigg(\boldsymbol{0}_{L\times(n-1)L}\ \ \sum_{i=1}^{N}\tilde{\boldsymbol{R}}_{x_i x_i}(k+1) \\ &\qquad \boldsymbol{0}_{L\times(N-n)L}\Bigg)_{L\times NL}\,, \end{aligned} \tag{13.19}$$

$$\begin{aligned}&\boldsymbol{X}(k+1)\boldsymbol{D}_n(k+1) \\ &\quad = (\tilde{\boldsymbol{R}}_{x_1 x_n}(k+1)\ \ \tilde{\boldsymbol{R}}_{x_2 x_n}(k+1) \\ &\qquad \cdots\ \tilde{\boldsymbol{R}}_{x_N x_n}(k+1))_{L\times NL}\,, \end{aligned} \tag{13.20}$$

where

$$\begin{aligned}\tilde{\boldsymbol{R}}_{x_i x_j}(k+1) &= \boldsymbol{x}_i(k+1)\boldsymbol{x}_j^{\mathrm{T}}(k+1)\,, \\ & i, j = 1, 2, \cdots, N. \end{aligned}$$

Here we put a tilde on $\tilde{\boldsymbol{R}}_{x_i x_j}$ to distinguish this instantaneous value from its mathematical expectation $\boldsymbol{R}_{x_i x_j}$.

Next, substituting (13.19) and (13.20) into (13.18) yields

$$\begin{aligned}\frac{\partial \chi(k+1)}{\partial \hat{\boldsymbol{h}}_n(k)} &= 2\Big(-\tilde{\boldsymbol{R}}_{x_1 x_n}(k+1) \\ &\qquad \cdots \textstyle\sum_{i\neq n}\tilde{\boldsymbol{R}}_{x_i x_i}(k+1) \\ &\qquad \cdots -\tilde{\boldsymbol{R}}_{x_N x_n}(k+1)\Big)\hat{\boldsymbol{h}}(k)\,. \end{aligned} \tag{13.21}$$



Amazon EX1010 Page 290

Therefore, we incorporate (13.21) into (13.16) and obtain

$$\frac{\partial \chi(k+1)}{\partial \hat{h}(k)} = 2\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\,, \tag{13.22}$$

$$\nabla J(k+1) = \frac{1}{\|\hat{\boldsymbol{h}}(k)\|^2} \cdot [2\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - 2J(k+1)\hat{\boldsymbol{h}}(k)]\,, \tag{13.23}$$

where

$$\tilde{\boldsymbol{R}}_{x+}(k) = \begin{pmatrix} \sum_{n\neq 1}\tilde{\boldsymbol{R}}_{x_n x_n}(k) & -\tilde{\boldsymbol{R}}_{x_2 x_1}(k) & \cdots & -\tilde{\boldsymbol{R}}_{x_N x_1}(k) \\ -\tilde{\boldsymbol{R}}_{x_1 x_2}(k) & \sum_{n\neq 2}\tilde{\boldsymbol{R}}_{x_n x_n}(k) & \cdots & -\tilde{\boldsymbol{R}}_{x_N x_2}(k) \\ \vdots & \vdots & \ddots & \vdots \\ -\tilde{\boldsymbol{R}}_{x_1 x_N}(k) & -\tilde{\boldsymbol{R}}_{x_2 x_N}(k) & \cdots & \sum_{n\neq N}\tilde{\boldsymbol{R}}_{x_n x_n}(k) \end{pmatrix}.$$

Finally, we substitute (13.23) into (13.15) and deduce the update equation of the unit-norm-constrained MCLMS algorithm:

$$\hat{\boldsymbol{h}}(k+1) = \hat{\boldsymbol{h}}(k) - \frac{2\mu}{\|\hat{\boldsymbol{h}}(k)\|^2} [\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - J(k+1)\hat{\boldsymbol{h}}(k)]\,. \tag{13.24}$$

If the channel estimate is always normalized after each update, then we have the simplified, unit-norm-constrained MCLMS algorithm [13.15]:

$$\hat{\boldsymbol{h}}(k+1) = \{\hat{\boldsymbol{h}}(k) - 2\mu[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]\} \Big/ \{\|\hat{\boldsymbol{h}}(k) - 2\mu[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]\|\}\,. \tag{13.25}$$

The time-domain, unit-norm-constrained MCLMS adaptive algorithm is summarized in Table 13.1.

Now we would like to comment briefly on the convergence of the unit-norm-constrained MCLMS algorithm. Assuming that the independence assumption [13.17] holds, it can be easily shown that the unit-norm-constrained MCLMS algorithm converges in the mean if the step size satisfies

$$0 < \mu < \frac{1}{\lambda_{\max}}\,, \tag{13.26}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $E\{\tilde{\boldsymbol{R}}_{x+}(k) - J(k)\boldsymbol{I}_{NL\times NL}\}$ and $\boldsymbol{I}_{NL\times NL}$ is the identity matrix of size $NL$ by $NL$. After convergence, taking the expectation of (13.24) produces

$$\boldsymbol{R}_{x+}\frac{\hat{\boldsymbol{h}}(\infty)}{\|\hat{\boldsymbol{h}}(\infty)\|} = E\{J(\infty)\}\frac{\hat{\boldsymbol{h}}(\infty)}{\|\hat{\boldsymbol{h}}(\infty)\|}\,, \tag{13.27}$$

which is the desired result: $\hat{\boldsymbol{h}}$ converges in the mean to the eigenvector of $\boldsymbol{R}_{x+}$ corresponding to the smallest eigenvalue $E\{J(\infty)\}$.

**Table 13.1** The unit-norm-constrained multichannel LMS adaptive algorithm for the blind identification of an SIMO FIR system

| | |
|---|---|
| Parameters: | $\hat{\boldsymbol{h}} = [\hat{\boldsymbol{h}}_1^{\mathrm{T}}\ \hat{\boldsymbol{h}}_2^{\mathrm{T}}\ \cdots\ \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}$, model filter |
| | $\mu > 0$, step size |
| Initialization: | $\hat{\boldsymbol{h}}_n(0) = [1\ \ 0\ \ \cdots\ \ 0]^{\mathrm{T}}$, $n = 1, 2, \cdots, N$ |
| | $\hat{\boldsymbol{h}}(0) = \hat{\boldsymbol{h}}(0)/\sqrt{N}$ (normalization) |
| Computation: | For $k = 0, 1, 2, \cdots$, compute |
| (a) | $e_{ij}(k+1) = \boldsymbol{x}_i^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k) - \boldsymbol{x}_j^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_i(k)$, $i, j = 1, 2, \cdots, N$ |
| (b) | $\chi(k+1) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} e_{ij}^2(k+1)$ |
| (c) | Construct the matrix $\tilde{\mathbf{R}}_{x+}(k+1)$ following Sect. 13.4.2 |
| (d) | $\hat{\boldsymbol{h}}(k+1) = \dfrac{\hat{\boldsymbol{h}}(k) - 2\mu[\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]}{\left\|\hat{\boldsymbol{h}}(k) - 2\mu[\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]\right\|}$ |



Amazon EX1010 Page 291

### 13.4.2 Unit-Norm Constrained Multichannel Newton Algorithm

It has been proved that the unit-norm-constrained MCLMS algorithm developed above can converge in the mean to the desired channel impulse responses. However, one of the difficulties in the design and implementation of the adaptive multichannel LMS filters is the selection of the step size $\mu$. To select the step size $\mu$ in an LMS algorithm, there is a trade-off, as pointed out in many studies, between the rate of convergence, the amount of excess mean-square error, and the ability of the algorithm to track the system as its impulse responses change. In order to achieve a good balance of the three competing design objectives, we present here the unit-norm-constrained multichannel Newton algorithm (see [13.18] for the Newton method) with a variable step size during adaptation:

$$\hat{\boldsymbol{h}}(k+1) = \hat{\boldsymbol{h}}(k) - E^{-1}\{\nabla^2 J(k+1)\}\nabla J(k+1)\,, \tag{13.28}$$

where $\nabla^2 J(k+1)$ is the Hessian matrix of $J(k+1)$ with respect to $\hat{\boldsymbol{h}}(k)$. Taking derivative of (13.23) with respect to $\hat{\boldsymbol{h}}(k)$ and using the formula

$$\begin{aligned}
&\frac{\partial}{\partial \hat{\boldsymbol{h}}(k)}[J(k+1)\hat{\boldsymbol{h}}(k)] \\
&= \hat{\boldsymbol{h}}(k)\left[\frac{\partial J(k+1)}{\partial \hat{\boldsymbol{h}}(k)}\right]^{\mathrm{T}} + J(k+1)\boldsymbol{I}_{NL\times NL} \\
&= \hat{\boldsymbol{h}}(k)[\nabla J(k+1)]^{\mathrm{T}} + J(k+1)\boldsymbol{I}_{NL\times NL}\,,
\end{aligned} \tag{13.29}$$

we obtain

$$\begin{aligned}
&\nabla^2 J(k+1) \\
&= \frac{2\tilde{\boldsymbol{R}}_{x+}(k+1)}{\|\hat{\boldsymbol{h}}(k)\|^2} \\
&\quad - \frac{2\left[\hat{\boldsymbol{h}}(k)\nabla J(k+1))^{\mathrm{T}} + J(k+1)\boldsymbol{I}_{NL\times NL}\right]}{\|\hat{\boldsymbol{h}}(k)\|^2} \\
&\quad - \frac{4[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - J(k+1)\hat{\boldsymbol{h}}(k)]\hat{\boldsymbol{h}}^{\mathrm{T}}(k)}{\|\hat{\boldsymbol{h}}(k)\|^4}\,.
\end{aligned} \tag{13.30}$$

**Table 13.2** The unit-norm-constrained multichannel Newton adaptive algorithm for the blind identification of an SIMO FIR system

Parameters:

$\hat{\boldsymbol{h}} = [\hat{\boldsymbol{h}}_1^{\mathrm{T}}\ \hat{\boldsymbol{h}}_2^{\mathrm{T}}\ \cdots\ \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}$, model filter

$0 < \rho < 1$, step size

$0 < \lambda < 1$, forgetting factor

Initialization:

$\hat{\boldsymbol{h}}_n(0) = [1\ \ 0\ \ \cdots\ \ 0]^{\mathrm{T}}$, $n = 1, 2, \cdots, N$

$\hat{\boldsymbol{h}}(0) = \hat{\boldsymbol{h}}(0)/\sqrt{N}$ (normalization)

Computation:

For $k = 0, 1, 2, \cdots$, compute

(a) $e_{ij}(k+1) = \boldsymbol{x}_i^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k) - \boldsymbol{x}_j^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_i(k)\,, \quad i, j = 1, 2, \cdots, N$

(b) $\chi(k+1) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} e_{ij}^2(k+1)$

(c) Construct the matrix $\tilde{\mathbf{R}}_{x+}(k+1)$ following Sect. 13.4.2

(d) $\hat{\mathbf{R}}_{x+}(k+1) = \begin{cases} \mathrm{diag}\{\sigma_{x_1}^2, \cdots, \sigma_{x_1}^2, \cdots, \sigma_{x_N}^2, \cdots, \sigma_{x_N}^2\}\,, & k = 0 \\ \lambda\hat{\mathbf{R}}_{x+}(k) + \tilde{\mathbf{R}}_{x+}(k+1)\,, & k \geq 1 \end{cases}$

(e) $\mathbf{V}(k+1) = 2\hat{\mathbf{R}}_{x+}(k+1) - 4\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}(k)^{\mathrm{T}}\hat{\mathbf{R}}_{x+}(k+1) - 4\hat{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}(k)^{\mathrm{T}}$

(f) $\hat{\boldsymbol{h}}(k+1) = \dfrac{\hat{\boldsymbol{h}}(k) - 2\rho\mathbf{V}^{-1}(k+1)[\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]}{\left\|\hat{\boldsymbol{h}}(k) - 2\rho\mathbf{V}^{-1}(k+1)[\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) - \chi(k+1)\hat{\boldsymbol{h}}(k)]\right\|}$



Amazon EX1010 Page 292

With the unit-norm constraint $\|\hat{\boldsymbol{h}}(k)\| = 1$, (13.30) can be simplified as follows:

$$\begin{aligned}\nabla^2 J(k+1) = 2\{&\tilde{\boldsymbol{R}}_{x+}(k+1) - \hat{\boldsymbol{h}}(k)[\nabla J(k+1)]^{\mathrm{T}} \\ &- J(k+1)\boldsymbol{I}_{NL\times NL}\} \\ &- 4[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) \\ &- J(k+1)\hat{\boldsymbol{h}}(k)]\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\,. \end{aligned} \tag{13.31}$$

Taking the mathematical expectation of (13.31) and invoking the independence assumption [13.17] produces

$$\begin{aligned}E\{\nabla^2 J(k+1)\} = &2\boldsymbol{R}_{x+} - 4\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\boldsymbol{R}_{x+} \\ &- 4\boldsymbol{R}_{x+}\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}^{\mathrm{T}}(k) \\ &- 2E\{J(k+1)\} \\ &\cdot[\boldsymbol{I}_{NL\times NL} - 4\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}^{\mathrm{T}}(k)]\,. \end{aligned} \tag{13.32}$$

In practice, $\boldsymbol{R}_{x+}$ and $E\{J(k+1)\}$ are not known such that we have to estimate their values. Since $J(k+1)$ decreases as adaptation proceeds and is relatively small, particularly after convergence, we can neglect the term $E\{J(k+1)\}$ in (13.32) for simplicity and with appropriate accuracy, as suggested by simulations. The matrix $\boldsymbol{R}_{x+}$ is estimated recursively in a conventional way as follows:

$$\begin{aligned}\hat{\boldsymbol{R}}_{x+}(1) &= \mathrm{diag}\{\sigma_{x_1}^2, \cdots, \sigma_{x_1}^2, \sigma_{x_2}^2, \cdots, \sigma_{x_2}^2, \\ &\quad \cdots, \sigma_{x_N}^2, \cdots, \sigma_{x_N}^2\}\,, \\ \hat{\boldsymbol{R}}_{x+}(k+1) &= \lambda\hat{\boldsymbol{R}}_{x+}(k) + \tilde{\boldsymbol{R}}_{x+}(k+1)\,, \\ &\quad \text{for } k \geq 1\,, \end{aligned} \tag{13.33}$$

where $\sigma_{x_n}^2$ $(n = 1, 2, \cdots, N)$ is the power of $x_n(k)$ and $\lambda$ $(0 < \lambda < 1)$ is an exponential forgetting factor.

By using these approximations, we finally obtain an estimate $\boldsymbol{V}(k+1)$ for the mean Hessian matrix of $J(k+1)$

$$\begin{aligned}\boldsymbol{V}(k+1) = &2\hat{\boldsymbol{R}}_{x+}(k+1) - 4\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\hat{\boldsymbol{R}}_{x+}(k+1) \\ &- 4\hat{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\,, \end{aligned} \tag{13.34}$$

and hence deduce the unit-norm-constrained multichannel Newton algorithm [13.15]:

$$\begin{aligned}\hat{\boldsymbol{h}}(k+1) = &\{\hat{\boldsymbol{h}}(k) - 2\rho\boldsymbol{V}^{-1}(k+1)[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) \\ &- \chi(k+1)\hat{\boldsymbol{h}}(k)]\} \\ &\Big/\{\|\hat{\boldsymbol{h}}(k) - 2\rho\boldsymbol{V}^{-1}(k+1)[\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) \\ &- \chi(k+1)\hat{\boldsymbol{h}}(k)]\|\}\,, \end{aligned} \tag{13.35}$$

where $\rho$ is a new step size, close to but less than 1. The unit-norm-constrained MCN algorithm for blind SIMO identification is summarized in Table 13.2.

## 13.5 Unconstrained Multichannel LMS Algorithm with Optimal Step-Size Control

As explained in the previous sections, the constrained MCLMS algorithm converges slowly while the constrained MCN algorithm is computationally intensive. To improve performance, we can either find a way to accelerate the constrained MCLMS algorithm or reduce the complexity of the constrained MCN algorithm. We will develop algorithms along the former direction in this section and continue exploring this problem along the latter direction in the next section.

To accelerate the MCLMS algorithm, developing a scheme to determine the optimal step size during adaption is desirable and effective. We begin this endeavor with re-examining the update equation (13.15) or equivalently (13.24) of the unit-norm constrained MCLMS algorithm. As the adaptive algorithm proceeds, the cost function $J(k+1)$ diminishes and its gradient with respect to $\hat{\boldsymbol{h}}(k)$ can be approximated as

$$\nabla J(k+1) \approx \frac{2\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)}{\|\hat{\boldsymbol{h}}(k)\|^2}\,. \tag{13.36}$$

At this point, removing the unit-norm constraint leads to a simplified, *unconstrained* MCLMS adaptive algorithm:

$$\hat{\boldsymbol{h}}(k+1) = \hat{\boldsymbol{h}}(k) - 2\mu\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\,, \tag{13.37}$$

which is theoretically equivalent to the adaptive algorithm proposed in [13.19] although the cost functions are defined in different ways in these two adaptive blind SIMO identification algorithms.

With such a simplified adaptive algorithm, the primary concern is whether it will converge to the trivial all-zero estimate. Fortunately this will not happen as long as the initial estimate $\hat{\boldsymbol{h}}(0)$ is not orthogonal to the true channel impulse response vector $\boldsymbol{h}$, as shown in [13.19]. This can easily be demonstrated by premultiplying (13.37) with $\boldsymbol{h}^{\mathrm{T}}$:

$$\boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(k+1) = \boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(k) - 2\mu\boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\,. \tag{13.38}$$



Amazon EX1010 Page 293

**Table 13.3** The variable-step-size unconstrained multichannel LMS adaptive algorithm for the blind identification of an SIMO FIR system

| | |
|---|---|
| Parameters: | $\hat{\boldsymbol{h}} = [\hat{\boldsymbol{h}}_1^{\mathrm{T}} \ \hat{\boldsymbol{h}}_2^{\mathrm{T}} \ \cdots \ \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}$, model filter |
| Initialization: | $\hat{\boldsymbol{h}}_n(0) = [1 \ \ 1 \ \cdots \ 1]^{\mathrm{T}}$, $n = 1, 2, \cdots, N$ |
| Computation: | For $k = 0, 1, 2, \cdots$, compute |
| (a) | $e_{ij}(k+1) = \boldsymbol{x}_i^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k) - \boldsymbol{x}_j^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_i(k)$, $i, j = 1, 2, \cdots, N$ |
| (b) | $\chi(k+1) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} e_{ij}^2(k+1)$ |
| (c) | Construct the matrix $\tilde{\mathbf{R}}_{x+}(k+1)$ following Sect. 13.4.2 |
| (d) | $\nabla J(k+1) \approx \dfrac{2\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)}{\|\hat{\boldsymbol{h}}(k)\|^2}$ |
| (e) | $\mu_{\mathrm{opt}}(k+1) = \dfrac{\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\nabla J(k+1)}{\|\nabla J(k+1)\|^2}$ |
| (f) | $\hat{\boldsymbol{h}}(k+1) = \hat{\boldsymbol{h}}(k) - \mu_{\mathrm{opt}}(k+1)\nabla J(k+1)$ |

Using the cross relation (13.4), we know that, in the absence of noise,

$$\boldsymbol{h}^{\mathrm{T}}\tilde{\boldsymbol{R}}_{x+}(k+1) = \boldsymbol{0}^{\mathrm{T}} . \tag{13.39}$$

This implies that the gradient $\nabla J(k+1)$ is orthogonal to $\boldsymbol{h}$ at any time $k$. As a result, (13.38) turns out to be

$$\boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(k+1) = \boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(k) . \tag{13.40}$$

This indicates that $\boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(k)$ is time invariant for the unconstrained MCLMS algorithm. Provided that $\boldsymbol{h}^{\mathrm{T}}\hat{\boldsymbol{h}}(0) \neq 0$, $\hat{\boldsymbol{h}}(k)$ will not converge to zero.

Now we should feel comfortable to work on this unconstrained MCLMS algorithm and search for an optimal step size. Let us decompose the model filter $\hat{\boldsymbol{h}}(k)$ as follows:

$$\hat{\boldsymbol{h}}(k) = \hat{\boldsymbol{h}}_{\perp}(k) + \hat{\boldsymbol{h}}_{\parallel}(k) , \tag{13.41}$$

where $\hat{\boldsymbol{h}}_{\perp}(k)$ and $\hat{\boldsymbol{h}}_{\parallel}(k)$ are perpendicular and parallel to $\boldsymbol{h}$, respectively. Since the gradient $\nabla J(k+1)$ is orthogonal to $\boldsymbol{h}$ and $\boldsymbol{h}$ is parallel to $\boldsymbol{h}_{\parallel}(k)$, obviously $\nabla J(k+1)$ is orthogonal to $\boldsymbol{h}_{\parallel}(k)$ as well. Therefore, the update equation (13.37) of the unconstrained MCLMS algorithm can be decomposed into the following two separate equations:

$$\hat{\boldsymbol{h}}_{\perp}(k+1) = \hat{\boldsymbol{h}}_{\perp}(k) - \mu\nabla J(k+1) , \tag{13.42}$$

$$\hat{\boldsymbol{h}}_{\parallel}(k+1) = \hat{\boldsymbol{h}}_{\parallel}(k) . \tag{13.43}$$

From (13.42) and (13.43), it is clear that the unconstrained MCLMS algorithm adapts the model filter only in the direction perpendicular to $\boldsymbol{h}$. The component $\hat{\boldsymbol{h}}_{\parallel}(k)$ is not altered in the process of adaptation.

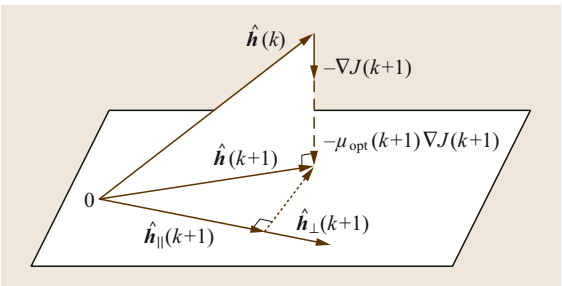



**Fig. 13.2** Illustration of the optimal step size $\mu_{\mathrm{opt}}(k+1)$ for the unconstrained MCLMS blind SIMO identification algorithm in a three-dimensional space

As far as a general system identification algorithm is concerned, the most important performance measure should apparently be the difference between the true channel impulse response and the estimate. Using a blind multichannel identification method, the channel impulse responses of a SIMO FIR system can be estimated up to a scale. Therefore, the misalignment of an estimate $\hat{\boldsymbol{h}}(k)$ with respect to the true channel impulse response vector $\boldsymbol{h}$ would be:

$$d(k) = \min_{\alpha} \left\| \boldsymbol{h} - \alpha\hat{\boldsymbol{h}}(k) \right\|^2 , \tag{13.44}$$

where $\alpha$ is an arbitrary scale. Substituting (13.41) into (13.44) and finding the minimum yields

$$\begin{aligned} d(k) &= \min_{\alpha}[\|\hat{\boldsymbol{h}}(k)\|^2\alpha^2 - 2\|\hat{\boldsymbol{h}}_{\parallel}(k)\|\|\boldsymbol{h}\|\alpha + \|\boldsymbol{h}\|^2] \\ &= \frac{\|\boldsymbol{h}\|^2}{1 + (\|\hat{\boldsymbol{h}}_{\parallel}(k)\|/\|\hat{\boldsymbol{h}}_{\perp}(k)\|)^2} . \end{aligned} \tag{13.45}$$



Amazon EX1010 Page 294

Clearly the ratio of $\|\hat{\boldsymbol{h}}_{\|}(k)\|$ over $\|\hat{\boldsymbol{h}}_{\perp}(k)\|$ reflects how close the estimate is to the desired solution. With this feature in mind, the optimal step size $\mu_{\text{opt}}(k+1)$ for the unconstrained MCLMS algorithm at time $k+1$ would be the one that makes $\hat{\boldsymbol{h}}_{\perp}(k+1)$ have a minimum norm, i. e.,

$$\begin{aligned}\mu_{\text{opt}}(n+1) &= \arg\min_{\mu} \|\hat{\boldsymbol{h}}_{\perp}(k+1)\| \\ &= \arg\min_{\mu} \|\hat{\boldsymbol{h}}_{\perp}(k) - \mu\nabla J(k+1)\| \,. \end{aligned} \tag{13.46}$$

In order to minimize the norm of $\hat{\boldsymbol{h}}(n+1) = \hat{\boldsymbol{h}}(k) - \mu(k+1)\nabla J(k+1)$, as illustrated in Fig. 13.2, $\mu(k+1)$ should be chosen such that $\hat{\boldsymbol{h}}(k+1)$ is orthogonal to $\nabla J(k+1)$. Therefore, we project $\hat{\boldsymbol{h}}(k)$ onto $\nabla J(k+1)$ and obtain the optimal step size:

$$\mu_{\text{opt}}(k+1) = \frac{\hat{\boldsymbol{h}}^{\mathrm{T}}(k)\nabla J(k+1)}{\|\nabla J(k+1)\|^2} \,. \tag{13.47}$$

Finally, this new adaptive algorithm with the optimal step size is referred to as the variable-step-size unconstrained MCLMS (VSS-UMCLMS) [13.20] for blind SIMO identification and is summarized in Table 13.3.

## 13.6 Frequency-Domain Blind Multichannel Identification Algorithms

In this section, we study the problem of adaptive blind SIMO identification in the frequency domain. Time–frequency analysis utilizing the fast Fourier transform (FFT) is an important mathematical tool in signal processing. Since its introduction by *Dentino* et al. [13.21], adaptive filtering in the frequency domain has attracted a great deal of research interest and has recently become an essential constituent of adaptive filtering theory. By taking advantage of the computational efficiency of the FFT, a convolution of two signals can be quickly calculated. Moreover, a discrete Fourier transform processes a time sequence like a filter bank, which orthogonalizes the data, and therefore the coefficients of a frequency-domain adaptive filter can converge independently or even uniformly if the update is normalized properly. For single-channel cases, the derivation and implementation of a frequency-domain adaptive filter is relatively simple. However, when multiple channels are considered, the algorithmic complexity increases significantly with the number of channels.

Deriving a frequency-domain adaptive algorithm can involve a large number of variables in the form of both vectors and matrices, in both the time and frequency domains. Therefore, before formulating the problem and developing adaptive algorithms in the frequency domain, it is beneficial to clarify the notation used in the following. The notation is conventional in the time domain, but is specifically defined in the frequency domain. In the frequency domain, matrices and vectors are represented respectively by uppercase and lowercase bold, *italic* letters, and are further emphasized by an arrow underneath. However, for a vector the arrow is single-barred while for a matrix the arrow is double-barred. The difference in their appearance is illustrated by the following example:

- $\underset{\rightarrow}{\mathsf{x}}$ a vector in the frequency domain (sans serif, bold lowercase, with a single-headed single-barred arrow underneath),
- $\underset{\Rightarrow}{\mathsf{X}}$ a matrix in the frequency domain (sans serif, bold, uppercase, with a single-headed double-barred arrow underneath).

### 13.6.1 Frequency-Domain Multichannel LMS Algorithm

We begin with the definition of a signal $y_{ij}(k+1)$ by convolving the $i$-th channel output $x_i(k+1)$ with the $j$-th model filter $\hat{h}_j(k)$ $(i, j = 1, 2, \cdots, N)$:

$$y_{ij}(k+1) = x_i(k+1) * \hat{h}_j(k) \,. \tag{13.48}$$

As can be clearly seen in Sect. 13.4, $y_{ij}(k)$ is essential to calculate the error signal $e_{ij}(k+1)$ in (13.11), but involves intensive computation in the time domain. Here, we intend to perform digital filtering efficiently in the frequency domain using the overlap-save technique [13.22].

In the overlap-save technique, the signal is processed on a frame-by-frame basis. We are going to indicate the index of a frame with $t$. Let the vector $\tilde{\boldsymbol{y}}_{ij}(t+1)$ of length $2L$ denote the result of the circular convolution of $\boldsymbol{x}_i(t+1)$ and $\hat{\boldsymbol{h}}_j(t)$:

$$\tilde{\boldsymbol{y}}_{ij}(t+1) = \boldsymbol{C}_{x_i}(t+1)\hat{\boldsymbol{h}}_j^{10}(t) \,, \tag{13.49}$$



Amazon EX1010 Page 295

where

$$\tilde{\boldsymbol{y}}_{ij}(t+1) = [\tilde{y}_{ij}(tL)\ \ \tilde{y}_{ij}(tL+1) \cdots\ \tilde{y}_{ij}(tL+2L-1)]^{\mathrm{T}},$$

$$\boldsymbol{C}_{x_i}(t+1) = \begin{pmatrix} x_i(tL) & x_i(tL+2L-1) & \cdots & x_i(tL+1) \\ x_i(tL+1) & x_i(tL) & \cdots & x_i(tL+2) \\ \vdots & \vdots & \ddots & \vdots \\ x_i(tL+2L-1) & x_i(tL+2L-2) & \cdots & x_i(tL) \end{pmatrix},$$

$$\hat{\boldsymbol{h}}_j^{10}(t) = \left[\hat{\boldsymbol{h}}_j^{\mathrm{T}}(t)\ \ \boldsymbol{0}_{L\times 1}^{\mathrm{T}}\right]^{\mathrm{T}} = [\hat{h}_{j,0}(t)\ \cdots \hat{h}_{j,L-1}(t)\ \ 0\ \cdots\ 0]^{\mathrm{T}}.$$

Note that $\boldsymbol{C}_{x_i}(t+1)$ is a circulant matrix. It can easily be shown that the last $L$ points in the circular convolution output, i. e., $\tilde{y}_{ij}(tL+L), \cdots, \tilde{y}_{ij}(tL+2L-1)$, are identical to the results of a linear convolution:

$$\begin{aligned} \boldsymbol{y}_{ij}(t+1) &= \boldsymbol{W}_{L\times 2L}^{01}\tilde{\boldsymbol{y}}_{ij}(t+1) \\ &= \boldsymbol{W}_{L\times 2L}^{01}\boldsymbol{C}_{x_i}(t+1)\hat{\boldsymbol{h}}_j^{10}(t) \\ &= \boldsymbol{W}_{L\times 2L}^{01}\boldsymbol{C}_{x_i}(t+1)\boldsymbol{W}_{2L\times L}^{10}\hat{\boldsymbol{h}}_j(t), \end{aligned} \tag{13.50}$$

where

$$\begin{aligned} \boldsymbol{y}_{ij}(t+1) &= [y_{ij}(tL)\ \ y_{ij}(tL+1) \cdots\ y_{ij}(tL+L-1)]^{\mathrm{T}}, \\ \boldsymbol{W}_{L\times 2L}^{01} &= (\boldsymbol{0}_{L\times L}\ \ \boldsymbol{I}_{L\times L}), \\ \boldsymbol{W}_{2L\times L}^{10} &= [\boldsymbol{I}_{L\times L}\ \ \boldsymbol{0}_{L\times L}]^{\mathrm{T}}, \\ \hat{\boldsymbol{h}}_j(t) &= [\hat{h}_{j,0}(t)\ \ \hat{h}_{j,1}(t)\ \cdots\ \hat{h}_{j,L-1}(t)]^{\mathrm{T}}. \end{aligned}$$

Therefore we overlap the input data blocks by $L$ points. For each block of length $L$, we have $2L$ data inputs, discard the first $L$ circular convolution results, and retain only the last $L$ results as outputs.

With $\boldsymbol{y}_{ij}(t+1)$ defined, a block of the a priori error signal based on the cross-relation between the $i$-th and $j$-th channel outputs is determined as:

$$\begin{aligned} \boldsymbol{e}_{ij}(t+1) &= \boldsymbol{y}_{ij}(t+1) - \boldsymbol{y}_{ji}(t+1) \\ &= \boldsymbol{W}_{L\times 2L}^{01}\left[\boldsymbol{C}_{x_i}(t+1)\boldsymbol{W}_{2L\times L}^{10}\hat{\boldsymbol{h}}_j(t) - \boldsymbol{C}_{x_j}(t+1)\boldsymbol{W}_{2L\times L}^{10}\hat{\boldsymbol{h}}_i(t)\right]. \end{aligned} \tag{13.51}$$

Next, we use the FFT technique to perform a circular convolution efficiently. Let $\boldsymbol{F}_{L\times L}$ be the Fourier matrix of size $L\times L$:

$$\boldsymbol{F}_{L\times L} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \mathrm{e}^{-\mathrm{i}\frac{2\pi}{L}} & \mathrm{e}^{-\mathrm{i}\frac{4\pi}{L}} & \cdots & \mathrm{e}^{-\mathrm{i}\frac{2\pi(L-1)}{L}} \\ 1 & \mathrm{e}^{-\mathrm{i}\frac{4\pi}{L}} & \mathrm{e}^{-\mathrm{i}\frac{8\pi}{L}} & \cdots & \mathrm{e}^{-\mathrm{i}\frac{4\pi(L-1)}{L}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mathrm{e}^{-\mathrm{i}\frac{2\pi(L-1)}{L}} & \mathrm{e}^{-\mathrm{i}\frac{4\pi(L-1)}{L}} & \cdots & \mathrm{e}^{-\mathrm{i}\frac{2\pi(L-1)^2}{L}} \end{pmatrix}, \tag{13.52}$$

where i is not the dummy variable we used above to indicate a system output, but stands for the square root of $-1$, i. e., $\mathrm{i} = \sqrt{-1}$. The matrix $\boldsymbol{F}_{L\times L}$ and its inverse are related by

$$\boldsymbol{F}_{L\times L}^{\mathrm{H}} = L\boldsymbol{F}_{L\times L}^{-1}, \tag{13.53}$$

where $(\cdot)^{\mathrm{H}}$ denotes the conjugate transpose or Hermitian transpose of a vector of a matrix. By using $\boldsymbol{F}_{2L\times 2L}$ and $\boldsymbol{F}_{2L\times 2L}^{-1}$, we can decompose the circulant matrix $\boldsymbol{C}_{x_i}(t+1)$:

$$\boldsymbol{C}_{x_i}(t+1) = \boldsymbol{F}_{2L\times 2L}^{-1}\underset{\Rightarrow x_i}{\mathsf{D}}(t+1)\boldsymbol{F}_{2L\times 2L}, \tag{13.54}$$

where $\underset{\Rightarrow x_i}{\mathsf{D}}(t+1)$ is a diagonal matrix whose diagonal elements are given by the DFT of the first column of $\boldsymbol{C}_{x_i}(t+1)$, i. e., the overlapped $i$-th channel output in the $(t+1)$-th block:

$$\boldsymbol{x}_i(t+1)_{2L\times 1} = [x_i(tL)\ \ x_i(tL+1) \cdots\ x_i(tL+2L-1)]^{\mathrm{T}}. \tag{13.55}$$

Now we multiply (13.51) by $\boldsymbol{F}_{L\times L}$ and use (13.54) to determine the block error sequence in the frequency domain:

$$\begin{aligned} \underset{\rightarrow ij}{\mathsf{e}}(t+1) &= \boldsymbol{F}_{L\times L}\boldsymbol{e}_{ij}(t+1) \\ &= \boldsymbol{F}_{L\times L}\boldsymbol{W}_{L\times 2L}^{01}\left[\boldsymbol{C}_{x_i}(t+1)\boldsymbol{W}_{2L\times L}^{10}\hat{\boldsymbol{h}}_j(t) - \boldsymbol{C}_{x_j}(t+1)\boldsymbol{W}_{2L\times L}^{10}\hat{\boldsymbol{h}}_i(t)\right] \\ &= \underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01}\left[\underset{\Rightarrow x_i}{\mathsf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\underset{\rightarrow j}{\hat{\mathsf{h}}}(t) - \underset{\Rightarrow x_j}{\mathsf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\underset{\rightarrow i}{\hat{\mathsf{h}}}(t)\right], \end{aligned} \tag{13.56}$$

where

$$\begin{aligned} \underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01} &= \boldsymbol{F}_{L\times L}\boldsymbol{W}_{L\times 2L}^{01}\boldsymbol{F}_{2L\times 2L}^{-1}, \\ \underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10} &= \boldsymbol{F}_{2L\times 2L}\boldsymbol{W}_{2L\times L}^{10}\boldsymbol{F}_{L\times L}^{-1}, \end{aligned}$$

and $\underset{\rightarrow i}{\hat{\mathsf{h}}}(t)$ consists of the $L$-point DFTs of the vector $\hat{\boldsymbol{h}}_i(t)$ at the $t$-th block, i. e., $\underset{\rightarrow i}{\hat{\mathsf{h}}}(t) = \boldsymbol{F}_{L\times L}\hat{\boldsymbol{h}}_i(t)$.

Having derived a frequency-domain block error signal (13.56), we can consequently construct



Amazon EX1010 Page 296

a frequency-domain mean square error criterion analogous to its time-domain counterpart:

$$J_{\mathrm{f}} = E\{J_{\mathrm{f}}(t)\}\,, \tag{13.57}$$

where

$$J_{\mathrm{f}}(t) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \underset{\rightarrow ij}{\mathsf{e}}^{\mathrm{H}}(t)\underset{\rightarrow ij}{\mathsf{e}}(t)$$

is the instantaneous square error at the $t$-th block.

Taking the partial derivative of $J_{\mathrm{f}}$ with respect to $\underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)$ (where $n = 1, 2, \cdots, N$ and $(\cdot)^{*}$ stands for complex conjugate) and pretending that $\underset{\rightarrow n}{\hat{\mathsf{h}}}(t)$ is a constant [13.23] produces

$$\frac{\partial J_{\mathrm{f}}}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)} = E\left\{\frac{\partial J_{\mathrm{f}}(t+1)}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)}\right\}\,. \tag{13.58}$$

In the LMS algorithm, the expectation is estimated with a one-point sample mean, i. e., the instantaneous value, which is given by:

$$\begin{aligned}
&\frac{\partial J_{\mathrm{f}}(t+1)}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)} \\
&= \frac{\partial}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)}\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \underset{\rightarrow ij}{\mathsf{e}}^{\mathrm{H}}(t+1)\underset{\rightarrow ij}{\mathsf{e}}(t+1)\right] \\
&= \frac{\partial}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)}\left[\sum_{i=1}^{n-1} \underset{\rightarrow in}{\mathsf{e}}^{\mathrm{H}}(t+1)\underset{\rightarrow in}{\mathsf{e}}(t+1)\right] \\
&\quad + \frac{\partial}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^{*}(t)}\left[\sum_{j=n+1}^{N} \underset{\rightarrow nj}{\mathsf{e}}^{\mathrm{H}}(t+1)\underset{\rightarrow nj}{\mathsf{e}}(t+1)\right] \\
&= \sum_{i=1}^{n-1}\left[\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01}\underset{\Rightarrow x_i}{\mathsf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\right]^{\mathrm{H}}\underset{\rightarrow in}{\mathsf{e}}(t+1) \\
&\quad - \sum_{j=n+1}^{N}\left[\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01}\underset{\Rightarrow x_j}{\mathsf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\right]^{\mathrm{H}}\underset{\rightarrow nj}{\mathsf{e}}(t+1) \\
&= \sum_{i=1}^{N}\left[\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01}\underset{\Rightarrow x_i}{\mathsf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\right]^{\mathrm{H}}\underset{\rightarrow in}{\mathsf{e}}(t+1)\,,
\end{aligned} \tag{13.59}$$

where the last step follows from the fact $\underset{\rightarrow nn}{\mathsf{e}}(t) = \mathbf{0}$. Using this gradient, we obtain the frequency-domain *unconstrained* multichannel LMS algorithm:

$$\begin{aligned}
\underset{\rightarrow n}{\hat{\mathsf{h}}}(t+1) &= \underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{10} \\
&\quad \cdot \sum_{i=1}^{N}\underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{01}\underset{\rightarrow in}{\mathsf{e}}(t+1)\,,
\end{aligned} \tag{13.60}$$

where $\mu_{\mathrm{f}}$ is a small positive step size and

$$\begin{aligned}
\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{10} &= \boldsymbol{F}_{L\times L}\boldsymbol{W}^{10}_{L\times 2L}\boldsymbol{F}^{-1}_{2L\times 2L} = \frac{1}{2}\left(\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{10}\right)^{\mathrm{H}}\,, \\
\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{01} &= \boldsymbol{F}_{2L\times 2L}\boldsymbol{W}^{01}_{2L\times L}\boldsymbol{F}^{-1}_{L\times L} = 2\left(\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{01}\right)^{\mathrm{H}}\,, \\
\boldsymbol{W}^{10}_{L\times 2L} &= (\boldsymbol{I}_{L\times L}\ \ \boldsymbol{0}_{L\times L})\,, \\
\boldsymbol{W}^{01}_{2L\times L} &= (\boldsymbol{0}_{L\times L}\ \ \boldsymbol{I}_{L\times L})^{\mathrm{T}}\,.
\end{aligned}$$

If the unit-norm constraint is employed, then we know that

$$\|\underset{\rightarrow}{\hat{\mathsf{h}}}(t)\|^2 = \frac{\|\hat{\boldsymbol{h}}(t)\|^2}{L} = \frac{1}{L}\,, \tag{13.61}$$

where

$$\underset{\rightarrow}{\hat{\mathsf{h}}}(t) = \left[\underset{\rightarrow 1}{\hat{\mathsf{h}}}^{\mathrm{T}}(t)\ \ \underset{\rightarrow 2}{\hat{\mathsf{h}}}^{\mathrm{T}}(t)\ \cdots\ \underset{\rightarrow N}{\hat{\mathsf{h}}}^{\mathrm{T}}(t)\right]^{\mathrm{T}}\,.$$

Enforcing the constraint on (13.60), we finally deduce the frequency-domain unit-norm-constrained multichannel LMS (FCMCLMS) algorithm [13.24]:

$$\begin{aligned}
&\underset{\rightarrow n}{\hat{\mathsf{h}}}(t+1) \\
&= \left\{\underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{10}\right. \\
&\quad \left.\cdot \sum_{i=1}^{N}\underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{01}\underset{\rightarrow in}{\mathsf{e}}(t+1)\right\} \\
&\quad \Big/ \left\{\sqrt{L}\left\|\underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow L\times 2L}{\mathsf{W}}^{10}\right.\right. \\
&\quad \left.\left.\cdot \sum_{i=1}^{N}\underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1)\underset{\Rightarrow 2L\times L}{\mathsf{W}}^{01}\underset{\rightarrow in}{\mathsf{e}}(t+1)\right\|\right\}\,, \\
&\quad n = 1, 2, \cdots, N\,.
\end{aligned} \tag{13.62}$$

This derivation is mathematically rigorous and the algorithm might look quite complicated to some readers since it involves a number of matrix multiplications. However, the implementation is not necessarily tedious. Actually many matrix multiplications are performed in the form of the Schur (element-by-element) product of two vectors. Here we want to briefly comment on the issue of how to skillfully and efficiently implement the



Amazon EX1010 Page 297

FCMCLMS algorithm. We will use (13.56) as an example, in which the error signal is calculated. The product $\underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}_{10}\,\underset{\rightarrow j}{\hat{\mathbf{h}}}(t)$ is obtained by appending $L$ zeros to the end of $\hat{\boldsymbol{h}}_j(t)$ and taking the $2L$-point FFT:

$$\underset{\rightarrow j}{\hat{\mathbf{h}}}^{10}(t) = \underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}\,\underset{\rightarrow j}{\hat{\mathbf{h}}}(t) = \mathrm{FFT}_{2L}\{\hat{\boldsymbol{h}}_j^{10}(t)\}\,, \quad (13.63)$$

where $\mathrm{FFT}_{2L}\{\cdot\}$ denotes the $2L$-point fast Fourier transform and $\hat{\boldsymbol{h}}_j^{10}(t)$ is given in (13.49). Note that $\underset{\Rightarrow x_i}{\mathbf{D}}(t+1)$ is a diagonal matrix whose diagonal elements are the DFT coefficients of $\boldsymbol{x}_i(t+1)_{2L\times 1}$ given in (13.55), i. e.,

$$\begin{aligned}\underset{\Rightarrow x_i}{\mathbf{D}}(t+1) &= \mathrm{diag}[\underset{\rightarrow i}{\mathbf{x}}(t+1)_{2L\times 1}] \\ &= \mathrm{diag}[\mathrm{FFT}_{2L}\{\boldsymbol{x}_i(t+1)_{2L\times 1}\}]\,. \end{aligned} \quad (13.64)$$

Therefore multiplying $\underset{\Rightarrow x_i}{\mathbf{D}}(t+1)$ with $\underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}\,\underset{\rightarrow j}{\hat{\mathbf{h}}}(t)$ would be straightforward:

$$\begin{aligned}\underset{\Rightarrow x_i}{\mathbf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}\,\underset{\rightarrow j}{\hat{\mathbf{h}}}(t) &= \underset{\Rightarrow x_i}{\mathbf{D}}(t+1)\,\underset{\rightarrow j}{\hat{\mathbf{h}}}^{10}(t) \\ &= \underset{\rightarrow i}{\mathbf{x}}(t+1)_{2L\times 1}\odot \underset{\rightarrow j}{\hat{\mathbf{h}}}^{10}(t)\,, \end{aligned} \quad (13.65)$$

where $\odot$ is the operator of the Schur product. Let $\boldsymbol{e}_{ij}(t+1)_{2L\times 1}$ be

$$\begin{aligned}\boldsymbol{e}_{ij}(t+1)_{2L\times 1} &= \mathrm{IFFT}_{2L}\Big\{\underset{\Rightarrow x_i}{\mathbf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}\,\underset{\rightarrow j}{\hat{\mathbf{h}}}(t) \\ &\quad - \underset{\Rightarrow x_j}{\mathbf{D}}(t+1)\underset{\Rightarrow 2L\times L}{\mathbf{W}}^{10}\,\underset{\rightarrow i}{\hat{\mathbf{h}}}(t)\Big\} \\ &= \mathrm{IFFT}_{2L}\Big\{\underset{\rightarrow i}{\mathbf{x}}(t+1)_{2L\times 1}\odot \underset{\rightarrow j}{\hat{\mathbf{h}}}^{10}(t) \\ &\quad - \underset{\rightarrow j}{\mathbf{x}}(t+1)_{2L\times 1}\odot \underset{\rightarrow i}{\hat{\mathbf{h}}}^{10}(t)\Big\}\,, \end{aligned} \quad (13.66)$$

**Table 13.4** The frequency-domain constrained multichannel LMS (FCMCLMS) adaptive algorithm for the blind identification of an SIMO FIR system

| | |
|---|---|
| Parameters: | $\hat{\boldsymbol{h}} = [\hat{\boldsymbol{h}}_1^{\mathrm{T}}\ \hat{\boldsymbol{h}}_2^{\mathrm{T}}\ \cdots\ \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}$, model filter |
| | $\mu_{\mathrm{f}} > 0$, step size |
| Initialization: | $\hat{\boldsymbol{h}}_n(0) = [1\ \ 0\ \ \cdots\ \ 0]^{\mathrm{T}}$, $n = 1, 2, \cdots, N$ |
| | $\hat{\boldsymbol{h}}(0) = \hat{\boldsymbol{h}}(0)/\sqrt{N}$ (normalization) |
| Computation: | For $t = 0, 1, 2, \cdots,$ compute |
| (a) | $\underset{\rightarrow n}{\hat{\mathbf{h}}}^{10}(t) = \mathrm{FFT}_{2L}\{[\hat{\boldsymbol{h}}_n^{\mathrm{T}}(t)\ \ \mathbf{0}_{1\times L}]^{\mathrm{T}}\}$, $n = 1, 2, \cdots, N$ |
| (b) | $\boldsymbol{x}_n(t+1)_{2L\times 1} = [x_n(tL)\ \ x_n(tL+1)\ \ \cdots\ \ x_n(tL+2L-1)]^{\mathrm{T}}$ |
| (c) | $\underset{\rightarrow n}{\mathbf{x}}(t+1)_{2L\times 1} = \mathrm{FFT}_{2L}\{\boldsymbol{x}_n(t+1)_{2L\times 1}\}$ |
| (d) | $\underset{\rightarrow ij}{\mathbf{e}}(t+1)_{2L\times 1} = \underset{\rightarrow i}{\mathbf{x}}(t+1)_{2L\times 1}\odot \underset{\rightarrow j}{\hat{\mathbf{h}}}^{10}(t) - \underset{\rightarrow j}{\mathbf{x}}(t+1)_{2L\times 1}\odot \underset{\rightarrow i}{\hat{\mathbf{h}}}^{10}(t)$, $i, j = 1, 2, \cdots, N$ |
| (e) | $\boldsymbol{e}_{ij}(t+1)_{2L\times 1} = \mathrm{IFFT}_{2L}\{\underset{\rightarrow ij}{\mathbf{e}}(t+1)_{2L\times 1}\}$ |
| (f) | Take the *last* $L$ elements of $\boldsymbol{e}_{ij}(t+1)_{2L\times 1}$ to form $\boldsymbol{e}_{ij}(t+1)$ |
| (g) | $\underset{\rightarrow ij}{\mathbf{e}}^{01}(t+1) = \mathrm{FFT}_{2L}\{[\mathbf{0}_{1\times L}\ \ \boldsymbol{e}_{ij}^{\mathrm{T}}(t+1)]^{\mathrm{T}}\}$ |
| (h) | $\Delta\underset{\rightarrow n}{\hat{\mathbf{h}}}(t)_{2L\times 1} = \mu_{\mathrm{f}}\sum_{i=1}^{N}\underset{\rightarrow i}{\mathbf{x}}^{*}(t+1)_{2L\times 1}\odot \underset{\rightarrow in}{\mathbf{e}}^{01}(t+1)$ |
| (i) | $\Delta\hat{\boldsymbol{h}}_n(t)_{2L\times 1} = \mathrm{IFFT}_{2L}\{\Delta\underset{\rightarrow n}{\hat{\mathbf{h}}}(t)_{2L\times 1}\}$ |
| (j) | Take the *first* $L$ elements of $\Delta\hat{\boldsymbol{h}}_n(t)_{2L\times 1}$ to form $\Delta\hat{\boldsymbol{h}}_n(t)$ |
| (k) | $\hat{\boldsymbol{h}}_n(t+1) = \dfrac{\hat{\boldsymbol{h}}_n(t) - \Delta\hat{\boldsymbol{h}}_n(t)}{\left\|\hat{\boldsymbol{h}}_n(t) - \Delta\hat{\boldsymbol{h}}_n(t)\right\|}$, $n = 1, 2, \cdots, N$ |



Amazon EX1010 Page 298

where $\text{IFFT}_{2L}\{\cdot\}$ denotes $2L$-point *inverse* fast Fourier transform. Then

$$\boldsymbol{e}_{ij}(t+1) = \boldsymbol{W}^{01}_{L\times 2L}\boldsymbol{e}_{ij}(t+1)_{2L\times 1} \tag{13.67}$$

can be simply carried out by discarding the first $L$ and keeping the last $L$ elements of $\boldsymbol{e}_{ij}(t+1)_{2L\times 1}$ to construct $\boldsymbol{e}_{ij}(t+1)$. Finally $\underset{\rightarrow ij}{\mathsf{e}}(t+1)$ is obtained by taking an $L$-point FFT of $\boldsymbol{e}_{ij}(t+1)$.

The tricks developed for (13.56) can also be applied to (13.60) and (13.62), which will be left to the reader. The FCMCLMS algorithm is summarized in Table 13.4. Compared to the time-domain frame-based multichannel LMS algorithm, the frequency-domain implementation is computationally more efficient since it uses the FFT to calculate block convolution and block correlation in the frequency domain. For each processed frame, it can be checked from Table 13.4 that $(N^2+2N)$ FFTs of $2L$ points are sufficient to implement exact linear convolutions between the channel outputs and the adaptive filter coefficients, and to implement exact correlations between the outputs and the error signals.

Having developed the frame-based MCLMS algorithm in the frequency domain, we need to demonstrate that it converges in the mean to the desired solution in a stationary environment. To establish this property, we begin by defining

$$\underset{\Rightarrow x_n}{\mathsf{S}}(t+1) = \mathsf{W}^{01}_{\Rightarrow L\times 2L}\,\underset{\Rightarrow x_n}{\mathsf{D}}(t+1)\mathsf{W}^{10}_{\Rightarrow 2L\times L}\,,$$
$$n = 1, 2, \cdots, N\,. \tag{13.68}$$

Then the block error signal in the frequency domain given by (13.56) can be rewritten as

$$\underset{\rightarrow ij}{\mathsf{e}}(t+1) = \underset{\Rightarrow x_i}{\mathsf{S}}(t+1)\underset{\rightarrow j}{\hat{\mathsf{h}}}(t) - \underset{\Rightarrow x_j}{\mathsf{S}}(t+1)\underset{\rightarrow i}{\hat{\mathsf{h}}}(t)\,, \tag{13.69}$$

and the instantaneous gradient (13.59) becomes

$$\frac{\partial J_{\text{f}}(t+1)}{\partial \underset{\rightarrow n}{\hat{\mathsf{h}}}^*(t)} = \sum_{i=1}^{N}\underset{\Rightarrow x_i}{\mathsf{S}}^{\text{H}}(t+1)\underset{\rightarrow in}{\mathsf{e}}(t+1)$$
$$= \underset{\Rightarrow}{\mathsf{S}}^{\text{H}}(t+1)\underset{\rightarrow n}{\mathsf{e}}(t+1)\,, \tag{13.70}$$

where

$$\underset{\Rightarrow}{\mathsf{S}}(t+1) = \left[\underset{\Rightarrow x_1}{\mathsf{S}}^{\text{H}}(t+1)\ \underset{\Rightarrow x_2}{\mathsf{S}}^{\text{H}}(t+1)\ \cdots\ \underset{\Rightarrow x_N}{\mathsf{S}}^{\text{H}}(t+1)\right]^{\text{H}},$$
$$\underset{\rightarrow n}{\mathsf{e}}(t+1) = \left[\underset{\rightarrow 1n}{\mathsf{e}}^{\text{T}}(t+1)\ \underset{\rightarrow 2n}{\mathsf{e}}^{\text{T}}(t+1)\ \cdots\ \underset{\rightarrow Nn}{\mathsf{e}}^{\text{T}}(t+1)\right]^{\text{T}}.$$

From (13.69), we then decompose the error signal $\underset{\rightarrow n}{\mathsf{e}}(t+1)$ associated with the $n$-th channel as follows:

$$\underset{\rightarrow n}{\mathsf{e}}(t+1) = \begin{pmatrix} \underset{\Rightarrow x_1}{\mathsf{S}}(t+1)\underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \underset{\Rightarrow x_n}{\mathsf{S}}(t+1)\underset{\rightarrow 1}{\hat{\mathsf{h}}}(t) \\ \underset{\Rightarrow x_2}{\mathsf{S}}(t+1)\underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \underset{\Rightarrow x_n}{\mathsf{S}}(t+1)\underset{\rightarrow 2}{\hat{\mathsf{h}}}(t) \\ \vdots \\ \underset{\Rightarrow x_N}{\mathsf{S}}(t+1)\underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \underset{\Rightarrow x_n}{\mathsf{S}}(t+1)\underset{\rightarrow N}{\hat{\mathsf{h}}}(t) \end{pmatrix}$$
$$= [\underset{\Rightarrow n}{\mathsf{U}}(t+1) - \underset{\Rightarrow n}{\mathsf{V}}(t+1)]\underset{\rightarrow}{\hat{\mathsf{h}}}(t)\,, \tag{13.71}$$

where

$$\underset{\Rightarrow n}{\mathsf{U}}(t+1) = \begin{pmatrix} \boldsymbol{0}_{L\times L} & \cdots & \boldsymbol{0}_{L\times L} & \underset{\times(t+1)}{\underset{\Rightarrow x_1}{\mathsf{S}}} & \boldsymbol{0}_{L\times L} & \cdots & \boldsymbol{0}_{L\times L} \\ \boldsymbol{0}_{L\times L} & \cdots & \boldsymbol{0}_{L\times L} & \underset{\times(t+1)}{\underset{\Rightarrow x_2}{\mathsf{S}}} & \boldsymbol{0}_{L\times L} & \cdots & \boldsymbol{0}_{L\times L} \\ \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \underbrace{\boldsymbol{0}_{L\times L} \cdots \boldsymbol{0}_{L\times L}}_{n-1\text{ blocks}} & & & \underset{\times(t+1)}{\underset{\Rightarrow x_N}{\mathsf{S}}} & \underbrace{\boldsymbol{0}_{L\times L} \cdots \boldsymbol{0}_{L\times L}}_{N-n\text{ blocks}} & & \end{pmatrix}_{NL\times NL}$$
$$= (\boldsymbol{0}_{NL\times(n-1)L}\ \underset{\Rightarrow}{\mathsf{S}}(t+1)\ \boldsymbol{0}_{NL\times(N-n)L})_{NL\times NL}\,,$$
$$\underset{\Rightarrow n}{\mathsf{V}}(t+1) = \begin{pmatrix} \underset{\Rightarrow x_n}{\mathsf{S}}(t+1) & \boldsymbol{0}_{L\times L} & \cdots & \boldsymbol{0}_{L\times L} \\ \boldsymbol{0}_{L\times L} & \underset{\Rightarrow x_n}{\mathsf{S}}(t+1) & \cdots & \boldsymbol{0}_{L\times L} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{L\times L} & \boldsymbol{0}_{L\times L} & \cdots & \underset{\Rightarrow x_n}{\mathsf{S}}(t+1) \end{pmatrix}_{NL\times NL}.$$

Continuing, we can write

$$\underset{\Rightarrow}{\mathsf{S}}^{\text{H}}(t+1)\underset{\Rightarrow n}{\mathsf{U}}(t+1)$$
$$= \underset{\Rightarrow}{\mathsf{S}}^{\text{H}}(t+1)(\boldsymbol{0}_{NL\times(n-1)L}\ \underset{\Rightarrow}{\mathsf{S}}(t+1)\ \boldsymbol{0}_{NL\times(N-n)L})$$
$$= (\boldsymbol{0}_{NL\times(n-1)L}\ \textstyle\sum_{i=1}^{N}\underset{\Rightarrow ii}{\tilde{\mathsf{R}}}(t+1)\ \boldsymbol{0}_{NL\times(N-n)L})\,, \tag{13.72}$$

$$\underset{\Rightarrow}{\mathsf{S}}^{\text{H}}(t+1)\underset{\Rightarrow n}{\mathsf{V}}(t+1)$$
$$= (\underset{\Rightarrow 1n}{\tilde{\mathsf{R}}}(t+1)\ \underset{\Rightarrow 2n}{\tilde{\mathsf{R}}}(t+1)\ \cdots\ \underset{\Rightarrow Nn}{\tilde{\mathsf{R}}}(t+1))\,, \tag{13.73}$$

where

$$\underset{\Rightarrow ij}{\tilde{\mathsf{R}}}(t+1) = \underset{\Rightarrow x_i}{\mathsf{S}}^{\text{H}}(t+1)\underset{\Rightarrow x_j}{\mathsf{S}}(t+1)\,,$$
$$i, j = 1, 2, \cdots, N\,.$$



Amazon EX1010 Page 299

Incorporating (13.72) and (13.73) into (13.70) and (13.71) yields

$$\frac{\partial J_{\mathrm{f}}(t+1)}{\partial \hat{\underset{\rightarrow}{\mathsf{h}}}_n^*(t)} = \underset{\Rightarrow}{\mathsf{S}}^{\mathrm{H}}(t+1)[\underset{\Rightarrow n}{\mathsf{U}}(t+1) - \underset{\Rightarrow n}{\mathsf{V}}(t+1)]\hat{\underset{\rightarrow}{\mathsf{h}}}(t)$$
$$= \Big(-\underset{\Rightarrow 1n}{\tilde{\mathsf{R}}}(t+1) \;\; -\underset{\Rightarrow 2n}{\tilde{\mathsf{R}}}(t+1) \;\; \cdots \;\; \sum_{i\neq n}\underset{\Rightarrow ii}{\tilde{\mathsf{R}}}(t+1) \;\; \cdots \;\; -\underset{\Rightarrow Nn}{\tilde{\mathsf{R}}}(t+1)\Big)\hat{\underset{\rightarrow}{\mathsf{h}}}(t)\,. \tag{13.74}$$

Substituting this gradient estimate into (13.60) and concatenating the $N$ impulse response vectors into a longer one produces a remarkably simple expression for the frequency-domain unconstrained MCLMS algorithm:

$$\hat{\underset{\rightarrow}{\mathsf{h}}}(t+1) = \hat{\underset{\rightarrow}{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t+1)\hat{\underset{\rightarrow}{\mathsf{h}}}(t)\,, \tag{13.75}$$

where

$$\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t+1) = \begin{pmatrix} \sum_{i\neq 1}\underset{\Rightarrow ii}{\tilde{\mathsf{R}}}(t+1) & -\underset{\Rightarrow 21}{\tilde{\mathsf{R}}}(t+1) & \cdots & -\underset{\Rightarrow N1}{\tilde{\mathsf{R}}}(t+1) \\ -\underset{\Rightarrow 12}{\tilde{\mathsf{R}}}(t+1) & \sum_{i\neq 2}\underset{\Rightarrow ii}{\tilde{\mathsf{R}}}(t+1) & \cdots & -\underset{\Rightarrow N2}{\tilde{\mathsf{R}}}(t+1) \\ \vdots & \vdots & \ddots & \vdots \\ -\underset{\Rightarrow 1N}{\tilde{\mathsf{R}}}(t+1) & -\underset{\Rightarrow 2N}{\tilde{\mathsf{R}}}(t+1) & \cdots & \sum_{i\neq N}\underset{\Rightarrow ii}{\tilde{\mathsf{R}}}(t+1) \end{pmatrix}.$$

The update equation of the frequency-domain, unit-norm-constrained MCLMS algorithm is then expressed as

$$\hat{\underset{\rightarrow}{\mathsf{h}}}(t+1) = \frac{\hat{\underset{\rightarrow}{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t+1)\hat{\underset{\rightarrow}{\mathsf{h}}}(t)}{\sqrt{L}\left\|\hat{\underset{\rightarrow}{\mathsf{h}}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t+1)\hat{\underset{\rightarrow}{\mathsf{h}}}(t)\right\|}\,. \tag{13.76}$$

Subtracting the true channel impulse responses $\underset{\rightarrow}{\mathsf{h}}$ from both sides of (13.75), we get the evolution of the misalignment in the frequency domain:

$$\delta\underset{\rightarrow}{\mathsf{h}}(t+1) = \delta\underset{\rightarrow}{\mathsf{h}}(t) - \mu_{\mathrm{f}}\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t+1)\hat{\underset{\rightarrow}{\mathsf{h}}}(t)\,, \tag{13.77}$$

where

$$\delta\underset{\rightarrow}{\mathsf{h}}(t) = \hat{\underset{\rightarrow}{\mathsf{h}}}(t) - \underset{\rightarrow}{\mathsf{h}}\,.$$

Taking the expectation of (13.77) and invoking the independence assumption [13.17], i. e., that the channel output $\boldsymbol{x}_n(t+1)$, $n = 1, 2, \cdots, N$, and the filter coefficients $\hat{\underset{\rightarrow}{\mathsf{h}}}(t)$ are independent, we have

$$E\{\delta\underset{\rightarrow}{\mathsf{h}}(t+1)\} = E\{\delta\underset{\rightarrow}{\mathsf{h}}(t)\} - \mu_{\mathrm{f}}\underset{\Rightarrow}{\mathsf{R}}E\{\hat{\underset{\rightarrow}{\mathsf{h}}}(t)\}\,, \tag{13.78}$$

where

$$\underset{\Rightarrow}{\mathsf{R}} = E\{\underset{\Rightarrow}{\tilde{\mathsf{R}}}(t)\}\,.$$

Recall that the gradient of the mean-square error is zero for the desired solution, i. e.,

$$\frac{\partial J_{\mathrm{f}}}{\partial \underset{\rightarrow}{\mathsf{h}}} = \underset{\Rightarrow}{\mathsf{R}}\underset{\rightarrow}{\mathsf{h}} = \mathbf{0}\,,$$

using which (13.78) can be written as

$$E\{\delta\underset{\rightarrow}{\mathsf{h}}(t+1)\} = (\boldsymbol{I}_{NL\times NL} - \mu_{\mathrm{f}}\underset{\Rightarrow}{\mathsf{R}})E\{\delta\underset{\rightarrow}{\mathsf{h}}(t)\}\,. \tag{13.79}$$

Therefore, if the step size satisfies

$$0 < \mu_{\mathrm{f}} < \frac{2}{\lambda_{\max}}\,, \tag{13.80}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\underset{\Rightarrow}{\mathsf{R}}$, then the misalignment mean $E\left\{\delta\underset{\rightarrow}{\mathsf{h}}(t)\right\}$ converges to zero and the estimated filter coefficients converge in the mean to the desired solutions.

### 13.6.2 Frequency-Domain Normalized Multichannel LMS Algorithm

The *unnormalized* frequency-domain MCLMS algorithm developed in the previous section is certainly computationally efficient. However, its convergence is still unsatisfactorily slow because of nonuniform convergence rates of the filter coefficients and cross-coupling between them. In this section, Newton's method will first be used and then necessary simplifications will be investigated to develop a frequency-domain *normalized* MCLMS algorithm for which the eigenvalue disparity is reduced and the convergence is accelerated.

To apply Newton's method, we need to evaluate the Hessian matrix of $J_{\mathrm{f}}(t+1)$ with respect to the filter coefficients, which can be computed by taking the row gradient of (13.70)

$$\underset{\Rightarrow n}{\mathsf{T}}(t+1) = \frac{\partial}{\partial \hat{\underset{\rightarrow}{\mathsf{h}}}_n^{\mathrm{T}}(t)}\left(\frac{\partial J_{\mathrm{f}}(t+1)}{\partial \hat{\underset{\rightarrow}{\mathsf{h}}}_n^*(t)}\right)$$
$$= \frac{\partial}{\partial \hat{\underset{\rightarrow}{\mathsf{h}}}_n^{\mathrm{T}}(t)}[\underset{\Rightarrow}{\mathsf{S}}^{\mathrm{H}}(t+1)\underset{\rightarrow n}{\mathsf{e}}(t+1)]$$
$$= \underset{\Rightarrow}{\mathsf{S}}^{\mathrm{H}}(t+1)\frac{\partial \underset{\rightarrow n}{\mathsf{e}}(t+1)}{\partial \hat{\underset{\rightarrow}{\mathsf{h}}}_n(t)}$$
$$= \sum_{i=1,i\neq n}^{N}\underset{\Rightarrow x_i}{\mathsf{S}}^{\mathrm{H}}(t+1)\underset{\Rightarrow x_i}{\mathsf{S}}(t+1)\,. \tag{13.81}$$



Amazon EX1010 Page 300

Then the filter coefficients will be updated as follows

$$\begin{aligned}\underset{\rightarrow n}{\hat{\mathsf{h}}}(t+1) &= \underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \rho_{\mathrm{f}} \underset{\Rightarrow n}{\mathsf{T}}^{-1}(t+1) \\ &\quad \times \underset{\Rightarrow}{\mathsf{S}}^{\mathrm{H}}(t+1) \underset{\rightarrow n}{\mathsf{e}}(t+1) \\ &= \underset{\rightarrow n}{\hat{\mathsf{h}}}(t) - \rho_{\mathrm{f}} \underset{\Rightarrow n}{\mathsf{T}}^{-1}(t+1) \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \\ &\quad \cdot \sum_{i=1}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\Rightarrow 2L\times L}{\mathsf{W}^{01}} \underset{\rightarrow in}{\mathsf{e}}(t+1)\,, \end{aligned} \tag{13.82}$$

where $\rho_{\mathrm{f}}$ is a new step size. Multiplying (13.82) from the left by $\underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}$ yields

$$\begin{aligned}\underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t+1) &= \underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t) - \rho_{\mathrm{f}} \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} \underset{\Rightarrow n}{\mathsf{T}}^{-1}(t+1) \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \\ &\quad \cdot \sum_{i=1}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\rightarrow in}{\mathsf{e}}^{01}(t+1)\,, \end{aligned} \tag{13.83}$$

where the symbols

$$\begin{aligned}\underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t) &= \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} \underset{\rightarrow n}{\hat{\mathsf{h}}}(t) = \boldsymbol{F}_{2L\times 2L}\left[\hat{\boldsymbol{h}}_n^{\mathrm{T}}(t)\ \ \boldsymbol{0}_{1\times L}\right]^{\mathrm{T}}, \\ \underset{\rightarrow in}{\mathsf{e}}^{01}(t+1) &= \underset{\Rightarrow 2L\times L}{\mathsf{W}^{01}} \underset{\rightarrow in}{\mathsf{e}}(t+1) \\ &= \boldsymbol{F}_{2L\times 2L}\left[\boldsymbol{0}_{1\times L}\ \ \boldsymbol{e}_{in}^{\mathrm{T}}(t+1)\right]^{\mathrm{T}}, \end{aligned}$$

are shown in Table 13.4.

The frequency-domain Newton algorithm is able to converge quickly, but unfortunately the matrix $\underset{\Rightarrow n}{\mathsf{T}}(t+1)$ is not diagonal and finding its inverse is computationally demanding, particularly when $L$ is large. It is desirable to reduce the complexity by approximation, which leads to the normalized frequency-domain MCLMS algorithm [13.24].

Let us expand the product in (13.81) using the definition (13.68) to find:

$$\begin{aligned}&\underset{\Rightarrow x_i}{\mathsf{S}}^{\mathrm{H}}(t+1) \underset{\Rightarrow x_i}{\mathsf{S}}(t+1) \\ &= \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{01}} \underset{\Rightarrow x_i}{\mathsf{D}}(t+1) \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}\,, \end{aligned} \tag{13.84}$$

where

$$\begin{aligned}\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{01}} &= \underset{\Rightarrow 2L\times L}{\mathsf{W}^{01}} \underset{\Rightarrow L\times 2L}{\mathsf{W}^{01}} \\ &= \boldsymbol{F}_{2L\times 2L} \begin{pmatrix} \boldsymbol{0}_{L\times L} & \boldsymbol{0}_{L\times L} \\ \boldsymbol{0}_{L\times L} & \boldsymbol{I}_{L\times L} \end{pmatrix} \boldsymbol{F}_{2L\times 2L}^{-1}\,, \end{aligned}$$

whose elements on the main diagonal dominate, as shown in [13.25]. When $L$ is large, $2\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{01}}$ can be well approximated by the identity matrix

$$2\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{01}} \approx \boldsymbol{I}_{2L\times 2L}\,. \tag{13.85}$$

Therefore, (13.84) becomes

$$\begin{aligned}&\underset{\Rightarrow x_i}{\mathsf{S}}^{\mathrm{H}}(t+1) \underset{\Rightarrow x_i}{\mathsf{S}}(t+1) \\ &\approx \frac{1}{2} \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\Rightarrow x_i}{\mathsf{D}}(t+1) \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}\,, \end{aligned} \tag{13.86}$$

and (13.81) becomes

$$\begin{aligned}\underset{\Rightarrow n}{\mathsf{T}}(t+1) &\approx \frac{1}{2} \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \left[\sum_{i=1,i\neq n}^{N} \underset{\Rightarrow x_i}{\mathsf{P}}(t+1)\right] \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} \\ &= \frac{1}{2} \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow \not{n}}{\mathsf{P}}(t+1) \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}\,, \end{aligned} \tag{13.87}$$

where

$$\begin{aligned}&\underset{\Rightarrow x_i}{\mathsf{P}}(t+1) = \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\Rightarrow x_i}{\mathsf{D}}(t+1)\,, \\ &\underset{\Rightarrow \not{n}}{\mathsf{P}}(t+1) = \sum_{i=1,i\neq n}^{N} \underset{\Rightarrow x_i}{\mathsf{P}}(t+1)\,, \quad n = 1, 2, \cdots, N\,, \end{aligned}$$

which are diagonal matrices.

There is a useful relation between the inverses of the $\underset{\Rightarrow n}{\mathsf{T}}(t+1)$ and $\underset{\Rightarrow \not{n}}{\mathsf{P}}(t+1)$ matrices given by [13.25]

$$\underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} \underset{\Rightarrow n}{\mathsf{T}}^{-1}(t+1) \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}} = 2\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow \not{n}}{\mathsf{P}}^{-1}(t+1)\,, \tag{13.88}$$

where

$$\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{10}} = \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} \underset{\Rightarrow L\times 2L}{\mathsf{W}^{10}}\,.$$

This relation can be justified by post-multiplying both sides of the expression by $\underset{\Rightarrow \not{n}}{\mathsf{P}}(t+1) \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}$, using (13.87), and recognizing that

$$\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}} = \underset{\Rightarrow 2L\times L}{\mathsf{W}^{10}}\,.$$

Substituting (13.88) into (13.83) produces

$$\begin{aligned}\underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t+1) &= \underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t) - 2\rho_{\mathrm{f}} \underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{10}} \underset{\Rightarrow \not{n}}{\mathsf{P}}^{-1}(t+1) \\ &\quad \cdot \sum_{i=1}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\rightarrow in}{\mathsf{e}}^{01}(t+1)\,. \end{aligned} \tag{13.89}$$

If the matrix $2\underset{\Rightarrow 2L\times 2L}{\mathsf{W}^{10}}$ is approximated by the identity matrix similar to (13.85), we finally deduce the the frequency-domain *normalized* MCLMS (FNMCLMS) algorithm [13.24]:

$$\begin{aligned}\underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t+1) &= \underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t) - \rho_{\mathrm{f}} \underset{\Rightarrow \not{n}}{\mathsf{P}}^{-1}(t+1) \\ &\quad \cdot \sum_{i=1}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\rightarrow in}{\mathsf{e}}^{01}(t+1)\,, \\ &\quad n = 1, 2, \cdots, N\,. \end{aligned} \tag{13.90}$$



Amazon EX1010 Page 301

An interesting interpretation follows if we compare the frequency-domain normalized (13.90) and unnormalized (13.60) MCLMS algorithms. In the frequency-domain unnormalized MCLMS algorithm, the correction that is applied to $\underset{\rightharpoonup n}{\hat{\mathsf{h}}}(t)$ is approximately proportional to the power spectrum $\underset{\Rightarrow \eta'}{\mathsf{P}}(t+1)$ of the multiple channel outputs; this can be seen from (13.60) and the definition of $\underset{\Rightarrow \eta'}{\mathsf{P}}(t+1)$, making the approximation $2\underset{\Rightarrow 2L\times 2L}{\mathsf{W}}^{01} \approx \underset{\Rightarrow \eta'}{\boldsymbol{I}_{2L\times 2L}}$. When the magnitudes of the channel outputs are large, the gradient noise amplification can be evident. With the normalization of the correction by $\underset{\Rightarrow \eta'}{\mathsf{P}}(t+1)$ as performed in the normalized FMCLMS algorithm, this noise amplification problem is diminished and the variability of the convergence rates due to

**Table 13.5** The frequency-domain normalized multichannel LMS (FNMCLMS) adaptive algorithm for the blind identification of an SIMO FIR system

Parameters:

$\hat{\boldsymbol{h}} = [\hat{\boldsymbol{h}}_1^{\mathrm{T}} \;\; \hat{\boldsymbol{h}}_2^{\mathrm{T}} \;\; \cdots \;\; \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}$, model filter

$\rho_{\mathrm{f}} > 0$, step size

$\lambda_{\mathrm{p}} = [1 - 1/(3L)]^L$, forgetting factor

$\delta > 0$, regularization factor

Initialization:

$\hat{\boldsymbol{h}}_n(0) = [1 \;\; 0 \;\; \cdots \;\; 0]^{\mathrm{T}}$, $n = 1, 2, \cdots, N$

$\hat{\boldsymbol{h}}(0) = \hat{\boldsymbol{h}}(0)/\sqrt{N}$ (normalization)

Computation:

For $t = 0, 1, 2, \cdots$, compute

(a) $\underset{\rightharpoonup n}{\hat{\mathsf{h}}}^{10}(t) = \mathrm{FFT}_{2L}\{[\hat{\boldsymbol{h}}_n^{\mathrm{T}}(t) \;\; \boldsymbol{0}_{1\times L}]^{\mathrm{T}}\}$, $n = 1, 2, \cdots, N$

(b) $\boldsymbol{x}_n(t+1)_{2L\times 1} = [x_n(tL) \;\; x_n(tL+1) \;\; \cdots \;\; x_n(tL+2L-1)]^{\mathrm{T}}$

(c) $\underset{\rightharpoonup n}{\mathsf{x}}(t+1)_{2L\times 1} = \mathrm{FFT}_{2L}[\boldsymbol{x}_n(t+1)_{2L\times 1}]$

(d) $\underset{\rightharpoonup ij}{\mathsf{e}}(t+1)_{2L\times 1} = \underset{\rightharpoonup i}{\mathsf{x}}(t+1)_{2L\times 1} \odot \underset{\rightharpoonup j}{\hat{\mathsf{h}}}^{10}(t) - \underset{\rightharpoonup j}{\mathsf{x}}(t+1)_{2L\times 1} \odot \underset{\rightharpoonup i}{\hat{\mathsf{h}}}^{10}(t)$, $i, j = 1, 2, \cdots, N$

(e) $\boldsymbol{e}_{ij}(t+1)_{2L\times 1} = \mathrm{IFFT}_{2L}\{\underset{\rightharpoonup ij}{\mathsf{e}}(t+1)_{2L\times 1}\}$

(f) Take the *last* $L$ elements of $\boldsymbol{e}_{ij}(t+1)_{2L\times 1}$ to form $\boldsymbol{e}_{ij}(t+1)$

(g) $\underset{\rightharpoonup ij}{\mathsf{e}}^{01}(t+1) = \mathrm{FFT}_{2L}\{[\boldsymbol{0}_{1\times L} \;\; \boldsymbol{e}_{ij}^{\mathrm{T}}(t+1)]^{\mathrm{T}}\}$

(h) $\underset{\rightharpoonup \eta'}{\tilde{\mathsf{p}}}(t+1) = \sum_{i=1, i\neq n}^{N} \underset{\rightharpoonup i}{\mathsf{x}}^*(t+1)_{2L\times 1} \odot \underset{\rightharpoonup i}{\mathsf{x}}(t+1)_{2L\times 1}$

(i) $\underset{\rightharpoonup \eta'}{\mathsf{p}}(t+1) = \begin{cases} \underset{\rightharpoonup \eta'}{\tilde{\mathsf{p}}}(t+1), & t = 0 \\ \lambda_{\mathrm{p}} \underset{\rightharpoonup \eta'}{\mathsf{p}}(t) + (1-\lambda_{\mathrm{p}}) \underset{\rightharpoonup \eta'}{\tilde{\mathsf{p}}}(t+1), & t > 0 \end{cases}$

(j) Take the element-by-element reciprocal of $[\underset{\rightharpoonup \eta'}{\mathsf{p}}(t+1) + \delta \boldsymbol{1}_{2L\times 1}]$ to form $\underset{\rightharpoonup \eta'}{\mathsf{p}}^{-1}(t+1)$

(k) $\triangle \underset{\rightharpoonup n}{\hat{\mathsf{h}}}(t)_{2L\times 1} = \rho_{\mathrm{f}} \underset{\rightharpoonup \eta'}{\mathsf{p}}^{-1}(t+1) \odot \left[ \sum_{i=1}^{N} \underset{\rightharpoonup i}{\mathsf{x}}^*(t+1)_{2L\times 1} \odot \underset{\rightharpoonup in}{\mathsf{e}}^{01}(t+1) \right]$

(l) $\triangle \hat{\boldsymbol{h}}_n(t)_{2L\times 1} = \mathrm{IFFT}_{2L}\{\triangle \underset{\rightharpoonup n}{\hat{\mathsf{h}}}(t)_{2L\times 1}\}$

(m) Take the *first* $L$ elements of $\triangle \hat{\boldsymbol{h}}_n(t)_{2L\times 1}$ to form $\triangle \hat{\boldsymbol{h}}_n(t)$

(n) $\hat{\boldsymbol{h}}_n(t+1) = \dfrac{\hat{\boldsymbol{h}}_n(t) - \triangle \hat{\boldsymbol{h}}_n(t)}{\left\| \hat{\boldsymbol{h}}_n(t) - \triangle \hat{\boldsymbol{h}}_n(t) \right\|}$, $n = 1, 2, \cdots, N$



Amazon EX1010 Page 302

the change of signal level is eliminated. In order to estimate a more-stable power spectrum, a recursive scheme is employed in practice [13.24]:

$$\underset{\Rightarrow n'}{\mathsf{P}}(t+1) = \lambda_{\mathrm{p}} \underset{\Rightarrow n'}{\mathsf{P}}(t) + (1-\lambda_{\mathrm{p}}) \cdot \sum_{i=1, i\neq n}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\Rightarrow x_i}{\mathsf{D}}(t+1) \,, \quad n = 1, 2, \cdots, N \,, \tag{13.91}$$

where $\lambda_{\mathrm{p}}$ is a forgetting factor that may appropriately be set as

$$\lambda_{\mathrm{p}} = \left(1 - \frac{1}{3L}\right)^{L}$$

for the FNMCLMS algorithm. Although the FNMCLMS algorithm overcomes the problem of noise amplification, we are now faced with a similar problem that occurs when the channel outputs becomes too small. An alternative approach, therefore, is to insert a positive number $\delta$ into the normalization, which leads to the following modification to the FNMCLMS algorithm:

$$\underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t+1) = \underset{\rightarrow n}{\hat{\mathsf{h}}}^{10}(t) - \rho_{\mathrm{f}} [\underset{\Rightarrow n'}{\mathsf{P}}(t+1) + \delta \boldsymbol{I}_{2L\times 2L}]^{-1} \cdot \sum_{i=1}^{N} \underset{\Rightarrow x_i}{\mathsf{D}}^{*}(t+1) \underset{\rightarrow in}{\mathsf{e}}^{01}(t+1) \,, \quad n = 1, 2, \cdots, N \,. \tag{13.92}$$

From a computational point of view, the modified FNMCLMS algorithm can easily be implemented even for a real-time application since the normalization matrix $\underset{\Rightarrow n'}{\mathsf{P}}(t+1) + \delta \boldsymbol{I}_{2L\times 2L}$ is diagonal and it is straightforward to find its inverse.

The FNMCLMS algorithm is summarized in Table 13.5.

## 13.7 Adaptive Multichannel Exponentiated Gradient Algorithm

We have developed in the previous Sections a number of adaptive algorithms for blind identification of SIMO FIR systems. In fact, all these algorithms are based on the classical stochastic gradient and their update rules do not discriminate against the model filter coefficients with significantly distinct magnitudes. Intuitively, however, it seems possible to develop a better adaptive blind SIMO identification algorithm by taking advantage of the sparseness in acoustic impulse responses.

It has only very recently been recognized that the sparseness of an impulse response can be exploited in adaptive algorithms to accelerate their initial convergence and improve their tracking performance. The proportionate normalized LMS (PNLMS) proposed by *Duttweiler* [13.26] was perhaps one of the first such algorithms, which was introduced for network echo cancelation. It was shown in [13.27] that the PNLMS is an approximation of the so-called exponentiated gradient algorithm with positive and negative weights (EG± algorithm). EG± was proposed by *Kivinen* and *Warmuth* in the context of computational learning theory [13.28]. In [13.29], a general expression of the mean-squared error (MSE) was derived for the EG± algorithm, illustrating that for sparse impulse responses, the EG± algorithm, like the PNLMS, converges more quickly than the LMS for a given asymptotic MSE. Both the PNLMS and EG± algorithms were developed for identifying a single channel with reference signals [13.27]. In this section, we will investigate how to apply the concept of exponentiated gradient for blind identification of sparse acoustic SIMO systems and review the multichannel EG± (MCEG±) algorithm [13.30].

We begin with the introduction of a new way to develop adaptive algorithms for blind SIMO identification, which is different from the gradient descent perspective described in Sect. 13.4. In addition to the unconstrained, a priori error signal (13.11), we define an a posteriori error:

$$\varepsilon_{ij}(k+1) = \boldsymbol{x}_i^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k+1) - \boldsymbol{x}_j^{\mathrm{T}}(k+1)\hat{\boldsymbol{h}}_j(k+1) \,, \quad i, j = 1, 2, \cdots, N \,. \tag{13.93}$$

Then an adaptive algorithm that adjusts the new model filter $\hat{\boldsymbol{h}}(k+1)$ from the old one $\hat{\boldsymbol{h}}(k)$ can be derived by minimizing the function

$$J[\hat{\boldsymbol{h}}(k+1)] = d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)] + \eta \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \varepsilon_{ij}^2(k+1) \,, \tag{13.94}$$

where $d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]$ is a measure of the distance from the old to the new model filter, and $\eta$ is a positive constant. The magnitude of $\eta$ represents the importance of



Amazon EX1010 Page 303

correctness compared to the importance of conservativeness [13.28]. If $\eta$ is very small, minimizing $J[\hat{\boldsymbol{h}}(k+1)]$ is very similar to minimizing $d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]$, so that the algorithm makes very small updates. On the other hand, if $\eta$ is very large, the minimization of $J[\hat{\boldsymbol{h}}(k+1)]$ is almost equivalent to minimizing $d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]$ subject to the constraint $\varepsilon_{ij}(k+1)=0, \forall i, j$.

To minimize $J[\hat{\boldsymbol{h}}(k+1)]$, we need to set the partial derivative $\partial J[\hat{\boldsymbol{h}}(k+1)]/\partial\hat{\boldsymbol{h}}(k+1)$ to zero. Hence, the vector of channel impulse responses $\hat{\boldsymbol{h}}(k+1)$ will be found by solving

$$\frac{\partial d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]}{\partial\hat{\boldsymbol{h}}(k+1)} + \eta\frac{\partial}{\partial\hat{\boldsymbol{h}}(k+1)}\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\varepsilon_{ij}^2(k+1)\right] = \boldsymbol{0}\,. \quad (13.95)$$

Similar to (13.22), it can be shown that

$$\frac{\partial}{\partial\hat{\boldsymbol{h}}(k+1)}\left[\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\varepsilon_{ij}^2(k+1)\right] = 2\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k+1)\,. \quad (13.96)$$

Then (13.95) becomes

$$\frac{\partial d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]}{\partial\hat{\boldsymbol{h}}(k+1)} + 2\eta\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k+1) = \boldsymbol{0}\,. \quad (13.97)$$

Solving (13.97) is in general very difficult. However, if the new weight vector $\hat{\boldsymbol{h}}(k+1)$ is close to the old weight vector $\hat{\boldsymbol{h}}(k)$, replacing $\hat{\boldsymbol{h}}(k+1)$ in the second term on the left-hand side of (13.97) with $\hat{\boldsymbol{h}}(k)$ is a reasonable approximation and the resulting equation

$$\frac{\partial d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]}{\partial\hat{\boldsymbol{h}}(k+1)} + 2\eta\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) = \boldsymbol{0} \quad (13.98)$$

is much easier to solve for all distance measures $d$.

The unconstrained multichannel LMS algorithm is easily obtained from (13.98) by using the squared Euclidean distance

$$d_{\mathrm{E}}[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)] = \left\|\hat{\boldsymbol{h}}(k+1) - \hat{\boldsymbol{h}}(k)\right\|_2^2\,. \quad (13.99)$$

If the unconstrained a posteriori error in (13.94) is replaced by a *constrained* a posteriori error signal, then the constrained multichannel LMS algorithm (13.24) can be deduced.

The exponentiated gradient (EG) algorithm with *positive* weights results from using the *relative entropy*, also known as the *Kullback–Leibler divergence*, for $d$:

$$d_{\mathrm{re}}[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)] = \sum_{l=0}^{NL-1}\hat{h}_l(k+1)\ln\frac{\hat{h}_l(k+1)}{\hat{h}_l(k)}\,, \quad (13.100)$$

with the constraint $\sum_{l=0}^{NL-1}\hat{h}_l(k+1)=u$, where $\hat{h}_l(k+1)$ is the $l$-th ($l=0, 1, \cdots, NL-1$) tap of the model filter $\hat{\boldsymbol{h}}(k+1)$ at time $k+1$, and $u$ is a positive constant (note that this is different from the unit-norm constraint used in the constrained multichannel LMS algorithm). We slightly modify the function (13.94) as follows

$$J[\hat{\boldsymbol{h}}(k+1)] = d[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)] + \frac{\eta}{u}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\varepsilon_{ij}^2(k+1)\,. \quad (13.101)$$

As a result, (13.98) becomes

$$\frac{\partial d_{\mathrm{re}}[\hat{\boldsymbol{h}}(k+1), \hat{\boldsymbol{h}}(k)]}{\partial\hat{\boldsymbol{h}}(k+1)} + \frac{2\eta}{u}\tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k) + \kappa\boldsymbol{1} = \boldsymbol{0}\,, \quad (13.102)$$

where $\kappa$ is a Lagrange multiplier and $\boldsymbol{1}=(1\ 1\ \cdots\ 1)^{\mathrm{T}}$ is a vector of ones. Substituting (13.100) into (13.102) and solving for $\hat{\boldsymbol{h}}(k+1)$ leads to the multichannel EG algorithm with positive weights (MCEG+):

$$\hat{h}_l(k+1) = u\frac{\hat{h}_l(k)r_l(n+1)}{\sum_{i=0}^{NL-1}\hat{h}_i(k)r_i(k+1)}, \quad l=0, 1, \cdots, NL-1\,, \quad (13.103)$$

where

$$r_l(k+1) = \exp\left[-\frac{2\eta}{u}g_l(k+1)\right]\,,$$

and $g_l(k+1)$ is the $l$-th element of the gradient vector

$$\boldsymbol{g}(k+1) = \tilde{\boldsymbol{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)\,. \quad (13.104)$$

We can clearly see the motivation for the EG algorithms by ignoring the normalization in (13.103) and taking the logarithm, which shows that the log weights use the same update as the MCLMS algorithm. Alternatively, this can be interpreted as exponentiating the update, hence the name EG. This has the effect of assigning larger relative updates to larger weights, thereby deemphasizing the effect of smaller weights. This is qualitatively similar to the idea that leads to the PNLMS algorithm, which makes the update *proportional* to the magnitude of the weight. This type of behavior is desirable for sparse impulse responses where small weights



Amazon EX1010 Page 304

do not contribute significantly to the *mean* solution but introduce an undesirable noise-like *variance*.

Since the MCEG+ algorithm works only for SIMO systems with positive channel impulse responses, it is useless for blind identification of an acoustic SIMO system. For an acoustic SIMO system with both positive and negative filter coefficients, we can decompose the model filter $\hat{\boldsymbol{h}}(k)$ into two components $\hat{\boldsymbol{h}}^+(k)$ and $\hat{\boldsymbol{h}}^-(k)$ (both with positive coefficients) such that $\hat{\boldsymbol{h}}(k)=\hat{\boldsymbol{h}}^+(k)-\hat{\boldsymbol{h}}^-(k)$. Then the function (13.101) becomes:

$$\begin{aligned}&J[\hat{\boldsymbol{h}}^+(k+1),\hat{\boldsymbol{h}}^-(k+1)]\\&=d_{\mathrm{re}}[\hat{\boldsymbol{h}}^+(k+1),\hat{\boldsymbol{h}}^+(k)]\\&\quad+d_{\mathrm{re}}[\hat{\boldsymbol{h}}^-(k+1),\hat{\boldsymbol{h}}^-(k)]\\&\quad+\frac{\eta}{u_1+u_2}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\varepsilon_{ij}^2(k+1)\,,\end{aligned}\tag{13.105}$$

where $u_1$ and $u_2$ are two positive constants. Since $\hat{\boldsymbol{h}}(k+1)=\boldsymbol{0}$ is an undesired solution, it is necessary to ensure that $\hat{\boldsymbol{h}}^+(k+1)$ and $\hat{\boldsymbol{h}}^-(k+1)$ are not equal to each other on initialization and throughout the process of adaptation. Among the many methods that can be used to do this, we introduce the following constraint proposed in [13.30]:

$$\sum_{l=0}^{NL-1}\hat{h}_l^+(k+1)=u_1\,,\qquad\sum_{l=0}^{NL-1}\hat{h}_l^-(k+1)=u_2\,,\tag{13.106}$$

where $u_1\neq u_2$. Using this constraint and taking derivatives of (13.105) with respect to $\hat{\boldsymbol{h}}^+(k+1)$ and $\hat{\boldsymbol{h}}^-(k+1)$ respectively produces:

$$\ln\frac{\hat{h}_l^+(k+1)}{\hat{h}_l^+(k)}+1+\frac{2\eta}{u_1+u_2}g_l(k+1)+\kappa_1=0\,,\tag{13.107}$$

$$\ln\frac{\hat{h}_l^-(k+1)}{\hat{h}_l^-(k)}+1-\frac{2\eta}{u_1+u_2}g_l(k+1)+\kappa_2=0\,,\tag{13.108}$$

where $\kappa_1$ and $\kappa_2$ are two Lagrange multipliers. Solving (13.107) and (13.108) for $\hat{h}_l^+(k+1)$ and $\hat{h}_l^-(k+1)$,

**Table 13.6** The multichannel exponentiated gradient algorithm with positive and negative weights (MCEG±) for the blind identification of a sparse SIMO FIR system

| | |
|---|---|
| Parameters: | $\hat{\boldsymbol{h}}=[\hat{\boldsymbol{h}}_1^{\mathrm{T}}\ \hat{\boldsymbol{h}}_2^{\mathrm{T}}\ \cdots\ \hat{\boldsymbol{h}}_N^{\mathrm{T}}]^{\mathrm{T}}=\hat{\boldsymbol{h}}^+-\hat{\boldsymbol{h}}^-$, model filter |
| | $\eta>0$, positive constant to balance correctness and conservativeness |
| | $u_1>0\,,u_2>0\,,$ and $u_1\neq u_2$ |
| Initialization: | $\hat{\boldsymbol{h}}_n^+(0)=[u_1/N\ \ 0\ \cdots\ 0]^{\mathrm{T}}\quad n=1,2,\cdots,N$ |
| | $\hat{\boldsymbol{h}}_n^-(0)=[u_2/N\ \ 0\ \cdots\ 0]^{\mathrm{T}}$ |
| | $\hat{\boldsymbol{h}}(0)=\hat{\boldsymbol{h}}^+(0)-\hat{\boldsymbol{h}}^-(0)$ |
| Computation: | For $k=0,1,2,\cdots,$ compute |
| | (a) Construct the matrix $\tilde{\mathbf{R}}_{x+}(k+1)$ following Sect. 13.4.2 |
| | (b) $\boldsymbol{g}(k+1)=\tilde{\mathbf{R}}_{x+}(k+1)\hat{\boldsymbol{h}}(k)$ |
| | (c) $r_l^+(k+1)=\exp\left[-\frac{2\eta}{u_1+u_2}g_l(k+1)\right]$ |
| | $r_l^-(k+1)=\frac{1}{r_l^+(k+1)}\,,\quad l=0,1,\cdots,NL-1$ |
| | (d) $\hat{\boldsymbol{h}}^+(k+1)=\frac{u_1[\hat{\boldsymbol{h}}^+(k)\odot\boldsymbol{r}^+(k+1)]}{[\hat{\boldsymbol{h}}^+(k)]^{\mathrm{T}}\boldsymbol{r}^+(k+1)}$ |
| | $\hat{\boldsymbol{h}}^-(k+1)=\frac{u_2[\hat{\boldsymbol{h}}^-(k)\odot\boldsymbol{r}^-(k+1)]}{\left[\hat{\boldsymbol{h}}^-(k)\right]^{\mathrm{T}}\boldsymbol{r}^-(k+1)}$ |
| | (e) $\hat{\boldsymbol{h}}(k+1)=\hat{\boldsymbol{h}}^+(k+1)-\hat{\boldsymbol{h}}^-(k+1)$ |



Amazon EX1010 Page 305

respectively, leads to the MCEG± algorithm:

$$\hat{h}_l^+(k+1) = u_1 \frac{\hat{h}_l^+(k) r_l^+(k+1)}{\sum_{i=0}^{NL-1} \hat{h}_i^+(k) r_i^+(k+1)} , \quad (13.109)$$

$$\hat{h}_l^-(k+1) = u_2 \frac{\hat{h}_l^-(k) r_l^-(k+1)}{\sum_{i=0}^{NL-1} \hat{h}_i^-(k) r_i^-(k+1)} , \quad (13.110)$$

where

$$r_l^+(k+1) = \exp\left[ -\frac{2\eta}{u_1+u_2} g_l(k+1) \right] ,$$

$$r_l^-(k+1) = \exp\left[ \frac{2\eta}{u_1+u_2} g_l(k+1) \right] = \frac{1}{r_l^+(k+1)} ,$$

which is summarized in Table 13.6.

Before wrapping up the development of the MCEG± algorithm, we would like to comment briefly on how to choose $u_1$ and $u_2$ in (13.106) properly. The constraint looks simple but, in practice, determining a proper set of $u_1$ and $u_2$ is actually quite difficult. Let $\boldsymbol{h}^{+0}$ be the non-negative component of $\boldsymbol{h}$, i. e.,

$$h_l^{+0} = \begin{cases} h_l, & h_l \geq 0 \\ 0, & h_l < 0 \end{cases} , \quad l = 0, 1, \cdots, NL-1 , \quad (13.111)$$

and

$$\boldsymbol{h}^{-0} = \boldsymbol{h}^{+0} - \boldsymbol{h} . \quad (13.112)$$

Then a decomposition of $\boldsymbol{h}$ can be expressed as

$$\boldsymbol{h}^+ = \boldsymbol{h}^{+0} + \boldsymbol{c} , \quad \boldsymbol{h}^- = \boldsymbol{h}^{-0} + \boldsymbol{c} , \quad (13.113)$$

where $\boldsymbol{c}$ is a constant vector. Since the coefficients of $\boldsymbol{h}^+$ and $\boldsymbol{h}^-$ are non-negative, $\boldsymbol{c}$ needs to satisfy

$$c_l \geq \max\left( -h_l^{+0}, -h_l^{-0} \right) , \quad l = 0, 1, \cdots, NL-1. \quad (13.114)$$

Therefore, the following values for $u_1$ and $u_2$ are all valid:

$$u_1 = \sum_{l=0}^{NL-1} h_l^+ = \sum_{l=0}^{NL-1} h_l^{+0} + \sum_{l=0}^{NL-1} c_l , \quad (13.115)$$

$$u_2 = \sum_{l=0}^{NL-1} h_l^- = \sum_{l=0}^{NL-1} h_l^{-0} + \sum_{l=0}^{NL-1} c_l . \quad (13.116)$$

## 13.8 Summary

In this chapter we have investigated blind identification of acoustic multichannel systems with emphasis on adaptive implementations. The necessary and sufficient conditions for an acoustic SIMO system to be blindly identifiable using only second-order statistics were explained and the principle of blind SIMO identification presented. We illustrated how to use the cross-relations between different microphone outputs to define an error signal and its corresponding cost function. It was then shown that, by minimizing the cost function, an adaptive algorithm could be developed and the acoustic channel impulse responses would be determined up to a scale. We discussed a number of state-of-the-art adaptive blind SIMO identification algorithms, in both the time and frequency domains, with or without optimal step-size controls. Finally, we also discussed how to take advantage of the sparseness in the acoustic impulse responses to accelerate initial convergence of an adaptive blind identification algorithm for acoustic multichannel systems.

### References


13.1 Y. Sato: A method of self-recovering equalization for multilevel amplitude-modulation, IEEE Trans. Commun. **COM-23**, 679–682 (1975)

13.2 D.N. Godard: Self-recovering equalization and carrier tracking in two-dimensional data communication systems, IEEE Trans. Commun. **COM-28**, 1867–1875 (1980)

13.3 J.R. Treichler, B.G. Agee: A new approach to multipath correction of constant modulus signals, IEEE Trans. Acoust. Speech **ASSP-31**, 459–472 (1983)

13.4 A. Benveniste, M. Goursat: Blind equalizers, IEEE Trans. Commun. **COM-32**, 871–883 (1984)

13.5 J.M. Mendel: Tutorial on higher-order statistics (spectra) in signal processing and system theory:




Amazon EX1010 Page 306

theoretical results and some applications, Proc. IEEE **79**, 278–305 (1991)
13.6 L. Tong, G. Xu, T. Kailath: A new approach to blind identification and equalization of multipath channels, Proc. 25th Asilomar Conf. on Signals, Systems, and Computers, Vol. 2 (1991) pp. 856–860
13.7 H. Liu, G. Xu, L. Tong: A deterministic approach to blind equalization, Proc. 27th Asilomar Conf. on Signals, Systems, and Computers, Vol. 1 (1993) pp. 751–755
13.8 M.I. Gürelli, C.L. Nikias: A new eigenvector-based algorithm for multichannel blind deconvolution of input clolored signals, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Vol. 4 (1993) pp. 448–451
13.9 L.A. Baccala, S. Roy: A new blind time-domain channel identification method based on cyclostationarity, IEEE Signal Process. Lett. **1**, 89–91 (1994)
13.10 G. Xu, H. Liu, L. Tong, T. Kailath: A least-squares approach to blind channel identification, IEEE Trans. Signal Process. **43**, 2982–2993 (1995)
13.11 E. Moulines, P. Duhamel, J.F. Cardoso, S. Mayrargue: Subspace methods for the blind identification of multichannel FIR filters, IEEE Trans. Signal Process. **43**, 516–525 (1995)
13.12 D. Slock: Blind fractionally-spaced equalization, prefect reconstruction filerbanks, and multilinear prediction, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Vol. 4 (1994) pp. 585–588
13.13 Y. Hua: Fast maximum likelihood for blind identification of multiple FIR channels, IEEE Trans. Signal Process. **44**, 661–672 (1996)
13.14 L. Tong, S. Perreau: Multichannel blind identification: from subspace to maximum likelihood methods, Proc. IEEE **86**, 1951–1968 (1998)
13.15 Y. Huang, J. Benesty: Adaptive multi-channel least mean square and Newton algorithms for blind channel identification, Signal Process. **82**, 1127–1138 (2002)
13.16 C. Avendano, J. Benesty, D.R. Morgan: A least squares component normalization approach to blind channel identification, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 4 (1999) pp. 1797–1800
13.17 S. Haykin: *Adaptive Filter Theory*, 4th edn. (Prentice Hall, Upper Saddle River 2002)
13.18 T.K. Moon, W.C. Stirling: *Mathematical Methods and Algorithms* (Prentice Hall, Upper Saddle River 1999)
13.19 H. Chen, X. Cao, J. Zhu: Convergence of stochastic-approximation-based algorithms for blind channel identification, IEEE Trans. Inform. Theory **48**, 1214–1225 (2002)
13.20 Y. Huang, J. Benesty, J. Chen: Optimal step size of the adaptive multichannel LMS algorithm for blind SIMO identification, IEEE Signal Process. Lett. **12**, 173–176 (2005)
13.21 M. Dentino, J. McCool, B. Widrow: Adaptive filtering in the frequency domain, Proc. IEEE **66**, 1658–1659 (1978)
13.22 A.V. Oppenheim, R.W. Schafer: *Discrete-Time Signal Processing* (Prentice Hall, Englewood Cliffs 1989)
13.23 D.H. Brandwood: A complex gradient operator and its application in adaptive array theory, Proc. IEE **130**, 11–16 (1983), Pts. F and H
13.24 Y. Huang, J. Benesty: A class of frequency-domain adaptive approaches to blind multichannel identification, IEEE Trans. Signal Process. **51**, 11–24 (2003)
13.25 J. Benesty, D. Morgan: Frequency-domain adaptive filtering revisited, generalization to the multi-channel case, and application to acoustic echo cancellation, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Vol. 2 (2000) pp. 789–792
13.26 D.L. Duttweiler: Proportionate normalized least-mean-square adaptation in echo cancelers, IEEE Trans. Speech Audio Process. **8**, 508–518 (2000)
13.27 J. Benesty, Y. Huang: The LMS, PNLMS, and exponentiated gradient algorithms, Proc. EUSIPCO (2004) pp. 721–724
13.28 J. Kivinen, M.K. Warmuth: Exponentiated gradient versus gradient descent for linear predictors, Inform. Comput. **132**, 1–64 (1997)
13.29 S.I. Hill, R.C. Williamson: Convergence of exponentiated gradient algorithms, IEEE Trans. Signal Process. **49**, 1208–1215 (2001)
13.30 J. Benesty, Y. Huang, J. Chen: An exponentiated gradient adaptive algorithm for blind identification of sparse SIMO systems, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Vol. 2 (2004) pp. 829–832



Amazon EX1010 Page 307

# Part C Speech Coding

Ed. by W. B. Kleijn


**14 Principles of Speech Coding**
W. B. Kleijn, Stockholm, Sweden

**15 Voice over IP: Speech Transmission over Packet Networks**
J. Skoglund, San Francisco, USA
E. Kozica, Stockholm, Sweden
J. Linden, San Francisco, USA
R. Hagen, Stockholm, Sweden
W. B. Kleijn, Stockholm, Sweden

**16 Low-Bit-Rate Speech Coding**
A. V. McCree, Lexington, USA

**17 Analysis-by-Synthesis Speech Coding**
J.-H. Chen, Irvine, USA
J. Thyssen, Irvine, USA

**18 Perceptual Audio Coding of Speech Signals**
J. Herre, Erlangen, Germany
M. Lutzky, Erlangen, Germany


Amazon EX1010 Page 308

# 14. Principles of Speech Coding

W. B. Kleijn

Speech coding is the art of reducing the bit rate required to describe a speech signal. In this chapter, we discuss the attributes of speech coders as well as the underlying principles that determine their behavior and their architecture. The ubiquitous class of linear-prediction-based coders is used as an illustration. Speech is generally modeled as a sequence of stationary signal segments, each having unique statistics. Segments are encoded using a two-step procedure: (1) find a model describing the speech segment, (2) encode the segment assuming it is generated by the model. We show that the bit allocation for the model (the predictor parameters) is independent of overall rate and of perception, which is consistent with existing experimental results. The modeling of perception is an important aspect of efficient coding and we discuss how various perceptual distortion measures can be integrated into speech coders.

14.1 **The Objective of Speech Coding** ..... 283
14.2 **Speech Coder Attributes** ..... 284
14.2.1 Rate ..... 284
14.2.2 Quality ..... 285
14.2.3 Robustness to Channel Imperfections ..... 285
14.2.4 Delay ..... 286
14.2.5 Computational and Memory Requirements ..... 286
14.3 **A Universal Coder for Speech** ..... 286
14.3.1 Speech Segment as Random Vector 286
14.3.2 Encoding Random Speech Vectors .. 287
14.3.3 A Model of Quantization ..... 288
14.3.4 Coding Speech with a Model Family 289
14.4 **Coding with Autoregressive Models** ..... 293
14.4.1 Spectral-Domain Index of Resolvability ..... 293
14.4.2 A Criterion for Model Selection ..... 294
14.4.3 Bit Allocation for the Model ..... 295
14.4.4 Remarks on Practical Coding ..... 296
14.5 **Distortion Measures and Coding Architecture** ..... 296
14.5.1 Squared Error ..... 297
14.5.2 Masking Models and Squared Error. 298
14.5.3 Auditory Models and Squared Error 299
14.5.4 Distortion Measure and Coding Architecture ..... 301
14.6 **Summary** ..... 302
**References** ..... 303

## 14.1 The Objective of Speech Coding

In modern communication systems, speech is represented by a sequence of bits. The main advantage of this *binary* representation is that it can be recovered exactly (without distortion) from a noisy channel (assuming proper system design), and does not suffer from decreasing quality when transmitted over many transmission legs. In contrast, analog transmission generally results in an increase of distortion with the number of legs.

An acoustic speech signal is inherently analog. Generally, the resulting analog microphone output is converted to a binary representation in a manner consistent with Shannon's sampling theorem. That is, the analog signal is first band-limited using an anti-aliasing filter, and then simultaneously sampled and quantized. The output of the *analog-to-digital (A/D) converter* is a digital speech signal that consists of a sequence of numbers of finite precision, each representing a sample of the band-limited speech signal. Common sampling rates are 8 and 16 kHz, rendering *narrowband speech* and *wideband speech*, respectively, usually with a precision of 16 bits per sample. For the 8 kHz sampling rate a logarithmic 8-bit-per-sample representation is also common.

Amazon EX1010 Page 309

Particularly at the time of the introduction of the binary speech representation, the bit rate produced by the A/D converter was too high for practical applications such as cost-effective mobile communications and secure telephony. A search ensued for more-efficient digital representations. Such representations are possible since the digital speech contains *irrelevancy* (the signal is described with a higher precision than is needed) and *redundancy* (the rate can be decreased without affecting precision). The aim was to trade off computational effort at the transmitter and receiver for the bit rate required for the speech representation. Efficient representations generally involve a *model* and a set of *model parameters*, and sometimes a set of coefficients that form the input to the model. The algorithms used to reduce the required rate are called speech-coding algorithms, or *speech codecs*.

The performance of speech codecs can be measured by a set of properties. The fundamental codec attributes are bit rate, speech quality, quality degradation due to channel errors and packet loss, delay, and computational effort. Good performance for one of the attributes generally leads to lower performance for the others. The interplay between the attributes is governed by the fundamental laws of information theory, the properties of the speech signal, limitations in our knowledge, and limitations of the equipment used.

To design a codec, we must know the desired values for its attributes. A common approach to develop a speech codec is to constrain all attributes but one quantitatively. The design objective is then to optimize the remaining attribute (usually quality or rate) subject to these constraints. A common objective is to maximize the average quality over a given set of channel conditions, given the rate, the delay, and the computational effort.

In this chapter, we attempt to discuss speech coding at a generic level and yet provide information useful for practical coder design and analysis. Section 14.2 describes the basic attributes of a speech codec. Section 14.3 discusses the underlying principles of coding and Sect. 14.4 applies these principles to a commonly used family of linear predictive (autoregressive model-based) coders. Section 14.5 discusses distortion criteria and how they affect the architecture of codecs. Section 14.6 provides a summary of the chapter.

## 14.2 Speech Coder Attributes

The usefulness of a speech coder is determined by its attributes. In this section we describe the most important attributes and the context in which they are relevant in some more detail. The attributes were earlier discussed in [14.1, 2].

### 14.2.1 Rate

The rate of a speech codec is generally measured as the average number of bits per second. For *fixed-rate* coders the bit rate is the same for each coding block, while for *variable-rate* coders it varies over time.

In traditional circuit-switched communication systems, a fixed rate is available for each communication direction. It is then natural to exploit this rate at all times, which has resulted in a large number of standardized fixed-rate speech codecs. In such coders each particular parameter or variable is encoded with the same number of bits for each block. This a priori knowledge of the bit allocation has a significant effect on the structure of the codec. For example, the mapping of the quantization indices to the transmitted codewords is trivial. In more-flexible circuit-switched networks (e.g., modern mobile-phone networks), codecs may have a variable number of modes, each mode having a different fixed rate [14.3,4]. Such codecs with a set of fixed coding rates should not be confused with true variable-rate coders.

In variable-rate coders, the bit allocation within a particular block for the parameters or variable depends on the signal. The bit allocation for a parameter varies with the quantization index and the mapping from the quantization index to the transmitted codeword is performed by means of a table lookup or computation, which can be very complex. The major benefit of variable-rate coding is that it leads to higher coding efficiency than fixed-rate coders because the rate constraint is less strict.

In general, network design evolves towards the facilitation of variable-rate coders. In packet-switched communication systems, both packet rate and size can vary, which naturally leads to variable-rate codecs. While variable-rate codecs are common for audio and video signals, they are not yet commonplace for speech. The requirements of low rates and delays lead to a small packet payload for speech signals. The relatively large packet header size limits the benefits of the low rate and,

Amazon EX1010 Page 310

consequently, the benefit of variable-rate speech coding. However, with the removal of the fixed-rate constraint, it is likely that variable-rate speech codecs will become increasingly common.

### 14.2.2 Quality

To achieve a significant rate reduction, the parameters used to represent the speech signal are generally transmitted at a reduced precision and the reconstructed speech signal is not a perfect copy of the original digital signal. It is therefore important to ensure that its quality meets a certain standard.

In speech coding, we distinguish two applications for quality measures. First, we need to evaluate the *overall quality* of a particular codec. Second, we need a *distortion measure* to decide how to encode each signal block (typically of duration 5–25 ms). The distortion measure is also used during the design of the coder (in the training of its codebooks). Naturally, these quality measures are not unrelated, but in practice their formulation has taken separate paths. Whereas overall quality can be obtained directly from scoring of speech utterances by humans, distortion measures used in coding algorithms have been defined (usually in an ad hoc manner) based on knowledge about the human auditory system.

The only true measure of the overall quality of a speech signal is its rating by humans. Standardized conversational and listening tests have been developed to obtain reliable and repeatable (at least to a certain accuracy) results. For speech coding, listening tests, where a panel of listeners evaluates performance for a given set of utterances, are most common. Commonly used standardized listening tests use either an absolute category rating, where listeners are asked to score an utterance on an absolute scale, or a degradation category rating, where listeners are asked to provide a relative score. The most common overall measure associated with the absolute category rating of speech quality is the mean opinion score (MOS) [14.5]. The MOS is the mean value of a numerical score given to an utterance by a panel of listeners, using a standardized procedure. To reduce the associated cost, subjective measures can be approximated by objective, repeatable algorithms for many practical purposes. Such measures can be helpful in the development of new speech coders. We refer to [14.6–8] and to Chap. 5 for more detail on the subject of overall speech quality.

As a distortion measure for speech segments variants of the squared-error criterion are most commonly used. The squared-error criterion facilitates fast evaluation for coding purposes. Section 14.5 discusses distortion measures in more detail. It is shown that adaptively weighted squared error criteria can be used for a large range of perceptual models.

### 14.2.3 Robustness to Channel Imperfections

Early terrestrial digital communication networks were generally designed to have very low error rates, obviating the need for measures to correct errors for the transmission of speech. In contrast, bit errors and packet loss are inherent in modern communication infrastructures.

Bit errors are common in wireless networks and are generally addressed by introducing channel codes. While the integration of source and channel codes can result in higher performance, this is not commonly used because it results in reduced modularity. Separate source and channel coding is particularly advantageous when a codec is faced with different network environments; different channel codes can then be used for different network conditions.

In packet networks, the open systems interconnection reference (OSI) model [14.9] provides a separation of various communication functionalities into seven layers. A speech coder resides in the application layer, which is the seventh and highest layer. Imperfections in the transmission are removed in both the physical layer (the first layer) and the transport layer (the fourth layer). The physical layer removes *soft* information, which consists of a probability for the allowed symbols, and renders a sequence of bits to the higher layers. Error control normally resides in the transport layer. However, the error control of the transport layer, as specified by the transmission control protocol (TCP) [14.10], and particularly the automatic repeat requests that TCP uses is generally not appropriate for real-time communication of audiovisual data because of delay. TCP is also rarely used for broadcast and multicast applications to reduce the load on the transmitter. Instead, the user datagram protocol (UDP) [14.11] is used, which means that the coded signal is handed up to the higher network layers without error correction. It is possible that in future systems cross-layer interactions will allow the application layer to receive information about the soft information available at the physical layer.

Handing the received coded signal with its defects directly to the application layer allows the usage of both the inherent redundancy in the signal and our knowledge of the perception of distortion by the user. This leads to coding systems that exhibit a graceful degrada-

tion with increasing error rate. We refer to the chapter on voice over internet protocol (IP) for more detail on techniques that lead to robustness against bit errors and packet loss.

### 14.2.4 Delay

From coding theory [14.12], we know that optimal coding performance generally requires a delay in the transfer of the message. Long delays are impractical because they are generally associated with methods with high computational and storage requirements, and because in real-time environments (common for speech) the user does not tolerate a long delay.

Significant delay directly affects the quality of a conversation. Impairment to conversations is measurable at one-way delays as low as 100 ms [14.13], although 200 ms is often considered a useful bound.

Echo is perceivable at delays down to 20 ms [14.14]. Imperfections in the network often lead to so-called network echo. Low-delay codecs have been designed to keep the effect of such echo to a minimum, e.g., [14.15]. However, echo cancelation has become commonplace in communication networks. Moreover, packet networks have an inherent delay that requires echo cancelation even for low-delay speech codecs. Thus, for most applications codecs can be designed without consideration of echo.

In certain applications the user may hear both an acoustic signal and a signal transmitted by a network. Examples are flight control rooms and wireless systems for hearing-impaired persons. In this class of applications, coding delays of less than 10 ms are needed to attain an acceptable overall delay.

### 14.2.5 Computational and Memory Requirements

Economic cost is generally a function of the computational and memory requirements of the coding system. A common measure of computational complexity used in applications is the number of instructions required on a particular silicon device. This is often translated into the number of channels that can be implemented on a single device.

A complicating factor is that speech codecs are commonly implemented on fixed-point signal processing devices. Implementation on a fixed-point device generally takes significant development effort beyond that of the development of the floating-point algorithm.

It is well known that vector quantization facilitates an optimal rate versus quality trade-off. Basic vector quantization techniques require very high computational effort and the introduction of vector quantization in speech coding resulted in promising but impractical codecs [14.16]. Accordingly, significant effort was spent to develop vector quantization structures that facilitate low computational complexity [14.17–19]. The continuous improvement in vector quantization methods and an improved understanding of the advantages of vector quantization over scalar quantization [14.20, 21] has meant that the computational effort of speech codecs has not changed significantly over the past two decades, despite significant improvement in codec performance. More effective usage of scalar quantization and the development of effective lattice vector quantization techniques make it unlikely that the computational complexity of speech codecs will increase significantly in the future.

## 14.3 A Universal Coder for Speech

In this section, we consider the encoding of a speech signal from a fundamental viewpoint. In information-theoretic terminology, speech is our *source* signal. We start with a discussion of the direct encoding of speech segments, without imposing any structure on the coder. This discussion is not meant to lead directly to a practical coding method (the computational effort would not be reasonable), but to provide an insight into the structure of existing coders. We then show how a signal model can be introduced. The signal model facilitates coding at a reasonable computational cost and the resulting coding paradigm is used by most speech codecs.

### 14.3.1 Speech Segment as Random Vector

Speech coders generally operate on a sequence of subsequent signal segments, which we refer to as *blocks* (also commonly known as *frames*). Blocks consist generally, but not always, of a fixed number of samples. In the present description of a basic coding system, we divide the speech signal into subsequent blocks of equal length and denote the block length in samples by $k$. We neglect dependencies across block boundaries, which is not always justified in a practical implementation, but simplifies the discussion; it is generally straightforward to cor-

Amazon EX1010 Page 312

rect this omission on implementation. We assume that the blocks can be described by $k$-dimensional random vectors $\boldsymbol{X}^k$ with a probability density function $p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)$ for any $\boldsymbol{x}^k \in \mathbb{R}^k$, the $k$-dimensional Euclidian space (following convention, we denote random variables by capital letters and realizations by lower case letters).

For the first part of our discussion (Sect. 14.3.2), it is sufficient to assume the existence of the probability density function. It is natural, however, to consider some structure of the probability density $p_{\boldsymbol{X}^k}(\cdot)$ based on the properties of speech. We commonly describe speech in terms of a particular set of sounds (a distinct set of phones). A speech vector then corresponds to one sound from a countable set of speech sounds. We impose the notion that speech consists of a set of sounds on our probabilistic speech description. We can think of each sound as having a particular probability density. A particular speech vector then has one of a set of possible probability densities. Each member probability density of the set has an a *prior* probability, denoted as $p_I(i)$, where $i$ indexes the set. The prior probability $p_I(i)$ is the probability that a random vector $\boldsymbol{X}^k$ is drawn from the particular member probability density $i$. The overall probability function of the random speech vector $p_{\boldsymbol{X}^k}(\cdot)$ is then a *mixture* of probability density functions

$$p_{\boldsymbol{X}^k}(\boldsymbol{x}^k) = \sum_{i \in \mathcal{A}} p_I(i) p_{\boldsymbol{X}^k|I}(\boldsymbol{x}^k|i) \,, \tag{14.1}$$

where $\mathcal{A}$ is the set of indexes for component densities and $p_{\boldsymbol{X}^k|I}(\cdot|\cdot)$ is the density of component $i$. These densities are commonly referred to as *mixture components*. If the set of mixture components is characterized by continuous parameters, then the summation must be replaced by an integral.

A common motivation for the mixture formulation of (14.1) is that a good approximation to the true probability density function can be achieved with a mixture of a finite set of probability densities from a particular family. This eliminates the need for the physical motivation. The family is usually derived from a single *kernel* function, such as a Gaussian. The kernel is selected for mathematical tractability.

If a mixture component does correspond to a physically reasonable speech sound, then it can be considered a statistical *model* of the signal. As described in Sect. 14.3.4, it is possible to interpret existing speech coding paradigms from this viewpoint. For example, linear prediction identifies a particular autoregressive model appropriate for a block. Each of the autoregressive models of speech has a certain prior probability and this in turn leads to an overall probability for the speech vector. According to this interpretation, mixture models have long been standard tools in speech coding, even if this was not explicitly stated.

The present formalism does not impose stationarity conditions on the signal within the block. In the mixture density, it is reasonable to include densities that correspond to signal transitions. In practice, this is not common, and the probability density functions are usually defined based on the definition that the signal is stationary within a block. On the other hand, the assumption that all speech blocks are drawn from the same distribution is implicit in the commonly used coding methods. It is consistent with our neglect of interblock dependencies. Thus, if we consider the speech signal to be a vector signal, then we assume stationarity for this vector signal (which is a rather inaccurate approximation). Strictly speaking, we do not assume ergodicity, as averaging over a database is best interpreted an averaging over an ensemble of signals, rather than time averaging over a single signal.

### 14.3.2 Encoding Random Speech Vectors

To encode observed speech vectors $\boldsymbol{x}^k$ that form *realizations* of the random vector $\boldsymbol{X}^k$, we use a speech codebook $C_{\boldsymbol{X}^k}$ that consists of a countable set of $k$-dimensional vectors (the code vectors). We can write $C_{\boldsymbol{X}^k} = \{\boldsymbol{c}_q^k\}_{q \in \mathcal{Q}}$, where $\boldsymbol{c}_q^k \in \mathbb{R}^k$ and $\mathcal{Q}$ is a countable (but not necessarily finite) set of indices. A decoded vector is simply the entry of the codebook that is pointed to by a transmitted index.

The encoding with codebook vectors results in the removal of both redundancy and irrelevancy. It removes irrelevancy by introducing a reduced precision version of the vector $\boldsymbol{x}^k$, i. e., by quantizing $\boldsymbol{x}^k$. The quantized vector requires fewer bits to encode than the unquantized vector. The mechanism of the redundancy removal depends on the coding method and will be discussed in Sect. 14.3.3.

We consider the speech vector, $\boldsymbol{X}^k$, to have a continuous probability density function in $\mathbb{R}^k$. Thus, coding based on the finite-size speech codebook $C_{\boldsymbol{X}^k}$ introduces *distortion*. To minimize the distortion associated with the coding, the encoder selects the code vector (codebook entry) $\boldsymbol{c}_q^k$ that is nearest to the observed vector $\boldsymbol{x}^k$ according to a particular distortion measure,

$$q = \operatorname{argmin}_{q' \in \mathcal{Q}} d\left(\boldsymbol{x}^k, \boldsymbol{c}_{q'}^k\right) . \tag{14.2}$$

*Quantization* is the operation of finding the nearest neighbor in the codebook. The set of speech vectors

that is mapped to a particular code vector $\boldsymbol{c}_q^k$ is called a *quantization cell* or *Voronoi region*. We denote the Voronoi region as $\mathcal{V}_q$,

$$\mathcal{V}_q = \left\{\boldsymbol{x}^k : d\left(\boldsymbol{x}^k, \boldsymbol{c}_q^k\right) < d\left(\boldsymbol{x}^k, \boldsymbol{c}_m^k\right) \forall_{m \neq q}\right\} , \qquad (14.3)$$

where we have ignored that generally points exist for which the inequality is not strict. These are boundary points that can be assigned to any of the cells that share the boundary.

Naturally, the average [averaged over $p_{\boldsymbol{X}^k}(\cdot)$] distortion of the decoded speech vectors differs for different codebooks. A method for designing a coder is to find the codebook, i. e., the set $C_{\boldsymbol{X}^k} = \{\boldsymbol{c}_q^k\}_{q \in \mathcal{Q}}$, that minimizes the average distortion over the speech probability density, given a constraint on the transmission rate. It is not known how to solve this problem in a general manner. Iterative methods (the Lloyd algorithm and its variants, e.g., [14.22–24]) have been developed for the case where $|\mathcal{Q}|$ (the *cardinality* or number of vectors in $C_{\boldsymbol{X}^k}$) is finite. The iterative approach is not appropriate for our present discussion for two reasons. First, we ultimately are interested in structured quantizers that allow us to approximate the optimal codebook and structure is difficult to determine from the iterative method. Second, as we will see below for the constrained-entropy case, practical codebooks do not necessarily have finite cardinality. Instead of the iterative approach, we use an approach where we make simplifying assumptions, which are asymptotically accurate for high coding rates.

### 14.3.3 A Model of Quantization

To analyze the behavior of the speech codebook, we construct a model of the quantization (encoding–decoding) operation. (This *quantization* model is not to be confused with the probabilistic *signal* model described in the next subsection.) Thus, we make the quantization problem mathematically tractable. For simplicity, we use the squared error criterion (Sect. 14.5 shows that this criterion can be used over a wide range of coding scenarios). We also make the standard assumption that the quantization cells are convex (for any two points in a cell, all points on the line segment connecting the two points are in the cell). To construct our encoding–decoding model, we make three additional assumptions that cannot always be justified:

1. The density $p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)$ is constant within each quantization cell. This implies that the probability that a speech vector is inside a cell with index $q$ is

$$p_Q(q) = V_q p_{\boldsymbol{X}^k}(\boldsymbol{x}^k), \boldsymbol{x}^k \in \mathcal{V}_q, \qquad (14.4)$$

where $V_q$ is the volume of the $k$-dimensional cell.

2. The average distortion for speech data falling within cell $q$ is

$$D_q = C V_q^{\frac{2}{k}} , \qquad (14.5)$$

where $C$ is a constant. The assumption made in (14.5) essentially means that the cell shape is fixed. *Gersho* [14.25], conjectured that this assumption is correct for optimal codebooks.

3. We assume that the countable set of code vectors $C_{\boldsymbol{X}^k}$ can be represented by a code-vector density, denoted as $g(\boldsymbol{x}^k)$. This means that the cell volume now becomes a function of $\boldsymbol{x}^k$ rather than the cell index $q$; we replace $V_q$ by $V(\boldsymbol{x}^k)$. To be consistent we must equate the density with the inverse of the cell volume:

$$g(\boldsymbol{x}^k) = \frac{1}{V(\boldsymbol{x}^k)} . \qquad (14.6)$$

The third assumption also implies that we can replace $D_q$ by $D(\boldsymbol{x}^k)$.

The three assumptions listed above lead to solutions that can generally be shown to hold asymptotically in the limit of infinite rate. The theory has been observed to make reasonable predictions of performance for practical quantizers at rates down to two bits per dimension [14.26, 27], but we do not claim accuracy here. The theory serves as a vehicle to understand quantizer behavior and *not* as an accurate predictor of performance.

The code-vector density $g(\boldsymbol{x}^k)$ of our quantization model replaces the set of code vectors as the description of the codebook. Our objective of finding the codebook that minimizes the average distortion subject to a rate constraint has become the objective of finding the optimal density $g(\boldsymbol{x}^k)$ that minimizes the distortion

$$\begin{aligned} D &= \int D(\boldsymbol{x}^k) p_{\boldsymbol{X}^k}(\boldsymbol{x}^k) \mathrm{d}\boldsymbol{x}^k \\ &= C \int V(\boldsymbol{x}^k)^{\frac{2}{k}} p_{\boldsymbol{X}^k}(\boldsymbol{x}^k) \mathrm{d}\boldsymbol{x}^k \\ &= C \int g(\boldsymbol{x}^k)^{-\frac{2}{k}} p_{\boldsymbol{X}^k}(\boldsymbol{x}^k) \mathrm{d}\boldsymbol{x}^k , \end{aligned} \qquad (14.7)$$

subject to a rate constraint.

Armed with our quantization model, we now attempt to find the optimal density $g(\boldsymbol{x}^k)$ (the optimal codebook) for encoding speech. We consider separately two commonly used constraints on the rate: a given fixed rate and a given average rate. As mentioned in Sect. 14.2.1, the former rate constraint applies to circuit-switched networks and the latter rate constraint represents situations

where the rate can be varied continuously, such as, for example, in storage applications and packet networks.

We start with the fixed-rate requirement, where each codebook vector $\boldsymbol{c}_q^k$ is encoded with a codeword of a fixed number of bits. This is called *constrained-resolution* coding. If we use a rate of $R$ bits per speech vector then we have a codebook cardinality of $N = 2^R$ and the density $g(\boldsymbol{x}^k)$ must be consistent with this cardinality:

$$N = \int_{\mathbb{R}^k} g(\boldsymbol{x}^k)\,\mathrm{d}\boldsymbol{x}^k \,. \tag{14.8}$$

We have to minimize the average distortion of (14.7) subject to the constraint (14.8) (i. e., subject to given $N$). This constrained optimization problem is readily solved with the calculus of variations. The solution is

$$\begin{aligned} g(\boldsymbol{x}^k) &= N \frac{p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)^{\frac{k}{k+2}}}{\int p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)^{\frac{k}{k+2}}\,\mathrm{d}\boldsymbol{x}^k} \\ &= 2^R \frac{p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)^{\frac{k}{k+2}}}{\int p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)^{\frac{k}{k+2}}\,\mathrm{d}\boldsymbol{x}^k} \\ &= 2^R \underline{p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)^{\frac{k}{k+2}}} \,, \end{aligned} \tag{14.9}$$

where the underlining denotes normalization to unit integral over $\mathbb{R}^k$ and where $R$ is the bit rate per speech vector. Thus, our encoding–decoding model suggests that, for constrained-resolution coding, the density of the code vectors varies with the data density. At dimensionalities $k >> 1$ the density of the code vectors approximates a simple scaling of the probability density of the speech vectors since $k/(k+2) \to 1$ with increasing $k$.

In the constrained-resolution case, *redundancy* is removed by placing the codebook vectors such that they reflect the density of the data vectors. For example, as shown by (14.9), regions of $\mathbb{R}^k$ without data have no vectors placed in them. This means no codewords are used for regions that have no data. If we had placed codebook vectors there, these would have been redundant. Note that scalar quantization of the $k$-dimensional random vector $\boldsymbol{X}^k$ would do precisely that. Similarly, regions of low data density get relatively few code vectors, reducing the number of codewords spent in such regions.

Next, we apply our quantization model to the case where the average rate is constrained. That is, the codeword length used to encode the cell indices $q$ varies. Let us denote the random index associated with the random vector $\boldsymbol{X}^k$ as $Q$. The source coding theorem [14.12] tells us the lowest possible average rate for uniquely (so it can be decoded) encoding the indices with separate codewords is within one bit of the index entropy (in bits)

$$H(Q) = -\sum_{q\in\mathcal{Q}} p_Q(q)\log_2[p_Q(q)] \,. \tag{14.10}$$

The entropy can be interpreted as the average of a bit allocation, $-\log_2[p_Q(q)]$, for each index $q$. Neglecting the aforementioned *within one bit*, the average rate constraint is $H(Q) = R$, where $R$ is the selected rate. For this reason, this coding method is known as *constrained-entropy* coding. This neglect is reasonable as the difference can be made arbitrarily small by encoding sequences of indices, as in arithmetic coding [14.28], rather than single indices. We minimize the distortion of (14.7) subject to the constraint (14.10), i. e., subject to given $H(Q) = R$. Again, the constrained optimization problem is readily solved with the calculus of variations. In this case the solution is

$$g(\boldsymbol{x}^k) = 2^{H(Q)-h(\boldsymbol{X}^k)} = 2^{R-h(\boldsymbol{X}^k)} \,, \tag{14.11}$$

where $h(\boldsymbol{X}^k) = -\int p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)\log_2[p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)]\,\mathrm{d}\boldsymbol{x}^k$ is the *differential entropy* of $\boldsymbol{X}^k$ in bits, and where $H(Q)$ is specified in bits. It is important to realize that special care must be taken if $p_{\boldsymbol{X}^k}(\cdot)$ is singular, i. e., if the data lie on a manifold.

Equation (14.11) implies that the the code vector density is uniform across $\mathbb{R}^k$. The number of code vectors is countably infinite despite the fact that the rate itself is finite. The codeword length $-\log_2[p_Q(q)]$ increases very slowly with decreasing probability $p_Q(q)$ and, roughly speaking, long codewords make no contribution to the mean rate.

In the constrained-entropy case, redundancy is removed through the lossless encoding of the indices. Given the probabilities of the code vectors, (ideal) lossless coding provides the most efficient bit assignment that allows unique decoding, and this rate is precisely the entropy of the indices. Code vectors in regions of high probability density receive short codewords and code vectors in regions of low probability density receive long codewords.

An important result that we have found for both the constrained-resolution and constrained-entropy cases is that the structure of the codebook is independent of the overall rate. The code-vector density simply increases as $2^R$ (cf. (14.9) and (14.11), respectively) anywhere in $\mathbb{R}^k$. Furthermore, for the constrained-entropy case, the code vector density depends only through the global variable $h(\boldsymbol{X}^k)$ on the probability density.

### 14.3.4 Coding Speech with a Model Family

Although the quantization model of Sect. 14.3.3 provides interesting results, a general implementation of

a codebook for the random speech vector $\boldsymbol{X}^k$ leads to practical problems, except for small $k$. For the constrained-resolution case, larger values of $k$ lead to codebook sizes that do not allow for practical training procedures for storage on conventional media. For the constrained-entropy case, the codebook itself need not be stored, but we need access to the probability density of the codebook entries to determine the corresponding codewords (either offline or through computation during encoding). We can resolve these practical coding problems by using a *model* of the density. Importantly, to simplify the computational effort, we do *not* assume that the model is an accurate representation of the density of the speech signal vector, we simply make a best effort given the tools we have.

The model-based approach towards reducing computational complexity is suggested by the mixture model that we discussed in Sect. 14.3.1. If we classify each speech vector first as corresponding to a particular sound, then we can specify a probability density for that sound. A signal model specifies the probability density, typically by means of a formula for the probability density. The probability densities of the models are typically selected to be relatively simple. The signal models reduce computational complexity, either because they reduce codebook size or because the structural simplicity of the model simplifies the lossless coder. We consider models of a similar structure to be member of a *model family*. The selection of a particular model from the family is made by specifying *model parameters*.

Statistical signal models are commonly used in speech coding, with autoregressive modeling (generally referred to as linear prediction coding methods) perhaps being the most common. In this section, we discuss the selection of a particular model from the model family (i.e., the selection of the model parameters) and the balance in bit allocation between the model and the specification of the speech vector.

Our starting point is that a family of signal models is available for the coding operation. The model family can be any model family that provides a probability assignment for the speech vector $\boldsymbol{x}^k$. We discuss relevant properties for coding with signal models. We do not make the assumption that the resulting coding method is close to a theoretical performance bound on the rate versus distortion trade-off. As seid, we also do not make an assumption about the appropriateness of the signal model family for the speech signal. The model probabilities may not be accurate. However it is likely that models that are based on knowledge of speech production result in better performance.

The reasoning below is based on the early descriptions of the minimum description length (MDL) principle for finding signal models [14.29–31]. These methods separate a code for the model and a code for the signal realization, making them relevant to practical speech coding methods whereas later MDL methods use a single code. Differences from the MDL work include a stronger focus on distortion, and the consideration of the constrained-resolution case, which is of no interest to modeling theory.

### Constrained-Entropy Case

First we consider the constrained-entropy case, i. e., we consider the case of a uniform codebook. Each speech vector is encoded with a codebook where each cell is of identical volume, which we denote as $V$. Let the model distribution be specified by a set of model parameters, $\theta$. We consider the models to have a probability density, which means that a particular parameter set $\theta$ corresponds to a particular realization of a random parameter vector $\Theta$. We write the probability density of $\boldsymbol{X}^k$ assuming the particular parameter set $\theta$ as $p_{\boldsymbol{X}^k;\Theta}(\boldsymbol{x}^k|\theta)$. The corresponding overall model density is

$$\tilde{p}_{\boldsymbol{X}^k}(\boldsymbol{x}^k) = -\sum_{\theta} p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta) \cdot p_{\Theta}(\theta)\,, \qquad (14.12)$$

where the summation is over all parameter sets. The advantage of selecting and then using models $p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)$ from the family over using the composite model density $\tilde{p}_{\boldsymbol{X}^k}(\boldsymbol{x}^k)$ is a decrease of the computational effort.

The quantization model of Sect. 14.3.3 and in particular (14.4) and (14.10), show that the constrained-entropy encoding of a vector $\boldsymbol{x}^k$ assuming the model with parameters $\theta$ requires $-\log_2[Vp_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)]$ bits. In addition, the decoder must receive side information specifying the model.

Let $\hat{\theta}(\boldsymbol{x}^k)$ be the parameter vector that maximizes $p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)$ and, thus, minimizes the bit allocation $-\log_2[Vp_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)]$. That is, $p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k)]$ is the maximum-likelihood model (from the family) for encoding the speech vector $\boldsymbol{x}^k$. The random speech vectors $\boldsymbol{X}^k$ do not form a countable set and as a result the random parameter vector $\hat{\Theta}(\boldsymbol{X}^k)$ generally does not form a countable set for conventional model families such as autoregressive models. To encode the model, we must discretize it.

To facilitate transmission of the random model index, $J$, the model parameters must be quantized and we write the random parameter set corresponding to random index $J$ as $\theta(J)$. If $p_J(j)$ is a prior probability of the model index, the overall bit allocation for the vector

$\boldsymbol{x}^k$ when encoded with model $j$ is

$$\begin{aligned} l &= -\log_2[p_J(j)] - \log_2\left\{V(\boldsymbol{x}^k)p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\theta(j)]\right\} \\ &= -\log_2[p_J(j)] + \log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k)]}{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\theta(j)]}\right) \\ &\quad - \log_2[V(\boldsymbol{x}^k)p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\hat{\theta})]\,, \end{aligned} \tag{14.13}$$

where the term $\log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k)]}{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\theta(j)]}\right)$ represents the additional (excess) bit allocation required to encode $\boldsymbol{x}^k$ with model $j$ over the bit allocation required to encode $\boldsymbol{x}^k$ with the true maximum-likelihood model from the model family.

With some abuse of notation, we denote by $j(\boldsymbol{x}^k)$ the function that provides the index for a given speech vector $\boldsymbol{x}^k$. In the following, we assume that the functions $\theta(j)$ and $j(\boldsymbol{x}^k)$ minimize $l$. That is, we quantize $\theta$ so as to minimize the total number of bits required to encode $\boldsymbol{x}^k$.

We are interested in the bit allocation that results from averaging over the probability density $p_{\boldsymbol{X}^k}(\cdot)$ of the speech vectors,

$$\begin{aligned} \mathrm{E}\{\mathsf{L}\} = &-\mathrm{E}\left\{\log_2[p_J(j(\boldsymbol{X}^k))]\right\} \\ &-\mathrm{E}\left\{\log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta(j(\boldsymbol{X}^k)))}{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\hat{\theta}(\boldsymbol{X}^k))}\right)\right\} \\ &-\mathrm{E}\left\{\log_2\left[V(\boldsymbol{X}^k)p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\hat{\theta}(\boldsymbol{X}^k))\right]\right\}, \end{aligned} \tag{14.14}$$

where $\mathrm{E}\{\cdot\}$ indicates averaging over the speech vector probability density and where $\mathsf{L}$ is the random bit allocation that has $I$ as realization. In (14.14), the first term describes the mean bit allocation to specify the model, the second term specifies the mean excess in bits required to encode $\boldsymbol{X}^k$ assuming $\theta(j(\boldsymbol{x}^k))$ instead of assuming the optimal $\hat{\theta}(\boldsymbol{x}^k)$, and the third term specifies the mean number of bits required to encode $\boldsymbol{X}^k$ if the optimal model is available. Importantly, only the third term contains the cell volume that determines the mean distortion of the speech vectors through (14.7).

Assuming validity of the encoding model of Sect. 14.3.3, the optimal trade-off between the bit allocation for the model index and the bit allocation for the speech vectors $\boldsymbol{X}^k$ depends only on the mean of

$$\begin{aligned} \eta = &-\log_2\{p_J[j(\boldsymbol{x}^k)]\} \\ &-\log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\theta(j(\boldsymbol{x}^k))]}{p_{\boldsymbol{X}^k|\Theta}[\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k)]}\right), \end{aligned} \tag{14.15}$$

which is referred to as the [14.32]. The goal is to find the functions $\theta(\cdot)$ and $j(\cdot)$ that minimize the index of resolvability over the ensemble of speech vectors. An important consequence of our logic is that these functions, and therefore the rate allocation for the model index, are dependent only on the excess rate and the probability of the quantized model. As the third term of (14.14) is missing, no relation to the speech distortion exists. That is *the rate allocation for the model index $J$ is independent of distortion and overall bit rate*. While the theory is based on assumptions that are accurate only for high bit rates, this suggests that the bit allocation for the parameters becomes proportionally more important at low rates.

The fixed entropy for the model index indicates, for example, that for the commonly used linear-prediction-based speech coders, the rate allocation for the linear prediction parameters is independent of the overall rate of the coder. As constrained-entropy coding is not commonly used for predictive coding, this result is not immediately applicable to conventional speech coders. However, the new result we derive below is applicable to such coders.

### Constrained-Resolution Case

Most current speech coders were designed with a constrained-resolution (fixed-rate) constraint, making it useful to study modeling in this context. We need some preliminary results. For a given model, with parameter set $\theta$, and optimal code vector density, the average distortion over a quantization cell centered at location $\boldsymbol{x}^k$ can be written

$$\begin{aligned} D(\boldsymbol{x}^k) &= CV(\boldsymbol{x}^k)^{\frac{2}{k}} \\ &= Cg(\boldsymbol{x}^k)^{-\frac{2}{k}} \\ &= CN^{-\frac{2}{k}}\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)^{\frac{k}{k+2}}\right]^{-\frac{2}{k}}, \end{aligned} \tag{14.16}$$

where we have used (14.7) and (14.9). We take the expectation of (14.16) with respect to the true probability density function $p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)$ and obtain the mean distortion for the constrained-resolution case:

$$D_{\mathrm{CR}} = CN^{-\frac{2}{k}}\mathrm{E}\left\{\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]^{-\frac{2}{k}}\right\}. \tag{14.17}$$

Equation (14.17) can be rewritten as

$$\begin{aligned} \frac{2}{k}\log_2(N) = &\log_2\left(\mathrm{E}\left\{\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]^{-\frac{2}{k}}\right\}\right) \\ &-\log_2\left(\frac{D_{\mathrm{CR}}}{C}\right). \end{aligned} \tag{14.18}$$

We assume that $k$ is sufficiently large that, in the region where $p_{\boldsymbol{X}^k}(\boldsymbol{x}^k)$ is significant, we can use the expansion

$u \approx 1 + \log(u)$ for the term $[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)^{k/(k+2)}]^{-2/k}$ and write

$$\begin{aligned}
&\log\left(\mathrm{E}\left\{\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]^{-\frac{2}{k}}\right\}\right)\\
&\approx \log\left(1-\frac{2}{k}\mathrm{E}\left\{\log\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]\right\}\right)\\
&\approx -\frac{2}{k}\mathrm{E}\left\{\log\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]\right\} . \qquad (14.19)
\end{aligned}$$

Having completed the preliminaries, we now consider the encoding of a speech vector $\boldsymbol{x}^k$. Let $L_{(m)}$ be the fixed bit allocation for the model index. The total rate is then

$$\begin{aligned}
L &= L_{(m)} + L(\boldsymbol{x}^k)\\
&= L_{(m)} + \log_2(N)\\
&= L_{(m)} - \mathrm{E}\left\{\log_2\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{\frac{k}{k+2}}\right]\right\}\\
&\quad - \frac{k}{2}\log_2\left(\frac{D_{\mathrm{CR}}}{C}\right) . \qquad (14.20)
\end{aligned}$$

The form of (14.20) shows that, given the assumptions made, we can define an *equivalent codeword length* $\log_2[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{X}^k|\theta)^{k/(k+2)}]$ for each speech codebook entry. The equivalent codeword length represents the spatial variation of the distortion. Note that this equivalent codeword length does *not* correspond to the true codeword length of the speech vector codebook, which is fixed for the constrained-resolution case. For a particular codebook vector $\boldsymbol{x}^k$, the equivalent codeword length is

$$\begin{aligned}
L &= L_{(m)} - \log_2\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)^{\frac{k}{k+2}}\right]\\
&\quad - \frac{k}{2}\log_2\left(\frac{D_{\mathrm{CR}}}{C}\right) . \qquad (14.21)
\end{aligned}$$

Similarly to the constrained-entropy case, we can decompose (14.21) into a rate component that relates to the encoding of the model parameters, a component that describes the excess equivalent rate resulting from limiting the precision of the model parameters, and a rate component that relates to optimal encoding with optimal (uncoded) model parameters:

$$\begin{aligned}
L &= L_{(m)} - \log_2(p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta(j))^{\frac{k}{k+2}}) - \frac{k}{2}\log_2\left(\frac{D_{\mathrm{CR}}}{C}\right)\\
&= L_{(m)} - \log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta(j))^{\frac{k}{k+2}}}{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k))^{\frac{k}{k+2}}}\right)\\
&\quad - \log_2\left[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k))^{\frac{k}{k+2}}\right] - \frac{k}{2}\log_2\left(\frac{D_{\mathrm{CR}}}{C}\right) ,
\end{aligned}$$

(14.22)

We can identify the last two terms as the bit allocation for $\boldsymbol{x}^k$ for the optimal constrained-resolution model for the speech vector $\boldsymbol{x}^k$. The second term is the excess equivalent bit allocation required to encode the speech vector with model $j$ over the bit allocation required for the optimal model from the model family. The first two terms determine the trade-off between the bits spent on the model, and the bits spent on the speech vectors. These two terms form the index of resolvability for the constrained-resolution case:

$$\eta = L_{(m)} - \log_2\left(\frac{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta(j))^{\frac{k}{k+2}}}{p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\hat{\theta}(\boldsymbol{x}^k))^{\frac{k}{k+2}}}\right) . \qquad (14.23)$$

As for the constrained-entropy case, the optimal set of functions $\theta(j)$ and $j(\boldsymbol{x}^k)$ (and, therefore, the bit allocation for the model) are dependent only on the speech vector density for the constrained-resolution case. The rate for the model is independent of the distortion selected for the speech vector and of the overall rate. With increasing $k$, the second term in (14.23) and (14.15) becomes identical. That is the expression for the excess rate for using the quantized model parameters corresponding to model $j$ instead of the optimal parameters is identical.

The independence of the model-parameter bit allocation of the overall codec rate for the constrained-entropy case is of great significance for practical coding systems. We emphasize again that this result is valid only under the assumptions made in Sect. 14.3.3. We expect the independence to break down at lower rates, where the codebook $C_{\boldsymbol{X}^k}$ describing the speech cannot be approximated by a density.

**Table 14.1** Bit rates of the AMR-WB coder [14.4]

| Rate (bits) | 6.6 | 8.85 | 12.65 | 14.25 | 15.85 | 18.25 | 19.85 | 23.05 |
|---|---|---|---|---|---|---|---|---|
| AR model | 36 | 46 | 46 | 46 | 46 | 46 | 46 | 46 |
| Pitch parameter | 23 | 26 | 30 | 30 | 30 | 30 | 30 | 30 |
| Excitation | 48 | 80 | 144 | 176 | 208 | 256 | 288 | 352 |

Amazon EX1010 Page 318

The results described in this section are indeed supported, at least qualitatively, by the configuration of practical coders. Table 14.1 shows the most important bit allocations used in the adaptive-multirate wideband (AMR-WB) speech coder [14.4]. The AMR-WB coder is a constrained-resolution coder. It is seen that the design of the codec satisfies the predicted behavior: the bit allocation for the model parameters is essentially independent of the rate of the codec, except at low rates.

#### Model-Based Coding

In signal-model-based coding we assume the family is known to the encoder and decoder. An index to the specific model is transmitted. Each model corresponds to a unique speech-domain codebook. The advantage of the model-based approach is that the structure of the density is simplified (which is advantageous for constrained-entropy coding) and that the required number of codebook entries for the constrained-resolution case is smaller. This facilitates searching through the codebook and/or the definition of the lossless coder.

The main result of this section is that we can determine the set of codebooks for the models independently of the overall rate (and speech-vector distortion). The result is consistent with existing results. The result of this section leads to fast codec design as there is no need to check the best trade-off in bit allocation between model and signal quantization.

When encoding with a model-based coding it is advantageous first to identify the *best* model, encode the model index $j$, and then encode the signal using codebook $C_{\boldsymbol{X}^k,j}$ that is associated with that particular model $j$. The model selection can be made based on the index of resolvability.

## 14.4 Coding with Autoregressive Models

We now apply the methods of Sect. 14.3 to a practical model family. Autoregressive model families are commonly used in speech coding. In speech coding this class of coders is generally referred to as being based on *linear prediction*. We discuss coding based on a family that consists of a set of autoregressive models of a particular order (denoted as $p$). To match current practice, we consider the constrained-resolution case.

We first formulate the index of resolvability in terms of a spectral formulation of the autoregressive model. We show that this corresponds to the definition of a distortion measure for the model parameters. The distortion measure is approximated by the commonly used Itakura–Saito and log spectral distortion measures. Thus, starting from a squared error criterion for the speech signal, we obtain the commonly used (e.g., [14.33–37]) distortion measures for the linear-prediction parameters. Finally, we show that our reasoning leads to an estimate for the bit allocation for the model. We discuss how this result relates to results on autoregressive model estimation.

### 14.4.1 Spectral-Domain Index of Resolvability

Our objective is to encode a particular speech vector $\boldsymbol{x}^k$ using the autoregressive model. To facilitate insight, it is beneficial to make a spectral formulation of the problem. To this purpose, we assume that $k$ is sufficiently large to neglect edge effects. Thus, we neglect the difference between circular and linear convolution.

The autoregressive model assumption implies that $\boldsymbol{x}^k$ has a multivariate Gaussian probability density

$$p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta) = \frac{1}{\sqrt{2\pi\det(\mathbf{R}_\theta)}}\exp\left(-\frac{1}{2}\boldsymbol{x}^{kT}\mathbf{R}_\theta^{-1}\boldsymbol{x}^k\right) . \quad (14.24)$$

$\mathbf{R}_\theta$ is the *model* autocorrelation matrix

$$\mathbf{R}_\theta = \mathbf{A}^{-1}\mathbf{A}^{-\mathrm{H}} , \quad (14.25)$$

where $\mathbf{A}$ a lower-triangular Toeplitz matrix with first column $\sigma[1, a_1, a_2, \cdots, a_p, 0, \cdots, 0]^{\mathrm{T}}$, where the $a_i$ are the autoregressive model parameters (linear-prediction parameters), and $p$ is the autoregressive model order and the superscript $H$ is the Hermitian transpose. Thus, the set of model parameters is $\theta = \{\sigma, a_1, \cdots, a_p\}_{i=1,\cdots,p}$. We note that typically $p = 10$ for 8 kHz sampling rate and $p = 16$ for 12 kHz and 16 kHz sampling rate.

When $k$ is sufficiently large, we can perform our analysis in terms of power spectral densities. The transfer function of the autoregressive model is

$$A(z)^{-1} = \frac{\sigma}{1 + a_1 z^{-1} + \cdots + a_p z^{-p}} , \quad (14.26)$$

where $\sigma$ is a gain. This corresponds to the model power spectral density

$$\mathbf{R}_\theta(z) = |A(z)|^{-2} \,. \tag{14.27}$$

In the following, we make the standard assumption that $A(z)$ is minimum-phase.

Next we approximate (14.24) in terms of power spectral densities and the transfer function of the autoregressive model. Using *Szegö*'s theorem [14.38], it is easy to show that, asymptotically in $k$,

$$\det(\mathbf{R}_\theta) = \exp\left\{\frac{k}{2\pi}\int_0^{2\pi}\log\left[\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})\right]\mathrm{d}\omega\right\} \,. \tag{14.28}$$

We also use the asymptotic equality

$$\begin{aligned}\frac{1}{2}\boldsymbol{x}^{kT}\mathbf{R}_\theta^{-1}\boldsymbol{x}^k &= \frac{1}{4\pi}\int_0^{2\pi}\frac{|x(\mathrm{e}^{\mathrm{i}\omega})|^2}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\mathrm{d}\omega \\ &= \frac{k}{4\pi}\int_0^{2\pi}\frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\mathrm{d}\omega \,, \end{aligned} \tag{14.29}$$

where $x(z) = \sum_{i=0}^{k-1} x_i z^{-i}$ for $\boldsymbol{x}^k = (x_1, \cdots, x_k)$ and $R_x(\mathrm{e}^{\mathrm{i}\omega}) = \frac{1}{k}|x(\mathrm{e}^{\mathrm{i}\omega})|^2$.

Equations (14.28) and (14.29) can be used to rewrite the multivariate density of (14.24) in terms of power spectral densities. It is convenient to write the log density:

$$\begin{aligned}&\log[p_{\boldsymbol{X}^k|\Theta}(\boldsymbol{x}^k|\theta)] \\ &= -\frac{1}{2}\log(2\pi) - \frac{k}{4\pi}\int_0^{2\pi}\log(\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega}))\mathrm{d}\omega \\ &\quad - \frac{k}{4\pi}\int_0^{2\pi}\frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\mathrm{d}\omega \,. \end{aligned} \tag{14.30}$$

We use (14.30) to find the index of resolvability for the constrained-resolution case. We make the approximation that $k$ is sufficiently large that it is reasonable to approximate the exponent $k/(k+2)$ by unity in (14.23). This implies that we do not have to consider the normalization in this equation. Inserting (14.30) into (14.23) results in

$$\begin{aligned}\eta = L_{(m)} + \frac{k}{4\pi}\int_0^{2\pi}\Bigg[&-\log\left(\frac{R_{\hat\theta}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right) \\ &+\frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})} - \frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{R_{\hat\theta}(\mathrm{e}^{\mathrm{i}\omega})}\Bigg]\mathrm{d}\omega \,. \end{aligned} \tag{14.31}$$

The maximum-likelihood estimate of the autoregressive model $\hat\theta$ given a data vector $\boldsymbol{x}^k$ is a well-understood problem, e.g., [14.39, 40]. The predictor parameter estimate of the standard Yule–Walker solution method has the same asymptotic density as the maximum-likelihood estimate [14.41].

To find the optimal bit allocation for the model we have to minimize the expectation of (14.31) over the ensemble of all speech vectors. We study the behavior of this minimization. For notational convenience we define a cost function

$$\begin{aligned}\psi(\theta,\hat\theta) = \frac{k}{4\pi}\int_0^{2\pi}\Bigg[&-\log\left(\frac{R_{\hat\theta}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right) \\ &+\frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})} - \frac{R_x(\mathrm{e}^{\mathrm{i}\omega})}{R_{\hat\theta}(\mathrm{e}^{\mathrm{i}\omega})}\Bigg]\mathrm{d}\omega \,. \end{aligned} \tag{14.32}$$

Let $\theta$ be a particular model from a countable model set $\mathcal{C}_\Theta(L_{(m)})$ with a bit allocation $L_{(m)}$ for the model. Finding the optimal model set $\mathcal{C}_\Theta(L_{(m)})$ is then equivalent to

$$\begin{aligned}&\min_{L_{(m)}\in\mathbb{N}} \mathrm{E}[\eta] \\ &= \min_{L_{(m)}\in\mathbb{N}}\left\{L_{(m)} + \min_{\mathcal{C}_\Theta(L_{(m)})}\mathrm{E}\Big[\min_{\theta\in\mathcal{C}_\Theta(L_{(m)})}\psi(\theta,\hat\theta)\Big]\right\} \,. \end{aligned} \tag{14.33}$$

If we write

$$D(L_{(m)}) = \min_{\mathcal{C}_\Theta(L_{(m)})}\mathrm{E}\Big[\min_{\theta\in\mathcal{C}_\Theta(L_{(m)})}\psi(\theta,\hat\theta)\Big] \tag{14.34}$$

then (14.33) becomes

$$\min_{L_{(m)}\in\mathbb{N}}\mathrm{E}[\eta] = \min_{L_{(m)}\in\mathbb{N}}\left[L_{(m)} + D(L_{(m)})\right] \,. \tag{14.35}$$

If we interpret $D(L_{(m)})$ as a minimum mean distortion, minimizing (14.35) is equivalent to finding a particular point on a rate-distortion curve. We can minimize the cost function of (14.34) for all $L_{(m)}$ and then select the $L_{(m)}$ that minimizes the overall expression of (14.35). Thus only one particular distortion level, corresponding to one particular rate, is relevant to our speech coding system. This distortion–rate pair for the model is dependent on the distribution of the speech models. Assuming that $D(L_{(m)})$ is once differentiable towards $L_{(m)}$, then (14.35) shows that its derivative should be $-1$ at the optimal rate for the model.

### 14.4.2 A Criterion for Model Selection

We started with the notion of using an autoregressive model family to quantize the speech signal. We found

Amazon EX1010 Page 320

that we could do so by first finding the maximum-likelihood estimate $\hat{\theta}$ of the autoregressive model parameters, then selecting from a set of models $\mathcal{C}_\Theta(L_{(m)})$ the model nearest to the maximum-likelihood model based on the cost function $\psi(\theta,\hat{\theta})$ and then quantizing the speech given the selected model. As quantization of the predictor parameters corresponds to our model selection, it is then relevant to compare the distortion measure of (14.32) with the distortion measures that are commonly used for the linear-prediction parameters in existing speech coders.

To provide insight, it is useful to write $R_x(\mathrm{e}^{\mathrm{i}\omega}) = R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})\, R_w(\mathrm{e}^{\mathrm{i}\omega})$, where $R_w(\mathrm{e}^{\mathrm{i}\omega})$ represents a *remainder* power-spectral density that captures the spectral error of the maximum likelihood model. If the model family is of low order, then $R_w(\mathrm{e}^{\mathrm{i}\omega})$ includes the spectral *fine structure*. We can rewrite (14.32) as

$$\psi(\theta,\hat{\theta}) = \frac{k}{4\pi}\int_0^{2\pi}\left[-\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right) + \left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})} - 1\right) R_w(\mathrm{e}^{\mathrm{i}\omega})\right] \mathrm{d}\omega\,. \quad (14.36)$$

Interestingly, (14.36) reduces to the well-known *Itakura–Saito criterion* [14.42] if $R_w(\mathrm{e}^{\mathrm{i}\omega})$ is set to unity.

It is common (e.g., [14.43]) to relate different criteria through the series expansion $u = 1+\log(u)+\frac{1}{2}[\log(u)]^2+\cdots$. Assuming small differences between the optimal model $\hat{\theta}$ and the model from the set $\theta$, (14.36) can be written

$$\psi(\theta,\hat{\theta}) \cong \frac{k}{4\pi}\int_0^{2\pi}\left\{\left[R_w(\mathrm{e}^{\mathrm{i}\omega})-1\right]\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right) + \frac{1}{2}R_w(\mathrm{e}^{\mathrm{i}\omega})\left[\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right)\right]^2\right\}\mathrm{d}\omega\,. \quad (14.37)$$

Equation (14.37) needs to be accurate only for nearest neighbors of $\hat{\theta}$.

We can simplify (14.37) further. With our assumptions for the autoregressive models, $\mathbf{R}_\theta(z)$ is related to monic minimum-phase polynomials through (14.27) and the further assumption that their gains $\sigma$ are identical (i. e., is not considered here), this implies that

$$\frac{1}{2\pi}\int_0^{2\pi}\log(\mathbf{R}_\theta)\,\mathrm{d}\omega = \frac{1}{2\pi}\int_0^{2\pi}\log(R_{\hat{\theta}})\,\mathrm{d}\omega = \log(\sigma^2)\,. \quad (14.38)$$

This means that we can rewrite (14.37) as

$$\psi(\theta,\hat{\theta}) \cong \frac{k}{4\pi}\int_0^{2\pi} R_w(\mathrm{e}^{\mathrm{i}\omega})\left\{\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right) + \frac{1}{2}\left[\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right)\right]^2\right\}\mathrm{d}\omega\,. \quad (14.39)$$

Equation (14.39) forms the basic measure that must be optimized for the selection of the model from a set of models, i. e., for the optimal quantization of the model parameters.

If we can neglect the impact of $R_w(z)$, then (using the result of (14.38)) minimizing (14.39) is equivalent to minimizing

$$\psi(\theta,\hat{\theta}) \cong \frac{k}{8\pi}\int_0^{2\pi}\left[\log\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{\mathrm{i}\omega})}{\mathbf{R}_\theta(\mathrm{e}^{\mathrm{i}\omega})}\right)\right]^2\mathrm{d}\omega\,, \quad (14.40)$$

which is the well-known *mean squared log spectral distortion*, scaled by the factor $k/4$. Except for this scaling factor, (14.39) is precisely the criterion that is commonly used (e.g., [14.35, 44, 45]) to evaluate performance of quantizers for autoregressive (AR) model parameters. This is not unreasonable as the neglected modeling error $R_w(\mathrm{e}^{\mathrm{i}\omega})$ is likely uncorrelated with the model quantization error.

### 14.4.3 Bit Allocation for the Model

The AR model is usually described with a small number of parameters (as mentioned, $p = 10$ is common for 8 kHz sampling rate). Thus, the spectral data must lie on a manifold of dimension $p$ or less in the log spectrum space. At high bit allocations, where measurement noise dominates (see also the end of Sect. 14.3.3), the manifold dimension is $p$ and the spectral distortion is expected to scale as

$$D(L_{(m)}) = \frac{k}{4}\beta^2 N_{\mathrm{AR}}^{-\frac{2}{p}} = \frac{k}{4}\beta^2\,\mathrm{e}^{-\frac{2}{p}L_{(m)}}\,, \quad (14.41)$$

where $\beta$ is a constant and $N_{\mathrm{AR}}$ is the number of spectral models in the family and $L_{(m)} = \log(N_{\mathrm{AR}})$. At higher spectral distortion levels, it has been observed that the physics of the vocal tract constrains the dimensionality of the manifold. This means that (14.41) is replaced by

$$D(L_{(m)}) = \frac{k}{4}\beta^2\,\mathrm{e}^{-\frac{2}{\kappa}L_{(m)}} \quad (14.42)$$

with $\kappa < p$. This behavior was observed for trained codebooks over a large range in [14.44] (similar behavior was

observed for cepstral parameters in [14.46]) and for specific vowels in [14.47]. The results of [14.44] correspond to $\kappa = 7.1$ and $\beta = 0.80$.

The mean of (14.31) becomes

$$\mathrm{E}[\eta] = L_{(m)} + \frac{k}{4}\beta^2 \mathrm{e}^{-\frac{2}{\kappa}L_{(m)}} \,. \tag{14.43}$$

Differentiating towards $L_{(m)}$ we find that the optimal bit allocation for the AR model selection to be

$$L_{(m)} = \frac{\kappa}{2}\log\left(\beta^2\frac{k}{2\kappa}\right) \,, \tag{14.44}$$

which is logarithmically dependent on $k$. Using the observed data of [14.44], we obtain an optimal rate of about 17 bits for 8 kHz sampled speech at a 20 ms block size. The corresponding mean spectral distortion is about 1.3 dB. The distortion is similar to the mean estimation errors found in experiments on linear predictive methods on speech sounds [14.48].

The 17 bit requirement for the prediction parameter quantizer is similar to that obtained by the best available prediction parameter quantizers that operate on single blocks and bounds obtained for these methods [14.35, 49–52]. In these systems the lowest bit allocation for 20 ms blocks is about 20 bits. However, the performance of these coders is entirely based on the often quoted 1 dB threshold for transparency [14.35]. The definition of this empirical threshold is consistent with the conventional two-step approach: the model parameters are first quantized using a separately defined criterion, and the speech signal is quantized thereafter based on a weighted squared error criterion. In contrast, we have shown that a single distortion measure operating on the speech vector suffices for this purpose.

We conclude that the definition of a squared-error criterion for the speech signal leads to a bit allocation for the autoregressive model. No need exists to introduce perception based thresholds on log spectral distortion.

### 14.4.4 Remarks on Practical Coding

The two-stage approach is standard practice in linear-prediction-based (autoregressive-model-based) speech codecs. In the selection stage, weighted squared error criteria in the so-called line-spectral frequency (LSF) representation of the prediction parameters are commonly used, e.g., [14.34–37]. If the proper weighting is used, then the criterion can be made to match the log spectral distortion measure [14.53] that we derived above.

The second stage is the selection of a speech codebook entry from a codebook corresponding to the selected model. The separation into a set of models simplifies this selection. In general, this means that a speech-domain codebook must be available for each model. It was recently shown that the computational or storage requirements for optimal speech-domain codebooks can be made reasonable by using a single codebook for each set of speech sounds that are similar except for a unitary transform [14.54]. The method takes advantage of the fact that different speech sounds may have similar statistics after a suitable unitary transform and can, therefore, share a codebook. As the unitary transform does not affect the Euclidian distance, it also does not affect the optimality of the codebook.

In the majority of codecs the speech codebooks are generated in real time, with the help of the model obtained in the first stage. This approach is the so-called *analysis-by-synthesis* approach. It can be interpreted as a method that requires the *synthesis* of candidate speech vectors (our speech codebook), hence the name. Particularly common is the usage of the analysis-by-synthesis approach for the autoregressive model [14.16,55]. While the analysis-by-synthesis approach has proven its merit and is used in hundreds of millions of communication devices, it is not optimal. It was pointed out in [14.54] that analysis-by-synthesis coding inherently results in a speech-domain codebook with quantization cells that have a suboptimal shape, limiting performance.

## 14.5 Distortion Measures and Coding Architecture

An objective of coding is the removal of irrelevancy. This means that precision is lost and that we introduce a difference between the original and the decoded signal, the error signal. So far we have considered basic quantization theory and how modeling can be introduced in this quantization structure. We based our discussion on a mean squared error distortion measure for the speech vector. As discussed in Sect. 14.2.2, the proper measure is the decrease in signal quality as perceived by human listeners. That is, the goal in speech coding is to minimize the perceived degradation resulting from an encoding at a particular rate. This section discusses

Amazon EX1010 Page 322

methods for integrating perceptually motivated criteria into a coding structure.

To base coding on perceived quality degradation, we must define an appropriate quantitative measure of the perceived distortion. Reasonable objectives for a good distortion measure for a speech codec are a good prediction of experimental data on human perception, mathematical tractability, low delay, and low computational requirements.

A major aspect in the definition of the criterion is the representation of the speech signal the distortion measure operates on. Most straightforward is to quantize the speech signal itself and use the distortion measure as a selection criterion for code vectors and as a means to design the quantizers. This coding structure is commonly used in speech coders based on linear-predictive coding. An alternative coding structure is to apply a transform towards a domain that facilitates a simple distortion criterion. Thus, in this approach, we first perform a mapping to a *perceptual domain* (preprocessing) and then quantize the mapped signal in that domain. At the decoder we apply the inverse mapping (postprocessing). This second architecture is common in transform coders aimed at encoding audio signals at high fidelity.

We start this section with a subsection discussing the squared error criterion, which is commonly used because of its mathematical simplicity. In Subsects. 14.5.2 and 14.5.3 we then discuss models of perception and how the squared error criterion can be used to represent these models. We end the section with a subsection discussing in some more detail the various coding architectures.

### 14.5.1 Squared Error

The squared-error criterion is commonly used in coding, often without proper physical motivation. Such usage results directly from its mathematical tractability. Given a data sequence, optimization of the model parameters for a model family often leads to a set of linear equations that is easily solved.

For the $k$-dimensional speech vector $\boldsymbol{x}^k$, the basic squared-error criterion is

$$\eta = (\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k)^{\mathrm{H}}(\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k)\,, \tag{14.45}$$

where the superscript 'H' denotes the Hermitian conjugate and $\hat{\boldsymbol{x}}^k$ is the reconstruction vector upon encoding and decoding. Equation (14.45) quantifies the variance of the signal error. Unfortunately, variance cannot be equated to loudness, which is the psychological correlate of variance. At most we can expect that, for a given original signal, a scaling of the error signal leads to a positive correlation between perceived distortion and squared error.

While the squared error in its basic form is not representative of human perception, adaptive weighting of the squared-error criterion can lead to improved correspondence. By means of weighting we can generalize the squared-error criterion to a form that allows inclusion of knowledge of perception (the formulation of the weighted squared error criterion for a specific perceptual model is described in Subsects. 14.5.2 and 14.5.3). To allow the introduction of perceptual effects, we linearly weight the error vector $\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k$ and obtain

$$\eta = (\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k)^{\mathrm{H}}\mathbf{H}^{\mathrm{H}}\mathbf{H}(\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k)\,, \tag{14.46}$$

where $\mathbf{H}$ is an $m \times k$ matrix, where $m$ depends on the weighting invoked. As we will see below, many different models of perception can be approximated with the simple weighted squared-error criterion of (14.46). In general, the weighting matrix $\mathbf{H}$ adapts to $x^k$, that is $\mathbf{H}(\boldsymbol{x}^k)$ and

$$\boldsymbol{y}^m = \mathbf{H}(\boldsymbol{x}^k)\,\boldsymbol{x}^k \tag{14.47}$$

can be interpreted as a perceptual-domain representation of the signal vector for a region of $\boldsymbol{x}^k$ where $\mathbf{H}(\boldsymbol{x}^k)$ is approximately constant.

The inclusion of the matrix $\mathbf{H}$ in the formulation of the squared-error criterion generally results in a significantly higher computational complexity for the evaluation of the criterion. Perhaps more importantly, when the weighted criterion of (14.46) is adaptive, then the optimal distribution of the code vectors (Sect. 14.3.3) for constrained-entropy coding is no longer uniform in the speech domain. This has significant implications for the computational effort of a coding system.

The formulation of (14.46) is commonly used in coders that are based on an autoregressive model family, i. e., linear-prediction-based analysis-by-synthesis coding [14.16]. (The matrix $\mathbf{H}$ then usually includes the autoregressive model, as the speech codebook is defined as a filtering of an excitation codebook.) Also in the context of this class of coders, the vector $\mathbf{H}\boldsymbol{x}^k$ can be interpreted as a perceptual-domain vector. However, because $\mathbf{H}$ is a function of $\boldsymbol{x}^k$ it is not straightforward to define a codebook in this domain.

The perceptual weighting matrix $\mathbf{H}$ often represents a filter operation. For a filter with impulse response $[h_0, h_1, h_2, \cdots]$, the matrix $\mathbf{H}$ has a Toeplitz structure:

$$\mathbf{H} = \begin{pmatrix} h_0 & 0 & \cdots \\ h_1 & h_0 & \cdots \\ h_2 & h_1 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} . \tag{14.48}$$

For computational reasons, it may be convenient to make the matrix $\mathbf{H}^{\mathrm{H}}\mathbf{H}$ Toeplitz. If the impulse response has time support $p$ then $\mathbf{H}^{\mathrm{H}}\mathbf{H}$ is Toeplitz if $\mathbf{H}$ is selected to have dimension $(m+p)\times m$ [14.19].

Let us consider how the impulse response $[h_0, h_1, h_2, \cdots]$ of (14.48) is typically constructed for the case of linear-predictive coding. The impulse response is constructed from the signal model. Let the transfer function of the corresponding autoregressive model be, as in (14.26)

$$A(z)^{-1} = \frac{\sigma}{1 + a_1 z^{-1} \cdots + a_p z^{-p}} , \tag{14.49}$$

where the $a_i$ are the prediction parameters and $\sigma$ is the gain. A weighting that is relatively flexible and has low computational complexity is then [14.56]

$$H(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} , \tag{14.50}$$

where $\gamma_1$ and $\gamma_2$ are parameters that are selected to accurately describe the impact of the distortion on perception. The sequence $[h_0, h_1, h_2, \cdots]$ of (14.48) is now simply the impulse response of $H(z)$. The parameters $\gamma_1$ and $\gamma_2$ are selected to approximate perception where $1 \geq \gamma_1 > \gamma_2 > 0$. The filter $A(z/\gamma_1)$ deemphasizes the envelope of the power spectral density, which corresponds to decreasing the importance of spectral peaks. The filter $1/A(z/\gamma_2)$ undoes some of this emphasis for a smoothed version of the spectral envelope. The effect is roughly that $1/A(z/\gamma_2)$ limits the spectral reach of the deemphasis $A(z/\gamma_1)$. In other words, the deemphasis of the spectrum is made into a local effect.

To understand coders of the transform model family, it is useful to interpret (14.46) in the frequency domain. We write the discrete Fourier transform (DFT) as the unitary matrix $\mathbf{F}$ and the define a frequency-domain weighting matrix $\mathbf{W}$ such that

$$\mathbf{H}\boldsymbol{x}^k = \mathbf{F}^{\mathrm{H}}\mathbf{W}\mathbf{F}\boldsymbol{x}^k , \tag{14.51}$$

The matrix $\mathbf{W}$ provides a weighting of the frequency-domain vector $\mathbf{F}\boldsymbol{x}^k$. If, for the purpose of our discussion, we neglect the difference between circular and linear convolution and if $\mathbf{H}$ represents a filtering operation (convolution) as in (14.48), then $\mathbf{W}$ is diagonal. To account for perception, we must adapt $\mathbf{W}$ to the input vector $\boldsymbol{x}^k$ (or equivalently, to the frequency-domain vector $\mathbf{F}\boldsymbol{x}^k$) and it becomes a function $\mathbf{W}(\boldsymbol{x}^k)$: Equation (14.50) could be used as a particular mechanism for such weighting. However, in the transform coding context, so-called *masking* methods, which are described in Sect. 14.5.2, are typically used to find $W(\boldsymbol{x}^k)$.

As mentioned before, the random vector $\boldsymbol{Y}^m = \mathbf{H}\boldsymbol{X}^k$ (or, equivalently, the vector $\mathbf{W}\mathbf{F}\boldsymbol{X}^k$) can be considered as a perceptual-domain description. Assuming smooth behavior of $\mathbf{H}(\boldsymbol{x}^k)$ as a function of $\boldsymbol{x}^k$, this domain can then be used as the domain for coding. A codebook must be defined for the perceptual-domain vector $\boldsymbol{Y}^m$ and we select entries from this codebook with the unweighted squared error criterion. This approach is common in transform coding. When this coding in the perceptual domain is used, the distortion measure does not vary with the vector $\boldsymbol{y}^m$, and a uniform quantizer is optimal for $\boldsymbol{Y}^m$ for the constrained-entropy case. If the mapping to the perceptual domain is unique and invertible (which is not guaranteed by the formulation), then $\boldsymbol{y}^m = \mathbf{H}(\boldsymbol{x}^k)\boldsymbol{x}^k$ ensures that $\boldsymbol{x}^k$ is specified when $\boldsymbol{y}^m$ is known and only indices to the codebook for $\boldsymbol{Y}^m$ need to be encoded. In practice, the inverse mapping may not be unique, resulting in problems at block boundaries and the inverse may be difficult to compute. As a result it is common practice to quantize and transmit the weighting $\mathbf{W}$, e.g., [14.57, 58].

### 14.5.2 Masking Models and Squared Error

Extensive quantitative knowledge of auditory perception exists and much of the literature on quantitative descriptions of auditory perception relates to the concept of *masking*, e.g., [14.59–63]. The masking-based description of the operation of the auditory periphery can be used to include the effect of auditory perception in speech and audio coding. Let us define an arbitrary signal that we call the *masker*. The masker implies a set of second signals, called *maskees*, which are defined as signals that are not audible when presented in the presence of the masker. That is, the maskee is below the *masking threshold*. Masking explains, for example, why a radio must be made louder in a noisy environment such as a car. We can think of masking as being

Amazon EX1010 Page 324

a manifestation of the internal precision of the auditory periphery.

In general, laws for the masking threshold are based on psychoacoustic measurements for the masker and maskee signals that are constructed independently and then added. However, it is clear that the coding error is correlated to the original signal. In the context of masking it is a commonly overlooked fact that, for ideal coding, the coding error signal is, under certain common conditions, independent of the reconstructed signal [14.12]. Thus, a reasonable objective of audio and speech coding is to ensure that the coding error signal is below the masking threshold of the *reconstructed* signal.

Masking is quantified in terms of a so-called masking curve. We provide a generalized definition of such a curve. Let us consider a signal vector $\boldsymbol{x}^k$ with $k$ samples that is defined in $\mathbb{R}^k$. We define a perceived-error measurement domain by any invertible mapping $\mathbb{R}^k \to \mathbb{R}^m$. Let $\{\boldsymbol{e}_i^m\}_{i\in\{0,\cdots,m-1\}}$ be the unit-length basis vectors that span $\mathbb{R}^m$. We then define the $m$-dimensional *masking curve* as [14.64] $JND_i$, $i \in 0, \cdots, m-1$, where the scalar $JND_i$ is the *just-noticeable difference* (JND) for the basis vector $\boldsymbol{e}_i^m$. That is, the vector $JND_i\,\boldsymbol{e}_i^m$ is precisely at the threshold of being audible for the given signal vector.

Examples of the masking curve can be observed in the time and the frequency domain. The frequency-domain representation of $\boldsymbol{x}^k$ is $\mathbf{F}\boldsymbol{x}^k$. *Simultaneous* masking is defined as the masking curve for $\mathbf{F}\boldsymbol{x}^k$, i. e., the just-noticeable amplitudes for the frequency unit vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, etc. In the time domain we refer to *nonsimultaneous* (or *forward* and *backward*) masking depending on whether the time index $i$ of the unit vectors $\boldsymbol{e}_i$ is prior to or after the main event in the masker (e.g., an onset). Both the time-domain (temporal) and frequency-domain masking curves are asymmetric and dependent on the loudness of the masker. A loud sound leads to a rapid decrease in auditory acuity, followed by a slow recovery to the default level. The recovery may take several hundreds of ms and causes forward masking. The decrease in auditory acuity before a loud sound, backward masking, extends only over very short durations (at most a few ms). Similar asymmetry occurs in the frequency domain, i. e., in simultaneous masking. Let us consider a tone. The auditory acuity is decreased mostly at frequencies higher than the tone. The acuity increases more rapidly from the masker when moving towards lower frequencies than when moving towards higher frequencies, which is related to the decrease in frequency resolution with increasing frequency. A significant difference exists in the masking between tonal and noise-like signals. We refer to [14.63, 65, 66] for further information on masking.

The usage of masking is particularly useful for coding in the perceptual domain with a constraint that the quality is to be transparent (at least according to the perceptual knowledge provided). For example, consider a transform coder (based on either the discrete cosine transform or the DFT). In this case, the quantization step size can be set to be the JND as provided by the simultaneous masking curve [14.57].

Coders are commonly subject to a bit-rate constraint, which means knowledge of the masking curve is not sufficient. A distortion criterion must be defined based on the perceptual knowledge given. A common strategy in audio coding to account for simultaneous masking is to use a weighted squared error criterion, with a diagonal weighting matrix $\mathbf{H}$ that is reciprocal of the masking threshold [14.27, 58, 67–70]. In fact, this is a general approach that is useful to convert a masking curve in any measurement domain:

$$\mathbf{H} = \begin{pmatrix} \frac{1}{JND_1} & 0 & \cdots \\ 0 & \frac{1}{JND_2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} , \tag{14.52}$$

where it is understood that the weighting matrix $\mathbf{H}$ is defined in the measurement domain. To see this consider the effect of the error vector $JND_i\,\boldsymbol{e}_i^m$ on the squared error:

$$\begin{aligned} \eta &= JND_i^2\,\boldsymbol{e}_i^{m\mathrm{H}}\mathbf{H}^{\mathrm{H}}\mathbf{H}\boldsymbol{e}_i^m \\ &= JND_i^2\;JND_i^{-2} = 1 . \end{aligned} \tag{14.53}$$

Thus, the points on the masking curve are defined as the amplitudes of basis vectors that lead to a unit distortion. This is a reasonable motivation for the commonly used reciprocal-weighting approach for the squared-error criterion defined by the weighting described in (14.52). However, it should be noted that for this formulation the distortion measure does not vanish below the masking threshold. A more-complex approach where the distortion measure does vanish below the JND is given in [14.71].

### 14.5.3 Auditory Models and Squared Error

The weighting procedure of (14.52) (possibly in combination with the transform to the measurement domain)

is an operation that transforms the signal to a perceptually relevant domain. Thus, the operation can be interpreted as a simple auditory model. Sophisticated models of the auditory periphery that directly predict the input to the auditory nerve have also been developed, e.g., [14.72–77]. Despite the existence of such quantitative models of perception, their application in speech coding has been limited. Only a few examples [14.78,79] of the explicit usage of existing quantitative knowledge of auditory perception in speech coding speech exist. In contrast, in the field of audio coding the usage of quantitative auditory knowledge is common. Transform coders can be interpreted as methods that perform coding in the perceptual domain, using a simple perceptual model, usually based on (simultaneous) masking results.

We can identify a number of likely causes for the lack of usage of auditory knowledge in speech coding. First, the structure of speech coders and the constraint on computational complexity naturally leads to speech-coding-specific models of auditory perception, such as (14.50). The parameters of these simple speech-coding-based models are optimized directly based on coding performance. Second, the perception of the periodicity nature of voiced speech, often referred to as the perception of *pitch*, is not well understood in a quantitative manner. It is precisely the distortion associated with the near-periodic nature of voiced speech that is often critical for the perceived quality of the reconstructed signal. An argument against using a quantitative model based on just-noticeable differences (JNDs) is that JNDs are often exceeded significantly in speech coding. While the weighting of (14.52) is reasonable near the JND threshold value, it may not be accurate in the actual operating region of the speech coder. Major drawbacks of using sophisticated models based on knowledge of the auditory periphery are that they tend to be computationally expensive, have significant latency, and often lead to a representation that has many more dimensions than the input signal. Moreover the complexity of the model structure makes inversion difficult, although not impossible [14.80].

The complex structure of auditory models that describe the functionality of the auditory periphery is time invariant. We can replace it by a much simpler structure at the cost of making it time variant. That is, the mapping from the speech domain to the perceptual domain can be simplified by approximating this mapping as locally linear [14.79]. Such an approximation leads to the *sensitivity matrix approach*, which was first introduced in a different context by [14.53] and described in a rigorous general manner in [14.81]. If a mapping from the speech domain vector $\boldsymbol{x}^k$ to an auditory domain vector $\boldsymbol{y}^m$ (as associated with a particular model of the auditory periphery) can be approximated as locally linear, then for a small coding error $\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k$, we can write $\boldsymbol{y}^m - \hat{\boldsymbol{y}}^m$ as a matrix multiplication of $\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k$:

$$\boldsymbol{y}^m - \hat{\boldsymbol{y}}^m \approx \mathbf{H}\left(\boldsymbol{x}^k - \hat{\boldsymbol{x}}^k\right), \tag{14.54}$$

where $\mathbf{H}$ is an $m \times k$ matrix. This means that, for the set of codebook vectors from $C_{\boldsymbol{X}^k}$ that is sufficiently close to $\boldsymbol{x}^k$, the squared-error criterion of (14.46) forms an approximation to the psychoacoustic measure. Moreover, selecting the nearest codebook entry from $C_{\boldsymbol{X}^k}$ using (14.46) results in the globally optimal codebook vector for the input vector $\boldsymbol{x}^k$. In the sensitivity matrix approach, the first step for each speech vector $\boldsymbol{x}^k$ is to find the $k \times k$ *sensitivity matrix* $\mathbf{H}^{\mathrm{T}}\mathbf{H}$. This operation is based on an analysis of the distortion criterion [14.79]. Once this has been done, the selection of the codebook entries is similar to that for a signal-invariant weighted squared-error distortion measure.

In the sensitivity matrix approach, the matrix $\mathbf{H}$ is a function of the past and future signal:

$$\mathbf{H} = \mathbf{H}(\cdots, \boldsymbol{x}_{i-1}^k, \boldsymbol{x}_i^k, \boldsymbol{x}_{i+1}^k, \cdots), \tag{14.55}$$

where $\boldsymbol{x}_i^k$ is the current speech vector, $\boldsymbol{x}_{i-1}^k$ is the previous speech vector, etc. To avoid the introduction of latency, the future speech vectors can be replaced by a prediction of these vectors from the present and past speech vectors.

The sensitivity matrix approach requires that the mapping from speech domain to perceptual domain is continuous and differentiable, which is not the case for psychoacoustic models. The approximation of such discontinuities by continuous functions generally leads to satisfactory results.

The sensitivity matrix approach is well motivated in the context of a speech-domain codebook $C_{\boldsymbol{X}^k}$ and a criterion that consists of a perceptual transform followed by the squared error criterion. The search through the speech-domain codebook with a perceptual criterion then reduces to searching with a weighted squared-error criterion.

The benefit of the sensitivity matrix approach is not so obvious if the signal vector codebook, $C_{\boldsymbol{X}^k}$, is defined in the perceptual domain. However, it can be useful if the perceptual transform is known to the decoder (by, for example, transmission of an index). The perceptual domain often has higher dimensionality than the corresponding

speech block. The singular-value decomposition of $\mathbf{H}$ can then be used to reduce the dimensionality of the perceptual domain error vector to $k$. Let $\mathbf{H} = \mathbf{VDU}$ be a singular value decomposition, where $\mathbf{V}$ is an $m \times m$ unitary matrix, $\mathbf{D}$ is a $m \times k$ diagonal matrix and $\mathbf{U}$ is a unitary $k \times k$ matrix

### 14.5.4 Distortion Measure and Coding Architecture

As we have seen, the distortion measure has a significant impact on the architecture of a speech coder. In this subsection, we summarize the above discussion from a codec-architecture viewpoint.

#### Speech-Domain Codebook

The most straightforward architecture for a speech coder is to define the codebook in the speech domain and use an appropriate distortion measure during encoding and during training of the codebook. In general, the measure is adaptive. This approach, which is shown in Fig. 14.1, is most common in speech coding. An advantage is that the decoder does not require knowledge of the time-varying distortion measure.

We saw in Sect. 14.3.4 that it can be advantageous to use speech codebooks that are associated with models. This simplifies the codebook structure and, in the case of constrained-resolution coding, its size. The codebooks can be generated in real time as, for example, in the case of linear-prediction (autoregressive model)-based analysis-by-synthesis coding [14.16, 55].

The underlying aim of the speech-domain codebook architecture is generally to approximate auditory perception by an adaptively weighted squared-error criterion. Usually, this criterion is heuristic and based on tuning within the context of the coder (as is typically done for (14.50)). However, as shown in Sect. 14.5.3, the sensitivity analysis method facilitates the usage of complex auditory models. This approach requires, at least in principle, no further tuning.

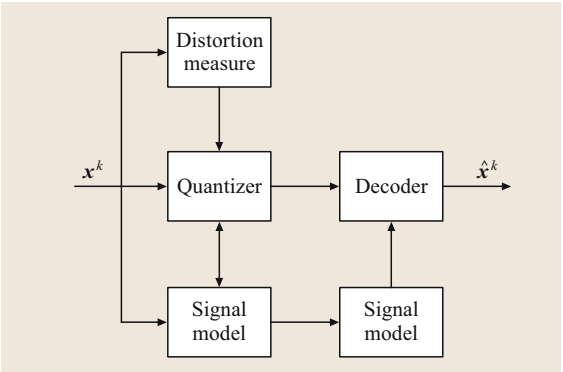



**Fig. 14.1** Common architecture for coding with a distortion measure. A signal quantization index and a signal model index are transmitted

Disadvantages of the weighted squared-error criterion are its computational complexity and that, in contrast to the unweighted squared-error measure, it does not lead to uniform codebooks for the constrained-entropy case. In the context of autoregressive-model-based analysis-by-synthesis coding, many procedures have been developed to reduce the computational complexity of the weighted squared-error criterion [14.18, 19, 82].

#### Perceptual-Domain Codebook

As an alternative to defining the codebook in the speech domain, we can define the codebook in a perceptual domain, as is shown in Fig. 14.2. We define a perceptual domain as a domain where the unweighted squared error criterion can be applied. The most elegant paradigm requires no information about the speech vector other than its index in the perceptual domain codebook. This elegance applies if the mapping to the perceptual domain is injective (one to one). Then if $\boldsymbol{y}^m$ is known, $\boldsymbol{x}^k$ is also known. An example is an auditory model that is a weighted DFT or DCT with a one-to-one function $\mathbb{R}^k \to \mathbb{R}^k$ that maps $\boldsymbol{x}^k$ into $\boldsymbol{y}^k$. An inverse func-

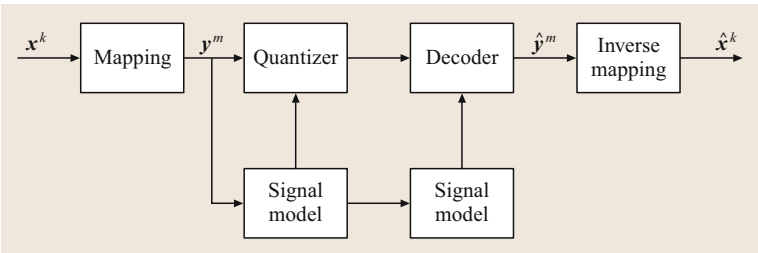



**Fig. 14.2** Ideal architecture for coding in the perceptual domain, with invertible mapping. A perceptual-domain quantization index and a signal model index are transmitted. The signal model can be omitted

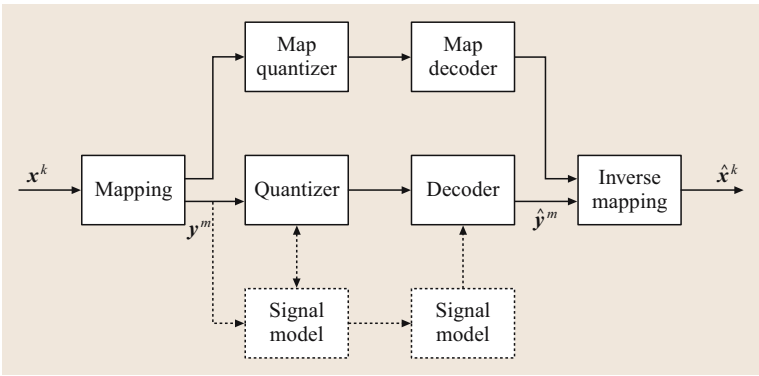



**Fig. 14.3** Architecture for coding in the perceptual domain with encoded mapping. This architecture is common in transform coding. A mapping index, a perceptual-domain quantization index, and a signal model index are transmitted. The signal model is commonly omitted

tion can be derived by the decoder from the quantized vector $\hat{\boldsymbol{y}}^{\mathrm{k}}$.

The non-uniqueness of the auditory mapping from the speech domain to the auditory domain results in practical problems. Particularly if the models are not accurate, the nonuniqueness can result in mismatches between coding blocks and severe audible distortion.

The uniqueness issue for the mapping can be solved, at the cost of increased rate, by transmitting information about the mapping as is shown in Fig. 14.3. For example, for transform coders used for audio signals, the masking curve is commonly transmitted, e.g., [14.52, 58, 67–69]. To reduce the rate required, this is commonly done on a per-frequency-band basis. The bands generally are uniformly spaced on an equivalent rectangular bandwidth (ERB) or mel scale. In practice the masking curve is not always transmitted directly, but a maximum amplitude, and the number of quantization levels for each band are transmitted, e.g., [14.27, 68, 70].

## 14.6 Summary

This chapter discussed the principles underlying the transmission of speech (and audio) signals. The main attributes of coding, rate, quality, robustness to channel errors, delay, and computational complexity were discussed first. We then provided a generic perspective of speech coding.

Each block of speech samples was described as a random vector. Information about the vector was transmitted in the form of a codebook index. The relation between the reconstruction vector density and the data density was given. We then modeled the probability density of the speech vector as a weighted sum of component probability density functions, each describing the speech vector probability density for a particular speech sound. Each such component density function corresponds to a *model*. In the case of linear-prediction-based (equivalent to autoregressive-model-based) coding, each component function (and thus model) is characterized by a set of predictor parameters. The approach results in the standard two-step speech coding approach, in which we first extract the model and then code the speech vector given the model. We showed that this approach leads to standard distortion measures used for the quantization of the predictor parameters.

We showed that the number of models (component probability density functions) is independent of the overall coding rate. Thus, the rate spent on linear-prediction parameters in linear-predictive coding should not vary with rate (at least when the rate is high). It was shown that practical coders indeed have this behavior. We emphasized also that the rate allocated to the model (the predictor parameters) is not a direct function of a perceptual threshold, but the result of an optimal trade-off between the rate allocated for the speech given the model and the rate allocated for the model. We showed that it is possible to calculate the rate allocation for the model (the bit allocation for the predictor parameters) and that the result provided is close to practical codec configurations. We discussed analysis-by-synthesis coding as a particular application of the two-stage coding method. We noted that analysis-by-synthesis coding is not optimal because the speech-domain codebook has suboptimal quantization cell shapes.

Amazon EX1010 Page 328

Finally we discussed how perception can be integrated into the coding structure. We distinguished coding in a perceptually relevant domain, which is commonly used in audio coding, from coding in the speech-signal domain, which is commonly used in speech coding. The advantage of coding in the perceptual domain is that a simple squared-error criterion can be used. However, in practice the method generally requires some form of encoding of the mapping, so that the decoder can perform an inverse mapping. Improved mapping procedures may change this requirement.

## References


14.1 W.B. Kleijn, K.K. Paliwal: An introduction to speech coding. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995) pp. 1–47

14.2 R.V. Cox: Speech coding standards. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995) pp. 49–78

14.3 R. Salami, C. Laflamme, J. Adoul, A. Kataoka, S. HAyashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, Y. Shoham: Design and description of CS-ACELP: a toll quality 8 kb/s speech coder, IEEE Trans. Speech Audio Process. **6**(2), 116–130 (1998)

14.4 B. Bessette, R. Salami, R. Lefebvre, M. Jelinek, J. Rotola-Pukkila, J. Vainio, H. Mikkola: The adaptive multirate wideband speech codec (amr-wb), IEEE Trans. Speech Audio Process. **6**(8), 620–636 (2002)

14.5 ITU-T Rec. P.800: *Methods for Subjective Determination of Transmission Quality* (1996)

14.6 A.W. Rix: Perceptual speech quality assessment – a review, Proc. IEEE ICASSP, Vol. 3 (2004) pp. 1056–1059

14.7 S. Möller: *Assessment and Prediction of Speech Quality in Telecommunications* (Kluwer Academic, Boston 2000)

14.8 P. Kroon: Evaluation of speech coders. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995) pp. 467–493

14.9 W. Stallings: *High-speed networks: TCP/IP and ATM design principles* (Prentice Hall, Englewood Cliffs 1998)

14.10 Information Sciences Institute: Transmission control protocol, IETF RFC793 (1981)

14.11 J. Postel: User datagram protocol, IETF RFC768 (1980)

14.12 T.M. Cover, J.A. Thomas: *Elements of Information Theory* (Wiley, New York 1991)

14.13 N. Kitawaki, K. Itoh: Pure delay effects on speech quality in telecommunications, IEEE J. Sel. Area. Comm. **9**(4), 586–593 (1991)

14.14 J. Cox: The minimum detectable delay of speech and music, Proc. IEEE ICASSP, Vol. 1 (1984) pp. 136–139

14.15 J. Chen: A robust low-delay CELP speech coder at 16 kb/s. In: *Advances in Speech Coding*, ed. by B.S. Atal, V. Cuperman, A. Gersho (Kluwer Academic, Dordrecht 1991) pp. 25–35

14.16 B.S. Atal, M.R. Schroeder: Stochastic coding of speech at very low bit rates, Proc. Int. Conf. Comm. (1984) pp. 1610–1613

14.17 J.-P. Adoul, P. Mabilleau, M. Delprat, S. Morisette: Fast CELP coding based on algebraic codes, Proc. IEEE ICASSP (1987) pp. 1957–1960

14.18 I.M. Trancoso, B.S. Atal: Efficient procedures for selecting the optimum innovation in stochastic coders, IEEE Trans. Acoust. Speech **38**(3), 385–396 (1990)

14.19 W.B. Kleijn, D.J. Krasinski, R.H. Ketchum: Fast methods for the CELP speech coding algorithm, IEEE Trans. Acoust. Speech **38**(8), 1330–1342 (1990)

14.20 T. Lookabough, R. Gray: High-resolution theory and the vector quantizer advantage, IEEE Trans. Inform. Theory **IT-35**(5), 1020–1033 (1989)

14.21 S. Na, D. Neuhoff: Bennett's integral for vector quantizers, IEEE Trans. Inform. Theory **41**(4), 886–900 (1995)

14.22 S.P. Lloyd: Least squares quantization in PCM, IEEE Trans. Inform. Theory **IT-28**, 129–137 (1982)

14.23 Y. Linde, A. Buzo, R.M. Gray: An algorithm for vector quantizer design, IEEE Trans. Commun. **COM-28**, 84–95 (1980)

14.24 P. Chou, T. Lookabough, R. Gray: Entropy-constrained vector quantization, IEEE Trans. Acoust. Speech **38**(1), 31–42 (1989)

14.25 A. Gersho: Asymptotically optimal block quantization, IEEE Trans. Inform. Theory **25**, 373–380 (1979)

14.26 P. Swaszek, T. Ku: Asymptotic performance of unrestricted polar quantizers, IEEE Trans. Inform. Theory **32**(2), 330–333 (1986)

14.27 R. Vafin, W.B. Kleijn: Entropy-constrained polar quantization and its application to audio coding, IEEE Trans. Speech Audio Process. **13**(2), 220–232 (2005)

14.28 J.J. Rissanen, G. Langdon: Arithmetic coding, IBM J. Res. Devel. **23**(2), 149–162 (1979)

14.29 J. Rissanen: Modeling by the shortest data description, Automatica **14**, 465–471 (1978)

14.30 J. Rissanen: A universal prior for integers and estimation by minimum description length, Ann. Stat. **11**(2), 416–431 (1983)


Amazon EX1010 Page 329

14.31 P. Grunwald: A tutorial introduction to the minimum description length principle. In: *Advances in Minimum Description Length: Theory and Applications*, ed. by P. Grunwald, I.J. Myung, M. Pitt (MIT, Boston 2005)

14.32 A. Barron, T.M. Cover: Minimum complexity density estimation, IEEE Trans. Inform. Theory **37**(4), 1034–1054 (1991)

14.33 A.H. Gray, J.D. Markel: Distance measures for speech process, IEEE Trans. Acoust. Speech Signal Process. **ASSP-24**(5), 380–391 (1976)

14.34 R. Hagen, P. Hedelin: Low bit-rate spectral coding in CELP a new LSP method, Proc. IEEE ICASSP (1990) pp. 189–192

14.35 K.K. Paliwal, B.S. Atal: Efficient vector quantization of LPC parameters at 24 bits/frame, IEEE Trans. Speech Audio Process. **1**(1), 3–14 (1993)

14.36 C. Xydeas, C. Papanastasiou: Split matrix quantization of lpc parameters, IEEE Trans. Speech Audio Process. **7**(2), 113–125 (1999)

14.37 A. Subramaniam, B. Rao: Speech LSF quantization with rate independent complexity, bit scalability, and learning, Proc. IEEE ICASSP (2001) pp. 705–708

14.38 U. Grenander, G. Szego: *Toeplitz Forms and their Applications* (Chelsea, New York 1984)

14.39 F. Itakura, S. Saito: Speech information compression based on the maximum likelihood estimation, J. Acoust. Soc. Jpn. **27**(9), 463 (1971)

14.40 S. Saito, K. Nakata: *Fundamentals of Speech Signal Process* (Academic, New York 1985)

14.41 P.J. Brockwell, R.A. Davis: *Time Series: Theory and Methods* (Springer, New York 1996)

14.42 F. Itakura, S. Saito: Analysis Synthesis Telephony Based Upon the Maximum Likelihood Method, Reports of 6th Int. Cong. Acoust.,C-5-5, C17-20, ed. by Y. Kohasi (1968)

14.43 R.M. Gray, A. Buzo, A.H. Gray, Y. Matsuyama: Distortion measures for speech process, IEEE Trans. Acoust. Speech Signal Process. **ASSP-28**(4), 367–376 (1980)

14.44 K.K. Paliwal, W.B. Kleijn: Quantization of LPC parameters. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995) pp. 433–466

14.45 W.R. Gardner, B.D. Rao: Noncausal all-pole modeling of voiced speech, IEEE Trans. Speech Audio Process. **5**(1), 1–10 (1997)

14.46 M. Nilsson, W.B. Kleijn: Shannon entropy estimation based on high-rate quantization theory, Proc. EUSIPCO (2004) pp. 1753–1756

14.47 M. Nilsson: *Entropy and Speech* (Royal Institute of Technology, Stockholm 2006), Ph.D. dissertation, KTH

14.48 C. Lamm: *Improved Spectral Estimation in Speech Coding* (Lund Institute of Technology (LTH), Lund 1998), Master's thesis

14.49 K.L.C. Chan: Split-dimension vector quantization of parcor coefficients for low bit rate speech coding, IEEE Trans. Speech Audio Process. **2**(3), 443–446 (1994)

14.50 A. Subramaniam, B.D. Rao: PDF optimized parametric vector quantization of speech line spectral freuencies, IEEE Speech Coding Workshop (Delavan 2000) pp. 87–89

14.51 P. Hedelin, J. Skoglund: Vector quantization based on Gaussian mixture models, IEEE Trans. Speech Audio Process. **8**(4), 385–401 (2000)

14.52 S. Srinivasan, J. Samuelsson, W.B. Kleijn: Speech enhancement using a-priori information with classified noise codebooks, Proc. EUSIPCO (2004) pp. 1461–1464

14.53 W.R. Gardner, B.D. Rao: Optimal distortion measures for the high rate vector quantization of LPC parameters, Proc. IEEE ICASSP (1995) pp. 752–755

14.54 M.Y. Kim, W.B. Kleijn: KLT-based adaptive classified vector quantization of the speech signal, IEEE Trans. Speech Audio Process. **12**(3), 277–289 (2004)

14.55 P. Kroon, E.F. Deprettere: A class of analysis-by-synthesis predictive coders for high quality speech coding at rates between 4.8 and 16 kbit/s, IEEE J. Sel. Area. Commun. **6**(2), 353–363 (1988)

14.56 J. Chen, A. Gersho: Real-time vector APC speech coding at 4–800 bps with adaptive postfiltering, Proc. IEEE ICASSP (1987) pp. 2185–2188

14.57 J. Johnston: Transform coding of audio signals using perceptual noise criteria, IEEE J. Sel. Area. Commun. **6**(2), 314–323 (1988)

14.58 H. Malvar: Enhancing the performance of subband audio coders for speech signals, Proc. IEEE Int. Symp. on Circ. Syst., Vol. 5 (1998) pp. 98–101

14.59 R. Veldhuis: Bit rates in audio source coding, IEEE J. Sel. Area. Commun. **10**(1), 86–96 (1992)

14.60 B.C.J. Moore: Masking in the human auditory system. In: *Collected papers on digital audio bit-rate reduction*, ed. by N. Gilchrist, C. Grewin (Audio Eng. Soc., New York 1996)

14.61 B.C.J. Moore: *An Introduction to the Psychology of Hearing* (Academic, London 1997)

14.62 E. Zwicker, H. Fastl: *Psychoacoustics* (Springer Verlag, Berlin, Heidelberg 1999)

14.63 T. Painter, A. Spanias: Perceptual coding of digital audio, Proc. IEEE **88**(4), 451–515 (2000)

14.64 J.H. Plasberg, W.B. Kleijn: The sensitivity matrix: Using advanced auditory models in speech and audio processing, IEEE Trans. Speech Audio Process. **15**, 310–319 (2007)

14.65 J.L. Hall: Auditory psychophysics for coding applications. In: *The Digital Signal Processing Handbook*, ed. by V.K. Madisetti, D. Williams (CRC, Boca Raton 1998) pp. 39.1–39.25

14.66 W. Jesteadt, S.P. Bacon, J.R. Lehman: Forward masking as a function of frequency, masker level and signal delay, J. Acoust. Soc. Am. **71**(4), 950–962 (1982)

14.67 D. Sinha, J.D. Johnston: Audio compression at low bit rates using a signal adaptive switched


Amazon EX1010 Page 330

filterbank, Proc. IEEE ICASSP, Vol. 2 (1996) pp. 1053–1056
14.68 T. Verma, T. Meng: A 6 kbps to 85 kbps scalable audio coder, Proc. IEEE ICASSP, Vol. 2 (2000) pp. II877–II880
14.69 A.S. Scheuble, Z. Xiong: Scalable audio coding using the nonuniform modulated complex lapped transform, Proc. IEEE ICASSP, Vol. 5 (2001) pp. 3257–3260
14.70 R. Heusdens, R. Vafin, W.B. Kleijn: Sinusoidal modeling using psychoacoustic-adaptive matching pursuits, IEEE Signal Proc. Lett. **9**(8), 262–265 (2002)
14.71 M.Y. Kim, W.B. Kleijn: Resolution-constrained quantization with JND based perceptual-distortion measures, IEEE Signal Proc. Lett. **13**(5), 304–307 (2006)
14.72 O. Ghitza: Auditory nerve representation as a basis for speech processing. In: *Advances in Speech Signal Processing* (Dekker, New York 1992) pp. 453–485
14.73 T. Dau, D. Püschel, A. Kohlrausch: A quantitative model of the effective signal processing in the auditory system. I. Model structure, J. Acoust. Soc. Am. **99**(6), 3615–3622 (1996)
14.74 T. Dau, B. Kollmeier, A. Kohlrausch: Modeling auditory processing of amplitude modulation. I. detection and masking with narrowband carriers, J. Acoust. Soc. Am. **102**(5), 2892–2905 (1997)
14.75 G. Kubin, W.B. Kleijn: On speech coding in a perceptual domain, Proc. IEEE ICASSP, Vol. I (1999) pp. 205–208
14.76 F. Baumgarte: *Ein psychophysiologisches Gehörmodell zur Nachbildung von Wahrnehmungsschwellen für die Audiocodierung* (Univ. Hannover, Hannover 2000), Ph.D. dissertation (in German)
14.77 S. van de Par, A. Kohlrausch, G. Charestan, R. Heusdens: A new psychoacoustical masking model for audio coding applications, Proc. IEEE ICASSP (2002) pp. 1805–1808
14.78 D. Sen, D. Irving, W. Holmes: Use of an auditory model to improve speech coders, Proc. IEEE ICASSP (1993) pp. II411–II414
14.79 J.H. Plasberg, D.Y. Zhao, W.B. Kleijn: The sensitivity matrix for a spectro-temporal auditory model, Proc. EUSIPCO (2004) pp. 1673–1676
14.80 X. Yang, K. Wang, S. Shamma: Auditory representation of acoustic signals, IEEE Trans. Inform. Theory **38**(2), 824–839 (1996)
14.81 T. Linder, R. Zamir, K. Zeger: High-resolution source coding for non-difference measures: the rate-distortion function, IEEE Trans. Inform. Theory **45**(2), 533–547 (1999)
14.82 I. Gerson, M. Jasiuk: Vector sum excited linear prediction (VSELP), Proc. IEEE ICASSP (1990) pp. 461–464

Amazon EX1010 Page 331

# 15. Voice over IP: Speech Transmission over Packet Networks

J. Skoglund, E. Kozica, J. Linden, R. Hagen, W. B. Kleijn



The emergence of packet networks for both data and voice traffic has introduced new challenges for speech transmission designs that differ significantly from those encountered and handled in traditional circuit-switched telephone networks, such as the public switched telephone network (PSTN). In this chapter, we present the many aspects that affect speech quality in a voice over IP (VoIP) conversation. We also present design techniques for coding systems that aim to overcome the deficiencies of the packet channel. By properly utilizing speech codecs tailored for packet networks, VoIP can in fact produce a quality higher than that possible with PSTN.

15.1 **Voice Communication** ..... 307
15.1.1 Limitations of PSTN ..... 307
15.1.2 The Promise of VoIP ..... 308
15.2 **Properties of the Network** ..... 308
15.2.1 Network Protocols ..... 308
15.2.2 Network Characteristics ..... 309
15.2.3 Typical Network Characteristics ..... 312
15.2.4 Quality-of-Service Techniques ..... 313
15.3 **Outline of a VoIP System** ..... 313
15.3.1 Echo Cancelation ..... 314
15.3.2 Speech Codec ..... 315
15.3.3 Jitter Buffer ..... 315
15.3.4 Packet Loss Recovery ..... 316
15.3.5 Joint Design of Jitter Buffer and Packet Loss Concealment ..... 316
15.3.6 Auxiliary Speech Processing Components ..... 316
15.3.7 Measuring the Quality of a VoIP System ..... 317
15.4 **Robust Encoding** ..... 317
15.4.1 Forward Error Correction ..... 317
15.4.2 Multiple Description Coding ..... 320
15.5 **Packet Loss Concealment** ..... 326
15.5.1 Nonparametric Concealment ..... 326
15.5.2 Parametric Concealment ..... 327
15.6 **Conclusion** ..... 327
**References** ..... 328

## 15.1 Voice Communication

Voice over internet protocol (IP), known as VoIP, represents a new voice communication paradigm that is rapidly establishing itself as an alternative to traditional telephony solutions. While VoIP generally leads to cost savings and facilitates improved services, its quality has not always been competitive. For over a century, voice communication systems have used virtually exclusively circuit-switched networks and this has led to a high level of maturity. The end-user has been accustomed to a telephone conversation that has *consistent* quality and low delay. Further, the user expects a signal that has a narrow-band character and, thus, accepts the limitations present in traditional solutions, limitations that VoIP systems lack.

A number of fundamental differences exist between traditional telephony systems and the emerging VoIP systems. These differences can severely affect voice quality if not handled properly. This chapter will discuss the major challenges specific to VoIP and show that, with proper design, the quality of a VoIP solution can be significantly better than that of the public switched telephone network (PSTN). We first provide a broad overview of the issues that affect end-to-end quality. We then present some general techniques for designing speech coders that are suited for the challenges imposed by VoIP. We emphasize multiple description coding, a powerful paradigm that has shown promising performance in practical systems, and also facilitates theoretical analysis.

### 15.1.1 Limitations of PSTN

Legacy telephony solutions are narrow-band. This property imposes severe limitations on the achievable quality. In fact, in traditional telephony applications, the speech bandwidth is restricted more than the inherent

limitations of narrow-band coding at an 8 kHz sampling rate. Typical telephony speech is band-limited to 300–3400 Hz. This bandwidth limitation explains why we are used to expect telephony speech to sound weak, unnatural, and lack crispness. The final connection to most households (the so-called local loop) is generally analog, by means of two-wire copper cables, while entirely digital connections are typically only found in enterprise environments. Due to poor connections or old wires, significant distortion may be generated in the analog part of the phone connection, a type of distortion that is entirely absent in VoIP implementations. Cordless phones also often generate significant analog distortion due to radio interference and other implementation issues.

### 15.1.2 The Promise of VoIP

It is clear that significant sources of quality degradation exist in the PSTN. VoIP can be used to avoid this distortion and, moreover, to remove the basic constraints imposed by the analog connection to the household.

As mentioned above, even without changing the sampling frequency, the bandwidth of the speech signal can be enhanced over telephony band speech. It is possible to extend the lower band down to about 50 Hz, which improves the base sound of the speech signal and has a major impact on the naturalness, presence, and comfort in a conversation. Extending the upper band to almost 4 kHz (a slight margin for sampling filter roll-off is necessary) improves the naturalness and *crispness* of the sound. All in all, a fuller, more-natural voice and higher intelligibility can be achieved just by extending the bandwidth within the limitations of narrow-band speech. This is the first step towards *face-to-face* communication quality offered by wide-band speech.

In addition to having an extended bandwidth, VoIP has fewer sources of analog distortion, resulting in the possibility to offer significantly better quality than PSTN within the constraint of an 8 kHz sampling rate. Even though this improvement is often clearly noticeable, far better quality can be achieved by taking the step to wide-band coding.

One of the great advantages of VoIP is that there is no need to settle for narrow-band speech. In principle, compact disc (CD) quality is a reasonable alternative, allowing for the best possible quality. However, a high sampling frequency results in a somewhat higher transmission bandwidth and, more importantly, imposes tough requirements on hardware components. The bandwidth of speech is around 10 kHz [15.1], implying a sampling frequency of 20 kHz for good quality. However, 16 kHz has been chosen in the industry as the best trade-off between bit rate and speech quality for wide-band speech coding.

By extending the upper band to 8 kHz, significant improvements in intelligibility and quality can be achieved. Most notably, fricative sounds such as [s] and [f], which are hard to distinguish in telephony band situations, sound natural in wide-band speech.

Many hardware factors in the design of VoIP devices affect speech quality as well. Obvious examples are microphones, speakers, and analog-to-digital converters. These issues are also faced in regular telephony, and as such are well understood. However, since the limited signal bandwidth imposed by the traditional network is the main factor affecting quality, most regular phones do not offer high-quality audio. Hence, this is another area of potential improvement over the current PSTN experience.

There are other important reasons why VoIP is rapidly replacing PSTN. These include cost and flexibility. VoIP extends the usage scenarios for voice communications. The convergence of voice, data, and other media presents a field of new possibilities. An example is web collaboration, which combines application sharing, voice, and video conferencing. Each of the components, transported over the same IP network, enhances the experience of the others.

## 15.2 Properties of the Network

### 15.2.1 Network Protocols

Internet communication is based on the internet protocol (IP) which is a network layer (layer 3) protocol according to the seven-layer open systems interconnection (OSI) model [15.2]. The physical and data link layers reside below the network layer. On top of the network layer protocol, a transport layer (OSI layer 4) protocol is deployed for the actual data transmission. Most internet applications are using the transmission control protocol (TCP) [15.3] as the transport protocol. TCP is very robust since it allows for retransmission

Amazon EX1010 Page 333

in the case that a packet has been lost or has not arrived within a specific time. However, there are obvious disadvantages of deploying this protocol for real-time, two-way communication. First and foremost, delays can become very long due to the retransmission process. Another major disadvantage of TCP is the increased traffic load due to transmission of acknowledgements and retransmitted packets. A better choice of transport layer protocol for real-time communication such as VoIP is the user datagram protocol (UDP) [15.4]. UDP does not implement any mechanism for retransmission of packets and is thus more efficient than TCP for real-time applications. On top of UDP, another Internet Engineering Task Force (IETF) protocol, the real-time transport protocol (RTP) [15.5], is typically deployed. This protocol includes all the necessary mechanisms to transport data generated by both standard codecs as well as proprietary codecs.

It should be mentioned that recently it has become common to transmit VoIP data over TCP to facilitate communication through firewalls that would normally not allow VoIP traffic. This is a good solution from a connectivity point of view, but introduces significant challenges for the VoIP software designer due to the disadvantages with deploying TCP for VoIP.

### 15.2.2 Network Characteristics

Three major factors associated with packet networks have a significant impact on perceived speech quality: delay, jitter, and packet loss. All three factors stem from the nature of a packet network, which provides no guarantee that a packet of speech data will arrive at the receiving end in time, or even that it will arrive at all. This contrasts with traditional telephony networks where data are rarely, or never, lost and the transmission delay is usually a fixed parameter that does not vary over time. These network effects are the most important factors distinguishing speech processing for VoIP from traditional solutions. If the VoIP device cannot address network degradation in a satisfactory manner, the quality can never be acceptable. Therefore, it is of utmost importance that the characteristics of the IP network are taken into account in the design and implementation of VoIP products as well as in the choice of components such as the speech codec. In the following sub-sections delay, jitter, and packet loss are discussed and methods to deal with these challenges are covered.

A fact often overlooked is that both sides of a call need to have robust solutions even if only one side is connected to a poor network. A typical example is a wireless device that has been properly designed to be able to cope with the challenges in terms of jitter and packet loss typical of a wireless (WiFi) network which is connecting through an enterprise PSTN gateway. Often the gateway has been designed and configured to handle network characteristics typical of a well-behaved wired local-area network (LAN) and not a challenging wireless LAN. The result can be that the quality is good in the wireless device but poor on the PSTN side. Therefore, it is crucial that all devices in a VoIP solution are designed to be robust against network degradation.

#### Delay

Many factors affect the perceived quality in two-way communication. An important parameter is the transmission delay between the two end-points. If the latency is high, it can severely affect the quality and ease of conversation. The two main effects caused by high latency are annoying talker overlap and echo, which both can cause significant reduction of the perceived conversation quality.

In traditional telephony, long delays are experienced only for satellite calls, other long-distance calls, and calls to mobile phones. This is not true for VoIP. The effects of excessive delay have often been overlooked in VoIP design, resulting in significant conversational quality degradation even in short-distance calls. Wireless VoIP, typically over a wireless LAN (WLAN), is becoming increasingly popular, but increases the challenges of delay management further.

The impact of latency on communication quality is not easily measured and varies significantly with the usage scenario. For example, long delays are not perceived as annoying in a cell-phone environment as for a regular wired phone because of the added value of mobility. The presence of echo also has a significant impact on our sensitivity to delay: the higher the latency, the lower the perceived quality. Hence, it is not possible to list a single number for how high latency is acceptable, but only some guidelines.

If the overall delay is more than about 40 ms, an echo is audible [15.6]. For lower delays, the echo is only perceived as an expected side-tone. For longer delays a well-designed echo canceler can remove the echo. For very long delays (greater than 200 ms), even if echo cancelation is used, it is hard to maintain a two-way conversation without talker overlap. This effect is often accentuated by shortcomings of the echo canceler design. If no echo is generated, a slightly higher delay is acceptable.

Amazon EX1010 Page 334

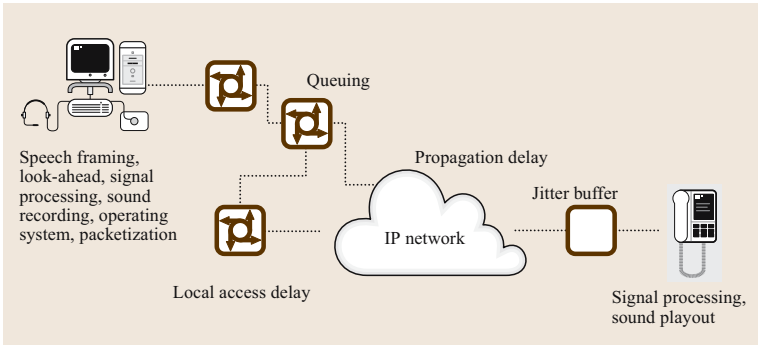



**Fig. 15.1** Main delay sources in VoIP

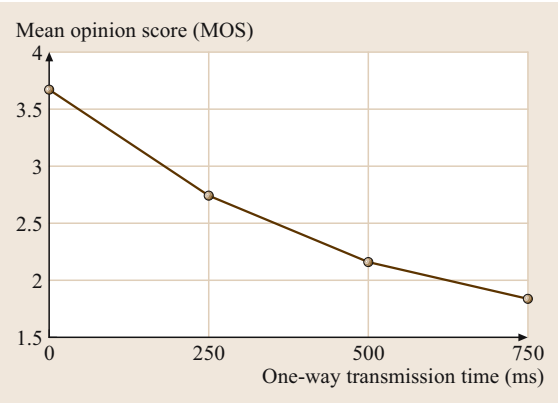



**Fig. 15.2** Effect of delay on conversational quality from ITU-T G.114

The International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) recommends in standard G.114 [15.7] that the one-way delay should be kept below 150 ms for acceptable conversation quality (Fig. 15.2 is from G.114 and shows the perceived effect on quality as a function of delay). Delays between 150 and 400 ms may be acceptable, but have an impact on the perceived quality of user applications. A latency larger than 400 ms is unacceptable.

### Packet Loss

Packet losses often occur in the routers, either due to high router load or to high link load. In both cases, packets in the queues may be dropped. Packet loss also occurs when there is a breakdown in a transmission link. The result is data link layer error and the incomplete packet is dropped. Configuration errors and collisions may also result in packet loss. In non-real-time applications, packet loss is solved at the transfer layer by retransmission using TCP. For telephony, this is not a viable solution since transmitted packets would arrive too late for use.

When a packet loss occurs some mechanism for filling in the missing speech must be incorporated. Such solutions are usually referred to as packet loss concealment (PLC) algorithms (Sect. 15.5). For best performance, these algorithms have to accurately predict the speech signal and make a smooth transition between the previous decoded speech and inserted segment.

Since packet losses occur mainly when the network is heavily loaded, it is not uncommon for packet losses to appear in bursts. A burst may consist of a series of consecutive lost packets or a period of high packet loss rate. When several consecutive packets are lost, even good PLC algorithms have problems producing acceptable speech quality.

To save transmission bandwidth, multiple speech frames are sometimes carried in a single packet, so a single lost packet may result in multiple lost frames. Even if the packet losses occur more spread out, the listening experience is then similar to that of having the packet losses occur in bursts.

### Network Jitter

The latency in a voice communication system can be attributed to algorithmic, processing, and transmission delays. All three delay contributions are constant in a conventional telephone network. In VoIP, the algorithmic and processing delays are constant, but the transmission delay varies over time. The transit time of a packet through an IP network varies due to queuing effects. The transmission delay is interpreted as consisting of two parts, one being the constant or slowly varying network delay and the other being the rapid variations on top of the basic network delay, usually referred to as jitter.

The jitter present in packet networks complicates the decoding process in the receiver device because the

decoder needs to have packets of data available at the right time instants. If the data is not available, the decoder cannot produce continuous speech. A jitter buffer is normally used to make sure that packets are available when needed.

### Clock Drift

Whether the communication end-points are gateways or other devices, low-frequency clock drift between the two can cause receiver buffer overflow or underflow. Simply speaking, this effect can be described as the two devices talking to each other having different time references. For example, the transmitter might send packets every 20 ms according to its perception of time, while the receiver's perception is that the packets arrive every 20.5 ms. In this case, for every 40th packet, the receiver has to perform a packet loss concealment to avoid buffer underflow. If the clock drift is not detected accurately, delay builds up during a call, so clock drift can have a significant impact on the speech quality. This is particularly difficult to mitigate in VoIP.

The traditional approach to address clock drift is to deploy a clock synchronization mechanism at the receiver to correct for clock drift by comparing the time stamps of the received RTP packets with the local clock. It is hard to obtain reliable clock drift estimates in VoIP because the estimates are based on averaging packet arrivals at a rate of typically 30–50 per second and because of the jitter in their arrival times. Consider for comparison the averaging on a per-sample basis at a rate of 8000 per second that is done in time-division multiplexing (TDM) networks [15.8]. In practice many algorithms designed to mitigate the clock drift effect fail to perform adequately.

### Wireless Networks

Traditionally, packet networks consisted of wired Ethernet solutions that are relatively straightforward to manage. However, the rapid growth of wireless LAN (WLAN) solutions is quickly changing the network landscape. WLAN, in particular the IEEE 802.11 family of standards [15.9], offers mobility for computer access and also the flexibility of wireless IP phones, and are hence of great interest for VoIP systems. Jitter and effective packet loss rates are significantly higher in WLAN than in a wired network, as mentioned in Sect. 15.2.3. Furthermore, the network characteristics often change rapidly over time. In addition, as the user moves physically, coverage and interference from other radio sources–such as cordless phones, Bluetooth [15.10] devices, and microwave ovens-varies. The result is that high-level voice quality is significantly harder to guarantee in a wireless network than a typical wired LAN.

WLANs are advertised as having very high throughput capacity (11 Mb/s for 802.11b and 54 Mb/s for 802.11a and 802.11g). However, field studies show that actual throughput is often only half of this, even when the client is close to the access point. It has been shown that these numbers are even worse for VoIP due to the high packet rate, with typical throughput values of 5–10% (Sect. 15.2.3).

When several users are connected to the same wireless access point, congestion is common. The result is jitter that can be significant, particularly if large data packets are sent over the same network. The efficiency of the system quickly deteriorates when the number of users increases.

When roaming in a wireless network, the mobile device has to switch between access points. In a traditional WLAN, it is common that such a hand-off introduces a 500 ms transmission gap, which has a clearly audible impact on the call quality. However, solutions are now available that cut that delay number to about 20 to 50 ms, if the user is not switching between two IP subnets. In the case of subnet roaming the handover is more complicated and no really good solutions exist currently. Therefore, it is common to plan the network in such a way that likelihood of subnet roaming is minimized.

Sensitivity to congestion is only one of the limitations of 802.11 networks. Degraded link quality, and consequently reduced available bandwidth, occurs due to a number of reasons. Some 802.11 systems operate in the unlicensed 2.4 GHz frequency range and share this spectrum with other wireless technologies, such as Bluetooth and cordless phones. This causes interference with potentially severe performance degradation since a lower connection speed than the maximum is chosen.

Poor link quality also leads to an increased number of retransmissions, which directly affects the delay and jitter. The link quality varies rapidly when moving around in a coverage area. This is a severe drawback, since a WLAN is introduced to add mobility and a wireless VoIP user can be expected to move around the coverage area. Hence, the introduction of VoIP into a WLAN environment puts higher requirements on network planning than for an all-data WLAN.

The result of the high delays that occur due to access-point congestion and bad link quality is that the packets often arrive too late to be useful. Therefore, the effective packet loss rate after the jitter buffer is typically significantly higher for WLANs than for wired LANs.



Amazon EX1010 Page 336

### 15.2.3 Typical Network Characteristics

Particularly VoIP communications that involve many *hops* are often negatively affected by delay, packet loss, and jitter. Little published work is available that describes network performance quantitatively. In one study towards improved understanding of network behavior, the transmission characteristics of the internet between several end-points were monitored over a four-week period in early 2002. Both packet loss and jitter were first measured as an average over 10 s periods. The maximum of these values over 5 min were then averaged over 7 d. Table 15.1 summarizes the results of these measurements. The table shows that significant jitter is present in long-distance IP communication, which affects speech quality significantly. Also, it was noted that, compared to Europe and the US, degradation in the quality of communications was more severe with calls to, from, and within Asia.

The ratio of infrastructure to traffic demand determines the level of resource contention. This, in turn, affects packet loss, delay, and jitter. With larger networks, packets can avoid bottlenecks and generally arrive within a reasonable time. When network conditions are less than ideal, communication quality deteriorates rapidly.

An informal test of the capacity of a wireless network was presented in [15.11]. The impact on packet loss and jitter of the number of simultaneous calls over a wireless access point with perfect coverage for all users is depicted in Figs. 15.3 and 15.4. Each call used the ITU-T G.711 codec [15.12] with a packet size of 20 ms which, including IP headers, results in a payload bandwidth of 80 kb/s. These results show that, for this access point, only five calls can be allowed if we do not allow packet loss. The results for this access point correspond to a bandwidth utilization factor of less than 10% percent. Interestingly, using a higher-compression speech codec did not increase the number of channels that can be handled. The reason is that access-point congestion depends much more on the number of packets the access point has to process than on the sum of the user bit rates. Voice packets are small and sent very frequently, which explains the low throughput for voice packets. Because of this limitation it is common to put several voice frames into the same packet, which reduces the number of packets and hence increases the throughput. However, this results both in

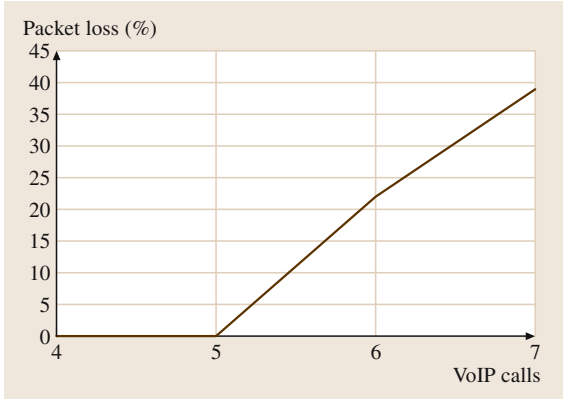



**Fig. 15.3** Effect of access point congestion on the amount of packet loss as a function of the number of simultaneous VoIP calls through one access point (after [15.11])

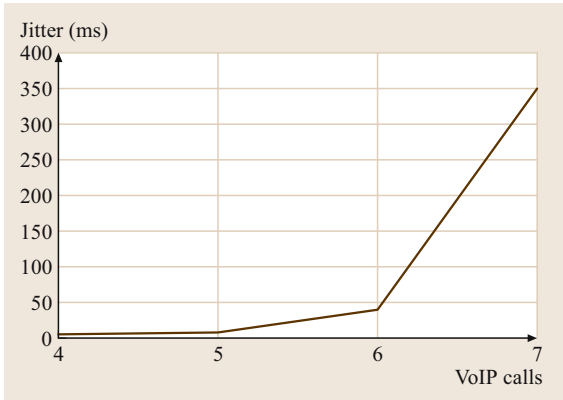



**Fig. 15.4** Effect of access point congestion on network jitter as a function of the number of simultaneous VoIP calls through one access point (after [15.11])

**Table 15.1** Results of international call performance monitoring. Long-term averages of short-term maximum values

| Connection | Packet loss (%) | Roundtrip delay (ms) | Jitter (ms) | Hops |
|---|---|---|---|---|
| Hong Kong – Urumgi, Xinjiang | 50 | 800 | 350 | 20 |
| Hong Kong – San Francisco | 25 | 240 | 250 | 17 |
| San Francisco – Stockholm | 8 | 190 | 200 | 16–18 |

Amazon EX1010 Page 337

increased delay and in an increased impact of packet loss.

### 15.2.4 Quality-of-Service Techniques

Since network imperfections have a direct impact on the voice quality, it is important that the network is properly designed and managed. Measures taken to ensure network performance are usually referred to as quality-of-service (QoS) techniques. High quality of service can be achieved by adjusting capacity, by packet prioritization, and by network monitoring.

#### Capacity Management

Generally capacity issues are related to the connection to a wide-area network or other access links. It is uncommon that a local-area network has significant problems in this respect.

#### Prioritization Techniques

By prioritizing voice over less time-critical data in routers and switches, delay and jitter can be reduced significantly without a noticeable effect on the data traffic. Many different standards are available for implementing differentiated services, including IEEE 802.1p/D [15.13] and DiffServ [15.14]. In addition, the resource reservation protocol (RSVP) [15.15] can be used to reserve end-to-end bandwidth for the voice connection.

#### Network Monitoring

As the needs and requirements of networks continuously change, it is important to implement ongoing monitoring and management of the network.

Clearly, QoS management is easily ensured in an isolated network. Challenges typically arise in cases where the traffic is routed through an unmanaged network. A particularly interesting and challenging scenario is a telecommuter connecting to the enterprise network through a virtual private network. For more information on QoS, we refer the reader to the vast literature available on this topic, e.g. [15.16].

The QoS supplement developed in the Institute of Electrical and Electronics Engineers (IEEE) is called 802.11e [15.17] and is applicable to 802.11a, 802.11b and 802.11g. The development of this standard has been quite slow, and it likely will take time before significant deployment is seen. In the meanwhile, several vendors have developed proprietary enhancements of the standards that are already deployed. The introduction of some QoS mechanisms in WLAN environments will have a significant impact on the number of channels that can be supported and the amount of jitter present in the system. However, it is clear that the VoIP conditions in WLANs will remain more challenging than those of the typical wired LAN. Hence, particularly in the case of WLAN environments, QoS has to be combined with efficient jitter buffer implementations and careful latency management in the system design.

## 15.3 Outline of a VoIP System

VoIP is used to provide telephony functionality to the end user, which can communicate through various devices. For traditional telephony replacement, regular phones are used while so-called media gateways to the IP network convert the calls to and from IP traffic. These gateways can be large trunking devices, situated in a telephony carrier's network, or smaller gateways, e.g., one-port gateways in an end user's home. An IP phone, on the other hand, is a device that looks very much like a regular phone, but it connects directly to an

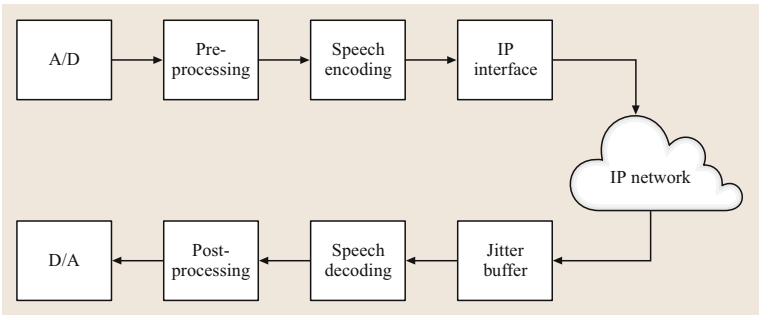



**Fig. 15.5** A typical VoIP system

IP network. Lately, PCs have become popular devices for VoIP through services like Google Talk, Skype, Yahoo! Messenger and others. In this case, the phone is replaced by an application running on the PC that provides the telephony functionality. Such applications also exist for WiFi personal digital assistants (PDAs) and even for cell phones that have IP network connections. Hence, it is not an overstatement to say that the devices used in VoIP represent a changing environment and impose challenges for the speech processing components. Therefore, the systems deployed need to be able to cope with the IP network as described in the previous section, as well as the characteristics of the different applications.

A simplified block diagram of the speech processing blocks of a typical VoIP system is depicted in Fig. 15.5. At the transmitting end, the speech signal is first digitized in an analog-to-digital (A/D) converter. Next, a preprocessing block performs functions such as echo cancelation, noise suppression, automatic gain control, and voice activity detection, depending on the needs of the system and the end user's environment. Thereafter, the speech signal is encoded by a speech codec and the resulting bit stream is transmitted in IP packets. After the packets have passed the IP network, the receiving end converts the packet stream to a voice signal using the following basic processing blocks: a jitter buffer receiving the packets, speech decoding and postprocessing, e.g., packet loss concealment.

Next, we look closer at the major speech processing components in a VoIP system. These are echo cancelation, speech coding, jitter buffering and packet loss recovery. Further, we provide an overview of a number of auxiliary speech processing components.

### 15.3.1 Echo Cancelation

One of the most important aspects in terms of effect on the end-to-end quality is the amount of echo present during a conversation. An end user experiences echo by hearing a delayed version of what he or she said played back. The artifact is at best annoying and sometimes even renders the communication useless. An echo cancelation algorithm that estimates and removes the echo is needed. The requirements on an echo canceler to achieve good voice quality are very challenging. The result of a poor design can show up in several ways, the most common being:

1. audible residual echo, due to imperfect echo cancelation,
2. clipping of the speech, where parts of or entire words disappear due to too much cancelation,
3. poor double-talk performance.

The latter problem occurs when both parties attempt to talk at the same time and the echo canceler suppresses one or both of them leading to unnatural conversation. A common trade-off for most algorithms is the performance in double-talk versus single-talk, e.g., a method can be very good at suppressing echo, but has clipping and double-talk artifacts or vice versa [15.18].

Echo is a severe distraction if the round trip delay is longer than 30–40 ms [15.6]. Since the delays in IP telephony systems are significantly higher, the echo is clearly audible to the speaker. Canceling echo is, therefore, essential to maintaining high quality. Two types of echo can deteriorate speech quality: network echo and acoustic echo.

#### Network Echo Cancelation

Network echo is the dominant source of echo in telephone networks. It results from the impedance mismatch at the hybrid circuits of a PSTN exchange, at which the two-wire subscriber loop lines are connected to the four-wire lines of the long-haul trunk. The echo path is stationary, except when the call is transferred to another handset or when a third party is connected to the phone call. In these cases, there is an abrupt change in the echo path. Network echo in a VoIP system occurs through gateways, where there is echo between the incoming and outgoing signal paths of a channel generated by the PSTN network. Network echo cancelers for VoIP are implemented in the gateways.

The common solutions to echo cancelation and other impairments in packet-switched networks are basically adaptations of techniques used for the circuit-switched network. To achieve the best possible quality, a systematic approach is necessary to address the quality-of-sound issues that are specific to packet networks. Therefore, there may be significant differences between the "repackaged" circuit-switched echo cancelers and echo cancelers optimized for packet networks.

#### Acoustic Echo Cancelation

Acoustic echo occurs when there is a feedback path between a telephone's microphone and speaker (a problem primarily associated with wireless and hands-free devices) or between the microphone and the speakers of a personal computer (PC)-based system. In addition to the direct coupling path from microphone to speaker, acoustic echo can be caused by multiple reflections of

the speaker's sound waves back to the microphone from walls, floor, ceiling, windows, furniture, a car's dashboard, and other objects. Hence, the acoustic echo path is nonstationary. Acoustic echo has become a major issue in VoIP systems as more direct end-to-end VoIP calls are being made, e.g., by the very popular PC-based services.

There are few differences between designing acoustic echo cancelation (AEC) algorithms for VoIP and for traditional telephony applications. However, due to the higher delays typically experienced in VoIP, the requirements on the AEC are often more demanding. Also, wide-band speech adds some new challenges in terms of quality and complexity. Large-scale deployments in PC environments create demanding challenges in terms of robustness to different environments (e.g., various microphones and speakers) as well as varying echo paths created by non real-time operating systems, like Windows.

### 15.3.2 Speech Codec

The basic algorithmic building block in a VoIP system, that is always needed, is the speech codec. When initiating a voice call, a session protocol is used for the call setup, where both sides agree on which codec to use. The endpoints normally have a list of available codecs to choose from. The most common codec used in VoIP is the ITU-T G.711 [15.12] standard codec, which is mandatory for every end point to support for interoperability reasons, i. e., to guarantee that one common codec always exist so that a call can be established. For low-bandwidth applications, ITU-T G.729 [15.19] has been the dominant codec.

The quality of speech produced by the speech codec defines the upper limit for achievable end-to-end quality. This determines the sound quality for perfect network conditions, in which there are no packet losses, delays, jitter, echoes or other quality-degrading factors. The bit rate delivered by the speech encoder determines the bandwidth load on the network. The packet headers (IP, UDP, RTP) also add a significant portion to the bandwidth. For 20 ms packets, these headers increase the bit rate by 16 kbits/s, while for 10 ms packets the overhead bit rate doubles to 32 kbit/s. The packet header overhead versus payload trade-off has resulted in 20 ms packets being the most common choice.

A speech codec used in a VoIP environment must be able to handle lost packets. Robustness to lost packets determines the sound quality in situations where network congestion is present and packet losses are likely. Traditional circuit switched codecs, e.g., G.729, are vulnerable to packet loss due to inter-frame dependencies caused by the encoding method. A new codec without interframe dependencies called the internet low-bit-rate codec (iLBC) has been standardized by the IETF for VoIP use [15.20]. Avoiding interframe dependencies is one step towards more robust speech coding. However, even codecs with low interframe dependencies need to handle inevitable packet losses and we will discuss that in more detail later.

Increasing the sampling frequency from 8 kHz, which is used for narrow-band products, to 16 kHz, which is used for wide-band speech coding, produces more natural, comfortable and intelligible speech. However, wide-band speech coding has thus far found limited use in applications such as video conferencing because speech coders mostly interact with the PSTN, which is inherently narrow-band. There is no such limitation when the call is initiated and terminated within the IP network. Therefore, because of the dramatic quality improvement attainable, the next generation of speech codecs for VoIP will be wide-band. This is seen in popular PC clients that have better telephony quality.

If the call has to traverse different types of networks, the speech sometimes needs to be transcoded. For example, if a user on an IP network that uses G.729 is calling a user on the PSTN, the speech packets need to be decoded and reencoded with G.711 at the media gateway. Transcoding should be avoided if possible, but is probably inevitable until there is a unified all-IP network.

It lies in the nature of a packet network that it, over short periods of time, has a variable-throughput bandwidth. Hence, speech coders that can handle variable bit rate are highly suitable for this type of channels. Variable bit rate can either be source-controlled, network-controlled, or both. If the encoder can get feedback from the network about current packet loss rates, it can adapt its rate to the available bandwidth. An example of an adaptive-rate codec is described in [15.21].

### 15.3.3 Jitter Buffer

A jitter buffer is required at the receiving end of a VoIP call. The buffer removes the jitter in the arrival times of the packets, so that there is data available for speech decoding when needed. The exception is if a packet is lost or delayed more than the length the jitter buffer is set to handle. The cost of a jitter buffer is an increase in the overall delay. The objective of a jitter buffer algorithm



Amazon EX1010 Page 340

is to keep the buffering delay as short as possible while minimizing the number of packets that arrive too late to be used. A large jitter buffer causes an increase in the delay and decreases the packet loss. A small jitter buffer decreases the delay but increases the resulting packet loss. The traditional approach is to store the incoming packets in a buffer (packet buffer) before sending them to the decoder. Because packets can arrive out of order, the jitter buffer is not a strict first-in first-out (FIFO) buffer, but it also reorders packets if necessary.

The most straightforward approach is to have a buffer of a fixed size that can handle a given fixed amount of jitter. This results in a constant buffer delay and requires no computations and provides minimum complexity. The drawback with this approach is that the length of the buffer has to be made sufficiently large that even the worst case can be accommodated or the quality will suffer.

To keep the delay as short as possible, it is important that the jitter buffer algorithm adapts rapidly to changing network conditions. Therefore, jitter buffers with dynamic size allocation, so-called adaptive jitter buffers, are now the most common [15.22]. The adaptation is achieved by inserting packets in the buffer when the delay needs to be increased and removing packets when the delay can be decreased. Packet insertion is usually done by repeating the previous packet. Unfortunately, this generally results in audible distortion. To avoid quality degradation, most adaptive jitter buffer algorithms are conservative when it comes to reducing the delay to lessen the chance of further delay increases. The traditional packet buffer approach is limited in its adaptation granularity by the packet size, since it can only change the buffer length by adding or discarding one or several packets. Another major limitation of traditional jitter buffers is that, to limit the audible distortion of removing packets, they typically only function during periods of silence. Hence, delay builds up during a talk spurt, and it can take several seconds before a reduction in the delay can be achieved.

### 15.3.4 Packet Loss Recovery

The available methods to recover from lost packets can be divided into two classes: sender-and-receiver-based and receiver-only-based techniques. In applications where delay is not a crucial factor automatic repeat request (ARQ) is a powerful and commonly used sender-and-receiver-based technique. However, the delay constraint restricts the use of ARQ in VoIP and other methods to recover the missing packets must be considered. Robust encoding refers to methods where redundant side-information is added to the transmitted data packets. In Sect. 15.4, we describe robust encoding in more detail. Receiver-only-based methods, often called packet loss concealment (PLC), utilize only the information in previously received packets to replace the missing packets. This can be done by simply inserting zeros, repeating signals, or by some more-sophisticated methods utilizing features of the speech signal (e.g., pitch periods). Section 15.5 provides an overview of different error concealment methods.

### 15.3.5 Joint Design of Jitter Buffer and Packet Loss Concealment

It is possible to combine an advanced adaptive jitter buffer control with packet loss concealment into one unit [15.23]. The speech decoder is used as a *slave* in the sense that it decodes data and delivers speech segments back when the control logic asks for it. The new architecture makes the algorithm capable of adapting the buffer size on a millisecond basis. The approach allows it to quickly adapt to changing network conditions and to ensure high speech quality with minimal buffer latency. This can be achieved because the algorithm is working together with the decoder and not in the packet buffer. In addition to minimizing jitter buffer delay, the packet loss concealment part of the algorithm in [15.23] is based on a novel approach, and is capable of producing higher quality than any of the standard PLC methods. Experiments show that, with this type of approach, one-way delay savings of 30–80 ms are achievable in a typical VoIP environment [15.23]. Similar approaches have also been presented in, e.g., [15.24] and [15.25].

### 15.3.6 Auxiliary Speech Processing Components

In addition to the most visible and well-known voice processing components in a VoIP device there are many other important components that are included either to enhance the user experience or for other reasons, such as, reducing bandwidth requirements. The exploitation of such components depends on system requirements and usage scenarios, e.g., noise suppression for hands-free usage in noisy environment or voice activity detection to reduce bandwidth on bandlimited links.

Since no chain is stronger than its weakest link, it is imperative that even these components are designed in an appropriate fashion. Examples of such components include automatic gain control (AGC), voice activity de-

Amazon EX1010 Page 341

tection (VAD), comfort noise generation (CNG), noise suppression, and signal mixing for multiparty calling features. Typically, there is not much difference in the design or requirements between traditional telecommunications solutions and VoIP solutions for this type of components. However, for example VAD and CNG are typically deployed more frequently in VoIP systems. The main reason for the increased usage of VAD is that the net saving in bandwidth is very significant, due to the protocol overhead in VoIP. Also, an IP network is well suited to utilize the resulting variable bit rate to transport other data while no voice packets are sent.

VAD is used to determine silent segments, where packets do not need to be transmitted or, alternatively, only a sparse noise description is transmitted. The CNG unit produces a natural sound noise signal at the receiver, based either on a noise description that it receives or on an estimate of the noise. Due to misclassifications in the VAD algorithm clipping of the speech signal and noise bursts can occur. Also, since only comfort noise is played out during silence periods, the background signal may sound artificial. The most common problem with CNG, is that the signal level is set too low, which results in the feeling that the other person has dropped out of the conversation. These performance issues mandate that VAD should be used with caution to avoid unnecessary quality degradation.

Implementing multiparty calling also faces some challenges due to the characteristics of the IP networks. For example, the requirements for delay, clock drift, and echo cancelation performance are tougher due to the fact that several signals are mixed together and that there are several listeners present. A jitter buffer with low delay and the capability of efficiently handling clock drift offers a significant improvement in such a scenario. Serious complexity issues arise since different codecs can be used for each of the parties in a call. Those codecs might use different sampling frequencies. Intelligent schemes to manage complexity are thus important. One way to reduce the complexity is by using VAD to determine what participants are active at each point in time and only mix those into the output signals.

### 15.3.7 Measuring the Quality of a VoIP System

Speech quality testing methods can be divided into two groups: subjective and objective. Since the ultimate goal of speech quality testing is to get a measure of the perceived quality by humans, subjective testing is the more relevant technique. The results of subjective tests are often presented as mean opinion scores (MOS) [15.26]. However, subjective tests are costly and quality ratings are relative from user to user. As a result, automatic or *objective* measuring techniques were developed to overcome these perceived shortcomings. The objective methods provide a lower cost and simple alternative, however they are highly sensitive to processing effects and other impairments [15.27, 28].

Perceptual evaluation of speech quality (PESQ), defined in ITU-T recommendation P.862 [15.29], is the most popular objective speech quality measurement tool. Even though PESQ is recognized as the most accurate objective method the likelihood that it will differ more than 0.5 MOS from subjective testing is 30% [15.30]. It is obvious that the objective methods do not offer the necessary level of accuracy in predicting the perceived speech quality and that subjective methods have to be used to achieve acceptable accuracy.

## 15.4 Robust Encoding

The performance of a speech coder over a packet loss channel can efficiently be improved by adding redundancy at the encoding side and utilizing the redundancy at the decoder to fully or partly recover lost packets. The amount of redundancy must obviously be a function of the amount of packet loss. In this section we distinguish two classes of such so-called robust encoding approaches, *forward error correction* (FEC) and *multiple description coding* (MDC), and describe them in more detail. MDC is the more powerful of the two in the sense that it is more likely to provide graceful degradation.

### 15.4.1 Forward Error Correction

In FEC, information in lost packets can be recovered by sending redundant data together with the information. Here we distinguish FEC methods from other methods that introduce redundancy in that the additional data does not contain information that can yield a better reconstruction at the decoder if no packets were lost. The typical characteristics of FEC are that the performance with some packet loss is the same as with no packet loss but that the performance rapidly deteriorates (a *cliff* effect) at losses higher than a certain critical packet loss

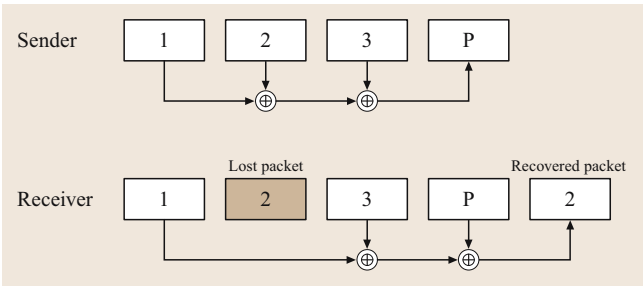



**Fig. 15.6** FEC by parity coding. Every $n$-th transmitted packet is a parity packet constructed by bitwise XOR on the previous $n-1$ packets

rate determined by the amount of redundancy. There are two classes of FEC schemes: *media-independent FEC* and *media-dependent FEC*.

### Media-Independent FEC

In media-independent FEC, methods known from channel coding theory are used to add blocks of parity bits. These methods are also known as erasure codes. The simplest of these, called parity codes, utilize exclusive-or (XOR) operations between packets to generate parity packets. One simple example is illustrated in Fig. 15.6, where every $n$-th packet contains bitwise XOR on the $n-1$ previous packets [15.31]. This method can exactly recover one lost packet if packet losses are at least separated by $n$ packets. More-elaborate schemes can be achieved by different combinations of packets for the XOR operation. By increasing the amount of redundancy and delay it is possible to correct a small set of consecutively lost packets [15.32]. More-powerful error correction can be achieved using erasure codes such as

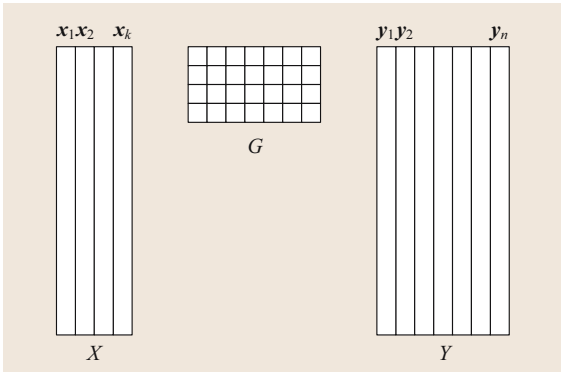



**Fig. 15.7** FEC by an RS$(n,k)$ code-sending side. A sequence of $k$ data packets is multiplied by a generator matrix to form $n$ encoded packets

Reed–Solomon (RS) codes. These codes were originally presented for streams of individual symbols, but can be utilized for blocks (packets) of symbols [15.33]. The channel model is in this case an erasure channel where packets are either fully received or erased (lost). From $k$ packets of data an RS$(n,k)$ code produces $n$ packets of data such that the original $k$ packets can be exactly recovered by receiving any subset of $k$ (or more) packets. Assume each packet $\boldsymbol{x} = [x_1, x_2, \ldots, x_B]^{\mathrm{T}}$ contains $B$ $r$-bit symbols $x_i = (b_i^{(1)}, b_i^{(2)}, \ldots, b_i^{(r)})$, $b_i^{(j)} \in [0,1]$.

The $n$ packets can then be generated by (Fig. 15.7)

$$\mathrm{Y} = \mathrm{XG}\,, \tag{15.1}$$

where $\mathrm{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$, $\mathrm{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n]$, and G is a generator matrix. All operations are performed in the finite extension field $GF(2^r)$.The generator matrix is specific for the particular RS code. It is often convenient to arrange the generator matrix in systematic form, which yields an output where the first $k$ packets are identical to the $k$ uncoded packets and the other $n-k$ packets contain parity symbols exclusively. An example of constructing a generator matrix in systematic form is [15.33]

$$\mathrm{G} = \mathrm{V}_{k,k}^{-1}\mathrm{V}_{k,n}\,, \tag{15.2}$$

using the Vandermonde matrix

$$\mathrm{V}_{i,j} = \alpha^{ij}\,, \tag{15.3}$$

where $\alpha$ is a generating element in $GF(2^r)$. The $r$-bit symbols are all elements in this field.

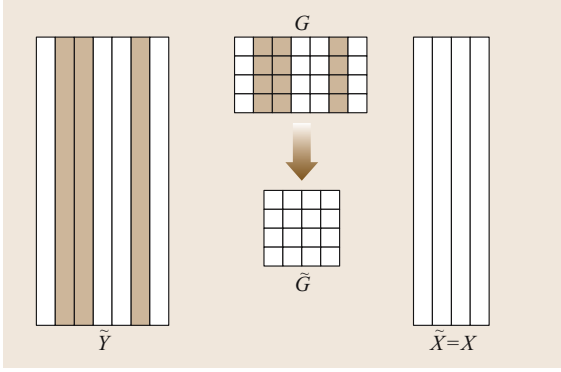



**Fig. 15.8** FEC by an RS$(n,k)$ code-receiving side. Some packets are lost during transmission (indicated by the gray color). The decoder forms a matrix $\tilde{G}$ from the columns corresponding to $k$ correctly received packets. These packets are then multiplied with the inverse of $\tilde{G}$ to recover $X$, the original sequence of $k$ data packets

From $k$ available packets the receiver forms $\tilde{Y}$, a subset of $Y$. The recovered packets are then calculated as

$$\tilde{X} = \tilde{Y}\tilde{G}^{-1} , \tag{15.4}$$

where $\tilde{G}$ is formed by the $k$ columns of G corresponding to the received packets. Figure 15.8 illustrates the decoding procedure. RS codes belong to a class of codes called maximum distance separable (MDS) codes that are generally powerful for many types of erasure channels but require a large $n$. For real-time interactive audio, delay is crucial and only short codes are feasible. Typical examples of RS codes utilized in VoIP are RS(5, 3) [15.34] and RS(3, 2) [15.35]. For the application of bursty erasure channels another type of codes, *maximally short codes* [15.36], have shown to require lower decoding delay than MDS codes.

A common way to decrease the packet overhead in FEC is to attach the redundant packets onto the information packet, a technique called *piggybacking*. Figure 15.9 depicts a case for an RS(5, 3) code, where the two parity packets are piggybacked onto the first two data packets of the next packet sequence.

### Media-Dependent FEC

In media-dependent FEC, the source signal is encoded more than once. A simple scheme is just to encode the information twice and send the data in two separate packets. The extra packet can be piggybacked onto the subsequent packet. To lower the overall bit rate, it is more common that the second description uses a lower rate-compression method, resulting in a lower quality in case a packet needs to be recovered. The latter method is sometimes referred to as low-bit-rate redundancy (LBR) and has been standardized by IETF [15.37], which actually also provides procedures for the first method. In the LBR context, the original (high-quality) coded description is referred to as the *primary encoding* and the lower-quality description is denoted the *redundant encoding*. Examples of primary and redundant encodings are G.711 (64 kbps)+ GSM (13 kbps) [15.38] and G.729 (8 kbps)+ LPC10 (2.4 kbps) [15.35]. Figure 15.10 depicts the method for the case when the secondary encoding is piggybacked on the next primary encoding in the following packet. To safeguard against burst errors, the secondary description can be attached to the $m$-th next packet instead of the immediate subsequent packet, at the cost of a decoding delay of $m$ packets. Even better protection is obtained with more redundancy, e.g., packet $n$ contains the primary encoding of block data block $n$ and redundant encodings of blocks $n-1, n-2, \ldots, n-m$ [15.38]. Although media-dependent FEC seems more popular, erasure codes are reported to have better subjective performance at comparable bit rates [15.35].

It is important to point out that most practical systems that today use FEC simply put the same encoded frame in two packets, as mentioned above. The rea-

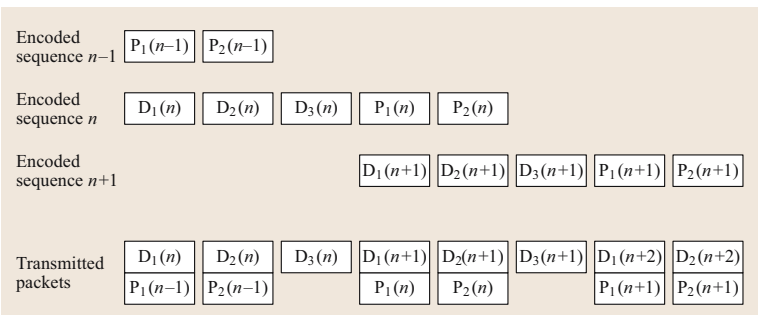



**Fig. 15.9** Piggybacking in an RS(5, 3) FEC system. The parity packets of sequence $n$ are attached to the data packets of sequence $n+1$.

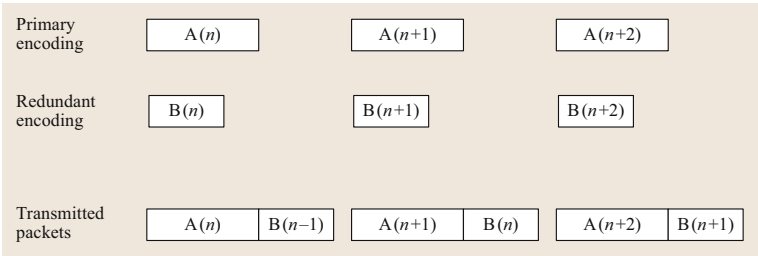



**Fig. 15.10** Media-dependent FEC proposed by IETF. The data is encoded twice, and the additional redundant encoding is piggybacked onto the primary encoding in the next packet

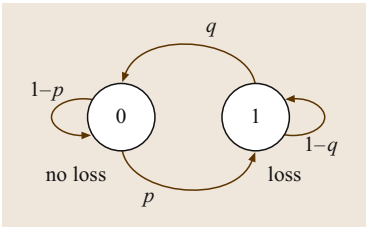



**Fig. 15.11** The Gilbert two-state model of a bursty packet loss channel

son for doing so is due to the fact that a lower bit rate secondary coder is typically more complex than the primary encoder and the increase in bit rate is preferred to an increase in complexity.

#### Adaptive FEC

The amount of redundancy needed is a function of the degree of packet loss. Some attempts have been made to model the loss process in order to assess the benefits of adding redundancy. In [15.38] *Bolot* et al. used LBR according to the IETF scheme mentioned in the last paragraph and a Gilbert channel model (a Markov chain model for the dependencies between consecutive packet losses, Fig. 15.11) to obtain expressions for the perceived quality at the decoder as a function of packet loss and redundancy. In the derivations the perceived quality is assumed to be an increasing function of the bit rate. They propose a design algorithm to allocate redundancy that functions even in the case that there are only a handful of low bit rate secondary encodings to choose from. In the case where the channel packet loss can be estimated somewhere in the network and signalled back to the sender, e.g., through RTCP (an out-of-band signaling for RTP [15.5]), their algorithm can be utilized for adaptive FEC. Due to the adaptation it is possible to obtain a more graceful degradation of performance than the typical collapsing behavior of FEC.

### 15.4.2 Multiple Description Coding

Multiple description coding (MDC) is another method to introduce redundancy into the transmitted description to counter packet loss. Compared to erasure codes, the method has the advantage that it naturally leads to graceful degradation. It optimizes an average distortion criterion assuming a particular packet loss scenario, which can in principle consist of an ensemble of scenarios. Disadvantages of the multiple description coding technique are that it is difficult to combine multiple description coding with legacy coding systems and that changing the robustness level implies changing the entire source coder.

#### MDC Problem Formulation

Consider a sampled speech signal that is transmitted with packets. To facilitate the derivations that follow below, the speech signal contained in a packet is approximated as a weakly stationary and ergodic process that is independent and identically distributed (i.i.d.). We denote this speech process by $X$ (thus, $X$ denotes a sequence of random variables that are i.i.d.). We note that a process $X$ with the aforementioned properties can be obtained from speech by performing an appropriate invertible transform. Optimal encoding of $X$ over a channel facilitating a given rate $R$ requires the *one* description of the source, which, when decoded at the receiver end, achieves the lowest achievable distortion $E[d(X, \hat{X})]$, where $\hat{X}$ is the reconstructed process. Varying the rate leads to a lower bound on all achievable distortions as a function of the rate, i. e., the distortion-rate function $D(R)$.

An extension of the problem formulation, to the case where several channels are available, is not entirely intuitive. We consider here only the case of two channels, $s \in \{1, 2\}$, each with rate $R_s$ and corresponding description $I_s$. The channels are such that they either do or do not work. (This corresponds to the cases where the packet either arrives or does not arrive.) Thus, we can distinguish four receiver scenarios: we receive both descriptions, we receive description $I_1$ solely, we receive description $I_2$ solely, or we receive no description. A particular formulation of the objective of multiple description coding is to find the two descriptions that minimize the central distortion $E[d(X, \hat{X}_0)]$, with constraints on the side distortions, $E[d(X, \hat{X}_s)] < D_s$, where $\hat{X}_0$ and $\hat{X}_s$ are *central* and *side* reconstruction values obtained from both descriptions and a single description, respectively.

Figure 15.12 illustrates the operation of multiple description coders. The encoders $f_s$ map the speech signal $X$ to two descriptions $I_s$, each of rate $R_s$. From these two descriptions, three reconstruction values from three decoders can be obtained, depending on which combi-

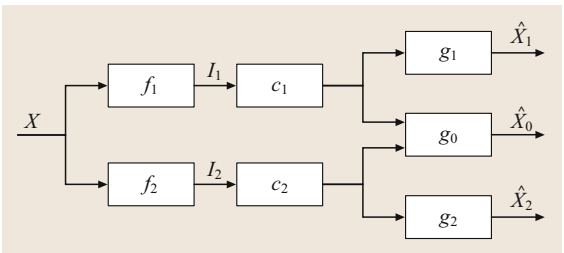



**Fig. 15.12** Block diagram of a multiple description system with two channels $c_s$, two encoders $f_s$, and three decoders $g_0$ and $g_s$. Each channel is denoted by an index $s \in \{1, 2\}$

nation of descriptions is used. When both descriptions are received, the central decoder is active and the reconstructed value is $\hat{X}_0 = g_0(I_1, I_2)$. When only one description is received, the corresponding side decoder is active and the reconstructed value is $\hat{X}_s = g_s(I_s)$. The design problem of a multiple description system is to find encoders $f_s$ and decoders $g_0$ and $g_s$ that minimize the central distortion $E[d(X, \hat{X}_0)]$, with given constraints on side distortions $E[d(X, \hat{X}_s)] < D_s$, for all combinations of rates $R_s$. The region of interest in $(R_1, R_2, D_0, D_1, D_2)$ is the performance region, which consists of all achievable combinations.

### Bounds on MDC Performance

For the single channel case, the rate-distortion theorem in rate-distortion theory, e.g., [15.39], provides the achievable rates for a given distortion. For the case of multiple descriptions with the given bounds $D_s$ on each side distortion, the achievable rates for the side descriptions are

$$R_s \geq I(X; \hat{X}_s) \,, \tag{15.5}$$

for each channel $s \in \{1, 2\}$, where $I(X; \hat{X})$ is the mutual information between $X$ and $\hat{X}$.

For the central description, when both channels are in the working state, the total rate is bounded as

$$R_1 + R_2 \geq I(X; \hat{X}_1, \hat{X}_2) + I(\hat{X}_1; \hat{X}_2) \,, \tag{15.6}$$

according to *El Gamal* and *Cover* in [15.40]. Unfortunately, this bound is usually loose.

Interpretation of the bounded multiple description region is straightforward. Any mutual information between the side descriptions increases the total rate of the system. If the design is such that no mutual information exists between the two side descriptions, then

$$R_1 + R_2 \geq I(X; \hat{X}_1, \hat{X}_2) = I(X; \hat{X}_1) + I(X; \hat{X}_2) \,, \tag{15.7}$$

and the effective rate is maximized. In this case, all redundancy is lost, which leads to the minimum possible central distortion $D_0 = D(R_1 + R_2)$.

In the other extreme, maximizing redundancy gives two identical descriptions and

$$R_1 + R_2 \geq 2I(X; \hat{X}_s) \,. \tag{15.8}$$

However, this increases the central distortion to $D_0 = D(R_s) = D_s$ and nothing is gained in terms of distortion. Note that this is the same setup as the simple FEC method in the beginning of the section on Media-Dependent FEC in Sect. 15.4.1.

### Bounds for Gaussian Sources

The bounds of the multiple description region are, as stated earlier, generally loose and the region is not fully known. Only for the case of memoryless Gaussian sources and the squared error distortion criterion is the region fully known. While this distribution is not representative of the speech itself, the result provides insight to the performance of multiple description coding. *Ozarow* showed in [15.41] that the region defined by the bounds of El Gamal and Cover are tight in this case and define all achievable combinations of rates and distortions. The bounds on distortions as functions of rate are

$$D_s \geq \sigma^2 2^{-2R_s} \tag{15.9}$$

$$D_0 \geq \sigma^2 2^{-2(R_1+R_2)} \cdot \gamma_D(R_1, R_2, D_1, D_2) \,, \tag{15.10}$$

where the variance $\sigma^2$ is used and

$$\gamma_D = \begin{cases} 1 & \text{if } D_1 + D_2 > \sigma^2 + D_0 \,, \\ \frac{1}{1-a^2} & \text{otherwise} \,, \end{cases} \tag{15.11}$$

$$a = \sqrt{(1-D_1)(1-D_2)} - \sqrt{D_1 D_2 - 2^{-2(R_1+R_2)}} \,. \tag{15.12}$$

Interpretation of the bounds on side distortion is trivial. The central distortion is increased by a factor of $\gamma_D$ for low side distortions. This implies that the best achievable distortion for rate $R_1 + R_2$, i. e., $D(R_1 + R_2)$, is obtained

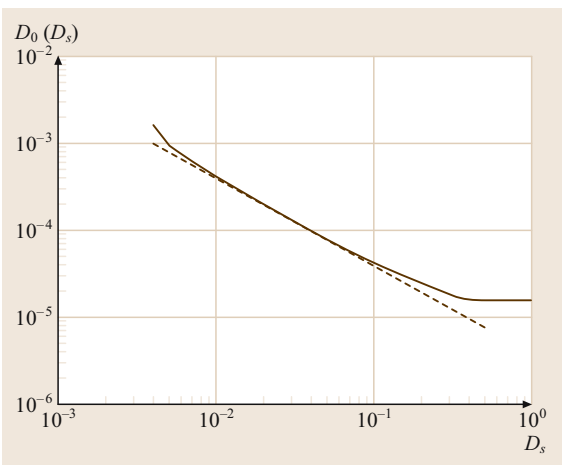



**Fig. 15.13** Central distortion $D_0$ as a function of the side distortion $D_s$ for a Gaussian i.i.d. source with unit variance and fixed per channel rate $R = 4$. Note the trade-off between side and central distortion. The *dashed line* represents the high-rate approximation

only if side distortions are allowed to be large. The relationship is illustrated in a plot of $D_0$ as a function of $D_s = D_1 = D_2$ for a fixed $R = R_1 = R_2$, in Fig. 15.13.

A high rate approximation to the bound derived by Ozarow is derived in [15.42] and is for the unit-variance Gaussian case given by

$$D_0 D_s = \frac{1}{4} 2^{-4R} . \tag{15.13}$$

### MDC Versus Channel Splitting

In modern MDC methods, the side distortions decrease with increasing rate. A technique relevant to speech that does not have this property but is generally included in discussions of MDC is channel splitting by odd-even separation. In this approach odd and even channels are transmitted in separate packets, and if a packet is lost interpolation (in some cases with the aid of an additional coarse quantizer) is used to counter the effect of the loss. Variants on this approach are found in [15.43–46] and a discussion on early unpublished work on this topic can be found in [15.47]. The method is perhaps more accurately classified as a Wyner–Ziv-type coding method (or distributed source coding method) [15.48, 49] with the odd and the even samples forming correlated sources that are encoded separately. In general, the performance of the odd–even separation based methods are suboptimal. The reason is that the redundancy between descriptions is highly dependent of the redundancy present in the speech signal. Hence, the trade-off between the side distortion and the central distortion is difficult to control.

### MDC Scalar Quantization

Two methods for design of multiple description scalar quantization, resolution-constrained and entropy-constrained, were described by *Vaishampayan* in [15.50] and [15.51]. In resolution-constrained quantization, the codeword length is fixed and its resolution is constrained. In entropy-constrained quantization the codeword length is variable and the index entropy is constrained. Vaishampayan assumes the source to be represented by a stationary and ergodic random process $X$. The design of the resolution-constrained coder is described next, followed by a description of the entropy-constrained coder.

The encoder first maps a source sample $x$ to a partition cell in the central partition $\mathcal{A} = \{A_1, \ldots, A_N\}$. The resulting cell with index $i_0$ in $\mathcal{A}$ is then assigned two indices via the index assignment function $a(i_0) = \{i_1, i_2\}$, where $i_s \in \mathcal{I}_s = \{1, 2, \ldots, M_s\}$ and $N \le M_1 M_2$. The index assignment function is such that an inverse $i_0 = a^{-1}(i_1, i_2)$ referring to cell $A_{i_0}$ always exists. The two indices, resolution-constrained, are sent over each channel to the receiver. As is appropriate for constrained-resolution coding, the codewords are of equal length.

At the receiver end, depending on which channels are in a working state, a decoder is engaged. In the case when one of the channels is not functioning, the received index of the other channel is decoded by itself to a value $\hat{x}_s = g_s(i_s)$ in the corresponding codebook $\hat{\mathcal{X}}_s = \{\hat{x}_{s,1}, \hat{x}_{s,2}, \ldots, \hat{x}_{s,M_s}\}$. In the case when both channels deliver an index, these are used to form the central reconstruction value $\hat{x}_0 = g_0(i_1, i_2)$ in codebook $\hat{\mathcal{X}}_0 = \{\hat{x}_{0,1}, \hat{x}_{0,2}, \ldots, \hat{x}_{0,N}\}$. The three decoders are referred to as $\boldsymbol{g} = \{g_0, g_1, g_2\}$.

The performance of the outlined coder is evaluated in terms of a Lagrangian, $L(\mathcal{A}, \boldsymbol{g}, \lambda_1, \lambda_2)$, which is dependent on the choice of partition $\mathcal{A}$, decoders $\boldsymbol{g}$, and Lagrange multipliers $\lambda_1$ and $\lambda_2$. The Lagrange multipliers weight the side distortions. The codeword length constraints are implicit in the partition definition and do not appear explicitly in the Lagrangian. Minimizing the Lagrangian $L$ is done with a training algorithm that is based on finding non-increasing values of $L$ by iteratively holding $\mathcal{A}$ and $\boldsymbol{g}$ constant while optimizing for $\boldsymbol{g}$ and $\mathcal{A}$, respectively.

The performance of the training algorithm presented is highly dependent on the index assignment $a(i_0) = \{i_1, i_2\}$. This mapping should be such that it minimizes the distortion for given channel rates $R_s = \log_2(M_s)$. The minimization is done by comparing all possible combinations of indices $i_1$ and $i_2$ for all possible cardinalities of $\mathcal{A}$, $N \le M_1 M_2$. This kind of search is too complex, as the total number of possible index assignments is $\sum_{N=1}^{M_1 M_2} (M_1 M_2)!/(M_1 M_2 - N)!$. A suboptimal search algorithm is presented in [15.52]. An illustration of an index assignment that is believed

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 3 | | | | | | |
| 5 | 2 | 4 | 5 | | | | | |
| 3 | | 6 | 7 | 9 | | | | |
| 4 | | | 8 | 10 | 11 | | | |
| 5 | | | | 12 | 13 | 15 | | |
| 6 | | | | | 14 | 16 | 17 | |
| 7 | | | | | | 18 | 19 | 21 |
| 8 | | | | | | | 20 | 22 |

**Fig. 15.14** Nested index assignment matrix for $R_s = 3$ bits. Vertical and horizontal axis represent the first and second channel, respectively

to be close to optimal, the *nested index assignment* [15.50], is found in Fig. 15.14. There, the index assignment matrix shows the procedure of inverse mapping $i_0 = a^{-1}(i_1, i_2)$. It is clear that the redundancy in the system is directly dependent on $N$. The more cells the central partition has, the lower central distortion is achievable at the expense of side distortions.

The performance of resolution-constrained multiple description scalar quantization was analyzed for high rates in [15.42]. It was shown that the central and side distortions for a squared distortion criterion are dependent of the source probability distribution function $p(x)$ as

$$D_0 = \frac{1}{48}\left(\int_{-\infty}^{\infty} p^{1/3}(x)\,\mathrm{d}x\right)^3 2^{-2R(1+a)} \qquad (15.14)$$

$$D_s = \frac{1}{12}\left(\int_{-\infty}^{\infty} p^{1/3}(x)\,\mathrm{d}x\right)^3 2^{-2R(1-a)}\,, \qquad (15.15)$$

where $a \in (0, 1)$. Writing this interdependency as the product of the distortions gives better understanding, since the equation then is independent of $a$. For the special case of a unit-variance Gaussian source we have

$$D_0 D_s = \frac{1}{4}\left(\frac{2\pi\,\mathrm{e}}{4\,\mathrm{e}3^{-1/2}}\right)^2 2^{-4R}\,. \qquad (15.16)$$

The gap of this high-rate scalar resolution-constrained quantizer to the rate-distortion bound computed by Ozarow is 8.69 dB.

The design of the entropy-constrained coder resembles the resolution-constrained coder described above. The differences are the replacement of fixed codeword length constraints by entropy constraints on index entropies, as well as added variable-length coders prior to transmission over each channel.

The Lagrangian function for the entropy-constrained case, $L(\mathcal{A}, \boldsymbol{g}, \lambda_1, \lambda_2, \lambda_3, \lambda_4)$, depends on six variables. These are the partition $\mathcal{A}$, decoders $\boldsymbol{g}$ and Lagrange multipliers $\lambda_1$ through $\lambda_4$, where two of the Lagrangian multipliers correspond to the constraints on the index entropies. Thus, in this case the rate constraints are explicitly in the Lagrangian. The Lagrangian is, as for the resolution-constrained case, minimized with a training algorithm. Before sending indices over each channel, variable length coders are applied to indices obtained by the index assignment.

Disregarding the requirement that codewords should have equal length results in improved performance. The high-rate approximation of the product between central and side distortions for the unit-variance Gaussian case is

$$D_0 D_s = \frac{1}{4}\left(\frac{2\pi\,\mathrm{e}}{12}\right)^2 2^{-4R}\,. \qquad (15.17)$$

The resulting gap of the scalar constrained-entropy quantizer to the rate-distortion bound is here 3.07 dB.

### MDC Vector Quantization

It is well known from classical quantization theory that the gap between the rate-distortion bound and the performance of the scalar quantizer can be closed by using vector quantization. The maximum gain of an optimal vector quantizer over an optimal scalar quantizer due to the space filling advantage is 1.53 dB [15.53]. This motivates the same approach in the case of multiple descriptions. One approach to multiple description vector quantization is described in the following. It uses lattice codebooks and is described in [15.54], where implementation details are provided for the $A_2$ and $\mathbb{Z}_i$ lattices, with $i = 1, 2, 4, 8$.

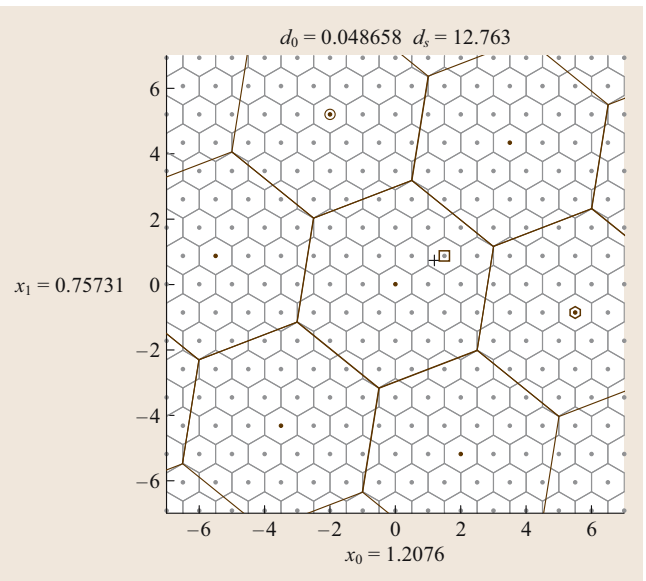



**Fig. 15.15** Encoding of the vector marked with a cross. The two descriptions are marked with brown circle and hex. When both descriptions are received the reconstruction point is chosen as the brown square. Central and side distortions are shown at the top

Multiple description vector quantization with lattice codebooks is a method for encoding a source vector $\boldsymbol{x} = [x_1, \ldots, x_n]^{\mathrm{T}}$ with two descriptions. In [15.54], an algorithm for two symmetric descriptions, equal side distortions, is described. These two descriptions are, as with multiple description scalar quantization, sent over different channels, providing a minimum fidelity in the case of failure of one of the channels. Figure 15.15 illustrates the encoded and decoded codewords for a two-dimensional source vector $\boldsymbol{x}$. The finer lattice $\Lambda$ represents the best resolution that can be obtained, that is with the central reconstruction value. Codewords that are sent over the channel are actually not a part of the fine lattice $\Lambda$, but of a geometrically similar sublattice $\Lambda'$. The sublattice $\Lambda'$ is obtained by scaling, rotating, and possibly reflecting $\Lambda$. This procedure is dependent on what trade-off one wants to obtain between central and side distortions. Which points in $\Lambda'$ to send and how to choose a reconstruction point in $\Lambda$ given these points is called the labeling problem, which is comparable to the index assignment problem in the one dimensional case. The labeling problem is solved by setting up a mapping $\Lambda \rightarrow \Lambda' \times \Lambda'$ for all cells of $\Lambda$ that are contained within the central cell of $\Lambda'$. This mapping is then extended to all cells in $\Lambda$ using the symmetry of the lattices.

Choosing distortions is done by scaling of the used lattices. However, the constraint that the side-distortions must be equal is a drawback of the method. This is improved in [15.55], where an asymmetric multiple description lattice vector quantizer is presented.

High-rate approximations for the setup described above with coding of infinitely large vectors result in a product of distortions given by

$$D_0 D_s = \frac{1}{4}\left(\frac{2\pi\,\mathrm{e}}{2\pi\,\mathrm{e}}\right)^2 2^{-4R} \tag{15.18}$$

for a unit-variance Gaussian source.

Looking closer at the equation, the gain compared to the approximation of the rate-distortion bound is 0 dB. This means that the high-rate approximation of the rate-distortion bound has been reached. In an implementation, however, finite-dimension vectors must be used. For the example provided above, with the hexagonal $A_2$ lattice, the distortion product is

$$D_0 D_s = \frac{1}{4}\left(\frac{2\pi\,\mathrm{e}}{5/36\sqrt{3}}\right)^2 2^{-4R} \tag{15.19}$$

and a gap of 1.53 dB remains compared to the approximation of the rate distortion bound. Using higher dimensions results in greater gains.

### Correlating Transforms

Multiple description coding with correlating transforms was first introduced by *Wang*, *Orchar*, and *Reibman* in [15.56]. Their approach to the MDC problem is based on a linear transformation of two random variables, which introduces correlation in the transform coefficients. This approach was later generalized by *Goyal* and *Kovacevic* in [15.57] to $M$ descriptions of an $N$-dimensional vector, where $M \leq N$.

Consider a vector $\boldsymbol{y} = [y_1, y_2]^{\mathrm{T}}$ with variances $\sigma_1$ and $\sigma_2$. Transforming the vector with an orthogonal transformation such as

$$\mathbf{z} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\boldsymbol{y}$$

produces transform coefficients $\mathbf{z} = [z_1, z_2]^{\mathrm{T}}$. Sending one transform coefficient over each channel results in central distortion

$$D_0 = \frac{\pi\,\mathrm{e}}{6}\left(\frac{\sigma_1^2+\sigma_2^2}{2}\right)2^{-2R} \tag{15.20}$$

and average side distortion

$$D_s = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} + \frac{\pi\,\mathrm{e}}{12}\left(\frac{\sigma_1^2+\sigma_2^2}{2}\right)2^{-2R} \tag{15.21}$$

on vector $\boldsymbol{y}$. When interpreting the results, we see that the relation of $\sigma_1$ and $\sigma_2$ determines the trade-off between central and side distortion. If $\sigma_1$ equals $\sigma_2$, the distortions obtained are equal to what would have been obtained if $\boldsymbol{y}$ was sent directly, i. e., the case with no redundancy between descriptions.

The example above leads to the conclusion that a transformation on the i.i.d. source vector $\boldsymbol{x}$ needs to be performed to get transform coefficients $\boldsymbol{y}$ of different variances. These are then handled as described above.

Extension to an $N$-dimensional source vector $\boldsymbol{x}$ is handled in the following manner. A square correlating transform produces $N$ coefficients in $\boldsymbol{y}$ with varying variances. These coefficients are quantized and transformed to $N$ transform coefficients, which in turn are placed into $M$ sets in an a priori fixed manner. These $M$ sets form the $M$ descriptions. Finally, entropy coding is applied to the descriptions.

High-rate approximations of multiple description coding with correlating transforms does not tell us anything about the decay of distortion. This is so since

independent of choice of transform $D_0 = O(2^{-2R})$ and $D_1 = D_2 = O(1)$. Simulations [15.57] show however that multiple description coding with correlating transforms performs well at low redundancies.

### Conclusions on MDC

This section has presented the bounds on achievable rate-distortion for multiple description coding and a number of specific algorithms have been discussed. We provided some useful comparisons.

Figure 15.16 shows an overview of the performance of the discussed algorithms for i.i.d. Gaussian signals and the squared error criterion. The figure is based on the high-rate approximations of achievable side and central distortions of resolution-constrained multiple description scalar quantization, entropy-constrained multiple description scalar quantization and multiple description vector quantization with dimension two. It is clear that the distortions achievable decrease in the order that the methods are mentioned. Using vectors of dimension larger than two reduces the distortions further, approaching the high-rate approximation of the Ozarow bound for vectors of infinite dimension. Note that multiple description with correlating transforms is not included in the figure, due to the reasons mentioned in the previous section.

Evaluating the performance of the three methods proposed by *Vaishampayan* [15.50, 51, 54], i. e., constrained-resolution MDC scalar quantization, constrained-entropy MDC scalar quantization and MDC vector quantization with lattice codebooks for constrained-entropy coding, the results are not surprising. They are consistent with what is known in equivalent single description implementations. Entropy-constrained quantization performs better than resolution-constrained quantization because the average rate constraint is less stringent than the fixed rate constraint. Even for the i.i.d. case, vector quantization outperforms scalar quantization because of the space filling and shape (constrained resolution only) advantages [15.58].

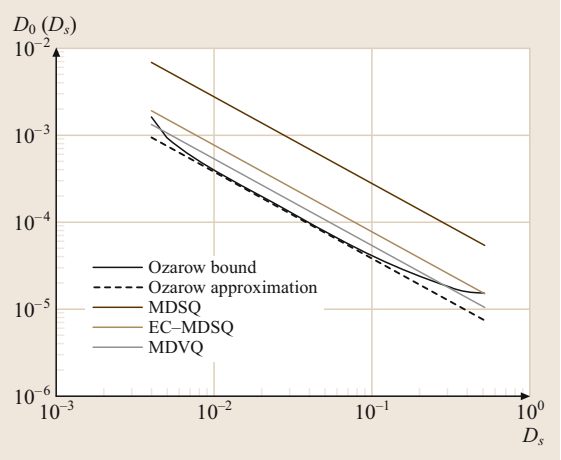



**Fig. 15.16** A comparison of the Ozarow bound, its high-rate approximation, and the high-rate approximations of three coders

Improved performance has a price. While the quantization step is often of lower computational complexity for entropy-constrained quantization, it requires an additional lossless encoding and decoding step. The drawbacks of vector quantization are that it introduces delay in the signal and increases computational effort. For constrained-entropy quantizers the added cost generally resides in the variable-length encoding of the quantization indices. For constrained-resolution coding the codebook search procedure generally increases rapidly in computational effort with increasing vector dimension.

The usage of correlating transforms to provide multiple descriptions has been shown to work well for low redundancy rates [15.57]. However, better performance is obtained with the approaches using an index assignment when more redundancy is required. Hence, one might choose not to implement correlating transforms even for low-redundancy applications to avoid changes in the coder design if additional redundancy is required at a later time.

When it comes to speech applications, the described multiple description coding techniques are in general not directly applicable. The speech source is not i.i.d., which was the basic assumption in this section. Thus, the assumption that speech is an i.i.d. source is associated with performance loss. Only pulse-code modulation (PCM) systems are based on the i.i.d. assumption, or perhaps more correctly, ignore the memory that is existent in speech. It is straightforward to adapt the PCM coding systems to using the described multiple description theory, and a practical example is [15.59]. Other speech coding systems are generally based on prediction [e.g., differential PCM (DPCM), adaptive DPCM (ADPCM), and code excited linear prediction (CELP)] making the application of MDC less straightforward. The MDC theory can be applied to the encoding of the prediction parameters (if they are transmitted as side information) and to the excitation. However, the prediction loops at encoder and decoder are prone to mismatch and, hence, error propagation. An alternative approach to exploiting the memory of the source has



Amazon EX1010 Page 350

been described in [15.60]. A Karhunen–Loève transform is applied to decorrelate a source vector prior to usage of MDC. However, since transform coding is not commonly used in speech coders, this method is not applicable either.

Multiple description theory has made large advances in the last two decades. The results in artificial settings are good. However, much work remains to be done before its promise is fulfilled in practical speech coding applications.



## 15.5 Packet Loss Concealment

If no redundancy is added at the encoder and all processing to handle packet loss is performed at the decoder the approach is often referred to as packet loss concealment (PLC), or more generally error concealment. Sometimes, a robust encoding scheme is combined with a PLC, where the latter operates as a safety net, set up to handle lost packets that the former did not succeed in recovering.

Until recently, two simple approaches to dealing with lost packets have prevailed. The first method, referred to as zero stuffing, involves simply replacing a lost packet with a period of silence of the same duration as the lost packet. Naturally, this method does not provide a high-quality output and, already for packet loss rates as low as 1%, very annoying artifacts are apparent. The second method, referred to as packet repetition, assumes that the difference between two consecutive speech frames is quite small and replaces the lost packet by simply repeating the previous packet. In practice, it is virtually impossible to achieve smooth transitions between the packets with this approach. The ear is very sensitive to discontinuities which leads to audible artifacts as a result of the discontinuities. Furthermore, even a minor change in pitch frequency is easily detected by the human ear. However, this approach performs fairly well at low packet loss probabilities (less than 3%).

More-advanced approaches to packet loss design are based on signal analysis where the speech signal is extrapolated or interpolated to produce a more natural-sounding concealment. These approaches can be divided into two basic classes: nonparametric and parametric methods. If the jitter buffer contains one or more future packets in addition to the past packets, the missing signal can be generated by interpolation. Otherwise, the waveform in the previous frame is extrapolated into the missing frame. Two-sided interpolation generally gives better results than extrapolation.

### 15.5.1 Nonparametric Concealment

Overlap-and-add (OLA) techniques, originally developed for time-scale modification, belong to the class of nonparametric concealment methods used in VoIP. The missing frame is in OLA generated by time-stretching the signal in the adjacent frames. The steps of the basic OLA [15.61] are:

1. Extract regularly spaced windowed speech segments around sampling instants $\tau(kS)$.
2. Space windowed segments according to time scaling (regularly spaced $S$ samples apart).
3. Normalize the sum of segments as

$$y(n) = \frac{\sum_k v(n-kS)x(n+\tau(kS)-kS)}{\sum_k v(n-kS)} , \quad (15.22)$$

where $v(n)$ is a window function. Due to the regular spacing and with a proper choice of symmetric window the denominator becomes constant, i. e., $\sum_k v(n-kS) = C$, and the synthesis is thus simplified.

OLA adds uncorrelated segments with similar short-term correlations, thus retaining short-term correlations. However, the pitch structure, i. e., spectral fine structure, has correlations that are long compared to the extraction window and these correlations are destroyed, resulting in poor performance. Two significant improvement approaches are synchronized OLA (SOLA) [15.62] and waveform similarity OLA (WSOLA) [15.63].

In SOLA the extracted windowed speech segments are regularly spaced as in OLA, but when spacing the output segments they are repositioned such that they have high correlation with the already formed portion. The generated segment is formed as

$$y(n) = \frac{\sum_k v(n-kS+\delta_k)x(n+\tau(kS)-kS+\delta_k)}{\sum_k v(n-kS+\delta_k)} . \quad (15.23)$$

A renormalization is needed after placing the segment due to the nonconstant denominator. The window shift $\delta_k$ is searched and selected such that the cross-correlation

between the windowed segment $v(n-kS+\delta_k)x(n+\tau(kS)-kS+\delta_k)$ and the previously generated output

$$y_{k-1}(n) = \frac{\sum_{m=-\infty}^{k-1} v(n-mS+\delta_m)x(n+\tau(mS)-mS+\delta_m)}{\sum_{m=-\infty}^{k-1} v(n-mS+\delta_m)} \quad (15.24)$$

is maximized.

WSOLA, instead, extracts windowed segments that are selected for maximum cross-correlation with the last played out segment and regularly space these at the output. The constant denominator implies that there is no need to re-normalize and WSOLA is thus simpler than SOLA.

$$y(n) = \frac{\sum_k v(n-kS)x(n+\tau(kS)-kS+\delta_k)}{\sum_k v(n-kS)} = \sum_k v(n-kS)x(n+\tau(kS)-kS+\delta_k)\,. \quad (15.25)$$

An example of PLC using WSOLA is presented in [15.64].

The two techniques can efficiently be utilized in compensating for packet delays, as well as for pure PLC as mentioned in Sect. 15.3.5. For example, if a packet is not lost and only delayed less than a full frame interval the signal can be stretched this time amount until the play-out of the delayed frame starts. Examples of this are [15.25] for SOLA and [15.24] for WSOLA.

### 15.5.2 Parametric Concealment

Waveform substitution methods were early PLC methods that tried to find a pitch cycle waveform in the previous frames and repeat it. *Goodman* et al. [15.65] introduced two approaches to waveform substitution, a pattern matching approach and a pitch detection approach. The pattern matching approach used the last samples of the previous frame as a template and searched for a matching segment earlier in the received signal. The segment following this match was then used to generate the missing frame. The pitch detection approach estimated the pitch period and repeated the last pitch cycle. According to [15.66], the pitch detection approach yielded better results.

Voicing, power spectrum and energy are other example of features, besides the pitch period, that can be estimated for the previous speech segments and extrapolated in the concealed segment. The packet loss concealment method for the waveform codec G.711, as specified in annex I [15.67] to the ITU standard, is an example of such a method.

If the PLC is designed for a specific codec and has access to the decoder and its internal states, better performance can generally be obtained than if only the decoded waveform is available. Many codecs, such as G.729, have a built-in packet loss concealment algorithm that is based on knowledge of the decoder parameters used for extrapolation.

Recent approaches to PLC include different types of speech modeling such as linear prediction [15.68], sinusoidal extrapolation [15.69], multiband excitation [15.70], and nonlinear oscillator models [15.71]. The latter uses an oscillator model of the speech signal, consisting of a transition codebook built from the existing signal, which predicts future samples. It can advantageously be used for adaptive play-out similar to the OLA methods. Further, in [15.72], a PLC method is presented for LPC-based coders, where the parameter evolution in the missing frames is determined by hidden Markov models.

Concealment systems that are not constrained by producing an integer number of whole frames, thus becoming more flexible in the play-out, have the potential to produce higher-quality recovered speech. This has been confirmed by practical implementations [15.23].

## 15.6 Conclusion

Designing a speech transmission system for VoIP imposes many technical challenges, some similar to those of traditional telecommunications design, and some specific to VoIP. The most important challenges for VoIP are a direct result of the characteristics of the transport media — IP networks. We showed that overall delay of such networks can be a problem for VoIP and that, particularly for the small packets common in VoIP, significant packet loss often occurs at networks loads that are far below the nominal capacity of the network. It can be concluded that, if a VoIP system is to provide the end user with low transmission delay and with high voice quality, it must be able to handle packet loss and transmission time jitter efficiently.



Amazon EX1010 Page 352

In this chapter, we discussed a number of techniques to address the technical challenges imposed by the network on VoIP. We concluded that, with proper design, it is generally possible to achieve VoIP voice quality that is equal or even better than PSTN. The extension of the signal bandwidth to 8 kHz is a major contribution towards improved speech quality. Multiple description coding is a powerful technique to address packet loss and delay in an efficient manner. It has been proven in practical applications and provides a theoretical framework that facilitates further improvement. Significant contributions towards robustness and minimizing overall delay can also be made by the usage of adaptive jitter buffers that provide flexible packet loss concealment. The combination of wide-band, multiple description coding, and packet loss concealment facilitates VoIP with high speech quality and a reasonable latency.

## References


15.1 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice Hall, Englewood Cliffs 1978)

15.2 W. Stallings: *High-Speed Networks: TCP/IP and ATM Design Principles* (Prentice Hall, Englewood Cliffs 1998)

15.3 Information Sciences Institute: Transmission control protocol, IETF **RFC793** (1981)

15.4 J. Postel: User datagram protocol, IETF **RFC768** (1980)

15.5 H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP a transport protocol for real-time applications, IETF **RFC3550** (2003)

15.6 ITU-T: G.131: Talker echo and its control (2003)

15.7 ITU-T: G.114: One-way transmission time (2003)

15.8 C.G. Davis: An experimental pulse code modulation system for short haul trunks, Bell Syst. Tech. J. **41**, 25–97 (1962)

15.9 IEEE: 802.11: Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications (2003)

15.10 IEEE: 802.15.1: Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs) (2005)

15.11 E. Dimitriou, P. Sörqvist: Internet telephony over WLANs, 2003 USTAs Telecom Eng. Conf. Supercomm (2003)

15.12 ITU-T: G.711: Pulse code modulation (PCM) of voice frequencies (1988)

15.13 IEEE: 802.1D Media access control (MAC) bridges (2004)

15.14 D. Grossman: New terminology and clarifications for diffserv, IETF **RFC3260** (2002)

15.15 R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: Resource ReSerVation Protocol (RSVP) – Version 1 Functional specification, IETF **RFC2205** (1997)

15.16 C. Aurrecoechea, A.T. Campbell, L. Hauw: A survey of QoS architectures, Multimedia Syst. **6**(3), 138–151 (1998)

15.17 IEEE: 802.11e: Medium Access Control (MAC) Quality of Service (QoS) Enhancements (2005)

15.18 E. Hänsler, G. Schmidt: *Acoustic Echo and Noise Control – A Practical Approach* (Wiley, New York 2004)

15.19 ITU-T: G.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP) (1996)

15.20 S. Andersen, A. Duric, H. Astrom, R. Hagen, W.B. Kleijn, J. Linden: Internet Low Bit Rate Codec (iLBC), IETF **RFC3951** (2004)

15.21 Ajay Bakre: www.globalipsound.com/datasheets/isac.pdf (2006)

15.22 S.B. Moon, J.F. Kurose, D.F. Towsley: Packet audio playout delay adjustment: Performance bounds and algorithms, Multimedia Syst. **6**(1), 17–28 (1998)

15.23 Ajay Bakre: www.globalipsound.com/datasheets/neteq.pdf (2006)

15.24 Y. Liang, N. Farber, B. Girod: Adaptive playout scheduling and loss concealment for voice communication over IP networks, IEEE Trans. Multimedia **5**(4), 257–259 (2003)

15.25 F. Liu, J. Kim, C.-C.J. Kuo: Adaptive delay concealment for internet voice applications with packet-based time-scale modification, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001)

15.26 ITU-T: P.800: Methods for subjective determination of transmission quality (1996)

15.27 S. Pennock: Accuracy of the perceptual evaluation of speech quality (PESQ) algorithm, Proc. Measurement of Speech and Audio Quality in Networks (2002)

15.28 M. Varela, I. Marsh, B. Grönvall: A systematic study of PESQs behavior (from a networking perspective), Proc. Measurement of Speech and Audio Quality in Networks (2006)

15.29 ITU-T: P.862: Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs (2001)

15.30 ITU-T: P.862.1: Mapping function for transforming P.862 raw result scores to MOS-LQO (2003)

15.31 C. Perkins, O. Hodson, V. Hardman: A survey of packet loss recovery techniques for streaming audio, IEEE Network **12**, 40–48 (1998)

15.32 J. Rosenberg, H. Schulzrinne: An RTP payload format for generic forward error correction, IETF **RFC2733** (1999)


Amazon EX1010 Page 353

15.33 J. Lacan, V. Roca, J. Peltotalo, S. Peltotalo: Reed–Solomon forward error correction (FEC), IETF (2007), work in progress

15.34 J. Rosenberg, L. Qiu, H. Schulzrinne: Integrating packet FEC into adaptive voice playout buffer algorithms on the internet, Proc. Conf. Comp. Comm. (IEEE INFOCOM 2000) (2000) pp. 1705–1714

15.35 W. Jiang, H. Schulzrinne: Comparison and optimization of packet loss repair methods on VoIP perceived quality under bursty loss, Proc. Int. Workshop on Network and Operating System Support for Digital Audio and Video (2002)

15.36 E. Martinian, C.-E.W. Sundberg: Burst erasure correction codes with low decoding delay, IEEE Trans. Inform. Theory **50**(10), 2494–2502 (2004)

15.37 C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. Bolot, A. Vega-Garcia, S. Fosse-Parisis: RTP payload format for redundant audio data, IETF **RFC2198** (1997)

15.38 J.-C. Bolot, S. Fosse-Parisis, D. Towsley: Adaptive FEC-based error control for internet telephony, Proc. Conf. Comp. Comm. (IEEE INFOCOMM '99) (IEEE, New York 1999) p. 1453-1460

15.39 T.M. Cover, J.A. Thomas: *Elements of Information Theory* (Wiley, New York 1991)

15.40 A.A.E. Gamal, T.M. Cover: Achievable rates for multiple descriptions, IEEE Trans. Inform. Theory **IT-28**(1), 851–857 (1982)

15.41 L. Ozarow: On a source coding prolem with two channels and three receivers, Bell Syst. Tech. J. **59**, 1909–1921 (1980)

15.42 V.A. Vaishampayan, J. Batllo: Asymptotic analysis of multiple description quantizers, IEEE Trans. Inform. Theory **44**(1), 278–284 (1998)

15.43 N.S. Jayant, S.W. Christensen: Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure, IEEE Trans. Commun. **COM-29**(2), 101–109 (1981)

15.44 N.S. Jayant: Subsampling of a DPCM speech channel to provide two self-contained half-rate channels, Bell Syst. Tech. J. **60**(4), 501–509 (1981)

15.45 A. Ingle, V.A. Vaishampayan: DPCM system design for diversity systems with applications to packetized speech, IEEE Trans. Speech Audio Process. **3**(1), 48–58 (1995)

15.46 A.O.W. Jiang: Multiple description speech coding for robust communication over lossy packet networks, IEEE Int. Conf. Multimedia and Expo (2000) pp. 444–447

15.47 V.K. Goyal: Multiple description coding: Compression meets the network, IEEE Signal Process. Mag. **18**, 74–93 (2001)

15.48 A.D. Wyner: Recent results in the Shannon theory, IEEE Trans. Inform. Theory **20**(1), 2–10 (1974)

15.49 A.D. Wyner, J. Ziv: The rate-distortion function for source coding with side information at the decoder, IEEE Trans. Inform. Theory **22**(1), 1–10 (1976)

15.50 V.A. Vaishampayan: Design of multiple description scalar quantizers, IEEE Trans. Inform. Theory **IT-39**(4), 821–834 (1993)

15.51 V.A. Vaishampayan, J. Domaszewicz: Design of entropy-constrained multiple-description scalar quantizers, IEEE Trans. Inform. Theory **IT-40**(4), 245–250 (1994)

15.52 N. Görtz, P. Leelapornchai: Optimization of the index assignments for multiple description vector quantizers, IEEE Trans. Commun. **51**(3), 336–340 (2003)

15.53 R.M. Gray: *Source Coding Theory* (Kluwer, Dordrecht 1990)

15.54 V.A. Vaishampayan, N.J.A. Sloane, S.D. Servetto: Multiple-description vector quantization with lattice codebooks: Design and analysis, IEEE Trans. Inform. Theory **47**(1), 1718–1734 (2001)

15.55 S.N. Diggavi, N. Sloane, V.A. Vaishampayan: Asymmetric multiple description lattice vector quantizers, IEEE Trans. Inform. Theory **48**(1), 174–191 (2002)

15.56 Y. Wang, M.T. Orchard, A.R. Reibman: Multiple description image coding for noisy channels by pairing transform coefficients, IEEE Workshop on Multimedia Signal Processing (1997) pp. 419–424

15.57 V.K. Goyal, J. Kovacevic: Generalized multiple description coding with correlating transforms, IEEE Trans. Inform. Theory **47**(6), 2199–2224 (2001)

15.58 T. Lookabough, R. Gray: High-resolution theory and the vector quantizer advantage, IEEE Trans. Inform. Theory **IT-35**(5), 1020–1033 (1989)

15.59 Ajay Bakre: www.globalipsound.com/datasheets/ipcm-wb.pdf (2006)

15.60 J. Batllo, V.A. Vaishampayan: Asymptotic performance of multiple description transform codes, IEEE Trans. Inform. Theory **43**(1), 703–707 (1997)

15.61 D.W. Griffin, J.S. Lim: Signal estimation from modified short-time Fourier transform, IEEE Trans. Acoust. Speech Signal Process. **32**, 236–243 (1984)

15.62 S. Roucos, A. Wilgus: High quality time-scale modification for speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 493–496

15.63 W. Verhelst, M. Roelands: An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1993) pp. 554–557

15.64 H. Sanneck, A. Stenger, K. Ben Younes, B. Girod: A new technique for audio packet loss concealment, Proc. Global Telecomm. Conf GLOBECOM (1996) pp. 48–52

15.65 D.J. Goodman, G.B. Lockhart, O.J. Wasem, W.C. Wong: Waveform substitution techniques for recovering missing speech segments in packet voice communications, IEEE Trans. Acoust. Speech Signal Process. **34**, 1440–1448 (1986)

15.66 O.J. Wasem, D.J. Goodman, C.A. Dvorak, H.G. Page: The effect of waveform substitution on the qual-




Amazon EX1010 Page 354

ity of PCM packet communications, IEEE Trans. Acoust. Speech Signal Process. **36**(3), 342–348 (1988)
15.67 ITU-T: G.711 Appendix I: A high quality low-complexity algorithm for packet loss concealment with G.711 (1999)
15.68 E. Gündüzhan, K. Momtahan: A linear prediction based packet loss concealment algorithm for PCM coded speech, IEEE Trans. Acoust. Speech Signal Process. **9**(8), 778–785 (2001)
15.69 J. Lindblom, P. Hedelin: Packet loss concealment based on sinusoidal extrapolation, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (2002) pp. 173–176
15.70 K. Clüver, P. Noll: Reconstruction of missing speech frames using sub-band excitation, Int. Symp. Time-Frequency and Time-Scale Analysis (1996) pp. 277–280
15.71 G. Kubin: Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (1996) pp. 267–270
15.72 C.A. Rodbro, M.N. Murthi, S.V. Andersen, S.H. Jensen: Hidden Markov Model-based packet loss concealment for voice over IP, IEEE Trans. Speech Audio Process. **14**(5), 1609–1623 (2006 )



Amazon EX1010 Page 355

# 16. Low-Bit-Rate Speech Coding

A. V. McCree

Low-bit-rate speech coding, at rates below 4 kb/s, is needed for both communication and voice storage applications. At such low rates, full encoding of the speech waveform is not possible; therefore, low-rate coders rely instead on parametric models to represent only the most perceptually relevant aspects of speech. While there are a number of different approaches for this modeling, all can be related to the basic linear model of speech production, where an excitation signal drives a vocal-tract filter.

The basic properties of the speech signal and of human speech perception can explain the principles of parametric speech coding as applied in early vocoders. Current speech modeling approaches, such as mixed excitation linear prediction, sinusoidal coding, and waveform interpolation, use more-sophisticated versions of these same concepts. Modern techniques for encoding the model parameters, in particular using the theory of vector quantization, allow the encoding of the model information with very few bits per speech frame.

Successful standardization of low-rate coders has enabled their widespread use for both military and satellite communications, at rates from 4 kb/s all the way down to 600 b/s. However, the goal of toll-quality low-rate coding continues to provide a research challenge.

16.1 **Speech Coding** ..... 331

16.2 **Fundamentals: Parametric Modeling of Speech Signals** ..... 332
16.2.1 Speech Production ..... 332
16.2.2 Human Speech Perception ..... 334
16.2.3 Vocoders ..... 335

16.3 **Flexible Parametric Models** ..... 337
16.3.1 Mixed Excitation Linear Prediction (MELP) ..... 337
16.3.2 Sinusoidal Coding ..... 341
16.3.3 Waveform Interpolation ..... 342
16.3.4 Comparison and Contrast of Modeling Approaches ..... 343

16.4 **Efficient Quantization of Model Parameters** ..... 344
16.4.1 Vector Quantization ..... 344
16.4.2 Exploiting Temporal Properties ..... 344
16.4.3 LPC Filter Quantization ..... 345

16.5 **Low-Rate Speech Coding Standards** ..... 345
16.5.1 MIL-STD 3005 ..... 345
16.5.2 The NATO STANAG 4591 ..... 346
16.5.3 Satellite Communications ..... 346
16.5.4 ITU 4 kb/s Standardization ..... 347

16.6 **Summary** ..... 347

**References** ..... 347



## 16.1 Speech Coding

As digital computers and communication systems continue to spread through our modern society, the use of digitized speech signals is increasingly common. The large number of bits required for accurate reproduction of the speech waveform makes many of these systems complex and expensive, so more-efficient encoding of speech signals is desirable. For example, limited radio bandwidth is a major constraint in the design of public mobile telephone systems, and the speech data rate directly influences the bandwidth requirement. In military tactical communications, a system with a lower speech data rate can use less transmitter power to make detection more difficult, or it can allow higher signal-to-noise ratios to improve performance in a hostile jamming environment. Also, computer storage of speech, such as in voice mail or voice response systems, becomes cheaper if the number of bits required for speech storage can be reduced. These are just some of the applications that can benefit from the development of algorithms that significantly reduce the speech data rate.

Amazon EX1010 Page 356

At bit rates above 4 kb/s, speech-specific waveform coders based on code excited linear prediction (CELP) [16.1] can produce good-quality speech. But at lower rates, it becomes very difficult to encode all of the information in the speech signal. Therefore, most low-rate coders model only the key perceptual features of speech, rather than the entire waveform. Since this is typically done by encoding the parameters of a linear model for speech, these are called parametric speech coders.

In Sect. 16.2, this presentation of low-rate speech coding begins with a review of the basics of human speech production and perception, and then introduces the linear model of speech as the basis for parametric speech coding. Section 16.3 then presents modern low-rate speech coding models using mixed excitation linear prediction, sinusoidal coding, and waveform interpolation. After a discussion of techniques for quantizing the model parameters in Sect. 16.4, complete low-rate coder designs that have been standardized for use in a range of applications are presented in Sect. 16.5, followed by conclusions and a summary in Sect. 16.6.



## 16.2 Fundamentals: Parametric Modeling of Speech Signals

For parametric coding, it is important to understand the properties of speech signals and which of their characteristics need to be preserved to provide high quality to a human listener. Therefore, in this section, we review human speech production and perception and then introduce the classical vocoder algorithms.

### 16.2.1 Speech Production

Speech is a sequence of sounds generated by the human vocal system [16.2, 3]. The acoustic energy necessary to produce speech is generated by exhaling air from the lungs. This air stream is used to produce sound in two different ways: by vibrating the vocal cords or by forcing air turbulence. If the vocal cords are used, the speech is referred to as voiced speech; otherwise, the speech is called unvoiced. In voiced speech, the opening and closing of the vocal cords at the glottis produces quasiperiodic puffs of air called glottal pulses, which excite the acoustic tubes of the vocal and nasal tracts. The average spacing between glottal pulses is called the pitch period. The frequency content of the resulting acoustic wave propagating from the mouth depends on the spectrum of the glottal pulses and the configuration of the vocal tract. Typically, glottal pulses are roughly triangular in shape with a sharp discontinuity at the closure of the glottis, as shown in Fig. 16.1. As this stylized example illustrates, glottal pulses have most of their energy concentrated at lower frequencies.

Unvoiced sounds result from turbulent noise exciting the vocal and nasal tracts. The turbulence can come from simply breathing air out of the lungs in a process called aspiration. This is the excitation source for whispered speech and for the /h/ sound. Turbulence can also be

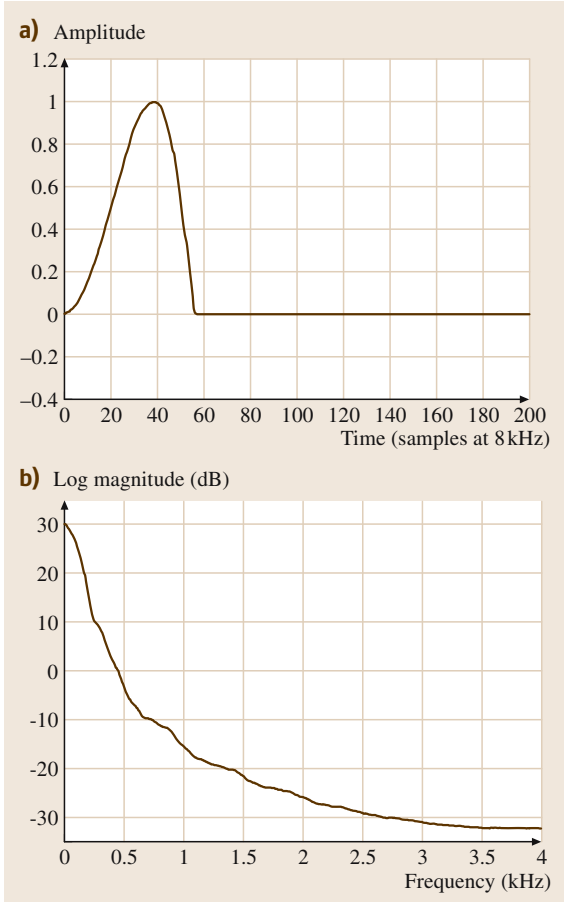



**Fig. 16.1** **(a)** Stylized glottal pulse and **(b)** corresponding Fourier transform (after *Rabiner* and *Schafer* [16.2])

Amazon EX1010 Page 357

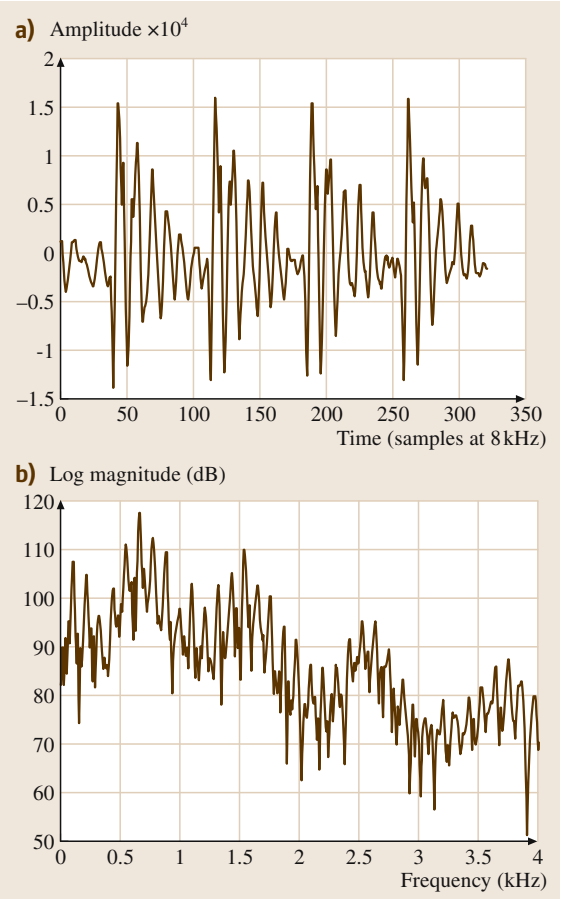



**Fig. 16.2a,b** Sustained vowel /æ/: **(a)** waveform and **(b)** Fourier spectrum

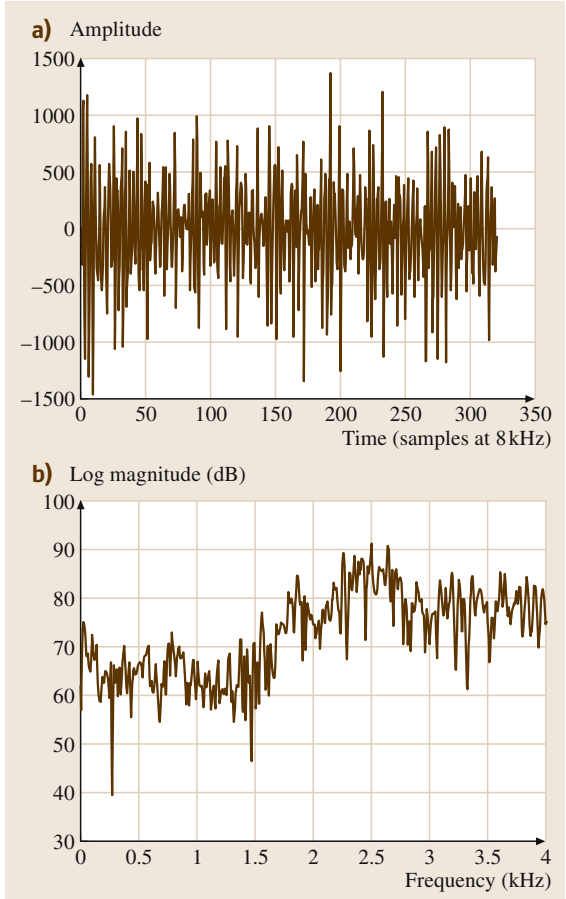



**Fig. 16.3a,b** Sustained fricative /sh/: **(a)** waveform and **(b)** Fourier spectrum

generated by forcing the air through a constriction in the vocal tract formed with the tongue or lips. This process, called frication, is used to make sounds such as /sh/.

Figures 16.2–16.4 show time waveforms and Fourier spectra for some typical speech sounds. Figure 16.2 shows the waveform and spectrum from a sustained vowel. Since this vowel was generated by periodic glottal pulses, the spectrum consists of harmonics of the pitch fundamental, in this case 110 Hz. The spectral peaks at 600, 1500, 2500, and 3600 Hz represent the formant frequencies in this example. The relatively high amplitudes of the first few harmonics are a result of the glottal pulse excitation. Figure 16.3 shows the waveform and spectrum of a sustained fricative. This sound is mainly high-frequency turbulent noise. An example of a plosive sound is shown in Fig. 16.4. Since this is an unvoiced plosive, the signal consists of silence during the vocal tract closure followed by a broadband noise burst.

The basic features of the speech production process can be captured in a simple time-varying linear model, as shown in Fig. 16.5. In this model, glottal pulses or random noise pass through a linear filter representing the vocal-tract frequency response. The parameters describing the excitation and vocal tract change as a function of time, tracking changes in the speech waveform. Key parameters include the voiced/unvoiced decision, pitch period, glottal pulse shape, vocal-tract filter coefficients, and power level. This model can directly mimic stationary sounds, whether voiced or unvoiced, by appropriate choice of excitation signal and vocal tract frequency response. Also, it can approximate nonstationary sounds if the model parameters are changed as rapidly as in natural speech.



Amazon EX1010 Page 358

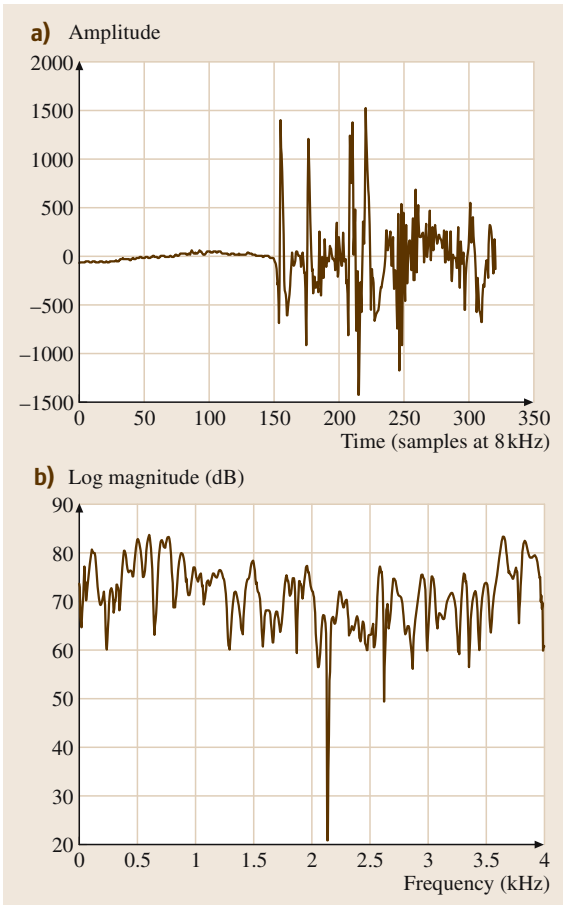



**Fig. 16.4a,b** Plosive /t/: **(a)** waveform and **(b)** Fourier spectrum

## 16.2.2 Human Speech Perception

Human perception of speech is determined by the capabilities of the human auditory system, which consists of the ear, the auditory nerve, and the brain [16.4]. The ear serves as a transducer, converting the acoustic input at the outer ear first to bone vibrations in the middle ear, then to fluid motion in the cochlea of the inner ear, and finally to electrical pulses generated by the inner hair cells in the cochlea. The location of maximum fluid vibration in the cochlea varies systematically with the input signal frequency, and this frequency response varies with the strength of the input signal. Also, the inner hair cells only detect motion in one direction. Thus, the ear acts like a very large bank of bandpass filters with dynamic range compression, and the output of each filter undergoes half-wave rectification. The half-wave-rectified bandpass-filtered outputs are transmitted across the auditory nerve to the lower levels of the brain, where specialized neurons can perform basic signal-processing operations. For example, there are neurons that respond to onsets and to decays, while other neurons may be able to estimate autocorrelations by comparing a signal to a delayed version of itself. The outputs of these neurons are then passed to higher levels of the brain for more-sophisticated processing. This results in the final analysis of the acoustic signal based on context and additional knowledge, such as classification of sound source and interpretation of pattern. In speech processing, this includes recognition of words as well as analysis of speaker identity and emotional state.

For a typical speech signal such as the vowel shown in Fig. 16.2, a human listener should be able to distin-

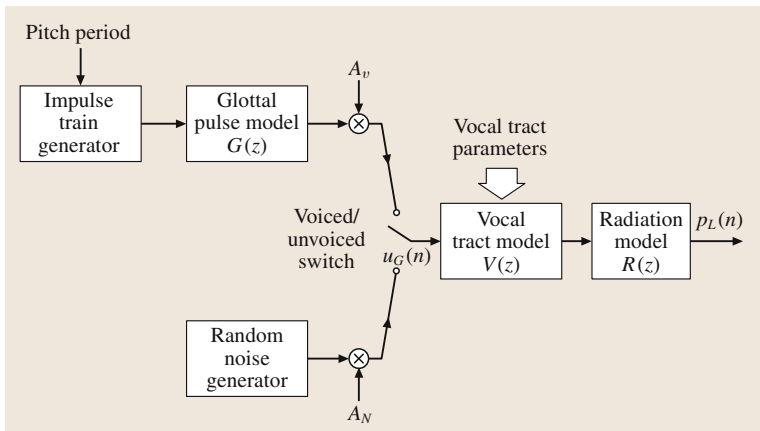



**Fig. 16.5** Simple linear model of speech production (after *Rabiner* and *Schafer* [16.2])

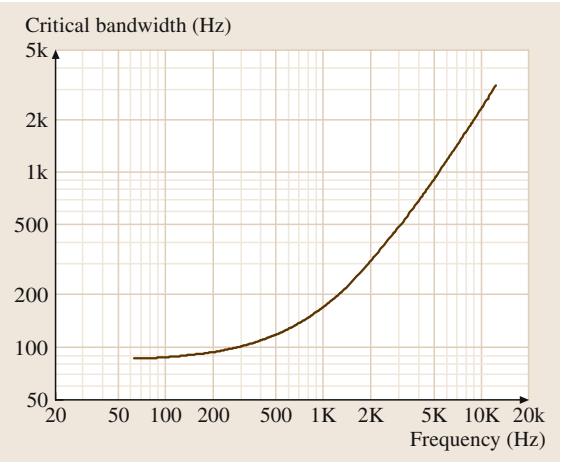



**Fig. 16.6** Critical bands as a function of center frequency

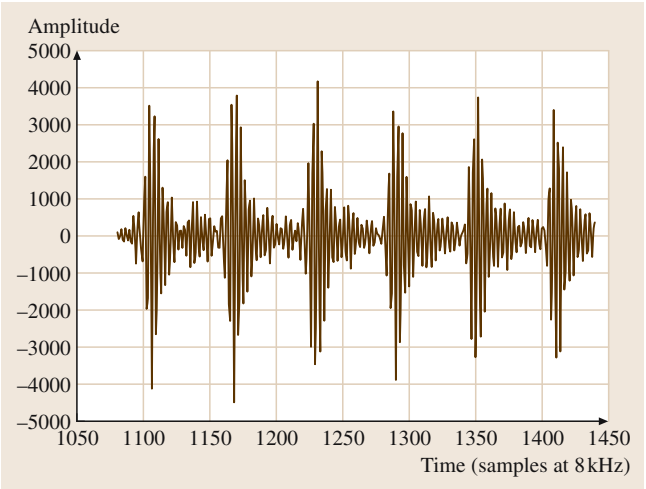



**Fig. 16.7** Example bandpass filter output

guish a number of features. A comparison of the average power in each bandpass filter gives an overall estimate of the power spectrum of the speech signal. This spectral analysis will also yield the exact frequency of the lower pitch harmonics. However, human listeners cannot distinguish individual frequencies present in a 12 tone signal unless the tones are separated by more than a measure of bandwidth called the critical band [16.5], and these bandwidths increase with frequency as shown in Fig. 16.6. Therefore, for a typical male pitch frequency of 100 Hz, individual pitch harmonics will not be distinguished above about 500 Hz. Higher-frequency harmonics near the formant frequencies may also be detected by this spectral analysis if they are much stronger than their immediate neighbors.

At higher frequencies, individual pitch harmonics cannot be resolved since each bandpass filter contains multiple pitch harmonics. However, there is a great deal of information in the time variation of the higher-frequency bandpass filter outputs. An example of a bandpass filter output centered at 2500 Hz from a sustained vowel is shown in Fig. 16.7. Notice that the individual pitch pulses can be seen in the rise and fall of this bandpass signal. It is reasonable to presume that the brain can detect pitch pulses as abrupt power transitions in the bandpass signals, and may also estimate pitch frequency from the average pulse spacing and voicing strength from the abruptness of the pulse onset. In fact, an experiment has shown that higher-frequency onset neurons in the cat respond with every pitch pulse in a sustained synthetic speech vowel [16.6].

Overall, the combination of spectral and bandpass waveform analysis allows the perception of many aspects of speech signals. For sustained or continuant sounds, the human listener should be able to estimate the spectral envelope, gain, harmonic content at lower frequencies, and pulse characteristics at higher frequencies. For noncontinuants, the human auditory system can detect rapid transitions in these characteristics, especially at high frequencies. These are the characteristics of the speech signal that we would expect low-rate coders to be able to reproduce.

### 16.2.3 Vocoders

Knowledge of speech production and speech perception can be combined to gain insight into efficient transmission and storage of speech signals. The ensemble of possible speech waveforms is fundamentally limited by the capabilities of the speech production process, and not all details of these waveforms are perceptually important. One particularly efficient method of speech compression is to analyze the parameters of the linear speech model, transmit these parameters across the communication channel, and synthesize a reproduction of the speech signal with a linear model. Speech coders based on this approach are called *vocoders* or *parametric* coders. If the parameters are updated 20 to 100 times per second, the synthesizer can reproduce much of the information of the speech signal using significantly fewer bits than direct waveform coding.

The first vocoder was developed by Homer *Dudley* in the 1930s [16.7]. In this method, the input speech is divided into 10 frequency channels, and the spectral envelope information representing the vocal tract is



Amazon EX1010 Page 360

transmitted as the average power in each channel. The synthesizer applies either a pulse train or random noise to an identical bank of bandpass filters. This approach, now called a channel vocoder, can produce speech that is intelligible but not of very high quality.

Because it uses only a fairly small number of frequency values to represent the spectral envelope, the channel vocoder cannot accurately reproduce the perceptually important formant structure of voiced speech. A better approach, called the formant vocoder [16.8, 9], transmits information about the formant frequencies directly and synthesizes speech with a small number of damped resonators or poles. The poles can be connected either in parallel or in cascade. Since they better model the spectral envelope of voiced speech, formant vocoders can produce speech of higher quality than channel vocoders. Unfortunately, the formants are difficult to estimate reliably in the analysis procedure, so formant vocoders may make annoying errors in tracking the formant frequencies. Formant synthesizers are more often used in text-to-speech applications where the formant frequencies can be generated by fixed rules.

The difficulty of formant tracking can be avoided if more poles are used to model the entire speech spectrum, without explicit identification of formant frequencies. In this way, the formants will be automatically captured when they dominate the spectrum, but more-subtle spectral features due to glottal pulse shape or nasal zeros can still be approximately reproduced. This principle is used in vocoders based on *linear predictive coding* (LPC), first introduced around 1970 [16.10, 11]. The term linear prediction is used because the all-pole coefficients are derived from the best predictor of the current input speech sample from a linear combination of previous samples:

$$\hat{s}(n) = \sum_{m=1}^{P} a_m s(n-m) \,, \tag{16.1}$$

where $a_m$ are the set of $P$ predictor coefficients. Minimum mean-square error estimation results in a highly efficient time-domain algorithm to generate these filter coefficients [16.2]. A block diagram of the LPC synthesizer is shown in Fig. 16.8. Either a periodic impulse train or white noise is used to excite an all-pole filter, and the output is scaled to match the level of the input speech. The voicing decision, pitch period, filter coefficients, and gain are updated for every block of input speech (called a speech frame) to track changes in the input speech.

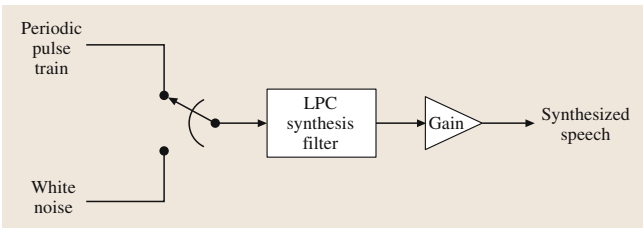



**Fig. 16.8** LPC vocoder synthesizer

Since the basic LPC vocoder does not produce high-quality speech, there has been significant effort aimed at improving the standard model. One well-known problem with vocoder speech output is a strong buzzy quality. Formant synthesizers use a *mixed excitation* with simultaneous pulse and noise components to address this problem [16.8, 9], an idea which has also been used in channel vocoders [16.12] and LPC vocoders [16.13–15]. Most of these early mixed excitation systems used a low-pass-filtered pulse combined with high-pass-filtered noise, although a multiband mixture algorithm with separate voicing decisions in each of three bands was used in [16.12]. Other attempts at vocoder improvement included using more realistic excitation pulses [16.16] and pitch-synchronous LPC analysis [16.17]. *Atal* and *David* proposed encoding the Fourier coefficients of the LPC excitation signal [16.18]. Mathematically, this can be written as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \exp(\mathrm{j}\omega_k n) \,, \tag{16.2}$$

where $x(n)$ is the excitation signal, $N$ is the pitch period, $k$ is a frequency index, $X(k)$ is a complex amplitude, and $\omega_k = \frac{2\pi k}{N}$ is the frequency of the $k$-th harmonic. Since $x(n)$ is real, this can equivalently be written as:

$$x(n) = \sum_{k=0}^{N/2} [A_\mathrm{c}(k) \cos(\omega_k n) + A_\mathrm{s}(k) \sin(\omega_k n)] \,, \tag{16.3}$$

where $A_\mathrm{c}(k)$ and $A_\mathrm{s}(k)$ are real amplitudes, or as:

$$x(n) = \sum_{k=0}^{N/2} A(k) \cos(\omega_k n + \phi_k) \,, \tag{16.4}$$

where $A(k)$ and $\phi_k$ represent the magnitudes and phases of the harmonics. These coefficients were analyzed with either a pitch-synchronous Fourier series analysis or a frame-based Fourier transform, and the excitation was synthesized as a sum of harmonic sine waves.

While these techniques provided some improvement, as of the early 1980s these enhanced LPC vocoders

Amazon EX1010 Page 361

were still unable to provide reliable communications in all environments. However, they did provide the background for newer approaches that have provided significant improvements in speech quality at low rates. The more-recent technical progress enabling this success has come in two areas: flexible parametric models and efficient quantization techniques. These are the subjects of the next two sections.

## 16.3 Flexible Parametric Models

Three more-recent speech models have been shown to provide significant performance improvements over the traditional LPC vocoder model: mixed excitation linear prediction, sinusoidal coding, and waveform interpolation. In this section, we review each of these approaches.

### 16.3.1 Mixed Excitation Linear Prediction (MELP)

The primary feature of the MELP model, shown in Fig. 16.9, is the use of mixed excitation for more-robust LPC synthesis [16.19]. Other key features are the use of aperiodic pulses, adaptive spectral enhancement to match formant waveforms, and a pulse dispersion filter to better match natural excitation waveform characteristics. Finally, MELP also includes representation of the Fourier magnitudes of the excitation signal.

#### Mixed Excitation

MELP generates an excitation signal with different mixtures of pulse and noise in each of a number (typically five) of frequency bands [16.20]. As shown in Fig. 16.9, the pulse train and noise sequence are each passed through time-varying spectral shaping filters and then added together to give a full-band excitation. For each frame, the frequency shaping filter coefficients are generated by a weighted sum of fixed bandpass filters. The pulse filter is calculated as the sum of each of the bandpass filters weighted by the voicing strength in that band. The noise filter is generated by a similar weighted sum, with weights set to keep the total pulse and noise power constant in each frequency band. These two frequency shaping filters combine to give a spectrally flat excitation signal with a staircase approximation to any desired noise spectrum. Since only two filters are required regardless of the number of frequency bands, this structure is more computationally efficient than using a bank of bandpass filters [16.19].

To make full use of this mixed excitation synthesizer, the desired mixture spectral shaping must be estimated accurately for each frame. In MELP, the relative pulse and noise power in each frequency band is determined by an estimate of the voicing strength at that frequency in the input speech. An algorithm to estimate these voicing strengths combines two methods of analysis of the bandpass-filtered input speech. First, the periodicity in each band is estimated using the strength of the normalized correlation coefficients around the pitch lag, where the correlation coefficient for a delay $t$ is defined by

$$c(t) = \frac{\sum_{n=0}^{N-1} s(n)s(n+t)}{\sqrt{\sum_{n=0}^{N-1} s^2(n) \sum_{n=0}^{N-1} s^2(n+t)}} . \tag{16.5}$$

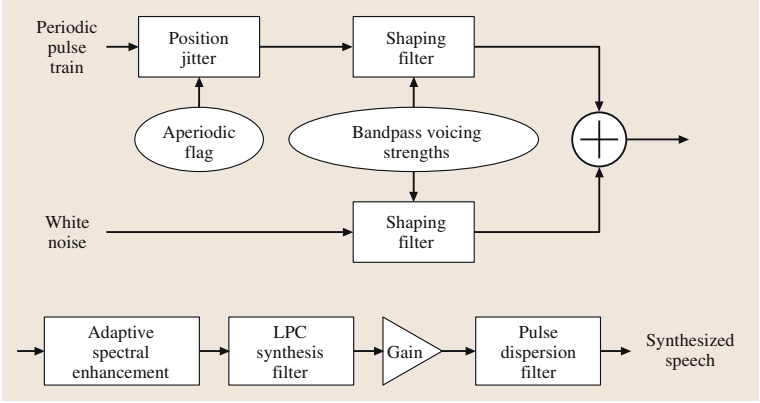



**Fig. 16.9** MELP synthesizer

This technique works well for stationary speech, but the correlation values can be too low in regions of varying pitch. The problem is worst at higher frequencies, and results in a slightly whispered quality to the synthetic speech. The second method is similar to the time-domain analysis of a wideband spectrogram. The envelopes of the bandpass-filtered speech are generated by full-wave rectification and smoothing with a one-pole low-pass filter, while using a first-order notch filter to remove the direct-current (DC) term from the output. At higher frequencies, these envelopes can be seen to rise and fall with each pitch pulse, just as in the spectrogram display. Autocorrelation analysis of these bandpass filter envelopes yields an estimate of the amount of pitch periodicity. The overall voicing strength in each frequency band is chosen as the largest of the correlation of the bandpass filtered input speech and the correlation of the envelope of the bandpass-filtered speech.



### Aperiodic Pulses

This mixed excitation can remove the buzzy quality from the speech output, but another distortion is sometimes apparent. This is the presence of short isolated tones in the synthesized speech, especially for female speakers. The tones can be eliminated by adding noise in the lower frequencies, but so much noise is required that the output speech sounds rough and noisy. A more-effective solution is to destroy the periodicity in the voiced excitation by varying each pitch period length with a pulse position jitter uniformly distributed up to ±25%. This allows the synthesizer to mimic the erratic glottal pulses that are often encountered in voicing transitions or in vocal fry [16.21]. This cannot be done for strongly voiced frames without introducing a hoarse quality, so a control algorithm is needed to determine when the jitter should be added.

Therefore, MELP adds a third voicing state to the voicing decision that is made at the transmitter [16.22]. The input speech is now classified as either voiced, jittery voiced, or unvoiced. In both voiced states, the synthesizer uses a mixed pulse/noise excitation, but in the jittery voiced state the synthesizer uses aperiodic pulses as shown in Fig. 16.9. This makes the problem of voicing detection easier, since strong voicing is defined by periodicity and is easily detected from the strength of the normalized correlation coefficient of the pitch search algorithm. Jittery voicing corresponds to erratic glottal pulses, so it can be detected by either marginal correlation or peakiness in the input speech. The peakiness $p$ is defined by the ratio of the root-mean-square (RMS) power to the average value of the full-wave-rectified LPC residual signal [16.23]:

$$p = \frac{\sqrt{\frac{1}{N}\sum_{n=0}^{N-1} s^2(n)}}{\frac{1}{N}\sum_{n=0}^{N-1} |s(n)|} \,. \tag{16.6}$$

This peakiness detector will detect unvoiced plosives as well as jittery voicing, but this is not a problem since the use of randomly spaced pulses has previously been suggested to improve the synthesis for plosives [16.15].

### Adaptive Spectral Enhancement

The third feature in the MELP model is adaptive spectral enhancement. This adaptive filter helps the bandpass-filtered synthetic speech to match natural speech waveforms in the formant regions. Typical formant resonances usually do not completely decay in the time between pitch pulses in either natural or synthetic speech, but the synthetic speech waveforms reach a lower valley between the peaks. This is probably caused by the inability of the poles in the LPC synthesis filter to reproduce the features of formant resonances in natural human speech. Here, there are two possible causes. First, the problem could be simply due to improper LPC pole bandwidth. The synthetic time signal may decay too quickly because the LPC pole has a weaker resonance than the true formant. Another possible explanation is that the true formant bandwidth may vary somewhat within the pitch period [16.8], and the synthetic speech cannot mimic this behavior. Informal experiments have shown that a time-varying LPC synthesis pole bandwidth can improve speech quality by modeling this effect, but it can be difficult to control [16.19].

The adaptive spectral enhancement filter provides a simpler solution to the problem of matching formant waveforms. This adaptive pole/zero filter was originally developed to reduce quantization noise between the formant frequencies in CELP coders [16.24]. The poles are generated by a bandwidth-expanded version of the LPC synthesis filter, where each $z^{-1}$ term in the $z$-transform of the LPC filter is replaced by $\alpha z^{-1}$, with $\alpha = 0.8$. Since this all-pole filter introduces a disturbing low-pass filtering effect by increasing the spectral tilt, a weaker all-zero filter calculated with $\alpha$ equal to 0.5 is used to decrease the tilt of the overall filter without reducing the formant enhancement. In addition, a simple first-order finite impulse response (FIR) filter is used to further reduce the low-pass muffling effect [16.25].

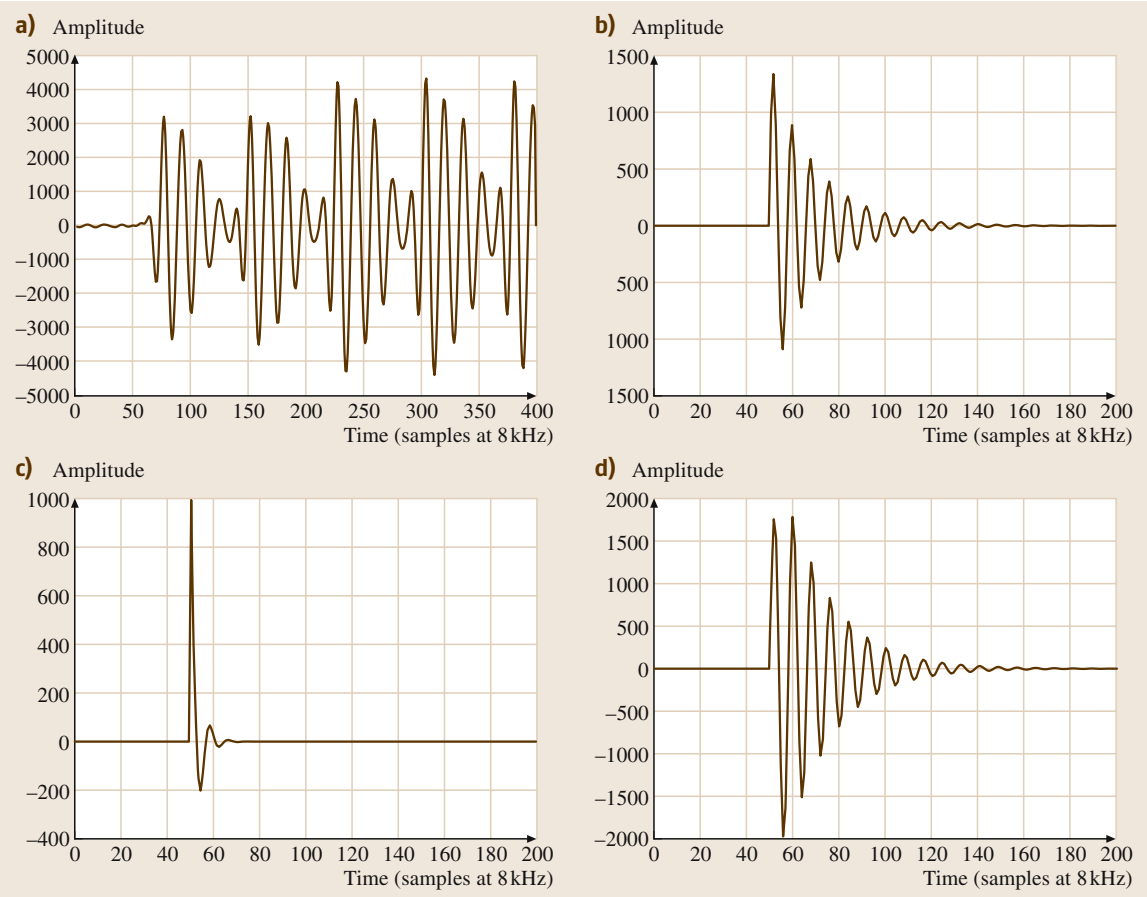



**Fig. 16.10a–d** Natural speech versus decaying resonance waveforms: **(a)** first formant of natural speech vowel, **(b)** synthetic exponentially decaying resonance, **(c)** pole/zero enhancement filter impulse response for this resonance, and **(d)** enhanced decaying resonance

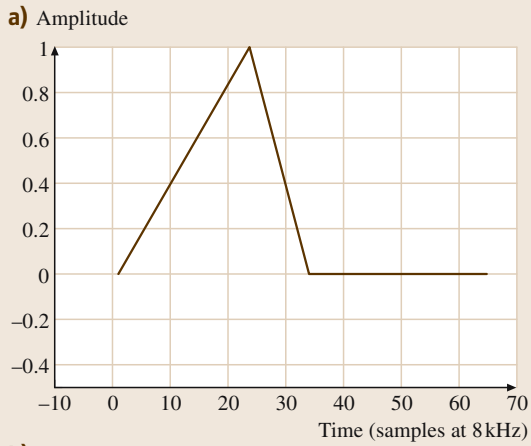
In the MELP coder, reducing quantization noise is not a concern, but the time-domain properties of this filter produce an effect similar to pitch-synchronous pole bandwidth modulation. As shown in Fig. 16.10, a simple decaying resonance has a less abrupt time-domain attack when this enhancement filter is applied. This feature allows the speech output to better match the bandpass waveform properties of natural speech in formant regions, and increases the perceived quality of the synthetic speech.

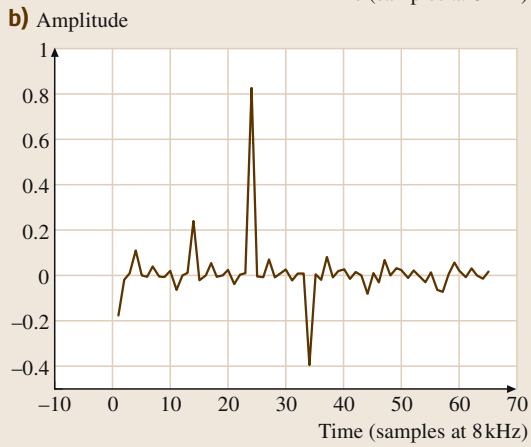
### Pulse Dispersion Filter

The pulse dispersion filter shown in Fig. 16.9 improves the match of bandpass-filtered synthetic and natural speech waveforms in frequency bands that do not contain a formant resonance. At these frequencies, the synthetic speech often decays to a very small value between the pitch pulses. This is also true for frequencies near the higher formants, since these resonances decay significantly between excitation points, especially for the longer pitch periods of male speakers. In these cases, the bandpass-filtered natural speech has a smaller peak-to-valley ratio than synthetic speech. In natural speech, the excitation may not all be concentrated at the point in time corresponding to the closure of the glottis [16.26]. This additional excitation prevents the natural bandpass envelope from falling as low as the synthetic version. This could be due to a secondary excitation peak from the opening of the glottis, aspiration noise resulting from incomplete glottal closure, or a small amount of acoustic background noise, which is visible in between the excitation peaks. In all these cases, there is a greater difference



Amazon EX1010 Page 364



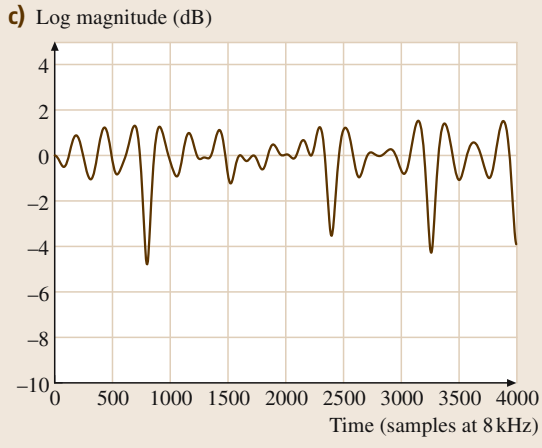




**Fig. 16.11a–c** Synthetic triangle pulse and FIR filter: **(a)** triangle waveform, **(b)** filter coefficients after spectral flattening with length 65 DFT, and **(c)** Fourier transform after spectral flattening

between the peak and valley levels of the bandpass-filtered waveform envelopes for the LPC speech than for natural human speech.

The pulse dispersion filter is a fixed FIR filter, based on a spectrally flattened synthetic glottal pulse, which introduces time-domain spread into the synthetic speech. In [16.19], a fixed triangle pulse [16.16, 27] based on a typical male pitch period is used, with the low-pass character removed from its frequency response. The filter coefficients are generated by taking a discrete Fourier transform (DFT) of the triangle pulse, setting the magnitudes to unity, and taking the inverse DFT. The dispersion filter is applied to the entire synthetic speech signal to avoid introducing delay into the excitation signal prior to synthesis. Figure 16.11 shows some properties of this triangle pulse and the resulting dispersion filter. The pulse has considerable time-domain spread, as well as some fine detail in its Fourier magnitude spectrum. Using this dispersion filter decreases the synthetic bandpass waveform peakiness in frequencies away from the formants and results in more natural sounding LPC speech output.

### Fourier Series Modeling

The final feature of the MELP model is a Fourier series expansion of the voiced excitation signal. Implementing a Fourier series expansion in MELP synthesis is straightforward, using the pitch-synchronous Fourier series synthesis of (16.4). Instead of using a simple time-domain digital impulse, each pitch pulse of the mixed excitation signal shown in Fig. 16.9 is generated by an inverse DFT of exactly one period in length. If the magnitude values of this DFT are all equal to one and the phases are all set to zero, this is simply an alternate representation for a digital impulse. As in [16.18], these magnitudes can be estimated using either a pitch-synchronous Fourier series analysis or a DFT with a longer window.

### MELP Model Improvements

A number of refinements to the basic MELP model have been demonstrated. Replacement of the frame-based autocorrelation pitch algorithm with a subframe-based approach that allows time-varying pitch within an analysis frame was shown to improve the reliability of pitch estimation in [16.28]. The use of a sliding pitch window, a specially designed plosive speech analysis and synthesis algorithm, and an excitation magnitude post-filter combined to improve performance in [16.29]. A wideband MELP coder, extended to 8 kHz

bandwidth, provided improved quality in [16.30]. An intelligibility enhancement preprocessor that emphasizes the perceptually important formant transitions by combining adaptive spectral enhancement with variable-rate timescale modification improved intelligibility in [16.31]. Finally, a number of enhancements, including subframe pitch estimation, pitch-synchronous LPC and Fourier series analysis, and oversampled Fourier synthesis for fractional pitch values, showed significant performance improvement for female speakers [16.32].

### 16.3.2 Sinusoidal Coding

Another successful approach to low-rate speech coding is based on modeling speech as a sum of sine waves [16.33,34]. Initial encouraging work in this direction [16.35,36] led to the development of two successful speech coding techniques: the sinusoidal transform coder (STC) [16.37] and the multiband excited (MBE) vocoder [16.38].

#### The Sinusoidal Model of Speech

Many signals can be locally modeled as a sum of sine waves:

$$s(n) = \sum_{l=0}^{L} A(l) \cos(\omega_l n + \phi_l) , \qquad (16.7)$$

where $A(l)$, $\omega_l$, and $\phi_l$ are the magnitude, frequency, and phase of the $l$-th sine wave. Note that, while this equation is similar to (16.4), in this case we are not constraining the frequencies to be harmonically related, and we are analyzing the speech signal itself rather than the excitation signal. While it is clear that such a model is appropriate for quasiperiodic voiced speech, it can be shown that this representation can also provide a perceptually sufficient representation of unvoiced speech as long as the sinusoids are closely spaced in frequency [16.37]. The principle of sinusoidal speech coding is to estimate and transmit the sine wave parameters for each speech frame, and then to synthesize speech using these parameters.

#### Sinusoidal Parameter Estimation

The general problem of estimating the time-varying sine wave parameters such that the reconstructed speech signal is as close as possible to the original speech is difficult to solve analytically. However, if a speech signal is locally periodic, then minimizing the squared estimation error over a window spanning a number of consecutive pitch periods produces a straightforward solution: the optimal estimates of the complex amplitudes are given by the DFT values at the harmonics of the pitch fundamental frequency [16.33]. Also, since the nonharmonic DFT coefficients will be zero in this case, the sine wave frequencies correspond to the peaks of this spectral estimate. Based on these insights, a more-general sinusoidal analysis method has been developed and shown to be very effective in STC [16.37]:

- window the input speech signal, for example with a Hamming window with a duration of approximately 2–3 pitch periods
- compute the DFT
- find the sine wave frequencies as the locations of the peaks of the DFT magnitude
- estimate the magnitude and phase of each sinusoid based on the corresponding complex DFT coefficients.

An alternative approach, used in the MBE vocoder, is to assume that the speech signal is locally periodic, but to model the DFT distortion due to windowing explicitly as a frequency-domain convolution of the desired sine wave coefficients with the DFT of the window function [16.38]. Then the sinusoid frequencies are the harmonics of the pitch fundamental, and the minimum squared error estimates of the amplitudes and phases come from the speech spectrum at frequencies around the harmonic along with the spectrum of the window function.

Finally, the sine wave parameters can be estimated using analysis by synthesis, either in the frequency or time domain [16.39–41].

#### Synthesis

At first glance, it would seem that sinusoidal synthesis using (16.7) is straightforward. However, each frame of speech has different sine wave parameters, so the synthesizer must perform smoothing to avoid discontinuities at the frame boundaries.

One form of parameter interpolation involves matching sinusoids from one frame to the next and then continuously interpolating their amplitudes, phases, and frequencies [16.37]. For well-behaved signals this matching process is not difficult, but a general solution typically involves an additional algorithm for birth and death of sine wave tracks to handle difficult cases. Phase interpolation also presents difficulties, since the phase values are only relative within a period, and the number of periods from one frame to the next must also be estimated. An elegant solution is to assume that the phase trajectory can be modeled with a cubic polynomial as



Amazon EX1010 Page 366

a function of time; this is the simplest solution meeting endpoint constraints for both frequency and phase.

A less-complex synthesis technique is the overlap/add method. Each frame is synthesized once with the constant parameters from the previous frame, and again with the new parameter set. The two synthesized signals are appropriately weighted by windowing functions and then summed together to produce the final synthesis output. These windows are designed such that the previous parameter synthesis tapers down to zero, the new parameter synthesis tapers up from zero, and the sum of the windows is one for every sample. Despite performing synthesis more than once for at least some of the speech samples, overlap/add synthesis can reduce complexity since there is no need to compute interpolated values of the parameters for every sample. In addition, it is readily amenable to the use of efficient fast Fourier transform methods.

#### Low-Rate Coding Constraints

The general sinusoidal model has been used successfully in many applications. Unfortunately, for low-rate speech coding, the complete set of sine wave amplitudes, phases, and frequencies typically contains too much information for efficient quantization. Therefore, the linear speech model is used to impose constraints on the sine wave parameters. First, the voiced speech frequencies are assumed to be harmonics of the fundamental frequency. This eliminates the need to estimate and transmit the exact frequency of each sinusoid, and incorporating this constraint into the sine wave analysis procedure naturally leads to integrated frequency-domain pitch and voicing estimation procedures. In both STC and MBE, pitch estimation is performed by measuring the optimal harmonic sinusoidal fit over a range of possible fundamental frequencies, and then selecting the pitch value as the one providing the best performance by a distance metric.

The linear speech model also necessitates the explicit modeling of unvoiced speech. Voicing decisions can be made based on the fit of the harmonic sinusoidal model to the original speech, since voiced speech frames should produce a better model fit. Typically better performance is obtained with a *soft* voicing decision, where partial voicing is allowed with both voiced and unvoiced components. As in earlier mixed excitation work, this is done either using a cutoff frequency, where frequencies below the cutoff are voiced and those above are unvoiced [16.37], or with separate decisions for a number of frequency bands [16.38]. The unvoiced component can be synthesized either with sinusoidal synthesis with randomized phases, or with white-noise generation.

Since encoding of the phases can require a significant number of bits, low-rate sinusoidal coders use a parametric model for the phase as well. Since the voiced excitation signal should look like a pulse, it can be assumed to be zero phase with a flat magnitude. In this case, the overall magnitude and phase responses will come from the vocal-tract filter.

In addition, the vocal tract can be modeled parametrically. In particular, LPC coefficients can be fitted to the sine wave amplitudes. This can be done with a frequency-domain algorithm, which can also incorporate a mel-scale frequency warping to improve perceptual modeling [16.42, 43], or with a time-domain LPC analysis [16.44]. The resulting model will have a minimum phase output corresponding to this all-pole filter. Alternatively, the sinusoidal amplitudes can be quantized using nonparametric techniques such adaptive transform coding [16.38] or direct vector quantization [16.45–47].

#### Spectral Post-filtering

Since the synthetic speech from sinusoidal coders sometimes exhibits a muffling effect, a postfilter can be applied to deepen the spectral valleys between formant frequencies [16.33]. Motivated by the time-domain LPC post-filter of [16.24], a frequency-domain post-filter design method has been developed [16.43]. For each frequency, these post-filter weights correspond to the unity-gain spectrally flattened sine-wave amplitudes raised to a power of less than one for spectral compression. This post-filtering approach is used in both STC and MBE coders to improve speech quality.

### 16.3.3 Waveform Interpolation

For voiced speech, it is natural to think of the excitation signal as a sequence of pitch periods or cycles, with the waveform for each cycle representing the ideal excitation for the vocal-tract filter. From one pitch period to the next, these excitation waveforms will be similar but not identical, since the speech content is gradually changing. Therefore, we can extract these pitch cycle excitation waveforms at a slower rate, quantize them for transmission, and reconstruct the missing waveforms at the receiver. This is the principle behind another successful low-rate speech technique called *waveform interpolation* (WI) [16.48].

#### Analysis

The fundamental operation of WI is the extraction of the excitation pitch cycle waveform, called the characteristic waveform [16.49]. Each waveform is extracted as one period of the signal, so minimizing edge effects is important for accurate representation. Therefore, this analysis is performed on the LPC residual signal; in addition, some local adjustment of the window position is allowed to minimize the signal power near the boundaries.

To ensure that the characteristic waveforms evolve smoothly in time, an alignment process is performed, in which each extracted waveform is circularly shifted to maximize its correlation with the previous one. The result of this alignment process is a sequence of time-aligned characteristic waveforms, each with a similar shape.

#### Synthesis

Synthesis in WI coders is typically done using harmonic sine wave synthesis of the excitation signal followed by LPC synthesis. The fundamental frequency and the complex sine wave coefficients are interpolated for every sample, while the LPC coefficients are interpolated on a subframe basis. However, it is sometimes more convenient to perform the LPC synthesis filtering in the frequency domain prior to the sine wave synthesis [16.48].

Note that, since the alignment phase (the circular time shift of each characteristic waveform) is not preserved, the precise time offset from the beginning to the end of a frame of synthesized speech is not explicitly controlled and will depend upon the interpolated pitch contour for the frame. Therefore, WI coders are typically not time-synchronized with the input speech signal; instead they exhibit a slow time drifting which is not perceptually relevant.

#### Low-Rate WI Speech Models

At low bit rates, WI coders also exploit the linear speech model. Besides the concepts of pitch and vocal-tract filtering mentioned in the basic WI formulation, additional bit savings can be introduced by using a soft voicing concept and a parametric model for system phase.

In WI, voicing is represented by the decomposition of the characteristic waveform into two additive components: the slowly evolving waveform (SEW) and the rapidly evolving waveform (REW) [16.50]. The underlying idea is that the quasiperiodic voiced component changes slowly over time, while the unvoiced noisy component changes quickly. Analysis of the SEW/REW decomposition is done by low-pass and high-pass filtering the aligned characteristic waveforms over time, typically with a cutoff frequency of around 20 Hz. The SEW can then be carefully quantized at an update rate of 40 Hz, while the REW can be replaced by a noise signal with similar spectral characteristics (typically by randomizing the phase). Since the full SEW waveform contains a great deal of information, at low rates it is typically quantized by coding only the lower harmonic amplitudes, setting the other amplitudes to unity, and setting the phases to zero so that the LPC filter will provide an overall minimum phase response. For the REW, only a crude amplitude envelope is necessary, but it should be updated quite frequently.

#### High-Rate Improvements

More recent work has improved the performance of WI modeling at higher bit rates. A WI method with asymptotically perfect reconstruction properties was presented in [16.51], enabling very high quality output from the WI analysis/synthesis system. Similarly, a scalable WI coder using pitch-synchronous wavelets allows a wide range of bit rates to be utilized [16.52]. In this coder, higher bit rates achieve better speech quality by encoding higher resolutions of the wavelet decomposition.

### 16.3.4 Comparison and Contrast of Modeling Approaches

All of the low-rate coding models described can be viewed as representing the linear speech model of Fig. 16.5, where the parameters for transmission for each frame of speech include the pitch period (or frequency), gain, voicing information, and vocal-tract filter. Common elements of most MELP, sinusoidal, and WI coders are:

- vocal-tract filter modeling with LPC
- pitch extraction algorithm for excitation analysis
- incorporation of excitation Fourier amplitude spectrum
- zero-phase excitation
- soft voicing decision, allowing mixed excitation synthesis
- spectral enhancement filtering to sharpen formant frequencies

However, each coder has its own strengths since it is based on a different perspective of the speech signal. The MELP model extends the traditional vocoder view of the excitation as a sequence of pulses, so it is able to exploit pitch pulse characteristics such as waveform shape. In contrast, sinusoidal modeling is based on a Fourier transform view of the speech signal, allowing straight-



Amazon EX1010 Page 368

forward frame-based processing. Finally, WI views the excitation signal as an evolving Fourier series expansion, allowing smooth analysis and synthesis of a signal with changing characteristics including pitch.

## 16.4 Efficient Quantization of Model Parameters

For low-rate speech coding, a flexible parametric model provides a necessary first step, but efficient quantization is critical to an overall coder design. In this section, we review two very important quantization topics: vector quantization, and LPC coefficient quantization.



### 16.4.1 Vector Quantization

The traditional approach to the quantization of speech parameters, such as harmonic amplitudes, is to encode each one separately. In contrast to this *scalar quantization* approach, significant performance improvements can be attained by encoding multiple values simultaneously, by searching a codebook of possible vectors. Since both encoder and decoder have copies of the same codebook, the only information to be transmitted is the index of the selected code vector. This process, called *vector quantization* (VQ) [16.53], has a long history in low-rate speech coding. The earliest applications, in pattern-matching vocoders of the 1950s, were based on tables designed for specific speech phonemes [16.54]. Starting around the same time, a more-rigorous information-theoretic approach was developing based on *Shannon*'s work on block source coding and rate-distortion theory [16.55].

Since the complexity of vector quantization can become prohibitive when the number of code vectors is large, specially structured codebooks are often used. Theoretically, such approaches will lose coding efficiency compared to a full VQ, but they can have significant advantages in codebook storage size and search complexity. In one common approach, called *split* VQ, the vector is divided into subvectors, and each of these is vector quantized. As the number of subvectors becomes large, split VQ becomes scalar quantization, so this approach provides a straightforward way to adjust coding performance versus complexity. Another approach, *multistage* VQ (MSVQ), maintains full dimensionality but reconstructs a quantized vector as the sum of code vectors from a number of different codebooks, each of smaller size. A simple MSVQ search algorithm, called sequential search, processes the codebooks one stage at at time. The input vector is quantized using the first codebook, then the error from the this first stage is quantized using the second codebook, and this process continues through all of the codebooks. However, much better performance is typically achieved with a joint search MSVQ, where the best joint set of code vectors over all stages is selected.

### 16.4.2 Exploiting Temporal Properties

Further coding efficiency can be attained by exploiting the slowly varying characteristics of speech over time. One simple approach uses *predictive* VQ, where the parameter values for a current frame are first predicted from past quantized values, and only the error between this predicted value and the true input is quantized. Typically a linear combination of past values is used, resulting in vector linear prediction equations similar to the traditional linear prediction of (16.1), but with prediction across frames rather than samples. These prediction coefficients can be designed offline based on training data; alternatively in *switched predictive VQ* the prediction coefficients can be adjusted and transmitted for each frame based on the speech characteristics or to optimize quantizer performance. By using simple models of speech evolution, predictive quantization can provide significant bit-rate reduction without introducing additional delay.

More-general coding schemes quantize a sequence of speech frames together. The advantage of joint encoding is that redundancies between neighboring frames can be fully exploited, but the disadvantage can be a significant increase in delay. A fixed block of parameter vectors, often called a *superframe*, can be encoded simultaneously. The spectral vectors can be quantized directly using *matrix quantization*, where each codebook entry corresponds to a sequence of vectors. Alternatively, adaptive interpolation can be used to transmit only a subset of frames with each block, replacing nontransmitted frames using interpolation. More flexibility can be introduced by replacing fixed blocks with variable-length segments. In particular, in a phonetic vocoder, each segment corresponds to a speech phoneme [16.56]. In this case, the only information transmitted is what was said (the phonetic content) and the way it was said (prosody).

Amazon EX1010 Page 369

### 16.4.3 LPC Filter Quantization

In a low-rate coder utilizing linear prediction, the LPC filter coefficients must be quantized. It is well known that these coefficients are not suitable for scalar quantization because they require a large number of bits to maintain an accurate representation and the quantized filter tends to be unstable. For this reason, early quantization work used alternative representations such as the reflection coefficients [16.2, 57]; however, almost all recent work on this topic uses the more-powerful line spectral frequency (LSF) representation [16.58].

Originally developed as a speech synthesis structure, the LSFs are generated by finding the roots of the two $z$-transform polynomials corresponding to the LPC coefficients with additional reflection coefficients of 1 or $-1$. LSFs have many useful properties for both quantization and synthesizer interpolation. They have a natural ordering, and a valid set of LSFs guarantees a stable LPC filter. In addition, the frequency representation allows quantization to take advantage of known properties of human perception. The higher frequencies can be quantized less accurately than lower frequencies, and LSFs corresponding to sharp LPC poles at the perceptually important speech formants can be selected for more-accurate representation.

In modern low-rate coders, spectral quantization is typically done with some form of vector quantization of the LSFs. A common performance measure is spectral distortion (SD) between unquantized and quantized LPC spectra, with a performance goal of average distortion of 1 dB while minimizing outlier frames [16.59]. However, there is evidence that a critical-band weighted version of SD is a more-accurate predictor of perceived distortion [16.60]. Typically, more than 20 bits are required per frame to achieve this performance, and the complexity of full VQ with $2^{20}$ codewords becomes too high for reasonable applications. Therefore, most low-rate coders use either split [16.59] or [16.61] multistage VQ. Finally, to implement VQ in the LSF domain, a weighted Euclidean LSF distance measure is needed. Commonly used functions weight LSFs in the vicinity of formants more strongly based on the LPC power spectrum [16.59], but the optimal LSF weighting to optimize SD performance has been derived [16.62]. This has also been modified to incorporate critical-band weighting [16.63].



## 16.5 Low-Rate Speech Coding Standards

In this section, we present a number of successful low-rate coder designs that have been standardized for communication applications. First, we describe low-rate coders that have been standardized for military communication applications: the US MIL-STD 3005 2.4 kb/s MELP and the North Atlantic Treaty Organization (NATO) STANAG 4591 MELPe coding suite at 2400, 1200, and 600 b/s. Then, we present an overview of speech coding standards for satellite telephony based on MBE. Finally, we discuss more-recent efforts by the International Telecommunications Union (ITU) to standardize a toll-quality low-rate coder at 4 kb/s.

**Table 16.1** 2.4 kb/s MELP coder bit allocation

| Parameters | Voiced | Unvoiced |
|---|---|---|
| LSF | 25 | 25 |
| Fourier magnitudes | 8 | – |
| Gain (2 per frame) | 8 | 8 |
| Pitch and overall voicing | 7 | 7 |
| Bandpass voicing | 4 | – |
| Aperiodic flag | 1 | – |
| Error protection | – | 13 |
| Sync bit | 1 | 1 |
| Total bits/22.5 ms | 54 | 54 |

### 16.5.1 MIL-STD 3005

In the early 1993, the US government Department of Defense (DoD) digital voice processing consortium initiated a three-year project to select a new 2.4 kb/s coder to replace the older LPC-10e standard [16.64] for secure communications. This selection process consisted of two parts: evaluation of six minimum performance requirements and overall composite rating by figure of merit [16.65]. Intelligibility, quality, speaker recognizability, communicability, and complexity were all measured as part of this selection process.

This project led to the selection in 1996 of a 2.4 kb/s MELP coder as the new MIL-STD 3005, as it had the highest figure of merit among the coders that passed all the requirements. This coder uses a 22.5 ms speech frame, with the bit allocation shown in Table 16.1 [16.66,67]. The spectral envelope is represented

Amazon EX1010 Page 370

by a 10th-order LPC, quantized using MSVQ of the LSFs with four stages and an M-best search algorithm [16.61]. Additional spectral modeling is achieved using the first 10 residual Fourier harmonic magnitudes, which are estimated from the spectral peaks of a frame-based DFT and quantized with an 8 bit VQ. The gain is estimated twice per frame, and quantized with a joint 8 bit coding scheme. The logarithm of the pitch period is quantized with a scalar quantizer spanning lags from 20 to 160 samples, with a reserved all-zero code for unvoiced frames. Of the five bandpass voicing decisions, the lowest one is represented by the overall voiced/unvoiced decision, and the remaining four are encoded with one bit each.

In terms of intelligibility, quality, speaker recognizability, and communicability, the performance of the 2.4 kb/s MELP standard was much better than the earlier 2.4 kb/s LPC-10e [16.68]. This coder also met the project goal of performance at least as good as the older 4.8 kb/s FS1016 CELP standard [16.69], at only half the bit rate. Interestingly, despite their differences in design, three other coders, based on STC, WI, and MBE, provided similar performance, with the first two of these also able to meet all six performance requirements.

### 16.5.2 The NATO STANAG 4591

These impressive results led the North Atlantic Treaty Organization (NATO) to undertake a similar standardization process in 1997, this time targeting the dual bit rates of 2.4 kb/s and 1.2 kb/s. Candidates from three NATO nations were evaluated using tests for speech quality, intelligibility, speaker recognition, and language dependency [16.70–72]. An enhanced MELP coder, called MELPe, was then selected as the 2.4/1.2 kb/s NATO standard STANAG 4591 in 2001 [16.73].

This 2.4 kb/s MELPe coder is based on MIL-STD 3005 MELP, using the same quantization techniques and bit allocation, and is therefore completely interoperable with it. However, performance has been improved, primarily by the addition of a robust noise suppression front-end based on the minimum mean-square error of the log spectral amplitudes [16.74]. This 2.4 kb/s MELPe coder was extended to also operate at 1.2 kb/s, using a superframe of three consecutive 22.5 ms frames [16.75]. This approach allows the 1.2 kb/s MELPe coder to perform nearly as well as the 2.4 kb/s version, at the price of additional coding delay.

In 1.2 kb/s MELPe, the pitch trajectory and voicing pattern are jointly vector quantized using 12 bits for each superframe. The LSF quantization depends upon the voicing pattern, but for the most challenging case of all voiced frames a forward/backward interpolation scheme is used. This algorithm has three components: the last frame is quantized with the single-frame 25 bit MSVQ from the 2.4 kb/s coder, the optimal interpolation patterns for the first two frames are vector quantized with four bits, and finally the remaining error of these two frames is jointly quantized with 14 bit MSVQ. The Fourier magnitudes for the last frame are quantized with the same 8 bit VQ as in 2.4 kb/s, and the remaining frame magnitudes are regenerated using interpolation. The six gain values are vector quantized with 10 bits, and the four bandpass voicing decisions for each voiced frame are quantized with a two-bit codebook. The aperiodic flag is quantized with one bit per superframe, with codebooks selected by the overall voicing pattern.

In 2006, the NATO STANAG 4591 was extended to also operate at 600 b/s [16.76]. As with the 1.2 kb/s version, this coder is based on the 2.4 kb/s coder with a longer analysis superframe. In this case, four consecutive frames are grouped into a 90 ms superframe. For each superframe, the overall voicing and bandpass voicing decisions are first jointly vector quantized with five bits. The remaining quantization algorithms are mode-dependent, based on this voicing pattern. The pitch trajectory is quantized using up to 8 bits, using scalar quantization with adaptive interpolation. The LSFs are quantized using a matrix extension of MSVQ, with each matrix representing the concatenation of two consecutive input vectors. Finally, the eight gain values in a superframe are jointly quantized using either full VQ or MSVQ, depending on the mode. The aperiodic flag and Fourier magnitudes are not used at this low rate.

Overall, the NATO STANAG 4591 MELPe coder family provides communication-quality speech at bit rates of 2.4 kb/s, 1.2 kb/s, and 600 b/s. MELPe performance at 2.4 kb/s is better than 4.8 kb/s FS1016, and there is a graceful degradation in performance with decreasing bit rate. Even at 600 b/s, MELPe performance is still better than 2.4 kb/s LPC-10e [16.76].

### 16.5.3 Satellite Communications

There are a number of commercial satellite communication systems for use in situations where cellular telephony may be impractical, such as for maritime applications [16.77]. These include the Inmarsat system, as well as other satellite communication systems such as ICO Global Communications, Iridium, Asia Cellular Satellite (ACeS), Optus, and American Mo-

bile Satellite Corporation Telesat Mobile Incorporated (AMSC-TMI). While the published literature on speech coding standards in such systems is limited, many use versions of MBE tailored for bit rates around 4 kb/s, for example the IMBE coder described in [16.78] or the more-recent proprietary AMBE coder. These systems are designed to provide communications quality in potentially high bit error rates. According to one assessment [16.79], the AMBE coder intended for use in the Inmarsat Aeronautical system provides speech quality approximately equivalent to first-generation digital cellular (specifically the US time-division multiple-access (TDMA) full-rate standard vector sum excited linear prediction (VSELP) algorithm [16.80]).

In addition to these satellite communication systems, a similar technology has been standardized for North American digital land mobile radio systems by the Association of Public-Safety Communications Officials (APCO) Project 25.

### 16.5.4 ITU 4 kb/s Standardization

We conclude this section with a discussion of an effort by the telecommunications standardization sector of the International Telecommunications Union (ITU-T) to standardize a toll-quality 4 kb/s speech coder for a wide range of communication applications. The ambitious goals of this project were announced in 1994 [16.81].

Unfortunately, achieving these goals has proven more difficult than initially anticipated. In parametric low-rate coders, 4 kb/s improvements have focussed on better representation of the current model parameters, i. e., faster frame update, more-accurate modeling of spectral amplitudes, and the use of complex Fourier coefficients (or equivalently waveform phase information), as in [16.82–85].

Other approaches to the 4 kb/s problem abandoned the parametric model entirely and worked with higher-rate waveform coders, based on CELP, to lower their bit rate. Key to these approaches is the idea of relaxing the waveform matching constraint using the generalized analysis-by-synthesis coding paradigm, also known as RCELP [16.86]. Significant benefits are achieved by allowing the CELP synthesis to drift in time relative to the input signal according to the best pitch prediction path. This approach was taken in one of the most promising ITU 4 kb/s candidates [16.87].

A third alternative is a *hybrid* coder, using both waveform and parametric coding at different times [16.23, 49, 88, 89]. Since parametric coders are good at producing periodic voiced speech and waveform coders are good at modeling transitional frames, a hybrid coder uses each where it is best suited, and switches between the two as the speech characteristics changes. A hybrid MELP/CELP coder was the basis for the other top-performing candidate [16.90].

After almost a decade of effort for the ITU 4 kb/s standardization, it is clear that, while multiple low-rate coding approaches can sometimes produce very high speech quality in formal evaluations, it is difficult to achieve toll-quality performance in all conditions. The top candidates were able to improve their performance in multiple rounds of testing, but none were able to achieve toll quality reliably in all testing laboratories and languages, and as a result no ITU 4 kb/s standard has yet been named.

## 16.6 Summary

Low-rate speech coding can now provide reliable communications-quality speech at bit rates well below 4 kb/s. While there are a number of different approaches to achieve these rates, such as MELP, STC, MBE, and WI, all rely on flexible parametric models combined with sophisticated quantization techniques to achieve this performance. Successful standardization of low-rate coders has enabled their widespread use for military and satellite communications. However, the goal of toll-quality low-rate coding continues to provide a research challenge.

### References

16.1 M.R. Schroeder, B.S. Atal: Code excited linear prediction (CELP): High quality speech at very low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (Tampa 1985) pp. 937–940



Amazon EX1010 Page 372

16.2 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice Hall, Englewood Cliffs 1978)

16.3 G. Fant: *Acoustic Theory of Speech Production* (Mouton, The Hague 1960)

16.4 L.E. Kinsler: *Fundamentals of Acoustics*, 3rd edn. (Wiley, New York 1982)

16.5 B. Scharf: Critical bands. In: *Foundations of Modern Auditory Theory*, ed. by J.V. Tobias (Academic, New York 1970), Chap. 5

16.6 D.O. Kim, W.S. Rhode, S.R. Greenberg: Responses of cochlear nucleus neurons to speech signals: Neural encoding of pitch, intensity, and other parameters. In: *Auditory Frequency Selectivity*, ed. by B.C.J. Moore, R.D. Patterson (Plenum, New York 1986) pp. 281–288

16.7 H. Dudley: Remaking speech, J. Acoust. Soc. Am. **11**, 169–177 (1939)

16.8 J.N. Holmes: The influence of glottal waveform on the naturalness of speech from a parallel formant synthesizer, IEEE Trans. Audio Electroacoust. **21**, 298–305 (1973)

16.9 D.H. Klatt: Review of text-to-speech conversion for english, J. Acoust. Soc. Am. **82**, 737–793 (1987)

16.10 F. Itakura, S. Saito: Analysis synthesis telephony based on the maximum likelihood method, Rep. 6th Int. Congr. Acoustics (1968) pp. C17–C20

16.11 B.S. Atal, S.L. Hanauer: Speech analysis and synthesis by linear prediction of the speech wave, J. Acoust. Soc. Am. **50**(2), 637–655 (1971)

16.12 O. Fujimura: An approximation to voice aperiodicity, IEEE Trans. Audio Electroacoust. **16**, 68–72 (1968)

16.13 J. Makhoul, R. Viswanathan, R. Schwartz, A.W.F. Huggins: A mixed-source model for speech compression and synthesis, J. Acoust. Soc. Am. **64**(6), 1577–1581 (1978)

16.14 S.Y. Kwon, A.J. Goldberg: An enhanced LPC vocoder with no voiced/unvoiced switch, IEEE Trans. Acoust. Speech Signal Process. **32**, 851–858 (1984)

16.15 G.S. Kang, S.S. Everett: Improvement of the excitation source in the narrow-band linear prediction vocoder, IEEE Trans. Acoust. Speech Signal Process. **33**, 377–386 (1985)

16.16 M.R. Sambur, A.E. Rosenberg, L.R. Rabiner, C.A. McGonegal: On reducing the buzz in LPC synthesis, J. Acoust. Soc. Am. **63**, 918–924 (1978)

16.17 D.Y. Wong: On understanding the quality problems of LPC speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1980) pp. 725–728

16.18 B.S. Atal, N. David: On synthesizing natural-sounding speech by linear prediction, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1979) pp. 44–47

16.19 A. McCree, T.P. Barnwell III: A mixed excitation LPC vocoder model for low bit rate speech coding, IEEE Trans. Speech Audio Process. **3**(4), 242–250 (1995)

16.20 A. McCree, T.P. Barnwell III: Improving the performance of a mixed excitation LPC vocoder in acoustic noise, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1992) pp. II137–II140

16.21 W. Hess: *Pitch Determination of Speech Signals* (Springer, Berlin, Heidelberg 1983)

16.22 A. McCree, T.P. Barnwell III: A new mixed excitation LPC vocoder, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1991) pp. 593–596

16.23 D.L. Thomson, D.P. Prezas: Selective modeling of the LPC residual during unvoiced frames: White noise or pulse excitation, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1986) pp. 3087–3090

16.24 J.H. Chen, A. Gersho: Real-time vector APC speech coding at 4800 bps with adaptive postfiltering, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1987) pp. 2185–2188

16.25 W.B. Kleijn, D.J. Krasinski, R.H. Ketchum: Fast methods for the CELP speech coding algorithm, IEEE Trans. Acoust. Speech Signal Process. **38**(8), 1330–1342 (1990)

16.26 J.N. Holmes: Formant excitation before and after glottal closure, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1976) pp. 39–42

16.27 A.E. Rosenberg: Effect of glottal pulse shape on the quality of natural vowels, J. Acoust. Soc. Am. **49**, 583–590 (1971)

16.28 A. McCree, J.C. DeMartin: A 1.7 kb/s MELP coder with improved analysis and quantization, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1998) pp. 593–596

16.29 T. Unno, T.P. Barnwell III, K. Truong: An improved mixed excitation linear prediction (MELP) coder, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1999) pp. 245–248

16.30 W. Lin, S.N. Koh, X. Lin: Mixed excitation linear prediction coding of wideband speech at 8 kbps, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (2000) pp. II1137–II1140

16.31 N.R. Chong-White, R.V. Cox: An intelligibility enhancement for the mixed excitation linear prediction speech coder, IEEE Signal Process. Lett. **10**(9), 263–266 (2003)

16.32 A.E. Ertan, T.P. Barnwell III: Improving the 2.4 kb/s military standard MELP (MS-MELP) coder using pitch-synchronous analysis and synthesis techniques, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (2005) pp. 761–764

16.33 R.J. McAulay, T.F. Quatieri: Sinusoidal coding. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995), Chap. 4

16.34 T.F. Quatieri: *Discrete Time Speech Signal Processing: Principles and Practice* (Prentice Hall, Englewood Cliffs 2002), Chap. 9

16.35 P. Hedelin: A tone-oriented voice-excited vocoder, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1981) pp. 205–208




Amazon EX1010 Page 373

16.36 L.B. Almeida, F.M. Silva: Variable-frequency synthesis: An improved harmonic coding scheme, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1984), Sects. 27.5.1–27.5.4.

16.37 R.J. McAulay, T.F. Quatieri: Speech analysis/synthesis based on a sinusoidal representation, IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(4), 744–754 (1986)

16.38 D.W. Griffin, J.S. Lim: Multiband excitation vocoder, IEEE Trans. Acoust. Speech Signal Process. **36**(8), 1223–1235 (1988)

16.39 E.B. George, M.J.T. Smith: Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model, IEEE Trans. Speech Audio Process. **5**(5), 389–406 (1997)

16.40 C. Li, V. Cuperman: Analysis-by-synthesis multimode harmonic speech coding at 4 kb/s, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 3 (2000) pp. 1367–1370

16.41 C.O. Etemoglu, V. Cuperman, A. Gersho: Speech coding with an analysis-by-synthesis sinusoidal model, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 3 (2000) pp. 1371–1374

16.42 M.S. Brandstein: *A 1.5 kbps multi-band excitation speech coder, M.S. Thesis* (Massachusetts Institute of Technology, Cambridge 1990)

16.43 R. McAulay, T. Parks, T. Quatieri, M. Sabin: Sine-wave amplitude coding at low data rates. In: *Advances in Speech Coding*, ed. by B.S. Atal, V. Cuperman, A. Gersho (Kluwer Academic, Boston 1991) pp. 203–214

16.44 S. Yeldener, A.M. Kondoz, B.G. Evans: High quality multiband LPC coding of speech at 2.4 kbit/s, Electron. Lett. **27**(14), 1287–1289 (1991)

16.45 M. Nishiguchi, J. Matsumoto, R. Wakatsuki, S. Ono: Vector quantized MBE with simplified V/UV division at 3.0 kbit/s, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 2 (1993) pp. 151–154

16.46 A. Das, A.V. Rao, A. Gersho: Variable-dimension vector quantization, IEEE Signal Process. Lett. **3**(7), 200–202 (1996)

16.47 P. Lupini, V. Cuperman: Nonsquare transform vector quantization, IEEE Signal Process. Lett. **3**(1), 1–3 (1996)

16.48 W.B. Kleijn, J. Haagen: Waveform interpolation for coding and synthesis. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, Amsterdam 1995), Chap. 5

16.49 W.B. Kleijn: Encoding speech using prototype waveforms, IEEE Trans. Speech Audio Process. **1**(4), 386–399 (1993)

16.50 W.B. Kleijn, J. Haagen: Transformation and decomposition of the speech signal for coding, IEEE Signal Process. Lett. **1**, 136–138 (1994)

16.51 T. Eriksson, W.B. Kleijn: On waveform-interpolation coding with asymptotically perfect reconstruction, Proc. IEEE Workshop on Speech Coding (1999) pp. 93–95

16.52 N.R. Chong, I.S. Burnett, J.F. Chicharo: A new waveform interpolation coding scheme based on pitch synchronous wavelet transform decomposition, IEEE Trans. Speech Audio Process. **8**(3), 345–348 (2000)

16.53 A. Gersho, R.M. Gray: *Vector Quantization and Signal Compression* (Kluwer, Dordrecht 1992)

16.54 H. Dudley: Phonetic pattern recognition vocoder for narrow-band speech transmission, J. Acoust. Soc. Am. **30**, 733–739 (1958)

16.55 C.E. Shannon: A mathematical theory of communication, Bell Syst. Tech. J. **27**, 379–423,623–656 (1948)

16.56 J. Picone, G. Doddington: A phonetic vocoder, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 580–583

16.57 J. Makhoul: Linear prediction: A tutorial review, IEEE Proc. **63**, 561–579 (1975)

16.58 F. Itakura: Line spectrum representation of linear predictive coefficients of speech signals, J. Acoust. Soc. Am. **57**, S35(A) (1975)

16.59 K.K. Paliwal, B.S. Atal: Efficient vector quantization of LPC parameters at 24 bits/frame, IEEE Trans. Speech Audio Process. **1**(1), 3–14 (1993)

16.60 J.S. Collura, A. McCree, T.E. Tremain: Perceptually based distortion measures for spectrum quantization, Proc. IEEE Workshop on Speech Coding for Telecommunications (1995) pp. 49–50

16.61 W.P. LeBlanc, B. Bhattacharya, S.A. Mahmoud, V. Cuperman: Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding, IEEE Trans. Speech Audio Process. **1**(4), 373–385 (1993)

16.62 W. Gardner, B. Rao: Theoretical analysis of the high-rate vector quantization of LPC parameters, IEEE Trans. Speech Audio Process. **3**, 367–381 (1995)

16.63 A. McCree, J.C. DeMartin: A 1.6 kb/s MELP coder for wireless communications, Proc. IEEE Workshop on Speech Coding for Telecommunications (1997) pp. 23–24

16.64 T.E. Tremain: The government standard linear predictive coding algorithm: LPC-10, Speech Technol. **1**, 40–49 (1982)

16.65 T.E. Tremain, M.A. Kohler, T.G. Champion: Philosophy and goals of the DoD 2400 bps vocoder selection process, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 2 (1996) pp. 1137–1140

16.66 A. McCree, K. Truong, E.B. George, T.P. Barnwell III, V.R. Viswanathan: A 2.4 kbit/s MELP coder candidate for the new U.S. Federal Standard, Proc. IEEE Int. Conf. Acoust. Speech Signal Processing, Vol. 1 (1996) pp. 200–203

16.67 L.M. Supplee, R.P. Cohn, J.S. Collura, A. McCree: MELP: the new Federal Standard at 2400 bps, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 2 (1997) pp. 1591–1594

16.68 M.A. Kohler: A comparison of the new 2400 bps MELP federal standard with other standard coders,




Amazon EX1010 Page 374

Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (1997) pp. 1587–1590
16.69 J.P. Campbell Jr., T.E. Tremain, V.C. Welch: The DoD 4.8 kbps Standard (Proposed Federal Standard (1016)). In: *Advances in Speech Coding*, ed. by B.S. Atal, V. Cuperman, A. Gersho (Kluwer Academic, Boston 1991) pp. 121–133
16.70 S. Villette, K.T. Al Naimi, C. Sturt, A.M. Kondoz, H. Palaz: A 2.4/1.2 kbps SB-LPC based speech coder: the Turkish NATO STANAG candidate, Proc. IEEE Workshop on Speech Coding (2002) pp. 87–89
16.71 G. Guilmin, P. Gournay, F. Chartier: Description of the French NATO candidate, Proc. IEEE Workshop on Speech Coding (2002) pp. 84–86
16.72 T. Wang, K. Koishida, V. Cuperman, A. Gersho, J.S. Collura: A 1200/2400 bps coding suite based on MELP, Proc. IEEE Workshop on Speech Coding (2002) pp. 90–92
16.73 J.S. Collura, D.F. Brandt, D.J. Rahikka: The 1.2 kbps/2.4 kbps MELP speech coding suite with integrated noise pre-processing, IEEE Mil. Commun. Conf. Proc., Vol. 2 (1999) pp. 1449–1453
16.74 R. Martin, R.V. Cox: New speech enhancement techniques for low bit rate speech coding, Proc. IEEE Workshop on Speech Coding (1999) pp. 165–167
16.75 T. Wang, K. Koishida, V. Cuperman, A. Gersho, J.S. Collura: A 1200 bps speech coder based on MELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2000) pp. 1375–1378
16.76 G. Guilmin, F. Capman, B. Ravera, F. Chartier: New NATO STANAG narrow band voice coder at 600 bits/s, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2006) pp. 689–693
16.77 J.V. Evans: Satellite systems for personal communications, Proc. IEEE, Vol. 86 (1998) pp. 1325–1341
16.78 J.C. Hardwick, J.S. Lim: The application of the IMBE speech coder to mobile communications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (1991) pp. 249–252
16.79 S.F.C. Neto, F.L. Corcoran, J. Phipps, S. Dimolitsas: Performance assessment of 4.8 kbit/s AMBE coding under aeronautical environmental conditions, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (1996) pp. 499–502
16.80 I.A. Gerson, M.A. Jasiuk: VSELP). In: *Advances in Speech Coding*, ed. by B.S. Atal, V. Cuperman, A. Gersho (Kluwer, Norwell 1991) pp. 69–79
16.81 S. Dimolitsas, C. Ravishankar, G. Schroder: Current objectives in 4-kb/s wireline-quality speech coding standardization, IEEE Signal Process. Lett. **1**(11), 157–159 (1994)
16.82 E.L.T. Choy: *Waveform interpolation speech coder at 4 kb/s*, M.S. Thesis (McGill University, Montreal 1998)
16.83 O. Gottesman, A. Gersho: Enhanced waveform interpolative coding at low bit-rate, IEEE Trans. Speech Audio Process. **9**(8), 786–798 (2001)
16.84 J. Stachurski, A. McCree, V. Viswanathan: High quality MELP coding at bit-rates around 4 kb/s, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1999) pp. 485–488
16.85 S. Yeldener: A 4 kb/s toll quality harmonic excitation linear predictive speech coder, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (1999) pp. 481–484
16.86 W.B. Kleijn, R.P. Ramachandran, P. Kroon: Generalized analysis-by-synthesis coding and its application to pitch rediction, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (1992) pp. 337–340
16.87 J. Thyssen, Y. Gao, A. Benyassine, E. Shlomot, C. Murgia, H. Su, K. Mano, Y. Hiwasaki, H. Ehara, K. Yasunaga, C. Lamblin, B. Kovesi, J. Stegmann, H. Kang: A candidate for the ITU-T 4 kbit/s speech coding standard, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 2 (2001) pp. 681–684
16.88 I.M. Trancoso, L. Almeida, J.M. Tribolet: A study on the relationships between stochastic and harmonic coding, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1986) pp. 1709–1712
16.89 E. Shlomot, V. Cuperman, A. Gersho: Combined harmonic and waveform coding of speech at low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1998) pp. 585–588
16.90 A. McCree, J. Stachurski, T. Unno, E. Ertan, E. Paksoy, V. Viswanathan, A. Heikkinen, A. Ramo, S. Himanen, P. Blocher, O. Dressler: A 4 kb/s hybrid MELP/CELP speech coding candidate for ITU standardization, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 1 (2002) pp. 629–632



Amazon EX1010 Page 375

# 17. Analysis-by-Synthesis Speech Coding

J.-H. Chen, J. Thyssen

Since the early 1980s, advances in speech coding technologies have enabled speech coders to achieve bit-rate reductions of a factor of 4 to 8 while maintaining roughly the same high speech quality. One of the most important driving forces behind this feat is the so-called *analysis-by-synthesis* paradigm for coding the excitation signal of predictive speech coders. In this chapter, we give an overview of many variations of the analysis-by-synthesis excitation coding paradigm as exemplified by various speech coding standards around the world. We describe the variations of the same basic theme in the context of different coder structures where these techniques are employed. We also attempt to show the relationship between them in the form of a *family tree*. The goal of this chapter is to give the readers a big-picture understanding of the dominant types of analysis-by-synthesis excitation coding techniques for predictive speech coding.

17.1 **Overview** ..... 352

17.2 **Basic Concepts of Analysis-by-Synthesis Coding** ..... 353
17.2.1 Definition of Analysis-by-Synthesis 353
17.2.2 From Conventional Predictive Waveform Coding to a Speech Synthesis Model ..... 353
17.2.3 Basic Principle of Analysis by Synthesis ..... 354
17.2.4 Generic Analysis-by-Synthesis Encoder Structure ..... 355
17.2.5 Reasons for the Coding Efficiency of Analysis by Synthesis ..... 357

17.3 **Overview of Prominent Analysis-by-Synthesis Speech Coders** ..... 357

17.4 **Multipulse Linear Predictive Coding (MPLPC)** ..... 360

17.5 **Regular-Pulse Excitation with Long-Term Prediction (RPE-LTP)** ..... 362

17.6 **The Original Code Excited Linear Prediction (CELP) Coder** ..... 363

17.7 **US Federal Standard FS1016 CELP** ..... 367

17.8 **Vector Sum Excited Linear Prediction (VSELP)** ..... 368

17.9 **Low-Delay CELP (LD-CELP)** ..... 370

17.10 **Pitch Synchronous Innovation CELP (PSI-CELP)** ..... 371

17.11 **Algebraic CELP (ACELP)** ..... 371
17.11.1 ACELP Background ..... 372
17.11.2 ACELP Efficient Search Methods ..... 373
17.11.3 ACELP in Standards ..... 377

17.12 **Conjugate Structure CELP (CS-CELP) and CS-ACELP** ..... 377

17.13 **Relaxed CELP (RCELP) – Generalized Analysis by Synthesis** ..... 378
17.13.1 Generalized Analysis by Synthesis Applied to the Pitch Parameters ..... 378
17.13.2 RCELP in Standards ..... 381

17.14 **eX-CELP** ..... 381
17.14.1 eX-CELP in Standards ..... 382

17.15 **iLBC** ..... 382

17.16 **TSNFC** ..... 383
17.16.1 Excitation VQ in TSNFC ..... 385
17.16.2 TSNFC in Standards ..... 386

17.17 **Embedded CELP** ..... 386

17.18 **Summary of Analysis-by-Synthesis Speech Coders** ..... 388

17.19 **Conclusion** ..... 390

**References** ..... 390



Amazon EX1010 Page 376

## 17.1 Overview

Historically, speech coding technology has been dominated by coders based on linear prediction. To achieve good speech quality, most speech coding standards are *waveform-approximating coders* (or *waveform coders* for short), and the majority use linear prediction to exploit the redundancy in the speech waveform.

Of course, the techniques in speech coding standards are not the only ones available, as there are many other speech coding techniques proposed in the literature that have not been used in standards. However, speech coding standards represent the dominant and most widely deployed speech coding techniques. Often they also include the best-performing techniques for the given requirements of bit rate, coding delay, and codec complexity at the time they were standardized. Therefore, it is useful to examine some of the techniques in speech coding standards to see how the dominant speech coding techniques have evolved over the years. For a comprehensive review of speech coding standards up to about 1995, see [17.1].

The first speech coding standard based on predictive waveform coding is probably 32 kb/s adaptive differential pulse code modulation (ADPCM), which was initially standardized by the Comité Consultatif International Téléphonique et Télégraphique (CCITT, the predecessor of the ITU-T) as recommendation G.721 in 1984 and later modified and restandardized as part of G.726 in 1986. The G.726 standard is a narrow-band (telephone-bandwidth) speech coder with an input sampling rate of 8 kHz and encoding bit rates of 16, 24, 32, and 40 kb/s, with 32 kb/s being the most widely used bit rate of G.726.

The basic idea of the G.726 ADPCM coder is to use an adaptive linear predictor to predict the input speech signal, and then use an adaptive scalar quantizer to quantize the difference signal between the input speech and the predicted version. Since statistically this difference signal tends to be smaller than the input speech signal, one can use a lower bit rate to quantize the difference signal to the same precision as direct quantization of the speech signal itself. This is how predictive coders achieve bit-rate reduction through the use of prediction to exploit the redundancy between nearby speech samples. The difference signal is often called the *prediction residual signal*, because it is the residual of the speech signal after the predictable portion is removed. In analysis-by-synthesis speech coders, this prediction residual signal is often referred to as the *excitation signal*. The reason will become clear during later discussions in this chapter.

In 1982, at about the same time that G.721 ADPCM was being developed, *Atal* and *Remde* proposed the first analysis-by-synthesis speech waveform coding technique called *multipulse linear predictive coding* (MPLPC) [17.2]. This work led to *Atal* and *Schroeder*'s 1984 proposal of another analysis-by-synthesis speech coding technique called *stochastic coding* [17.3], which was renamed *code-excited linear prediction* (CELP) [17.4] in 1985. From then on, CELP became the dominant speech coding technique for the next two decades, with hundreds or perhaps even thousands of technical papers published on variations of CELP techniques.

Due to their high coding efficiency, variations of CELP coding techniques together with other advancements have enabled speech waveform coders to halve the bit rate of 32 kb/s G.726 ADPCM three times while maintaining roughly the same speech quality. This is evidenced by the ITU-T's speech coding standardization effort after standardizing 32 kb/s ADPCM in 1984.

The first halving of bit rate to 16 kb/s happened with the ITU-T G.728 low-delay CELP (LD-CELP) coder [17.5, 6]. At the cost of increasing the buffering delay from one to five samples (0.625 ms at 8 kHz sampling rate), the G.728 coder halved the bit rate of 32 kb/s G.726 ADPCM while maintaining equivalent or better speech quality for all conditions tested.

The second halving of bit rate to 8 kb/s happened with the ITU-T G.729 conjugate-structure algebraic CELP (CS-ACELP) [17.7]. At the cost of increasing the buffering delay further to 80 samples of frame size plus 40 samples of look-ahead (or 15 ms total), the G.729 coder halved the bit rate again to 8 kb/s while maintaining the speech quality for most test conditions except tandeming and speech in background noise.

The third halving of bit rate to 4 kb/s happened with candidate coders [17.8, 9] submitted for the ITU-T's 4 kb/s speech coding standardization. A promising candidate 4 kb/s coder [17.8] was based on CELP. Although the ITU-T eventually did not standardize a 4 kb/s coder, this CELP-based candidate coder did achieve equivalent speech quality as 32 kb/s G.726 ADPCM at least for clean speech conditions at the cost of further increasing the buffering delay to 30–35 ms.

Of course, the G.728, G.729, and ITU-T 4 kb/s candidate coders are merely three example coders that help to make a point that the analysis-by-synthesis



Amazon EX1010 Page 377

speech coding technique was the main driving force behind the feat of reducing the bit rate by a factor of 4 to 8 within two decades while maintaining equivalent speech quality, at least for clean speech. In the last two decades, researchers have proposed numerous other speech waveform coders in the bit-rate range of 4–16 kb/s that achieve speech quality equivalent to or better than the speech quality of the 32 kb/s G.726 ADPCM coder. The vast majority of these are analysis-by-synthesis speech coders based on either CELP or MPLPC. Therefore, it is fair to say that *analysis-by-synthesis* is undeniably the most important speech coding method in modern low-bit-rate speech waveform coding.

The main emphasis of this chapter is to describe many of the major flavors of analysis-by-synthesis excitation coding for linear predictive speech waveform coders, as exemplified by various speech coding standards around the world. Due to the space limitation, it is not possible to cover all speech coding standards or even all aspects of the standards discussed in this chapter. Readers interested in learning more about other aspects of analysis-by-synthesis speech coders are referred to [17.10].

The rest of this chapter is organized as follows. Section 17.2 describes the basic concepts of the analysis-by-synthesis paradigm in the context of speech coding. Section 17.3 gives an overview of the different flavors of analysis-by-synthesis excitation coding and shows the relationship between them in the form of a *family tree*. Sections 17.4–17.17 then provide somewhat more-detailed descriptions of these different flavors of analysis-by-synthesis speech coders one-by-one, including MPLPC, original CELP, regular-pulse excitation (RPE), federal standard 1016 (FS1016) CELP, vector sum excited linear prediction (VSELP), LD-CELP, pitch synchronous innovation CELP (PSI-CELP), ACELP, CS-ACELP, relaxed CELP (RCELP), extended CELP (eX-CELP), forward backward linear predictive coding (FB-LPC), two-stage noise feedback coding (TSNFC), and embedded CELP, roughly in a chronological order. Section 17.18 summarizes these analysis-by-synthesis speech coders in a table format. Finally, Sect. 17.19 concludes this chapter.



## 17.2 Basic Concepts of Analysis-by-Synthesis Coding

### 17.2.1 Definition of Analysis-by-Synthesis

What exactly is *analysis-by-synthesis*? The phrase "*analysis-by-synthesis*" can be traced back to at least 1959 in a paper by *Halle* and *Stevens* [17.11]. In another 1961 paper by *Bell* et al. [17.12], a detailed definition of analysis-by-synthesis is given as follows:

> *The term analysis-by-synthesis is used to refer to an active analysis process that can be applied to signals that are produced by a generator whose properties are known. The heart of an analysis-by-synthesis system is a signal generator capable of synthesizing all and only the signals to be analyzed. The signals synthesized by the generator are compared with the signals to be analyzed, and a measure of error is computed. Different signals are generated until one is found that causes the error to reach some smallest value, at which time the analyzer indicates the properties of the internally generated signal.*

Although this definition was given four and a half decades ago in a paper discussing the subject of speech analysis, it is still equally valid today as a description of the basic concepts behind analysis-by-synthesis speech waveform coding.

### 17.2.2 From Conventional Predictive Waveform Coding to a Speech Synthesis Model

To understand analysis-by-synthesis speech waveform coding, it is useful to look first at conventional predictive speech waveform coding. In conventional predictive speech coders based on linear prediction, the prediction is typically performed using a weighted sum of previously quantized speech samples rather than unquantized input speech samples. This is because the quantized speech samples are available at the decoder while the input speech samples are not. Performing linear prediction based on the quantized speech in the encoder enables the decoder to produce the same predicted speech and track the encoder states in the absence of transmission errors. Prediction based on previously quantized speech signal is called *closed-loop* prediction, while prediction based on unquantized input speech signal is called *open-loop* prediction. For in-depth discussion of predictive waveform coding principles, see [17.13].

The encoder and decoder of the most basic form of closed-loop predictive waveform coding is shown in Fig. 17.1. In the encoder in Fig. 17.1a, the linear predictor, represented by the transfer function $P(z)$,

Amazon EX1010 Page 378

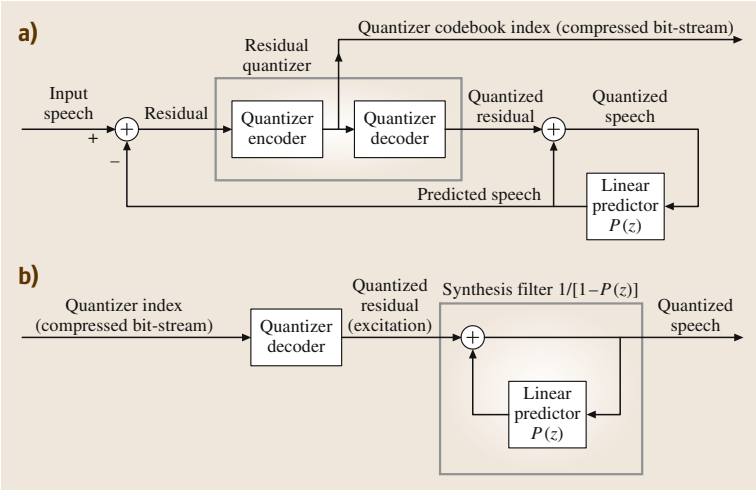



**Fig. 17.1a,b** Basic structure of conventional linear predictive speech waveform coder. **(a)** Speech encoder, and **(b)** speech decoder

uses previously quantized speech signal as its input to produce the predicted speech signal, which is subtracted from the input speech signal to get the prediction residual signal. After the prediction residual signal is quantized by the quantizer, the same predicted speech signal is added back to the quantized prediction residual signal to get the quantized speech signal. The sequence of quantizer codebook indices corresponding to the sequence of selected quantizer output levels is transmitted to the decoder as the compressed bitstream.

Note that the decoder structure in Fig. 17.1b is basically just a replica of the right half of the encoder structure. Therefore, if the compressed bitstream is received without transmission errors, then the decoder can decode the same quantized speech signal as the quantized speech signal in the encoder. Note that the feedback loop in the decoder that contains the linear predictor can be regarded as a *synthesis filter* with a transfer function of $1/[1-P(z)]$. This synthesis filter corresponds to an autoregressive model. It should also be noted that the quantized prediction residual at the output of the quantizer decoder can be regarded as the *excitation source* for this synthesis filter. From this perspective, the decoder of a conventional linear predictive speech coder can be viewed as a *source-filter model* [17.14] for speech production or speech synthesis. Therefore, the task of the corresponding encoder is to determine the quantized excitation signal and the parameters of the synthesis filter, either on a sample-by-sample or frame-by-frame basis. In other words, the task of the encoder is simply to identify the model parameters of this source-filter model as represented by the decoder structure.

### 17.2.3 Basic Principle of Analysis by Synthesis

In conventional predictive speech waveform coders, the identification of such model parameters at the encoder is not performed using the analysis-by-synthesis method as defined above by *Bell* et al. in their 1961 paper [17.12]. The encoder in Fig. 17.1a identifies the excitation signal by first calculating the closed-loop prediction residual signal and then directly quantizing this residual signal sample by sample to get the quantized residual (or excitation) signal. The encoder does not perform a synthesis operation when attempting to analyze one of the model parameters, namely, the excitation signal.

In an analysis-by-synthesis speech waveform coder, on the other hand, to achieve better coding efficiency, the prediction residual signal, or excitation signal, is usually quantized block by block rather than sample by sample, where each block is typically 0.5–10 ms long (4–80 samples at 8 kHz sampling). Each block of samples is often called a *vector*. Let the dimension of the excitation vector be $K$ samples and let the excitation encoding bitrate be $r$ bits per sample, then each excitation vector will be represented by $Kr$ bits. Therefore, each vector of the excitation signal can only be quantized into one of $2^{Kr}$ possible candidate excitation vectors.

An analysis-by-synthesis speech waveform coder does not directly quantize the prediction residual signal as in Fig. 17.1a. Instead, each of the finite $2^{Kr}$ possible candidates for the excitation signal is passed through the synthesis filter and the resulting synthesized speech signal is compared with the input speech signal. The candidate excitation signal that minimizes a predeter-

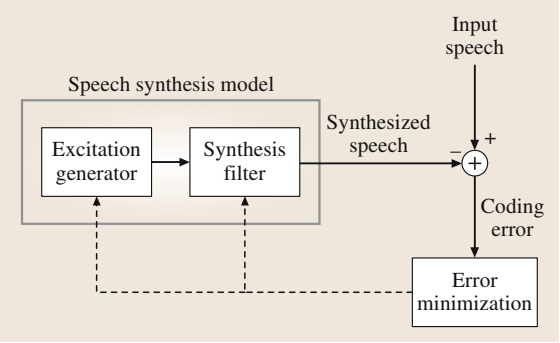



**Fig. 17.2** Simplified analysis-by-synthesis speech waveform coder

mined distortion measure between the input speech and the synthesized speech is selected by the encoder as the final excitation signal to be transmitted to the decoder and to be used in the decoder to *excite* the synthesis filter.

The greatly simplified block diagram in Fig. 17.2 illustrates this analysis-by-synthesis coding principle. The gray rectangular box in Fig. 17.2 represents the source-filter speech synthesis model, consisting of an excitation generator followed by a synthesis filter. Note that this speech synthesis model is in direct correspondence to the decoder structure shown in Fig. 17.1b. The difference now is that rather than calculating the prediction residual first and then quantizing it as in Fig. 17.1a, the analysis-by-synthesis coder in Fig. 17.2 passes each of the possible candidate excitation signals through the synthesis filter and identifies the candidate excitation signal that minimizes the error between the corresponding synthesized speech and the input speech. Note that what is shown in Fig. 17.2 is only the encoder structure. The corresponding decoder is simply the excitation generator and synthesis filter enclosed by the gray rectangle.

The encoder in Fig. 17.2 attempts to perform *analysis* of the input speech signal in order to identify the model parameters of the speech synthesis model. The name *analysis-by-synthesis* comes from the fact that the encoder does not try to calculate unquantized model parameters and then quantize them, but instead perform the *analysis* to get the model parameters *by* the *synthesis* of candidate output speech signals using all possible model parameters and identify the set of model parameters that minimizes the error between the synthesized speech and the input speech.

The identification of the best candidate excitation signal that minimizes the coding error is indicated by the dashed arrow going from the *error minimization* block to the *excitation generator* block. This analysis-by-synthesis method of identifying the best model parameters is often called *closed-loop* optimization or quantization, which should not be confused with closed-loop prediction. Closed-loop prediction means the predictor uses previously quantized signal to perform prediction, whereas closed-loop quantization in the current context means the quantization is performed in a way that minimizes the distortion in the speech domain rather than the domain of the parameter to be quantized. In contrast, open-loop quantization means deriving or extracting a parameter and then directly quantize that parameter by minimizing the quantization distortion in the parameter domain.

Strictly speaking, a true analysis-by-synthesis coder will synthesize all possible candidate output speech signals using all combinations of the candidate excitation signals and the candidate synthesis filters and then identify the particular combination that gives the minimum coding error. However, in practice this is rarely done due to the impractically high search complexity that is required. The common approach is to derive the synthesis filter coefficients directly from the input speech signal and perform analysis-by-synthesis coding only for the excitation signal. This approach reduces the performance very little.

### 17.2.4 Generic Analysis-by-Synthesis Encoder Structure

Figure 17.3 shows a more detailed and more generic encoder structure of an analysis-by-synthesis speech waveform coder. This figure covers most of the analysis-by-synthesis speech coders, at least in the conceptual level. The actual encoder structures used in efficient implementations of such coders may differ, but they are

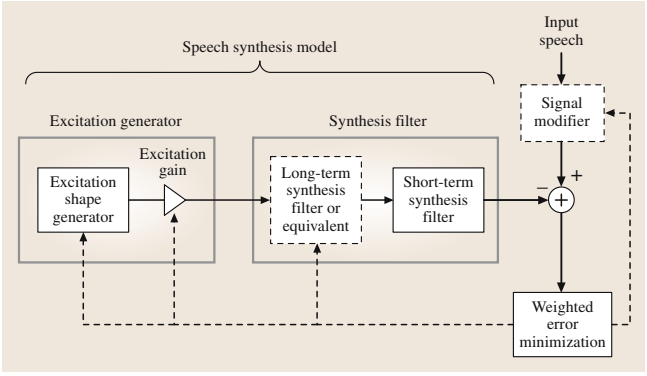



**Fig. 17.3** A more detailed generic analysis-by-synthesis speech waveform coder



Amazon EX1010 Page 380

usually mathematically equivalent to the encoder structure shown in Fig. 17.3. Compared with Fig. 17.2, this figure has

1. added an optional signal modifier for the input speech signal
2. used weighted error minimization
3. expanded the excitation generator and the synthesis filter into more blocks

Each of these differences is briefly explained below.

The addition of the signal modifier is optional; that is why the border of this block is in dashed lines. The purpose of this signal modifier is to modify the input signal in such a way that does not degrade the speech quality appreciably and yet makes the resulting modified speech signal easier for the analysis-by-synthesis coder to encode (i. e., requiring a lower bit rate to encode the signal to the same perceived speech quality) [17.15]. This technique is used to achieve better coding efficiency in some of the CELP coders, such as RCELP [17.15]. However, most analysis-by-synthesis speech coders do not use it because it adds complexity and may occasionally cause slightly audible degradation to speech quality.

The weighted error minimization block in Fig. 17.3 is typically used to shape the coding noise spectrum so that it follows the spectrum of the input signal to some extent – a process usually referred to as *noise spectral shaping* [17.16]. Due to the noise masking effect of the human auditory system, such spectrally shaped coding noise is less audible to human ears.

The excitation generator in Fig. 17.2 is expanded into the excitation shape generator and the excitation gain scaling unit in Fig. 17.3. The excitation shape generator generates excitation vectors with all kinds of possible vector shapes while having gains (as measured by a vector *norm*) either equal to a single value or lie within a narrow range around a single value. The excitation gain scaling unit then scales the excitation shape vectors so that the scaled excitation vector can have a wide dynamic range in terms of vector norm. The separation into generator and scaling blocks is not theoretically necessary, since it is possible for a single excitation generator block to generate excitation signals with all possible shapes and gains. Instead, such separation is motivated by the desire to keep the computational complexity low, since without it a very large excitation codebook is required.

The synthesis filter in Fig. 17.2 is expanded into the cascade of the long-term synthesis filter (or equivalent) and the short-term synthesis filter in Fig. 17.3. A long-term synthesis filter often contains a long-term predictor [17.16] in a feedback loop, and a short-term synthesis filter often contains a short-term predictor in a feedback loop [17.14]. Again, this separation is not theoretically necessary. However, there is an important reason to such decoupling – the long-term synthesis filter (or its equivalent) can model well the quasi-periodicity in voiced speech signals introduced by the periodic vibration of the human vocal cord, while the short-term synthesis filter can model the spectral envelope of speech signals as controlled by the human vocal tract. Thus, such decoupling is motivated by the way the speech signal is produced by a human talker.

Note that the long-term synthesis filter block in Fig. 17.3 is enclosed by dashed lines, signifying that this filter may or may not be used. Most analysis-by-synthesis speech coders do employ this long-term synthesis filter (or its equivalent) as well as the short-term synthesis filter. The G.728 LD-CELP coder [17.6] and the original MPLPC coder [17.2] are two exceptions.

Conceptually, on a block-by-block basis, the encoder in Fig. 17.3 attempts to find the best combination of the excitation shape vector, the excitation gain, the long-term synthesis filter parameters, the short-term synthesis filter parameters, and the signal modifier parameters (if the modifier is present), such that the resulting synthesized speech signal is closest to the modified input speech signal, where *closest* is in the sense of minimizing a weighted mean squared error criterion (distortion measure). Rather than exhaustively searching through all combinations of these parameters, practical analysis-by-synthesis speech coders usually use sequential optimization. In other words, these five sets of parameter values (or four if the signal modifier is not used) are usually optimized one set at a time in a sequential manner so that the computational complexity is manageable. In some speech coders, certain long-term synthesis filter parameters are jointly optimized with at least some of the excitation parameters in an attempt to achieve better coding efficiency. For example, see [17.17].

Essentially all practical analysis-by-synthesis speech coders quantize the short-term synthesis filter parameters in an *open-loop* manner. In other words, the unquantized short-term synthesis filter parameters are first derived from the input speech signal, and then they are directly quantized using a distortion measure in the filter parameter domain rather than using the weighted error in the speech domain as depicted in Fig. 17.3. That is why there is not a dashed arrow from the weight error minimization block to the short-term synthesis filter block in Fig. 17.3.



Amazon EX1010 Page 381

Exactly how many parameter sets should be closed-loop or jointly quantized depends on the coder design goals and requirements. In some analysis-by-synthesis speech coders that emphasize low computational complexity, only the excitation shape is closed-loop quantized and all other parameter sets are open-loop quantized. As long as the encoding bit rate is not overly low and the speech coder is carefully designed, even such an analysis-by-synthesis coder can achieve fairly high speech quality.

### 17.2.5 Reasons for the Coding Efficiency of Analysis by Synthesis

As mentioned earlier, the analysis-by-synthesis excitation coding is one of the most efficient speech coding techniques and is the main driving force behind the 4 to 8 times reduction in speech coder bit-rate in the last two decades. A fundamental question to ask is: why is this analysis-by-synthesis technique for predictive speech coders so powerful? One reason is that the speech synthesis model in Fig. 17.3 is a good fit to how speech signal is produced by a human talker.

Perhaps a more important reason is the analysis-by-synthesis method itself. Think of it this way: the task of the encoder is to find the quantized parameters of this speech synthesis model so that the synthesized output speech sounds closest to the input speech. Given this, then which other coding method is better than trying all possible quantized parameters and see which one gives an output speech signal closest to the input speech signal? This *trying all possible values and finding which one gives the best output speech* is the essence of analysis-by-synthesis coding. It is a stark contrast to earlier coding methods where the unquantized optimal parameter value is first derived and then directly quantized using a distortion measure in that parameter domain.

Yet a third reason is that such an analysis-by-synthesis approach naturally lends itself to enable the use of the so-called *vector quantization* (VQ) , which has a higher coding efficiency than conventional sample-by-sample *scalar quantization* as used in ADPCM. Some might argue that ADPCM and adaptive predictive coding (APC) [17.16] are analysis-by-synthesis coders since minimizing the quantization error in the prediction residual domain is mathematically equivalent to minimizing the coding error in the speech domain. While there is indeed such a mathematical equivalence in the degenerate case of scalar quantization, it is questionable, in our opinion, whether ADPCM and APC should be called analysis-by-synthesis coders. Our main objection is that these scalar-quantization-based coders never really perform the *synthesis* operation when trying to do the *analysis* of the model parameters (i. e., when performing residual quantization).

Conventional ADPCM and APC coders are restricted to using sample-by-sample scalar quantization and cannot be extended to vector quantization to reap the benefit of the higher coding efficiency of VQ. (If these coders indeed use VQ, then they will not be called ADPCM and APC anymore and will be called CELP.) These coders need to compute the unquantized prediction residual signal first and then quantize it, but with VQ the unquantized prediction residual signal depends on the quantized prediction residual signal due to the feedback filter structure in Fig. 17.1a. This creates a chicken-and-egg problem. In contrast, the analysis-by-synthesis approach completely avoid this chicken-and-egg problem by directly trying all possible quantized residual (excitation) signals without the need to compute the unquantized residual signal first. Thus, by enabling the use of VQ for the excitation signal, the analysis-by-synthesis approach reaps the benefit of the higher coding efficiency of VQ.

This chapter only explains the most fundamental basic ideas of analysis-by-synthesis coding. If the encoder structure in Fig. 17.3 were implemented as is, the resulting complexity would be quite high. In later sections, computationally much more efficient but mathematically equivalent encoder structures will be introduced. Most analysis-by-synthesis speech coders today are implemented based on the more efficient encoder structures.

## 17.3 Overview of Prominent Analysis-by-Synthesis Speech Coders

This section gives an overview of some prominent analysis-by-synthesis speech coders and shows the relationship between them in the form of a *family tree*. There are numerous analysis-by-synthesis speech coders proposed in the literature. It is not practical to describe all of them in this chapter. Instead, the intention here is to describe only those analysis-by-synthesis speech coders that are either standard coders or are the first of its kind and thus are definitive and seminal.



Amazon EX1010 Page 382

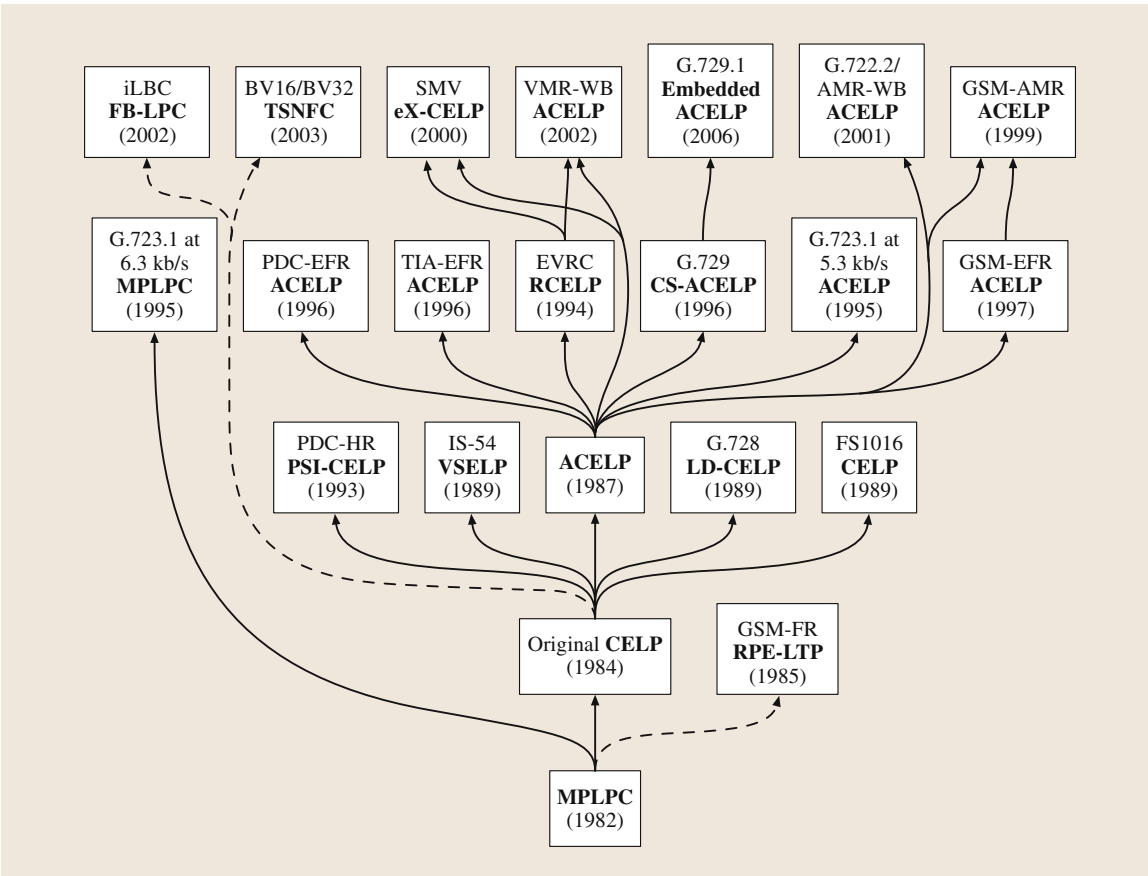



**Fig. 17.4** Family tree of prominent analysis-by-synthesis speech waveform coders

Figure 17.4 shows the family tree of these selected analysis-by-synthesis speech coders. Each rectangular block in the tree represents a particular speech coder or a type of speech coders. Within each rectangular block, the first row (sometimes the first two rows) specifies the name of the coding standard, the second row in boldface letters gives the name of the coding technique, and the third row provides the year of the first publication the authors can find for that coder (either as a technical paper or as a standard specification).

The speech coders in Fig. 17.4 are organized in five different rows, with each row roughly corresponding to a different generation of coders. Each generation represents a group of analysis-by-synthesis speech coders developed within a time period of a few years.

The MPLPC coder [17.2] proposed in 1982 is the first modern analysis-by-synthesis speech waveform coder. The original CELP coder [17.3] and the RPE-LTP coder [17.18] in the second row represent the second-generation analysis-by-synthesis coders developed in the mid 1980s. The five coders in the third row represent the third-generation analysis-by-synthesis coders developed in the late 1980s and early 1990s. The seven coders in the fourth row represent the fourth-generation analysis-by-synthesis coders developed in the mid 1990s. Finally, the seven coders in the fifth row represent the fifth-generation analysis-by-synthesis coders developed from the late 1990s to the present day. Strictly speaking, the distinctions between the fourth- and fifth-generation coders are not always clear; however, at least the fifth-generation coders were developed later than the fourth-generation coders.

A brief overview of this family tree of analysis-by-synthesis speech coders is given below. The MPLPC coder proposed in 1982 is at the root of the family tree, since it can be argued that all other coders in Fig. 17.4 can trace their roots back to this coder.



Amazon EX1010 Page 383

For the second-generation coders, the original CELP coder proposed in 1984 is a direct descendant of MPLPC with long-term prediction [17.19], as *Atal* worked on both coders and the two coders are similar except the multipulse model is replaced by VQ of the excitation in CELP. The excitation VQ codebook of this CELP coder is populated by Gaussian random numbers. The Groupe Spèciale Mobile (GSM) full-rate standard coder regular-pulse excitation with long-term prediction (RPE-LTP) [17.18] is inspired by the MPLPC coder; however, to reduce the computational complexity, it uses an excitation model of regularly spaced pulses which deviates substantially from the original multipulse model. That is why the arrow from MPLPC to RPE-LTP has a dashed line.

The third-generation coders saw the most variety of analysis-by-synthesis excitation coding techniques. The first standard coder based on CELP is the US federal standard FS1016 CELP coder at 4.8 kb/s [17.20]. It uses overlapping center-clipped Gaussian VQ codevectors for excitation coding. The second CELP coder that became a standard coder is the Telecommunications Industry Association (TIA) IS-54 standard vector sum excited linear prediction (VSELP) coder at 8 kb/s [17.17]. It uses sums of several basis vectors as the excitation VQ codevectors. A 6.7 kb/s VSELP coder later also became the Japanese personal digital cellular (PDC) full-rate standard coder, and a 5.6 kb/s VSELP coder became the GSM half-rate standard coder [17.21]. The third CELP coder that became a standard is the ITU-T G.728 standard low-delay CELP (LD-CELP) coder at 16 kb/s [17.6]. It uses a closed-loop trained-gain-shape VQ codebook structure with a small vector dimension and with joint optimization of VQ gain and shape. The fourth CELP coder that became a standard is the Japanese PDC half-rate standard pitch synchronous innovation CELP (PSI-CELP) coder at 3.45 kb/s [17.22]. This coder uses a fairly long vector dimension and repeats the first portion of the VQ codevectors in a pitch-synchronous manner if the pitch period is less than the vector dimension.

An influential class of third-generation coder is the algebraic CELP, or ACELP [17.23]. This coder reduces the VQ codebook search complexity by using an algebraic VQ codebook, with the elements of VQ codevectors having only $+1$, 0, and $-1$ as the possible sample values. As can be seen from Fig. 17.4, most of the fourth- and fifth-generation coders are derivatives of ACELP. However, it should be noted that ACELP-based speech coding standards use sparse algebraic codes [17.24] and not the original algebraic code proposed in [17.23]. The sparseness contributes to a low computational complexity and the reduction of granular type of coding noise.

All but one fourth-generation standard coders in Fig. 17.4 can trace their roots to ACELP. The single exception is the 6.3 kb/s version of the ITU-T G.723.1 standard [17.25]. It is an improved version of the first-generation MPLPC coder. One of the fourth-generation coders that deserves attention is the so-called relaxed CELP (RCELP) coder used in North American cellular standard enhanced variable-rate coder (EVRC) [17.26]. It is based on a generalization of analysis-by-synthesis [17.15] that adaptively modifies the input speech signal in such a way that the modified signal sounds essentially the same as the original input signal and yet it becomes easier for a CELP coder to encode the signal efficiently. This RCELP technique is also used in the North American cellular standard selectable mode vocoder (SMV) [17.27] and variable-rate multimode wide-band (VMR-WB) coder [17.28].

For fifth-generation standard coders in Fig. 17.4, all but two are based on ACELP. The exceptions are the BV16/BV32 coder [17.29], [17.30] and the internet low bit rate (iLBC) coder [17.31], [17.32]. The iLBC coder is an Internet Engineering Task Force (IETF) experimental standard and also a PacketCable standard. It is based on what is called forward backward linear predictive coding (FB-LPC), where a maximum-energy segment of the short-term prediction residual is independently coded, and then a dynamic codebook is constructed first forward in time and then backward in time. The 16 kb/s BroadVoice16 (BV16) coder [17.33] and its wide-band version 32 kb/s BroadVoice32 (BV32) coder use similar coding algorithms. BV16 is a PacketCable standard, SCTE (Society of Cable Telecommunications Engineers) standard, and American National Standards Institute (ANSI) standard for voice-over-internet protocol (VoIP) applications in cable telephony, and BV32 is a PacketCable 2.0 standard. The BV16/32 coder is based on two-stage noise feedback coding (TSNFC). This coder is not a CELP coder since its encoder structure is totally different from that of CELP; however, it employs many of the complexity reduction and even codebook design techniques of CELP within the TSNFC encoder structure. For this reason, the arrow from the original CELP to TSNFC has a dashed line, signifying that it is not a direct descendant of CELP but leverages nearly two decades of research on CELP.

As discussed above, recent speech coding standards are dominated by ACELP-related coders. For this

reason, ACELP and its derivatives will receive more-thorough treatment in this chapter. Nevertheless, there are still many other interesting kinds of analysis-by-synthesis coders in this family tree. In the sections that follow, the excitation coding methods for the more distinctive types of coders in this family tree will be described in more detail. Due to the similarity between many different ACELP coders in Fig. 17.4, many of them are lumped together and discussed in the same ACELP section.

## 17.4 Multipulse Linear Predictive Coding (MPLPC)

The MPLPC coder [17.2] was the first predictive waveform coder to abandon the sample-by-sample residual quantization procedure and to encode the entire block of excitation signal as a single entity in an analysis-by-synthesis manner. It achieves this by using an excitation signal with mostly zero samples and finding the optimal locations and amplitudes for a small number of nonzero pulses such that the resulting synthesized speech signal minimizes a weighted distortion measure in the speech domain.

Refer to the generic analysis-by-synthesis speech waveform coder shown in Fig. 17.3. The MPLPC coder does not use the signal modifier in Fig. 17.3. The original MPLPC coder proposed in [17.2] also does not use the long-term synthesis filter. In fact, this original MPLPC coder was developed to improve the linear predictive coding (LPC) vocoder [17.34], which does not use a long-term synthesis filter and have a rigid excitation model of either periodic pulse train or white noise.

The MPLPC coder does use the short-term synthesis filter in Fig. 17.3. This short-term synthesis filter is in the form of an all-pole filter given by the following transfer function

$$H_s(z) = \frac{1}{A(z)} , \tag{17.1}$$

where

$$A(z) = 1 - P(z) = 1 - \sum_{i=1}^{M} a_i z^{-i} \tag{17.2}$$

is the short-term prediction error filter, $P(z)$ is the short-term predictor, $a_i, i = 1, 2, \ldots, M$, are the predictor coefficients, and $M$ is the order of the short-term synthesis filter. The filter order $M$ is typically somewhere between 10 and 16 for 8 kHz sampled signals. The predictor coefficients are adaptive and are usually transmitted once every 10–20 ms.

In this first MPLPC coder the weighted error minimization in Fig. 17.3 is achieved through the use of a so-called perceptual weighting filter in the form of

$$W(z) = \frac{A(z)}{A(z/\gamma)} , \quad 0 < \gamma < 1 , \tag{17.3}$$

where

$$A\left(\frac{z}{\gamma}\right) = 1 - P\left(\frac{z}{\gamma}\right) = 1 - \sum_{i=1}^{M} a_i \gamma^i z^{-i} \tag{17.4}$$

is a modified version of $A(z)$ with the roots of the polynomial moved radially toward the origin (the radii of the roots are scaled by a factor of $\gamma$, which is typically 0.8). Since the roots are normally within the unit circle for a stable synthesis filter $H_s(z) = 1/A(z)$, the net effect is that the magnitude frequency response of the filter $1/A(z/\gamma)$ is a smoothed version of that of the filter $1/A(z)$.

For MPLPC, the weighted error minimization in Fig. 17.3 is achieved by passing the difference between the input speech signal and the synthesized speech signal through the perceptual weighting filter $W(z)$ and then identifying which combination of multipulse model parameters gives the minimum mean-squared error of this perceptually weighted difference signal. The perceptual weighting filter emphasizes the spectral valley regions and deemphasize the spectral peak regions of the input speech spectrum. This has an effect of allowing more coding noise under the spectral peaks and suppressing coding noise in the spectral valleys of speech. Thus, the perceptual weighting filter shapes the coding noise spectrum so that it follows the input speech spectrum to some extent. Due to the noise masking effect of the human auditory system, such a spectrally shaped coding noise is perceptually less audible than coding noise with a flat spectrum.

The short-term synthesis filter $H_s(z) = 1/A(z)$ has a magnitude frequency response corresponding to the spectral envelope of the input signal. Thus, the filter $H_s(z/\gamma) = 1/A(z/\gamma)$ has a magnitude response corresponding to a smoothed version of the spectral envelope of input speech. Subtracting one magnitude response



Amazon EX1010 Page 385

from another in the logarithmic domain is equivalent to dividing in the linear domain. Hence, the filter

$$\frac{1}{W(z)} = \frac{A(z/\gamma)}{A(z)} = \frac{H_s(z)}{H_s(z/\gamma)} \tag{17.5}$$

will have a magnitude response somewhat similar to the spectral envelope of the input speech but with a reduced spectral slope. Since this filter is the inverse of the perceptual weighting filter $W(z)$, given how the weighted error minimization works as described above, the magnitude response of this filter $1/W(z)$ is the spectral envelope of the coding noise that the MPLPC coder will achieve. For an example of what the magnitude responses of $H_s(z)$ and $W(z)$ look like, see [17.2].

The first MPLPC coder in [17.2] encoded the excitation signal block-by-block in an analysis-by-synthesis manner using a block size of 5 ms, or 40 samples at 8 kHz sampling. Most of the 40 samples are zeroes. Only about 10% of the 40 excitation samples (four samples) have nonzero amplitudes. These nonzero samples are called *pulses* because they look like pulses in the graphical representation of such a digital excitation signal. Since there are usually multiple of such nonzero samples in the block of excitation signal to be encoded, the resulting excitation model is called the *multipulse* model, and the resulting linear predictive coding scheme is called *multipulse linear predictive coding* (MPLPC).

Let $m$ be the number of pulses in each block of excitation signal. Each pulse is completely specified by its amplitude and its location within the block. Therefore, the multipulse excitation model has $2m$ model parameters that need to be determined. Jointly optimizing these $2m$ model parameters would cost too high computational complexity. A suboptimal sequential optimization approach is proposed in [17.2].

In the sequential optimization approach, the $m$ pulses are determined one at a time. For a given pulse location, the corresponding optimal amplitude can be obtained through a closed-form solution [17.2]. Before determining the location and amplitude of the first pulse, with the excitation signal set to zero, the output of the short-term synthesis filter during the current block (due to the nonzero filter memory left over at the end of the last block) is subtracted from the input speech. The result is the target signal for the search of the first pulse. Each of the 40 possible pulse locations is tried. With closed-loop optimal amplitude solved for each pulse location, the pulse is passed through the short-term synthesis filter. The filter output signal is subtracted from the target signal and the resulting difference signal is passed through the perceptual weighting filter. The mean-squared error (MSE) of the weighted signal is calculated. This process is repeated until the best pulse location and amplitude that gives the lowest weighted error is identified. Next, the short-term synthesis filter output signal due to this first pulse is subtracted from the target signal to form a new target signal for the search of the second pulse. This process is repeated until the pulse locations of all $m$ pulses are determined. Then, given the $m$ pulse locations, the $m$ pulse amplitudes can be jointly optimized in a single step [17.2].

Using this approach, it is reported in [17.2] that only minimal audio quality improvement are achieved after about eight pulses have been placed in a 10 ms interval. It is also reported [17.2] that the resulting output speech quality sounded natural and perceptually close to the input speech, without the common unnatural and buzzy characteristics of the LPC vocoder output speech.

Strictly speaking, since each pulse in the multipulse model is scaled by a different amplitude, the decomposition of the excitation generator into the excitation shape generator and the excitation gain in Fig. 17.3 is not a good description. This problem can be avoided if the excitation gain in Fig. 17.3 is understood to have the possibility of taking the form of a gain vector (with dimension $m$), where each element of the gain vector is individually used to scale one of the $m$ pulses.

Further improvements to this initial MPLPC coder was proposed in [17.19]. The most notable is the addition of a pitch predictor, or equivalently, the addition of the long-term synthesis filter in Fig. 17.3. Due to the quasiperiodicity in voiced speech such as vowels, it is found that the multipulse excitation signal shows significant correlation from one pitch period to the next. This correlation can be exploited by using a pitch predictor (also called long-term predictor). Let the pitch period be $T$ samples, and let $\beta$ be the long-term predictor coefficient. Then, with

$$H_l(z) = \frac{1}{1-\beta z^{-T}} \tag{17.6}$$

chosen as the transfer function of the long-term synthesis filter in Fig. 17.3, each pulse at the output of the long-term synthesis filter will be scaled by $\beta$ and then be added to the excitation signal $T$ samples later. This has the effect of creating a periodic pulse pattern at a period of $T$ samples. After adding this long-term synthesis filter, fewer pulses are needed to produce the same level of speech quality.

Another improvement proposed in [17.19] is related to pulse amplitude optimization. Rather than waiting until all pulse locations are determined and then jointly

optimize all the pulse amplitudes, the proposed procedure performs reoptimization of amplitudes along the way. In other words, during the sequential search for pulse locations, after the $j$-th pulse location is determined ($1 \le j \le m$), all pulse amplitudes from the first pulse to the $j$-th pulse are reoptimized jointly before continuing to search for the $(j+1)$-th pulse location. It is reported [17.19] that this approach improved coder output signal-to-noise ratio (SNR) by 2–10 dB, with an average of slightly over 3 dB.

There are many other improvements and variations proposed in the literature for the MPLPC coder. Due to the space limitation, they have to be omitted in the discussion here.

The multipulse excitation model is used in the ITU-T Recommendation G.723.1 speech coder. G.723.1 has two bit-rates. The lower-bit-rate 5.3 kb/s version is based on ACELP, while the higher-bit-rate 6.3 kb/s version is based on MPLPC with long-term prediction.

This 6.3 kb/s version of G.723.1 uses a frame size of 30 ms, which is equally divided into four subframes of 7.5 ms (60 samples) each. Multipulse excitation search is performed on each of the four subframes. The multipulse excitation model uses six pulses for even subframes and five pulses for odd subframes. The pulse location is restricted to be either all even or all odd, as indicated by a grid bit. Furthermore, all pulses are constrained to have the same magnitude, although different pulses can have different signs. The shared magnitude is first estimated and quantized. Then, four quantized magnitude values around the estimated magnitude are allowed in the analysis-by-synthesis multipulse search. For each of these four possible quantized magnitude values, the pulse locations and signs are sequentially optimized one pulse at a time. This procedure is repeated for both the even and odd pulse position grids. Finally, the best combination of all these parameters (quantized magnitude, pulse locations and signs, and even or odd grid) that gives the minimum weighted error of the synthesized speech is selected as the final multipulse excitation parameters to be transmitted to the G.723.1 decoder.

As can be seen, this G.723.1 multipulse coder has deviated substantially from the original MPLPC proposed in [17.2] and [17.19]. In fact, with all the constraints put on the G.723.1 multipulse model (due to the low-bit-rate limitation), the G.723.1 multipulse excitation signal starts to resemble to some extent the excitation signal of ACELP coders. However, such constraints aside, the spirit of the original multipulse excitation model can still be seen in this 6.3 kb/s G.723.1 multipulse coder.



## 17.5 Regular-Pulse Excitation with Long-Term Prediction (RPE-LTP)

The regular-pulse excitation (RPE) coder [17.35] can be regarded as a special low-complexity realization of the fundamental analysis-by-synthesis concept proposed in the original multipulse LPC coder [17.2]. A special version of it, the regular-pulse excitation with long-term prediction (RPE-LTP) coder at 13 kb/s [17.18], was selected as the first GSM standard coder for European digital cellular service.

The RPE excitation model consists of $J$ possible sequences of regularly spaced pulses, each with the pulses located at a different phase. The typical value of $J$ is 3 or 4. As an example, for the $k$-th sequence of pulses, the nonzero excitation samples (the pulses) are located at the positions of $k, k+J, k+2J, k+3J, \ldots$, and the sample values at other locations are zero. The task of excitation coding is to find the amplitude values for each of the pulses in each of the $J$ sequences of regularly spaced pulses, and then find the sequence of pulses that minimizes a weighted error measure.

Similar to the original MPLPC coder proposed in [17.2], the initial RPE coder [17.35] also did not use a long-term predictor. On a conceptual level, the encoder structure of the initial RPE coder is equivalent to the structure in Fig. 17.3 without the long-term synthesis filter and the signal modifier. However, in actual implementation, the LPC inverse filter $A(z)$, which is the numerator portion of the perceptual weighting filter $W(z) = A(z)/A(z/\gamma)$, is moved beyond the adder to the left and above. The one to the left of the adder cancels out the short-term synthesis filter, while the one above the adder stays there and filters the input speech to produce the short-term prediction residual. The remaining denominator portion of the weighting filter, or $1/A(z/\gamma)$, stays below the adder for weighted error minimization.

In [17.35], the procedure for the RPE analysis-by-synthesis excitation search is given. It basically amounts to solving $J$ sets of linear equations to find the optimal amplitude for each pulse of the $J$ possible regular pulse sequences. The weighted distortion measure for each of the $J$ pulse sequences are calculated and the sequence that minimizes the weighted distortion is selected as the final excitation sequence.

Amazon EX1010 Page 387

In the GSM full-rate (GSM-FR) standard RPE-LTP coder, a single-tap long-term predictor (LTP) is added to the initial RPE coder, so GSM-FR corresponds to the coder structure in Fig. 17.3 with the long-term synthesis filter enabled. The GSM-FR coder uses three possible sequences of regularly space pulses, that is, $J = 3$. It has a frame size of 20 ms and a subframe size of 5 ms. Therefore, for every 5 ms subframe of 40 samples, there are 3 possible sequences of regularly spaced pulses. Each sequence contains only 13 or 14 nonzero pulses, with the remaining samples having zero amplitudes. The selected sequence is identified by 2 bits, and the corresponding pulse amplitudes are encoded with a 3-bit block-adaptive PCM quantizer. The block maximum for each subframe is encoded with 6 bits. The pitch period and the pitch tap are derived once a subframe and quantized to 7 bits and 2 bits, respectively.

This GSM-FR RPE-LTP coder is the speech coder for the first-generation GSM digital cellular telephones. It became the first analysis-by-synthesis coder that was standardized and widely deployed across many countries.

## 17.6 The Original Code Excited Linear Prediction (CELP) Coder

By using an analysis-by-synthesis procedure, the multipulse excitation model in the MPLPC coder [17.2] opened the door for significantly more efficient excitation coding than was possible with earlier speech waveform coders. However, even with this multipulse model, there is still a limit on how low the excitation encoding bit-rate can go. For example, with the same 40-sample excitation block size and four pulses as in the original MPLPC, suppose it is desirable to encode the excitation signal at a bit rate of 1/4 bits/sample, this gives a mere 10 bits to represent the four pulse locations and four pulse amplitudes. It is virtually impossible to use 10 bits to quantize such eight parameters with sufficient accuracy. This problem led *Atal* and *Schroeder* to use a 10 bit vector quantization (VQ) codebook to quantize the excitation signal [17.3], leading to what is known today as code-excited linear prediction, or CELP [17.4].

To use 10-bit VQ to quantize 40 samples of the excitation signal, Atal and Schroeder constructed a VQ codebook containing $2^{10} = 1024$ codevectors, each of which is a 40-dimensional vector containing 40 white Gaussian random numbers with unit variance. Beside conceding that it is generally difficult to design an optimum deterministic codebook (with a total of 40 960 samples in it), in [17.4] *Schroeder* and *Atal* actually gave a justification for using the Gaussian random numbers. The justification is that the probability density function of the prediction residual signal after both short- and long-term prediction is nearly Gaussian.

The encoder structure of this initial CELP coder fits the structure shown in Fig. 17.3 exactly. Again, the signal modifier in Fig. 17.3 is not used in this CELP coder. The excitation shape generator is basically a table look-up of the 10-bit, 40-dimensional excitation VQ codebook described earlier. The excitation gain scales the unit-variance Gaussian codevectors to the proper gain level that matches the root mean square (RMS) value of the prediction residual signal after both short-term and long-term prediction. It is updated once every 5 ms together with the excitation codevector.

The synthesis filter of CELP contains both the long-term synthesis filter and the short-term synthesis filter. The short-term synthesis filter has the same form as given in (17.1), while the long-term synthesis filter uses a 3-tap pitch predictor rather than the single-tap pitch predictor given in (17.6). The weighted error minimization of CELP is also achieved with a perceptual weighting filter in exactly the same way as in MPLPC, described earlier in Sect. 17.4. The perceptual weighting filter $W(z)$ also has exactly the same form as defined in (17.3).

*Atal* and *Schroeder* reported [17.3, 4] that when the 40-sample excitation vector is quantized to 0.25 bit/sample in an analysis-by-synthesis manner using a 10-bit Gaussian VQ codebook, with all other speech synthesis model parameters (excitation gain and parameters for the long- and short-term synthesis filters) left at their open-loop-derived optimal values, the resulting synthesized speech sounded very close to the original input speech. Only small differences were noticeable even in close pairwise comparisons over headphones.

Achieving such a high level of output speech quality with such a low bit rate of merely 0.25 bit/sample for the excitation signal was considered a stunning breakthrough in 1984–1985. Almost immediately, a large number of speech coding researchers jumped in and engaged in the research of the CELP coding technique. In the subsequent years that followed, hundreds of CELP-related technical papers have been published and numerous advancements in CELP coding have been



Amazon EX1010 Page 388

made. As shown in the family tree of Fig. 17.4, this analysis-by-synthesis CELP coding idea eventually led to more than a dozen of modern low-bit-rate speech coding standards, and CELP-related coders dominate almost all speech coding standards established since the late 1980s.

The two initial CELP papers [17.3, 4] were more *proof of concept* papers than actual coder design papers. Not only were all model parameters except the excitation signal were left unquantized, but a major issue was the very high computational complexity required by the analysis-by-synthesis excitation VQ codebook search.

Actually, it is fairly easy to estimate the complexity of such a codebook search. Refer to Fig. 17.3. Given that the short-term filter order they used was $M = 16$, each of the 1024 excitation codevectors has to be scaled by the excitation gain (one multiply per sample), filtered by the long-term synthesis filter (three multiply-adds per sample), filtered by the short-term synthesis filter (16 multiply–adds per sample), subtracted from the input speech (one subtract per sample), filtered by the perceptual weighting filter ($2 \times 16 = 32$ multiply–adds per sample), squared (one multiply per sample), and then added (one add per sample) to get the weighted distortion value of that codevector. Therefore, at a sampling rate of 8000 Hz, the total complexity for searching through all 1024 excitation codevectors to identify the one that minimizes the weighted distortion measure would take at least $8000 \times 1024 \times (1+3+16+1+32+1+1) = 450\,560\,000$ operations per second, or 450.6 MFLOPS (million floating-point operations per second). Even the fastest supercomputer at that time could not perform computations that fast. No wonder *Atal* and *Schroeder* reported [17.3, 4] that their initial CELP simulation took 125 s of central processor unit (CPU) time on the then-supercomputer Cray-1 to process just one second of speech.

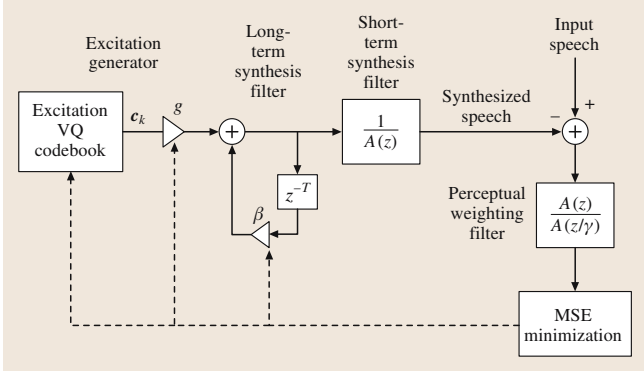



**Fig. 17.5** Encoder structure of the original CELP coder

However, even *Atal* and *Schroeder* pointed out [17.4] at that time that

> *a code book with sufficient structure amenable to fast search algorithms could lead to real-time implementation of code-excited coders.*

This turned out to be true. In subsequent years many specially structured excitation VQ codebooks that allowed fast codebook search were proposed – the excitation codebooks of ACELP, VSELP, and FS1016 CELP are all good examples.

However, even without such specially structured excitation codebooks, by just re-arranging the encoder structure in Fig. 17.3 and performing the CELP excitation codebook search in a mathematically equivalent way, the computational complexity can be reduced by almost an order of magnitude. This efficient encoder structure and excitation codebook search procedure is explained below. It forms the basis of essentially all practical CELP-based coders today.

Consider the encoder structure shown in Fig. 17.5. This structure is basically the same as the generic analysis-by-synthesis coder structure shown in Fig. 17.3, except that specific filter structure and transfer functions are given. For convenience of later discussion of the so-called *adaptive codebook*, the long-term synthesis filter is reverted from a three-tap filter back to a single-tap filter as used in the improved MPLPC [17.19]. The short-term synthesis filter is $1/A(z)$. The weighted error minimization is explicitly separated into a perceptual weighting filter as defined in (17.3) followed by MSE minimization. Even with the three-tap pitch filter replaced by a single-tap one, the complexity is still $8000 \times 1024 \times (1+1+16+1+32+1+1) = 434.2$ MFLOPS.

As suggested in [17.19], the perceptual weighting filter can be moved before the adder in Fig. 17.5 so that the input speech and the synthesized speech are each individually weighted before the difference of the two weighted signals are calculated. The cascade of the short-term synthesis filter and the perceptual weighting filter gives a weighted short-term synthesis filter in the form of

$$H(z) = \frac{W(z)}{A(z)} = \frac{1}{A(z)} \frac{A(z)}{A(z/\gamma)} = \frac{1}{A(z/\gamma)} \,. \qquad (17.7)$$

This is shown in Fig. 17.6. Even this step alone can cut the computational complexity by almost a factor of three. There is more that can be saved. However, before



Amazon EX1010 Page 389

continuing, let us briefly describe another new feature in Fig. 17.6 – the concept of the adaptive codebook as introduced in [17.36].

The adaptive codebook in Fig. 17.6 is a different way to achieve a similar effect as the long-term synthesis filter in Fig. 17.5. If the pitch period $T$ is not smaller than the excitation vector dimension, then once the effect of the filter memory is taken out, the long-term synthesis filter in Fig. 17.5 does not affect the excitation codebook search at all due to the bulk delay $z^{-T}$. However, if the pitch period $T$ is smaller than the vector dimension, then different excitation VQ codevectors will cause different contributions from the long-term synthesis filter to be added to the excitation to the short-term synthesis filter. This makes it more difficult to perform certain efficient codebook search methods.

To facilitate an efficient codebook search, the effect of the long-term synthesis filter in Fig. 17.5 is modeled by the adaptive codebook in Fig. 17.6. If the pitch period is not smaller than the excitation vector dimension, then the adaptive codebook simply contains different vector sections of the long delay line in the long-term synthesis filter defined by the range of the pitch period. In other words, the codevectors in the adaptive codebook is obtained by using a sliding rectangular window to extract different sections of the long delay line in the long-term synthesis filter, with different codevectors corresponding to different target pitch periods. In this case, the adaptive codebook, the adaptive codebook gain $\beta$, and the related adder in Fig. 17.6 is mathematically equivalent to the long-term synthesis filter in Fig. 17.5.

On the other hand, if the pitch period is smaller than the vector dimension, then at least a portion of the codevectors in the adaptive codebook will be beyond the long delay line of the long-term synthesis filter and will correspond to the samples in the current excitation vector that have not been determined yet. Rather than having incomplete codevectors, the adaptive codebook will periodically repeat the samples in the long delay line at the target pitch period.

Thus, the concept of adaptive codebook allows the use of large excitation vector dimension which tends to improve the coding efficiency. In addition, it allows the long-term synthesis filter to be modeled as just another stage of excitation VQ, thus simplifying the codebook search procedure.

The codevectors in an adaptive codebook is changing with time. That is why the codebook is called the *adaptive* codebook. In contrast, the original excitation VQ codebook in Fig. 17.5 does not change with time. Therefore, as shown in Fig. 17.6, it is often called the *fixed* codebook in a CELP coder where an adaptive codebook is used.

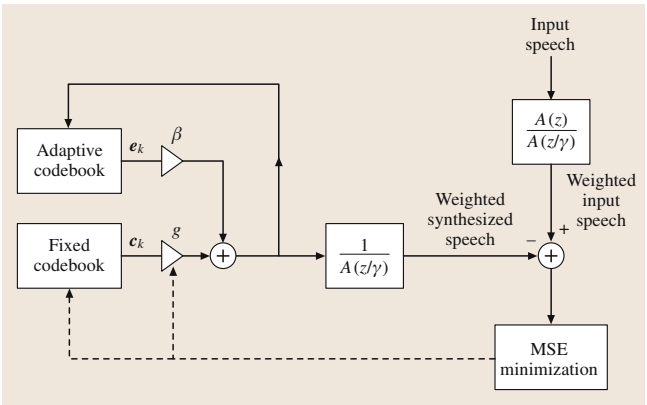



**Fig. 17.6** Intermediate encoder structure of the efficient CELP coder

The encoder structure in Fig. 17.6 can be changed to an even more efficient but mathematically equivalent structure as shown in Fig. 17.7. According to the linear system theory, the output of the filter $H(z)$ is the sum of two components

1. the zero-state response (ZSR), which is the output of the filter due to the input signal, with the initial filter memory set to zero,
2. the zero-input response (ZIR), which is the output of the filter due to only the filter memory, with the input signal set to zero.

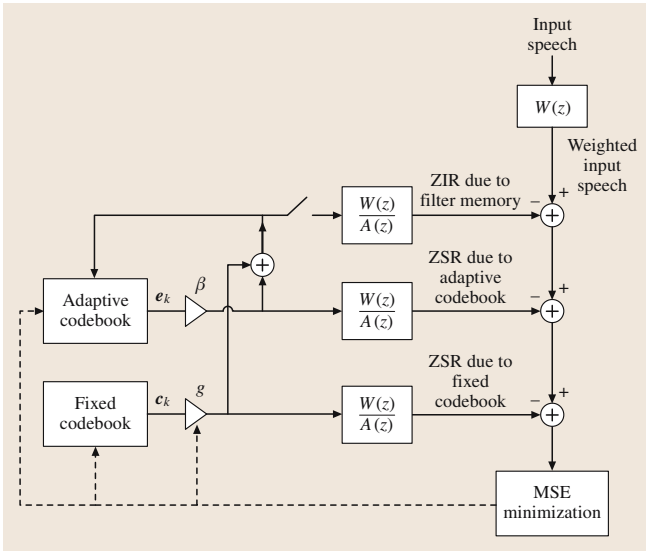



**Fig. 17.7** Encoder structure of the efficient CELP coder

Note that the ZIR does not depend on which excitation VQ codevector is used. Thus, by decomposing the output of the weighted short-term synthesis filter $H(z)$ into ZIR and ZSR, the ZIR component can first be subtracted out of the weighted speech signal, and the resulting signal becomes the *target signal* for the codebook search based on the ZSR component. In fact, this idea was already suggested by *Atal* and *Remde* in their original MPLPC paper [17.2], even though they did not use the terms of ZIR and ZSR.

In some later CELP coders, the perceptual weighting filter took a form different from what was specified in (17.3). Therefore, to keep Fig. 17.7 general, the weighted short-term synthesis filter is represented as $H(z) = W(z)/A(z)$.

In Fig. 17.6, the excitation signal to the short-term synthesis filter has two components: the scaled adaptive codebook vector $\beta \boldsymbol{e}_k$ and the scaled fixed codebook vector $g\boldsymbol{c}_k$. Therefore, the weighted synthesized speech signal in Fig. 17.6 can be decomposed into three components: the ZIR signal due to the memory of the filter $W(z)/A(z)$, the ZSR signal due to the adaptive codebook, and the ZSR signal due to the fixed codebook.

In the efficient CELP excitation codebook search method based on Fig. 17.7, the input speech vector is first passed through the perceptual weighting filter $W(z)$ to get the weighted input speech vector. Then, the ZIR vector due to the filter memory is calculated by setting the initial memory of the weighted short-term synthesis filter to the filter memory at the last sample of the last input speech vector and letting the filter *ring* without any input excitation signal (the *switch* in Fig. 17.7 is open). This ZIR vector is subtracted from the weighted input speech vector to get the target vector for the next stage. Next, the adaptive codebook index (the pitch period) and the adaptive codebook gain (the pitch gain) are usually determined in an analysis-by-synthesis manner to minimize the MSE between the target vector obtained above and the ZSR vector due to the adaptive codebook. The ZSR vector corresponding to the best combination of the pitch period and the pitch gain is then subtracted from the target vector to get the next target vector for the ZSR vector due to the fixed codebook. Finally, the fixed codebook index and the fixed codebook gain are determined in an analysis-by-synthesis manner to minimize the MSE between this next target vector and the ZSR vector due to the fixed codebook. After such a codebook search, the selected codebook vectors and gains are used to calculate the sum of the two excitation components $\beta \boldsymbol{e}_k + g\boldsymbol{c}_k$, and the resulting final excitation vector is used to update the adaptive codebook and the memory of the weighted short-term synthesis filter (the *switch* is closed). Then, this procedure is repeated for the next input speech vector.

The encoder structure in Fig. 17.7 looks much more complicated than the structure in Fig. 17.5, so why is it computationally more efficient? The key lies in the fact that by subtracting out the ZIR vector due to filter memory, the search through the adaptive codebook and the fixed codebook can be performed without the actual filtering operation. Let $h(n), n = 0, 1, \ldots, N-1$ be the truncated impulse response of the weighted short-term synthesis filter $H(z) = W(z)/A(z)$ with $N$ being the dimension (length) of the excitation vector. Then, with the initial filter states (memory) set to zero in the lower two branches in Fig. 17.7 containing the filter $W(z)/A(z)$, the filtering operation in these two branches can be replaced by convolution, which can be represented by a matrix–vector multiplication operation [17.37, 38].

Take the fixed codebook search as an example. Let the target vector for the fixed codebook search be $\boldsymbol{t} = [t(0), t(1), \cdots, t(N-1)]^{\mathrm{T}}$, let the $k$-th fixed codebook vector be $\boldsymbol{c}_k = [c_k(0), c_k(1), \cdots, c_k(N-1)]^{\mathrm{T}}$, and let

$$\boldsymbol{H} = \begin{pmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h(N-2) & h(N-3) & \cdots & 0 \\ h(N-1) & h(N-2) & \cdots & h(0) \end{pmatrix} \tag{17.8}$$

be the lower triangular Toeplitz matrix populated by the impulse response of the weighted short-term synthesis filter $H(z) = W(z)/A(z)$. Then, the ZSR vector due to the $k$-th fixed codebook vector can be represented as $g\boldsymbol{H}\boldsymbol{c}_k$, and the codebook search find the codebook index $k$ that minimizes the MSE distortion measure

$$E_k = \|\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k\|^2 , \tag{17.9}$$

where $\|\cdot\|^2$ indicates the Euclidean norm, or the sum of squares of the vector components. This distortion measure can be expanded as

$$E_k = \|\boldsymbol{t}\|^2 - 2g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k + g^2\|\boldsymbol{H}\boldsymbol{c}_k\|^2 . \tag{17.10}$$

Since the energy of target vector $\|\boldsymbol{t}\|^2$ is independent of the codebook index $k$, this term can be ignored when minimizing the distortion measure above. Thus, calculating the distortion measure amounts to calculating the energy of the ZSR vector $g\boldsymbol{H}\boldsymbol{c}_k$ and the correlation between this ZSR vector and the target vec-



Amazon EX1010 Page 391

tor $\boldsymbol{t}$. The adaptive codebook search can be formulated in the same way using the same kind of distortion measure.

Many efficient codebook search procedures have been proposed based on the distortion measure in (17.10). In addition, many special fixed codebook structures that allow efficient codebook search have been proposed in the literature. Together these techniques enabled the computational complexity to be reduced from the 430+ MFLOPS of the original CELP to 10 MFLOPS and below for many of the practical CELP coders deployed today.

## 17.7 US Federal Standard FS1016 CELP

The US Federal Standard FS1016 coder [17.20] is the first CELP coder ever standardized. It was developed by the US Department of Defense and is quite similar to the stochastically excited linear prediction (SELP) coder proposed in [17.36]. This coder uses a frame size of 30 ms. Each frame is divided into four subframes of 7.5 ms each (60 samples at 8 kHz sampling). The adaptive codebook approach described in [17.36] is used to determine the pitch period (also called the *pitch lag*) and the pitch gain once every subframe. (In this case, there is one excitation vector in each subframe.) For odd-numbered subframes, the pitch lag takes an integer value between 20 and 147 and thus can be encoded into 7 bits. For even-numbered subframes, the pitch lag is constrained to be within 32 samples relative to the pitch lag of the previous subframe. The pitch gain is coded using a five-bit nonuniform scalar quantizer. For the fixed codebook, the FS1016 standard uses a specially structured codebook where adjacent codevectors are essentially shifted version of each other and differ by only two samples at the end. This kind of shifted fixed codebook was first proposed in [17.39]. However, it was also independently studied in [17.36] using an improved distortion measure, and the optimal tradeoff of the two-sample shift was evaluated and proposed in [17.36].

This shifted fixed codebook introduces a certain constraint and structure into the codebook and allows an efficient codebook search. This is because when adjacent fixed codebook vectors are shifted version of each other, a large part of the ZSR computations in the matrix-vector multiplication $\boldsymbol{Hc}_k$ can be reused when going from one codevector to the next. Therefore, once the ZSR is calculated for the first codevector, the ZSR of the remaining codevectors can be obtained recursively with low complexity. This same principle also applies to the adaptive codebook search for those pitch lags that are not smaller than the excitation vector dimension (subframe size), since the corresponding adjacent adaptive codevectors are simply 1-sample shifted versions of each other.

To illustrate how a shifted codebook can reduce the codebook search complexity, first consider a fixed codebook with a one-sample shift between adjacent codevectors. In [17.39], a 1024-sample circular buffer $v(n), n = 0, 1, \ldots, 1023$ is used to store the 1024 codevectors of a 10-bit fixed codebook with 1-sample shift. The $j$-th codevector is defined to be $c_k(n) = v(k+n), 0 \le n \le N-1, 0 \le k \le 1023$, where $v(k+1024) = v(k)$ since $v(n)$ is in a circular buffer. Let $\boldsymbol{Hc}_k = \boldsymbol{r}_k = [r_k(0), r_k(1), \ldots, r_k(N-1)]^{\mathrm{T}}$. Then, due to the lower triangular Toeplitz structure of the $\boldsymbol{H}$ matrix, it is shown in [17.39] that $r_{k+1}(n)$ can be computed recursively from $r_k(n)$ as follows.

$$r_{k+1}(n-1) = r_k(n) - v(k)h(n)\,, \quad 1 \le n \le N-1\,,$$

$$r_{k+1}(N-1) = \sum_{n=0}^{N-1} v(k+1+n)h(N-1-n)\,.$$

This recursion is best understood with a graphical interpretation of the matrix-vector multiplication $\boldsymbol{Hc}_k$. In such a multiplication, the column vector $\boldsymbol{c}_k$ is turned sideways by 90° counterclockwise, then each of its elements is sample-by-sample multiplied with each element of the corresponding column of the matrix $\boldsymbol{H}$. Next, all the product terms along each row of the matrix is summed. The resulting column vector is $\boldsymbol{r}_k$. In the recursion above proposed in [17.39], when going from $\boldsymbol{c}_k$ to $\boldsymbol{c}_{k+1}$, the top element of $\boldsymbol{c}_k$ is removed, the remaining $N-1$ element is shifted up by one sample, and a new element is added at the bottom. Due to the lower triangular Toeplitz structure of the matrix $\boldsymbol{H}$, most of the product terms and the partial sums needed for $\boldsymbol{r}_{k+1}$ are already calculated for $\boldsymbol{r}_k$ and thus can be reused. However, the contribution to $r_k(n)$ due to the removed top element of $\boldsymbol{c}_k$ is already added to the element of $\boldsymbol{r}_k$. That is why in the first equation of the recursion above, the product term contribution due to the top element of $\boldsymbol{r}_k$ needs to be subtracted out again. None of the product terms in the second equation of the recursion above involving $r_{k+1}(N-1)$ has been calculated when calculating $\boldsymbol{r}_k$, so the entire summation in this second



Amazon EX1010 Page 392

equation needs to be calculated. This recursion above costs $(N-1)+N=2N-1$ multiply–add operations for each new $\boldsymbol{r}_{k+1}$ vector.

In fact, there is an even more-efficient recursion that requires only $N-1$ multiply–add operations for each new $\boldsymbol{r}_k$ vector. The trick is to reverse the order of the $\boldsymbol{r}_k$ vectors in the recursion. Using the graphical interpretation of the matrix-vector multiplication $\boldsymbol{H}\boldsymbol{c}_k$ explained above, one can see that if the recursion now starts at $\boldsymbol{r}_{1023}$ and going backward toward $\boldsymbol{r}_0$, then each time when going from $\boldsymbol{c}_{k+1}$ to $\boldsymbol{c}_k$, the bottom (last) element of $\boldsymbol{c}_{k+1}$ is removed, the remaining elements are shifted down by one sample, and a new element is added at the top. However, for the first $N-1$ element of $\boldsymbol{c}_{k+1}$ that are being shifted down, their partial sum contributions to $\boldsymbol{r}_k$ are exactly the first $N-1$ elements of $\boldsymbol{r}_{k+1}$. After these first $N-1$ elements of $\boldsymbol{c}_{k+1}$ have been shifted down and a new element added at the top to get $\boldsymbol{c}_k$, their partial sum contributions to $\boldsymbol{r}_k$ correspond to the product terms in the second through the $N$-th columns of the matrix $\boldsymbol{H}$. Thus, only the product terms in the first column need to be added to the partial sums already calculated and stored in the first $N-1$ elements of $\boldsymbol{r}_{k+1}$. Thus, the recursion can be described as

$$r_k(n) = r_{k+1}(n-1) + v(k)h(n)\,, \quad 1 \le n \le N-1\,,$$
$$r_k(0) = v(k)h(0) = v(k)\,.$$

The last equality above holds because $h(0)=1$. Therefore, this recursion performed in the reverse order requires only $N-1$ multiply-add operations for each new $\boldsymbol{r}_k$ vector. As mentioned above, this recursion can be used for the adaptive codebook search for those pitch lags not smaller than the subframe size.

The effect of shifting more than one sample between adjacent codevectors in the fixed codebook has been studied and reported in [17.36]. As the amount of shift between adjacent fixed codebook vectors increases, the coder performance increases, but the codebook search complexity also increases. It was found in [17.36] that for a large codebook (with 128 or more codevectors), a two-sample shift gives the same performance as a fully independent codebook. For this reason, the FS1016 coder uses such a fixed codebook with a two-sample shift between adjacent codevectors.

The same basic idea in the more efficient recursion above can easily be extended to the fixed codebook with a two-sample shift. In this case, each time when going from $\boldsymbol{c}_{k+1}$ to $\boldsymbol{c}_k$, the bottom two elements of $\boldsymbol{c}_{k+1}$ are removed, the remaining elements are shifted down by two samples, and two new elements are added at the top. For the first $N-2$ element of $\boldsymbol{c}_{k+1}$ that are being shifted down, their partial sum contributions to $\boldsymbol{r}_k$ are exactly the first $N-2$ elements of $\boldsymbol{r}_{k+1}$. These partial sum contributions to $\boldsymbol{r}_k$ correspond to the product terms in the third through the $N$-th columns of the matrix $\boldsymbol{H}$. Thus, only the product terms in the first two columns need to be added to the partial sums already calculated and stored in the first $N-2$ elements of $\boldsymbol{r}_{k+1}$. Thus, this recursion takes only $(N-1)+(N-2)=2N-3$ multiply–add operations to calculate each new $\boldsymbol{r}_k$.

In the case of the FS1016 coder, the elements of the fixed codebook are obtained by using a sequence of zero-mean, unit-variance, white Gaussian random numbers that are center-clipped at 1.2, which results in roughly 75% of the samples being zero. Thus, the fixed codebook is not only shifted, but also 75% *sparse*.

A sparse fixed codebook is said to produce slightly improved perceptual quality of the CELP output speech [17.36,38]. Furthermore, a sparse fixed codebook also reduces the codebook search complexity [17.38]. The reason is quite simple. Consider the graphical interpretation of the matrix–vector multiplication $\boldsymbol{H}\boldsymbol{c}_k$ again. When 75% to 90% of the elements in $\boldsymbol{c}_k$ are zero, the product terms in the corresponding columns of the matrix $\boldsymbol{H}$ do not need to be calculated, and thus the corresponding calculations can be saved. According to [17.20], the shifted sparse fixed codebook structure reduces the computation by a factor of 20.

In the FS1016 coder, the fixed codebook gain is quantized to 5 bits using a non-uniform scalar quantizer. An interoperable coder may use only a subset of the fixed codebook to reduce the computational complexity. The FS1016 coder also uses unequal forward error correction to protect perceptually most sensitive bits, and it uses extensive parameter smoothers to reduce the quality degrading effects of bit errors. At the time it was standardized, the output speech quality of the FS1016 coder was considered one of the best at around 4.8 kb/s.

## 17.8 Vector Sum Excited Linear Prediction (VSELP)

Vector sum excited linear prediction (VSELP) [17.17, 40] is another class of CELP coder that uses a specially structured fixed codebook to reduce the codebook search complexity. VSELP is the second type of CELP coder that was standardized.



Amazon EX1010 Page 393

Each codevector in the fixed codebook of VSELP is constructed as a weighted sum of $M$ independent basis vectors, with the weights taking only two possible values: $+1$ or $-1$. Thus, an $M$-bit fixed codebook uses exactly $M$ basis vectors. Let $\boldsymbol{v}_m = [v_m(0), v_m(1), \ldots, v_m(N-1)]^{\mathrm{T}}, m = 1, 2, \ldots, M$ be the $M$ basis vectors. Then, the $2^M$ codevectors in an $M$-bit VSELP fixed codebook is constructed as

$$\boldsymbol{c}_k = \sum_{m=1}^{M} \theta_{km} \boldsymbol{v}_m \tag{17.11}$$

for $k = 0, 1, \ldots, 2^M$, where $\theta_{km} = +1$ if bit $m$ of the $k$-th codeword is 1, and $\theta_{km} = -1$ if bit $m$ of the $k$-th codeword is 0.

An 8 kb/s VSELP coder [17.17] was selected as the TIA interim standard IS-54 for North American digital cellular telephone applications, although this IS-54 standard was later rescinded. A 6.7 kb/s VSELP coder was selected as the Japanese digital cellular PDC full-rate (PDC-FR) standard coder, and a 5.6 kb/s VSELP coder [17.21] also became the GSM half-rate (GSM-HR) standard coder.

In the IS-54 VSELP coder, there are two fixed codebooks sequentially searched, similar to a multistage VQ configuration. In the PDC-FR VSELP coder, only one fixed codebook is used. In the GSM-HR VSELP coder, there are two fixed codebooks, but how many fixed codebooks are used depends on the voicing mode. For example, for a voiced mode the adaptive codebook and one of the fixed codebooks is used, but for an unvoiced mode the adaptive codebook is not used and two fixed codebooks are used.

In the IS-54 VSELP coder, the adaptive codebook and the two fixed codebooks are treated like three successive stages of quantization. The codebook gains are left *floating* when searching for the codebook vectors of these three stages. The mathematics of the efficient codebook search is somewhat involved. Due to the space limitation, only the basic concepts will be described below. Interested readers are referred to [17.17] for more mathematical details.

The adaptive codebook is first searched using the method described in Sect. 17.7. Next, when searching each of the two fixed codebooks, each of the $M$ basis vectors are first filtered through the weighted short-term synthesis filter with zero initial memory to get the ZSR vector of that basis vector. Then, each of such ZSR vectors are orthogonalized with respect to each other and with respect to the ZSR vector due to the selected adaptive codebook vector. The ZSR vector due to each of the $2^M$ fixed codebook vectors can then be expressed as a linear combination of the $M$ orthogonalized ZSR vectors due to the $M$ basis vectors.

The search procedure needs to find the fixed codebook vector that minimizes a distortion measure that is the ratio of two quantities:

1. the square of the correlation between the target vector and the ZSR vector due to the fixed codebook vector
2. the energy of the ZSR vector due to the fixed codebook vector

These two quantities can each be evaluated in a recursive manner with low computational complexity if the codebook search procedure sequences through the codevectors using a binary Gray code so that the codewords of the adjacent codevectors differ by only one bit.

Another factor of two saving can be obtained by observing that the two fixed codevectors corresponding to the two codewords that are 1s complement of each other have the same shape but only differ in sign. Thus, the two quantities above for the distortion measure will be the same for these two complementary codevectors. The winner of the two complementary codevectors can be determined easily by just examining the sign of the correlation term in the first quantity. This means that only half as many distortion values need to be calculated.

Another novel feature of the IS-54 VSELP coder is the way it quantizes the gains of the adaptive codebook and the fixed codebooks. First the adaptive codebook vector and fixed codebook vectors are identified by the codebook search procedure outlined above. After that, rather than separate scalar quantization of each of the codebook gains as was the normal procedure in previous coders, this VSELP coder first encodes the speech energy of the entire 20 ms frame. Then, the approximate excitation signal energy in each of the 5 ms subframes is calculated based on the frame energy of speech and the reflection coefficients in each subframe. Next, the adaptive codebook gain and the two fixed codebook gains in each subframe are converted to the energy domain expressed as fractions of the total excitation energy in that subframe. The resulting three energy fractions are then jointly vector quantized using a trained codebook. At the VSELP decoder, the three decoded energy fractions are converted back to the codebook gain domain.

It is said [17.17] that such a gain VQ scheme in the energy-fraction domain not only makes the resulting energy fractions more correlated which allows more efficient VQ coding, but also the fact that all codebook gains are represented as energy fractions makes the en-



Amazon EX1010 Page 394

ergy of the decoded speech less susceptible to bit errors in the codebook gain codeword.

The GSM-HR VSELP coder also has another feature not yet discussed in this chapter – a long-term predictor with a fractional (non-integer) pitch period [17.41, 42]. Rather than using an integer pitch period as in previous predictive coders, an interpolation filter is used to achieve the effect of a non-integer pitch period. This allows the long-term synthesis filter to model the pitch periodicity of voiced speech with a higher temporal resolution, thus resulting in higher output speech quality.

In VSELP, the basis vectors for the fixed codebook can be optimized using a training database. It is reported [17.17] that such optimization of the basis vectors resulted in 0.64 dB improvement in weighted segmental SNR and significant improvement in subjective quality.

## 17.9 Low-Delay CELP (LD-CELP)

The low-delay CELP (LD-CELP) coder [17.5, 6, 43] is another different kind of CELP coder. It is the third kind of CELP coder that was selected as a standard coder. The ITU-T recommendation G.728 standard coder [17.6] is based on it. There are several important differences between G.728 and the previous CELP coders. These differences will be described in this section.

First of all, the excitation vector dimension used in G.728 LD-CELP is much shorter than that of earlier CELP coders. Most of the earlier CELP coders used an excitation vector dimension of 40 samples (5 ms) or more in order to get a better excitation VQ efficiency (since VQ performance generally increases with increasing vector dimension). However, using such a large vector dimension will not meet the ITU-T requirement of a very low coding delay. In order to achieve a one-way coding delay of less than 2 ms (16 samples at 8 kHz sampling), the G.728 coder is forced to use a frame size and vector dimension of merely five samples (0.625 ms).

With a frame size of five samples, basically the G.728 encoder is required to produce 10 bits as output for every five samples of input speech. This strict constraint, which is due to the low-delay requirement, greatly limits the flexibility of the coder design. While previous CELP coders could spend 20–40 bits per frame to encode the short-term predictor parameters and another 10 to 12 bits per subframe to encode the long-term predictor parameters, the G.728 coder cannot do any of these since it only has 10 bits total to encode everything in a frame.

This leads to the second major difference between G.728 and previous CELP coders – the predictor parameters are made *backward-adaptive*. Previous CELP coders all used *forward-adaptive* predictors, where the optimal predictor parameters are derived from the input speech and then quantized and transmitted to the decoder. With backward adaptation, no bits are spent in sending the parameters to the decoder; instead, the parameters are locally derived at both the encoder and the decoder from previously decoded speech signal or parameters. See [17.13] for a more-detailed discussion of backward adaptation versus forward adaptation. For G.728, both the short-term synthesis filter coefficients and the excitation gain are backward-adaptive and thus does not require any bit to transmit them to the decoder.

The third major difference between G.728 and previous CELP coders is the elimination of the long-term predictor, and in its place, the use of a high-order backward-adaptive short-term predictor. The reason is that a backward-adaptive long-term predictor has a strong tendency to diverge at the decoder due to bit errors or other reasons that cause a mismatch of the encoder states and the decoder states. On the other hand, when the backward-adaptive short-term predictor coefficients are obtained by performing LPC analysis on previously decoded speech signal, it is relatively easy to maintain the convergence at the decoder even with bit errors and states mismatch. It was found [17.5] that, with the short-term predictor order set at the conventional value of 10, the lack of a long-term predictor degrades female speech quality significantly. On the other hand, if the short-term predictor order is allowed to increase to 50, then a 50-th-order short-term predictor can exploit the pitch periodicity in female speech, since most female voices have a pitch period of less than 50 samples.

Using a 50-th-order short-term predictor in a conventional forward-adaptive CELP coder would be impractical due to the large number of predictor coefficients that need to be transmitted. However, since G.728 uses backward adaptation to update the short-term predictor, it doesn't cost any bit to increase the predictor order to 50. The only price paid is the increased computational complexity to derive the 50 predictor coefficients. In order to achieve toll quality



Amazon EX1010 Page 395

at 16 kb/s and with a frame size of only five samples, it was necessary to exploit the pitch periodicity in speech by using a 50-th-order short-term predictor. Thus, under the severe low-delay constraint, the G.728 had to make the complexity-quality trade-off – achieving toll quality through the use of a higher-complexity 50-th-order backward-adaptive short-term predictor.

The encoder structure of the G.728 LD-CELP coder is equivalent to the structure shown in Fig. 17.3 but without the long-term synthesis filter and the signal modifier. Both the excitation gain and the short-term synthesis filter are backward-adaptive. The excitation shape codebook (the fixed codebook) is a 10-bit codebook for five-dimensional VQ. However, to control the computational complexity of the codebook search, a gain-shape structured codebook is used, with the codebook constructed as the product of a 3-bit gain codebook and a 7-bit shape codebook. The three gain bits consists of one sign bit and two magnitude bits. The 7-bit shape codebook contains 128 independent codevectors.

Both the 7-bit shape codebook and the 2-bit magnitude codebook are closed-loop-trained based on the weighted distortion measure of LD-CELP using a large training speech file. Thus, the effects of backward adaptation of the predictor and gain are automatically taken into account in the training. Such closed-loop codebook training gives significant audio quality improvement and is crucial for achieving good speech quality in LD-CELP.

## 17.10 Pitch Synchronous Innovation CELP (PSI-CELP)

Pitch synchronous innovation CELP (PSI-CELP) [17.44] is the fourth kind of CELP coder that was selected as a standard coder. For the personal digital cellular (PDC) system of Japan, the half-rate standard coder (PDC-HR) is a 3.45 kb/s PSI-CELP coder [17.22].

The main distinguishing feature of PSI-CELP is that the fixed codebook is made adaptive by repeating the first portion of each codevector in a pitch-synchronous manner if the pitch period is smaller than the excitation vector dimension. Specifically, if the pitch period $T$ is less than the vector dimension $N$, then the first $T$ samples of the fixed codebook vectors are periodically repeated for later samples starting at the $(T+1)$-th sample. This is important for the PDC-HR coder because the coder uses a fairly large subframe size (vector dimension) of 80 samples (10 ms), which is larger than many typical pitch period values.

In place of the typical adaptive codebook in other CELP coders, the PDC-HR standard PSI-CELP coder actually uses either an adaptive codebook or a fixed codebook, with the fixed codebook mainly used in non-periodic regions of the speech signal. The adaptive codebook uses fractional pitch period [17.41, 42]. To save the codebook storage requirement, the fixed codebook has four basis vectors each rotated eight times to get 32 codevectors.

In place of the typical fixed codebook in other CELP coders, the PDC-HR standard PSI-CELP coder uses two stochastic codebooks in a conjugate structure [17.45] (which is similar to two-stage VQ). Each of the two stochastic codebooks contains 16 vectors. The selected codevector from each codebook is multiplied by a sign and the resulting two codevectors are added together. The pitch synchronous innovation (PSI) procedure is applied to the stochastic codebooks. To reduce the codebook search complexity, six out of the 16 codevectors from each stochastic codebook are preselected using a simplified method. Only the preselected codevectors go through full-complexity search.

The adaptive/fixed codebook gain and the stochastic codebook gain are jointly vector quantized to seven bits per subframe in a way somewhat similar to the codebook gain quantization of VSELP [17.17]. A delayed-decision coding technique is used to improve the PSI-CELP performance. Two best candidate codevectors are selected from the adaptive/fixed codebook search, and the remaining quantization procedures are performed for each candidate. The candidate that gives the lower distortion in the fully quantized version is selected as the winner.

## 17.11 Algebraic CELP (ACELP)

Algebraic code-excited linear prediction (ACELP) signifies the use of algebraic codes to make up the excitation in CELP. Due to the popularity of ITU-T recommendation G.729 [17.46] many people think of ACELP [17.24]



Amazon EX1010 Page 396

and G.729 as synonymous. However, in reality it is only the excitation of G.729 that is algebraic. Furthermore, G.729 and other speech coding standards utilizes sparse algebraic codes, (SACs) [17.24], and not an algebraic code as originally proposed in [17.23] in 1987 for CELP. The sparseness contributes significantly to a low computational complexity. Of further interest, a number of the techniques and structural changes proposed in [17.23] to the original CELP structure [17.3, 4] form the cornerstones of G.729 and many other speech coding standards. It should be noted that partly overlapping techniques were proposed in [17.38] in 1986. These techniques and structures are reviewed in other relevant sections, e.g., Sect. 17.6.

As indicated above, although the abbreviation ACELP was introduced in 1990 in [17.24], earlier work on algebraic codes for CELP appeared in 1987 in [17.23] and [17.47]. A significant advancement is the notion of sparse algebraic codes [17.24]. The idea of using sparse excitation in CELP coding was introduced in [17.38] in 1986, but in the context of Gaussian excitation vectors and not algebraic codes. Hence, the pulse amplitudes would be Gaussian distributed and not binary $(+1, -1)$ as in sparse algebraic codes. In [17.38], the use of sparse excitation for CELP is compared to MPLPC (multipulse linear predictive coding) [17.2] in terms of the number of nonzero pulses for voiced speech. Other early publications on sparse pulse excitation for CELP include [17.48] and [17.49].

ACELP has been a dominating form of excitation as the fixed codebook excitation in CELP speech coding standards from the mid 1990s until today (2006).

### 17.11.1 ACELP Background

ACELP appears to originate from the idea of using binary error correcting codes to represent $N$ points on an $M$ dimensional hyper sphere, i. e., a fixed codebook of size $N$ and vector dimension $M$ [17.47]. Even earlier, [17.50] proposed a spherical vector quantizer for encoding of the 1000 Hz base band of the short-term prediction residual after eighth-order LPC. The justification for the view of the fixed codebook representing points on a hyper sphere originates from the separate gain of the fixed codebook. Basically, the length of the residual vector, that is being approximated by the fixed codebook, can be considered normalized. Hence, the residual vectors will be points on a hypersphere. The task of the fixed codebook is to populate the hyper sphere in an optimal way. If the common assumption, that the samples of the residual vector after short- and long-term prediction are independent and identically distributed (i.i.d.) Gaussian, is reasonable, then the normalized residual vectors in terms of points on the hypersphere will be a uniform distribution. All $(+1, -1)M$-tuples will represent a uniform discrete sampling of the hypersphere with $2^M$ points, and it seems reasonable to pick the fixed codebook as a subset of the $2^M(+1, -1)M$-tuples. Error-correcting codes are attractive for picking the subset as they basically maximize the minimum distance between any two codewords, codevectors in the context of a fixed codebook, and spread out codewords for maximum coverage. Naturally, a mapping from the $(0, 1)M$ bits of a error correcting codeword to the $(+1, -1)M$ elements of the codevector of the fixed codebook is required. The benefit of such a fixed codebook is that no storage is required and it lends itself well to efficient methods.

An important additional feature is the realization that only relatively few nonzero pulses are required. For analysis-by-synthesis multipulse with only short-term prediction, the need for few nonzero pulses was reported in [17.2], where little improvement was observed after 8 pulses per 80 samples (10 ms). Similarly in the context of CELP, i. e., with long-term prediction as well, [17.38] reported the need for few nonzero elements. Although [17.38] includes long-term prediction an equivalent of four pulses per 40 samples (5 ms) was reported. However, neither [17.2] nor [17.38] were considering pulses of binary amplitude, but instead pulses of arbitrary amplitude. In [17.48,51,52], and [17.49] sparse binary pulse codevectors were discussed, and in [17.24] it was proposed along with the definition of the abbreviation ACELP and SAC. Technically, sparse $(+1, -1)$ binary excitation is really $(+1, 0, -1)$ ternary excitation.

The concept of pulse tracks and interleaved permutation codes [17.53, 54] contribute to further reducing complexity of searching and bit-rate of coding the sparse binary pulses. Partitioning the sample of an excitation vector into multiple tracks and applying a permutation code (resulting in sparse binary pulses) to each track has proven effective. Furthermore, interleaving the tracks is typical as it allows the flexibility of high local density of pulses in the final fixed codebook excitation.

**Table 17.1** Example ACELP ISPP excitation structure

| Track/code | Sample positions | Bits |
|---|---|---|
| 1 | 0, 5, 10, 15, 20, 25, 30, 35 | 3 + 1 = 4 |
| 2 | 1, 6, 11, 16, 21, 26, 31, 36 | 3 + 1 = 4 |
| 3 | 2, 7, 12, 17, 22, 27, 32, 37 | 3 + 1 = 4 |
| 4 | 3, 8, 13, 18, 23, 28, 33, 38 | 3 + 1 = 4 |
| 5 | 4, 9, 14, 19, 24, 29, 34, 39 | 3 + 1 = 4 |



Amazon EX1010 Page 397

Table 17.1 shows an example of a 40 sample ACELP fixed codebook made up of five single pulse interleaved permutation codes (5 tracks). Hence, it is a so-called interleaved single pulse permutation (ISPP) design. It results in a $5 \cdot (3+1) = 20$-bit fixed codebook excitation with five pulses. Other ACELP interleaved permutation codes employ multiple pulses per code/track [17.55].

Key advantages of ACELP are that it

- lends itself well to efficient methods,
- eliminates the need to store codebook,
- offers good speech quality,
- is flexible, and
- enables the use of large codebooks.

### 17.11.2 ACELP Efficient Search Methods

Modern ACELP is typically used in conjunction with the adaptive codebook implementation of the long-term (pitch) synthesis filter. Hence, the present section will present the search methods of ACELP in that context. Furthermore, the perceptual weighting filter of modern ACELP coders is frequently given by

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} , \tag{17.12}$$

where $A(z)$ is the unquantized prediction error filter and $0 < \gamma_2 < \gamma_1 < 1$. This form is used here without limitation, and all equations are easily extended to accommodate any perceptual weighting filter.

The efficient search methods for ACELP are based on a target signal, $t(n)$, which is basically the weighted input speech from which contributions from

1. ringing of filter memory
2. adaptive codebook

have been subtracted. Hence, the target signal, $t(n)$, is in the weighted speech domain. Searching the ACELP fixed codebook involves finding the entry that minimizes the MSE between the target signal and the ACELP codevector passed through the perceptual weighting filter and short-term synthesis filter, both with zero memory. This is illustrated in Fig. 17.8, and it is expressed as

$$I = \arg\min_k (E_k) = \arg\min_k \left\{ \sum_{n=0}^{N-1} [t(n) - y_k(n)]^2 \right\} , \tag{17.13}$$

where $y_k(n)$ is the output from passing the $k$-th ACELP codevector through the perceptual weighting filter and short-term synthesis filter. In terms of the ACELP codevector $c_k(n)$ (17.13) is expressed as

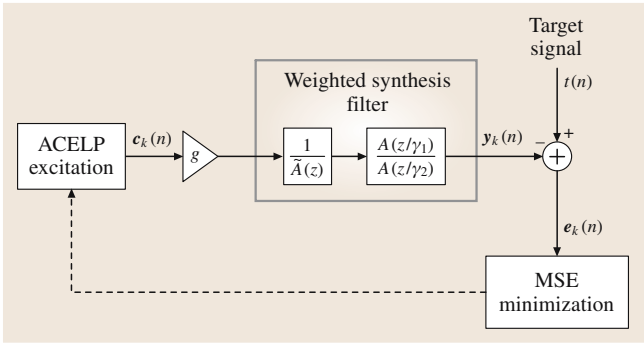



**Fig. 17.8** Basic ACELP fixed codebook excitation search

$$I = \arg\min_k \left[ \sum_{n=0}^{N-1} (t(n) - \{h(n) * [g \cdot c_k(n)]\})^2 \right] , \tag{17.14}$$

where $h(n)$ is the impulse response of the weighted synthesis filter, i. e., the inverse $z$-transform of

$$H(z) = \frac{1}{\tilde{A}(z)} W(z) = \frac{1}{\tilde{A}(z)} \frac{A(z/\gamma_1)}{A(z/\gamma_2)} , \tag{17.15}$$

where $1/\tilde{A}(z)$ is the quantized short-term synthesis filter. Note that enhancements to the ACELP codebook can be included into the impulse response, $h(n)$, without affecting the following search techniques. The enhancements are in the form of a prefilter, $F(z)$. One such example is the in-frame pitch enhancement for pitch lags shorter than the length of the ACELP codevector [17.25, 56]:

$$F(z) = \frac{1}{1 - \beta Z^{-T}} , \tag{17.16}$$

where $T$ is the pitch lag, and $\beta$ is a suitable, typically adaptive, filter coefficient related to the periodicity. This particular ACELP pitch prefilter is incorporated into $h(n)$ according to

$$\begin{aligned} h(n) &\leftarrow h(n) + \beta h(n-T) , \\ n &= T, T+1, \ldots , N-1 . \end{aligned} \tag{17.17}$$

In AMR-WB (adaptive multi-rate) [17.57], the ACELP prefilter additionally includes a tilt part:

$$F(z) = \frac{1}{1 - \beta Z^{-T}} (1 - \alpha z^{-1}) , \tag{17.18}$$

where also $\alpha$ is adaptive.

Equation (17.14) is written in vector form as

$$
\begin{aligned}
I &= \arg\min_k \left[ (\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k)^{\mathrm{T}} (\boldsymbol{t} - g\boldsymbol{H}\boldsymbol{c}_k) \right] \\
&= \arg\min_k \left[ \boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2 (\boldsymbol{H}\boldsymbol{c}_k)^T (\boldsymbol{H}\boldsymbol{c}_k) \right. \\
&\quad \left. - g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k - g (\boldsymbol{H}\boldsymbol{c}_k)^{\mathrm{T}} \boldsymbol{t} \right] \\
&= \arg\min_k \left( \boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2 \boldsymbol{c}_k^{\mathrm{T}} \boldsymbol{H}^{\mathrm{T}} \boldsymbol{H}\boldsymbol{c}_k - 2g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k \right) ,
\end{aligned}
\tag{17.19}
$$

where

$$
\begin{aligned}
\boldsymbol{t} &= [t(0)t(1)\cdots t(N-1)]^{\mathrm{T}} , \\
\boldsymbol{c}_k &= [c_k(0)c_k(1)\cdots c_k(N-1)]^{\mathrm{T}} , \\
\boldsymbol{H} &= \begin{pmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h(N-2) & h(N-3) & \cdots & 0 \\ h(N-1) & h(N-2) & \cdots & h(0) \end{pmatrix} .
\end{aligned}
$$

According to common procedure in ACELP, the excitation, $\boldsymbol{c}_k$, is found under the assumption of optimal gain $g$.

**Table 17.2** Key properties of ACELP excitation in standards

| Standard | Rate (kbps) | ACELP frame size (samples) | Pulses | Tracks | Bits |
|---|---|---|---|---|---|
| G.723.1 | 5.3 | 60 | 4 | 4 | 17 |
| G.729 | | 40 | 4 | 4 | 17 |
| G.722.2/AMR-WB | 23.85 / 23.05 | 64 | 24 | 4 | 88 |
| | 19.85 | | 18 | 4 | 72 |
| | 18.25 | | 16 | 4 | 64 |
| | 15.85 | | 12 | 4 | 52 |
| | 14.25 | | 10 | 4 | 44 |
| | 12.65 | | 8 | 4 | 36 |
| | 8.85 | | 4 | 4 | 20 |
| | 6.6 | | 2 | 2 | 12 |
| GSM AMR | 12.2 (GSM EFR) | 40 | 10 | 5 | 35 |
| | 10.2 | | 8 | 4 | 31 |
| | 7.95 / 7.4 (TIA EFR) | | 4 | 4 | 17 |
| | 6.7 (PDC EFR) | | 3 | 3 | 14 |
| | 5.9 | | 2 | 2 | 11 |
| | 5.15 / 4.75 | | 2 | 5 | 9 |
| EVRC | Full-rate | 53 / 54 | 8 | 5 | 35 |
| | Half-rate | | 3 | 3 | 10 |
| VMR-WB | Full-rate | 64 | 8 | 4 | 36 |
| | Voiced and generic half-rate | | 2 | 2 | 12 |
| MPEG-4 | 8 kHz core | | 3-12 | | |
| | 8 kHz enhancement | 40 | 2 | | |
| SMV | Rate 1/1 type 1 | 40 | 8 | 8 | 30 |
| | Rate 1/1 type 0 | 40 | 5 | 5 | 21 |
| | | | 5 | 5 | 20 ⇒ 22 |
| | | | 5 | 4 | 20 |
| | Rate 1/2 type 1 | 53/54 | 2 | 2 | 12 |
| | | | 3 | 3 | 11 ⇒ 13 |
| | | | 5 | 5 | 11 |
| | Rate 1/2 type 0 | 80 | 2 | 2 | 14 |
| | | | 3 | 3 | 13 ⇒ 15 |
| | | | Gaussian | NA | 13 |



Amazon EX1010 Page 399

The gain is determined by minimizing the error energy:

$$\frac{\partial E_k}{\partial g} = \frac{\partial}{\partial g}\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} + g^2\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k - 2g\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right) = 0$$

$$\Downarrow$$

$$g = \frac{\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k} \,. \tag{17.20}$$

Inserting this result in (17.19) yields

$$I = \arg\min_k \left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t} - \frac{\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) \,. \tag{17.21}$$

Since $\boldsymbol{t}$ is independent of the codevector, $\boldsymbol{c}_k$, this is equivalent to

$$I = \arg\max_k \left(\frac{\left(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) \,. \tag{17.22}$$

The numerator is often rewritten in terms of the backward filtered target vector [17.23,53], $\boldsymbol{t}_{\mathrm{b}} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{t}$, and the search is expressed as

$$I = \arg\max_k \left(\frac{\left(\boldsymbol{t}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{c}_k}\right) = \arg\max_k \left(\frac{\left(\boldsymbol{t}_{\mathrm{b}}^{\mathrm{T}}\boldsymbol{c}_k\right)^2}{\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{c}_k}\right) \,, \tag{17.23}$$

where $\boldsymbol{\Phi} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$ is symmetric and contains the autocorrelation of the impulse response of $H(z)$:

$$\Phi(i,\,j) = \begin{cases} \sum_{l=0}^{N-1-i} h(l)h(l+i-j) \,, & j \le i \\ \Phi(j,i) \,, & j > i \end{cases} \,. \tag{17.24}$$

The backward filtered target $\boldsymbol{t}_{\mathrm{b}}$ and the autocorrelation matrix $\boldsymbol{\Phi}$ are calculated prior to searching the ACELP codebook.

The efficient search methods all explore ways to most efficiently evaluate (17.23), either in a mathematically equivalent way, or in a slightly suboptimal way with only minor sacrifice in performance. The following subsections will present some of the methods typically used in ACELP.

### Sparse ACELP

In sparse ACELP only relatively few pulses are nonzero, and if they are binary, the nonzero pulses take on an amplitude of $\pm 1$. The $P$ pulses, $P < N$, are described uniquely by the location and amplitude, $m_i$ and $a_i$, respectively, $i = 0, 1, \ldots, P-1$. Note that $a_i = \pm 1$. With this notation, the search of (17.23) can be written as [17.53]

$$I = \arg\max_{\boldsymbol{m}\times\boldsymbol{a}} \left(\frac{\left(\sum_{i=0}^{P-1} t_{\mathrm{b}}(m_i)a_i\right)^2}{\sum_{i=0}^{P-1}\Phi(m_i,m_i) + 2\sum_{i=0}^{P-2}\sum_{j=i+1}^{P-1} a_i a_j \Phi(m_i,m_j)}\right) \,, \tag{17.25}$$

where the search is identifying the optimal $P$ pulse positions $\boldsymbol{m} = (m_0, m_1, \ldots, m_{P-1})$ and $P$ amplitudes $\boldsymbol{a} = (a_0, a_1, \ldots, a_{P-1})$. This search can be performed in a nested loop so that the addition of one more pulse is considered in each loop. At the inner most loop, the final cost function for a given candidate is calculated by one additional addition and the square to get the numerator, and $P$ additional additions and one multiplication to get the numerator. This provides for a very efficient exhaustive search, but as the size of the codebook increases this quickly becomes impractical [17.53]. Sparse ACELP is used in all ACELP standards.

### A Priori Sign

Studying the numerator of (17.25) it is evident that the cross correlation will be maximized by setting the pulse amplitudes, $\boldsymbol{a}$, basically the pulse signs, to be the same as the sign of the backward filtered target, $\boldsymbol{t}_{\mathrm{b}}$ [17.58]. Accordingly, by setting the pulse signs a priori only the pulse locations $\boldsymbol{m}$ need to searched. This is achieved by incorporating the a priori pulse signs into the backward filtering target and correlation matrix prior to the search [17.58] according to:

$$\tilde{t}_{\mathrm{b}}(n) = |t_{\mathrm{b}}(n)| \,, \tag{17.26}$$

$$\tilde{\Phi}(i,\,j) = \mathrm{sign}[t_{\mathrm{b}}(i)]\mathrm{sign}[t_{\mathrm{b}}(j)]\Phi(i,\,j) \,. \tag{17.27}$$

and performing the search for pulse positions according to

$$I = \arg\max_{\boldsymbol{m}} \left(\frac{\left(\sum_{i=0}^{P-1} \tilde{t}_{\mathrm{b}}(m_i)\right)^2}{\sum_{i=0}^{P-1}\tilde{\Phi}(m_i,m_i) + 2\sum_{i=0}^{P-2}\sum_{j=i+1}^{P-1}\tilde{\Phi}(m_i,m_j)}\right) \,, \tag{17.28}$$

where, again, the search identifies the optimal $P$ pulse positions $\boldsymbol{m} = (m_0, m_1, \ldots, m_{P-1})$. To eliminate the multiplication by 2 during the search, all off-diagonal elements of $\tilde{\Phi}(i,\,j)$ can be multiplied by 2 prior to the search. The a priori determination of the pulse signs is

**Table 17.3** Effectiveness of ACELP focused search [17.53]

| Percentage of exhaustive search | SNR (dB) |
|---|---|
| 100 | 22.20 |
| 4 | 22.14 |
| 1.6 | 22.00 |
| 0.2 | 22.05 |
| 0.15 | 21.83 |
| 0.05 | 21.80 |
| 0.03 | 21.50 |

not in general equivalent to the exhaustive search given by (17.25). It is used in G.729 [17.46], G.723.1 [17.25], EVRC [17.26], and GSM-EFR [17.56] among others. Note that the presetting of the sign can be based on the sign of a signal that is a linear combination of the backward filtered target and the long-term prediction residual [17.56] (or a similarly suitable signal), and hence not only based on the sign of the backward-filtered target.

### Focused Search

Various focused search methods exist [17.53, 54, 58]. Generally, the $P$ pulses are searched in nested loops and constraints on the correlation,

$$C = \sum_{i=0}^{p-2} \tilde{t}_b(m_i) , \quad p < P , \tag{17.29}$$

are set up to disregard subsections of the ACELP codebook (represented by entry into the $p$-th loop) if the correlation $C$ does not meet a predefined threshold. A table from [17.53] is reproduced in Table 17.3, illustrating this in the form of the relation between the percentage of exhaustive search and the SNR for a sentence spoken by a female talker. The threshold can be calculated in various ways. It can be based on the maximum possible correlation of the existing $p-1$ pulse candidates, or the average correlation over their possible positions, or a combination thereof as in [17.58]. Generalizing [17.58] leads to a threshold based on maximum pulse correlation

$$C_{\max} = \sum_{l=0}^{p-2} \max_{i \in \mathcal{T}_l} \left[\tilde{t}_b(i)\right] \tag{17.30}$$

and average pulse correlation

$$C_{\text{avg}} = \sum_{l=0}^{p-2} \text{avg}_{i \in \mathcal{T}_l} \left[\tilde{t}_b(i)\right] . \tag{17.31}$$

In (17.30) and (17.31), $\mathcal{T}_l$ is the set of possible pulse positions of pulse $l$. The threshold can then be constructed as a combination of the maximum and average correlation:

$$C_{\text{thr}} = C_{\text{avg}} + \alpha_{\text{thr}}(C_{\max} - C_{\text{avg}}) . \tag{17.32}$$

This will reduce the average complexity. To have a firm limit on the worst-case complexity a threshold on the number of times a certain loop can be entered may be enforced [17.58]. A very large part of the ACELP codebook can be ruled out this way.

### Depth-First Tree Search Procedure

As the number of pulses $P$ increases, additional methods can be used to search the effectively increasing codebook efficiently. One such method is used in G.729 Annex A [17.59], GSM-EFR [17.56], and AMR-WB [17.57]. The depth-first tree search procedure would typically be used instead of the focused search to further reduce complexity. The set of $P$ pulses is divided into $N_m$ subsets, typically with an equal number of $M$ pulses in each. While the subsets are searched sequentially, the pulses in each subset are searched jointly. First subset 1, then subset 2, etc. Subsequent sets are searched given the pulses of previous sets. This search is performed iteratively, circulating the assignment of pulses to tracks. For instance, at the first iteration the first two pulses can be assigned to tracks 0 and 1, while for the second iteration the first two pulses can be assigned to tracks 2 and 3, etc. In the second iteration the last two pulses can be assigned to tracks 0 and 1.

A reference signal may be constructed to assist in selecting a subset of pulse positions to consider in a given track. As an example, with pulses 0 and 1 being assigned to tracks 2 and 3 (each of eight positions), based on the reference signal, only four positions in track 2 are preselected for consideration for pulse 0. Such preselection reduces complexity. Techniques like this can be used to control the complexity and balance complexity of multiple rates in a speech coder. The AMR-WB speech coder [17.57] provides a good example of this technique. The reference signal is typically identical to the signal used to set the signs a priori (Sect. 17.11.2). It can be the backward filtered target signal as in [17.60], a linear combination of the backward filtered target and the long-term prediction residual signals as in [17.57], or some other suitable signal.

In G.729A, the depth-first tree search is used instead of the focused search of G.729 and provides a direct comparison. It results in a very significant saving of 5 MIPS (million instructions per second) at the cost of



Amazon EX1010 Page 401

a slight degradation in performance, about 0.2 dB SNR according to [17.60].

### 17.11.3 ACELP in Standards

A number of speech coding standards utilizes the ACELP excitation, that is, the sparse binary pulse excitation. In grouping by standards body: ITU-T G.723.1 [17.25] (ACELP for the 5.3 kbit/s rate, multipulse for the 6.3 kbit/s rate), ITU-T G.729 [17.46], ITU-T G.722.2 [17.61], ETSI GSM-EFR (European Telecommunications Standards Institute) [17.56], ETSI/3GPP GSM AMR (3-rd Generation Partnership Project) [17.62], 3GPP GSM AMR-WB [17.57], TIA-127 EVRC (enhanced variable rate codec) [17.26], TIA-136 EFR [17.63], TIA/3GPP2 SMV [17.64], 3GPP2 VMR-WB [17.28], ARIB PDC-EFR (Association of Radio Industries and Businesses), and MPEG-4 CELP (Moving Pictures Expert Group) narrow-band audio [17.65].

It should be noted that ITU-T G.722.2 and 3GPP GSM AMR WB is the same wide-band multirate speech coder. It was initially standardized for cellular by 3GPP and subsequently submitted to the ITU for consideration in an, at the time, ongoing standardization effort of wide-band coding of speech at around 16 kbit/s. It should also be noted that the AMR-WB rate of 12.65 kbit/s is interoperable with one of the rates of 3GPP2 VMR-WB. Furthermore, the GSM EFR is identical to the 12.2 kbit/s rate of ETSI/3GPP AMR, TIA-136 EFR is equivalent to the 7.4 kbit/s rate of AMR, and ARIB PDC-EFR is equivalent to the 6.7 kbit/s rate of AMR. Note that it can be somewhat confusing that multiple standards bodies use the same acronym, EFR. The MPEG-4 CELP is really somewhat of a hybrid between ACELP and multipulse as it doesn't use binary or ternary pulses, but instead apply a VQ of the pulse amplitudes. Similarly, TIA/3GPP2 SMV is a hybrid as it is based on sub-codebooks where most sub-codebooks are ACELP-like, but one is different, and the codebooks are weighted differently in selecting the overall best excitation.

Table 17.2 summarizes some of the key properties of the ACELP fixed codebook excitation structures used in the standards listed above.



## 17.12 Conjugate Structure CELP (CS-CELP) and CS-ACELP

Conjugate structure CELP (CS-CELP) introduces conjugate structure VQ for CELP. It was originally proposed in [17.66] for CELP. However, earlier the conjugate structure was used in other coders, e.g., in transform coding [17.67]. Generally, conjugate structure VQ constructs the output codevector $\boldsymbol{c}_{i,j}$ as a linear combination of the output codevectors from two codebooks $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ respectively:

$$\boldsymbol{c}_{i,j} = \alpha_1 \cdot \boldsymbol{c}_i + \alpha_2 \cdot \boldsymbol{c}_j \,. \tag{17.33}$$

The advantages of the conjugate structure compared to a single VQ is threefold [17.66]:

- improves resilience to bit-errors,
- reduces memory requirement,
- facilitates reduced complexity methods.

These advantages are demonstrated in [17.66] and [17.68], which also present methods to train the conjugate structure codebooks. Furthermore, [17.66] and [17.68] present an 8 kb/s CS-CELP speech coder where the conjugate structure is used for both the fixed codebook excitation and the joint quantization of the subframe based adaptive codebook gain and fixed codebook gain. This coder was submitted by NTT for the ITU-T (CCITT at the time) G.729 standardization. Although the coder was not standardized as G.729, techniques from multiple candidates were merged and eventually formed G.729. The resulting G.729 standard [17.46] uses the conjugate structure for the 2-dimensional joint VQ of the subframe based adaptive codebook gain and fixed codebook gain. In G.729 the linear combination of the codevectors from the two codebooks is a simple summation, and the conjugate structure codevector is

$$\boldsymbol{c}_{i,j} = \boldsymbol{c}_i + \boldsymbol{c}_j \,. \tag{17.34}$$

One codebook has a bias towards the element corresponding to the fixed codebook gain, while the other codebook has a bias towards the element corresponding to the adaptive codebook gain. This allows open-loop preselection of both codebooks based on the respective dominant parameter. This leaves only a subset (a quarter) for the more-complex joint closed-loop search without any noticeable degradation compared to an exhaustive closed-loop search [17.58].

Amazon EX1010 Page 402

## 17.13 Relaxed CELP (RCELP) – Generalized Analysis by Synthesis

Relaxed CELP (RCELP) [17.69] has become synonymous with a specific and practical usage of the generalized analysis-by-synthesis principle proposed in [17.15,70]. The basic idea of generalized analysis-by-synthesis is to relax the waveform matching constraint of CELP without affecting the speech quality. In principle, a relaxation of the waveform matching can be incorporated into the error criterion. However, generalized analysis-by-synthesis proposes a general signal modification function, that is applied to the original signal, constrained to provide a perceptually similar signal. The idea is to modify the signal into a signal that is simpler to represent (in the sense of minimizing distortion relative to the modified signal), yet perceptually indistinguishable from the original. The fundamental and the generalized analysis-by-synthesis paradigms as presented in [17.15] are depicted in Figs. 17.9 and 17.10, respectively.

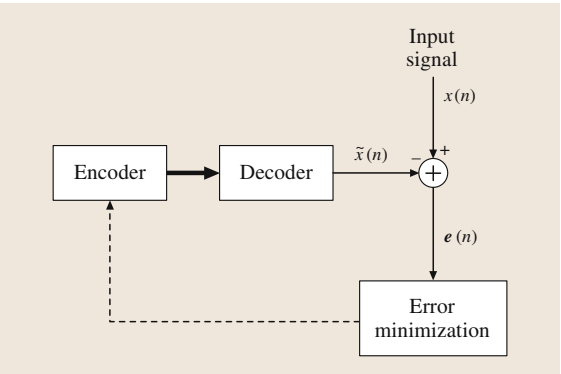



**Fig. 17.9** Basic analysis by synthesis

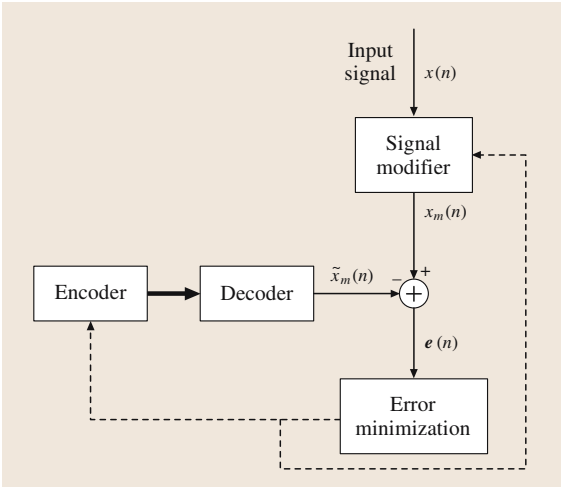



**Fig. 17.10** Generalized analysis by synthesis

### 17.13.1 Generalized Analysis by Synthesis Applied to the Pitch Parameters

Application of generalized analysis-by-synthesis to the pitch parameters implies that one or both of the pitch period contour and the pitch gain contour of the speech signal is modified for easier encoding. A detailed discussion of the application to the pitch period and gain is available in [17.70]. Typically, the pitch period and pitch gain are updated approximately every 5 ms. The idea of RCELP is to update and encode them far less frequently, e.g., every 20 ms, and then use an interpolated pitch period contour and pitch gain contour throughout the 20 ms. This is justified by the observation that the pitch period and periodicity of voiced speech evolve slowly. In order to maintain the coding efficiency when using the interpolated pitch period and gain (compared to the more-frequent update), the speech signal must be modified to follow these contours. Otherwise, the waveform matching of the analysis-by-synthesis principle will break down as the interpolated pitch period and gain will result in a pitch contribution from the adaptive codebook (or equivalently, the pitch predictor) that is misaligned with the reference signal for the analysis-by-synthesis. Since the pitch evolution in voiced speech is slow, typically only minor adjustments to the speech signal are necessary and without impact to the speech quality. This was demonstrated in [17.69] where subjective results showed that the modified speech received mean opinion scores (MOS) very close to those of the original speech, at a level similar to those of 64 kbit/s $\mu$-law (Table 17.4).

Besides exploiting the typical slow evolution of the pitch period to reduce the bits required to encode the pitch period, RCELP can also save the bits otherwise often used to specify fractional pitch lags. Basically, the speech can be modified to fit an interpolated pitch period contour specified by integer lags. Note that the interpolated pitch period contour will have fractional

**Table 17.4** MOS of RCELP signal modification [17.69]

| | MOS–IRS input | MOS–flat input |
|---|---|---|
| 64 kb/s $\mu$-law | 3.94 | 4.03 |
| modified speech | 3.90 | 3.99 |



Amazon EX1010 Page 403

lags, but at the points of transmission the pitch lag has integer values.

### Application to the Pitch Period

Although [17.70] discusses the application of generalized analysis-by-synthesis to both the pitch period and the pitch gain, in speech coding standards, the application to the pitch period has found the widest usage. Accordingly, the application of generalized analysis-by-synthesis to the pitch period is commonly referred as RCELP. In narrow-band speech coding this can bring the bit allocation for the pitch period from $8+5+8+5$ bits to seven bits for a 20 ms frame. This is a significant saving that frees up bits for improving quantization of other parameters of the coder. In practice the pitch period may be estimated open loop and used to create an interpolated continuous pitch track. The speech signal is subjected to minor adjustments in order to fit this interpolated pitch track. Often the adjustments are carried out through time warping and conveniently carried out in the short-term prediction residual domain [17.15]. The modified residual signal can be passed through the short-term synthesis filter or the weighted synthesis filter in order to obtain the modified speech signal or the modified weighted speech signal, respectively. For lower complexity, time shifting of sequential blocks can be applied instead of time warping [17.69]. Later, [17.71] proposed to carry out the signal modification in the weighted speech domain and use a combination of time warping and time shifting of blocks for modifying the speech signal. The method only shifts the high-energy blocks (supposedly containing the pitch pulse and immediate vicinity), while the low-energy blocks are time warped to adjust the pitch period. Effectively, this method does not alter the local waveform properties of the pitch pulses.

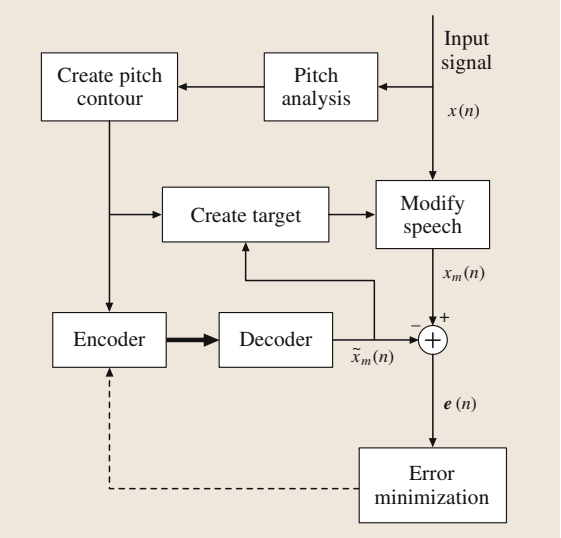



**Fig. 17.11** Principle of RCELP

The general sequence of operations for RCELP are:

1. estimate open loop pitch,
2. create pitch contour,
3. create target signal for modifying the speech,
4. modify the speech, and
5. generate the adaptive codebook contribution.

This is depicted in principle at a high level in Fig. 17.11, where the generation of the adaptive codebook contribution takes place inside the encoder like usual, except now according to a pitch contour as opposed to a fixed pitch. Note that the additional blocks in Fig. 17.11 as compared to Fig. 17.9 belong to the encoder just like the *signal modifier* in Fig. 17.10. Furthermore, the decoder would comprise identical functions to

1. create the pitch contour, and
2. generate the adaptive codebook contribution according to the pitch contour.

Note that Fig. 17.11 should be viewed at a conceptual level as the actual signal modification could take place in any domain, but with a corresponding modified input signal.

*Estimate Open-Loop Pitch.* It is critical for optimal performance of RCELP to get a good estimate of the pitch period. Typically, it is estimated in an open loop manner at the boundary of each frame. The pitch period towards the end of frame $m$ is denoted by $\mathrm{pp}(m)$.

*Create Pitch Contour.* In the general case where the pitch period of the previous frame, $\mathrm{pp}(m-1)$, and the current frame $\mathrm{pp}(m)$ are relatively close, the pitch period contour is created as a continuous interpolation between the two. A simple continuous interpolation is the linear interpolation between the two pitch periods:

$$\mathrm{ppc}(m,n) = \frac{n+1}{N+1}\mathrm{pp}(m) + \frac{N-n}{N+1}\mathrm{pp}(m-1)\,,$$
$$n = 0, 1, \ldots, N-1\,. \qquad (17.35)$$

The continuous interpolated pitch period contour is referred as the interpolated pitch contour in the following.

*Create Target Signal for Modifying the Speech.* In order to modify the speech signal to follow the in-

**Table 17.5** Properties of RCELP in standards

| Standard | RCELP mode | Time-scale search | | Time-scale modification | | Max. delay drift |
|---|---|---|---|---|---|---|
| | | Domain | Resolution | Domain | Method | |
| EVRC | Continuous | Short-term residual | (1/8)-th sample | Short-term residual | Interpolation | |
| SMV | Switched | Weighted speech | (1/10)-th sample | Weighted speech | Interpolation & block shifting | ±2.5 ms |
| VMR-WB | Switched | Weighted speech | (1/8)-th sample | Short-term residual | Block shifting | 0 ms |

terpolated pitch period contour a target signal can be constructed. One possibility is to extrapolate the previously modified speech signal according to the pitch contour and use this as a target for modifying the time-scale of the speech signal of the current frame. Note that in practice, the time-scale modification of the speech signal can take place in any domain. As an example, in EVRC [17.26] it takes place in the short-term residual signal domain, in SMV [17.64] it takes place in the weighted speech signal domain, and in VMR-WB [17.28] the actual time-scale modification is done in the short-term residual signal domain, but the time-scale modification is calculated in the weighted speech signal domain.

*Modify the Speech.* The speech signal can be modified in multiple ways, and in multiple domains as mentioned above. The goal is to modify the time-scale of the speech signal so that it follows the interpolated pitch contour. With the target signal described above, this can be achieved by modifying the time-scale of the speech signal so as to maximize the correlation between the modified speech signal and the target signal. If the time-scale modification is carried out on a pitch period by pitch period basis, then it becomes a matter of identifying the pitch pulses and perform time-scale modification so that the pitch pulses of the modified speech signal lines up with the pitch pulses of the target signal. The time-scale modification can be carried out either by time warping in the form of resampling, time shifting of blocks, or a combination thereof.

*Generate the Adaptive Codebook Contribution.* The final step is to generate the adaptive codebook (or equivalently, the pitch predictor) contribution. Based on the past short-term synthesis filter excitation the adaptive codebook contribution is generated by signal interpolation according to the interpolated pitch contour. Fractional resolution in the order of 1/8-th is generally used, and truncated sinc windows are used for the signal interpolation.

A number of practical issues need to be considered when using RCELP:

1. pitch dependence
2. delay drift
3. pitch doubling and halving
4. complexity

*Pitch Dependence.* Not surprisingly, there is a strong dependency on the pitch period estimation. If the pitch period estimation is not of sufficient accuracy, naturally, degradation in performance should be expected. Furthermore, RCELP works the best for voice speech with slow evolution of the pitch period. Accordingly, some coders have separate mode(s) with more frequent update of the pitch period for frames where a single interpolated pitch contour is not expected to provide satisfactory performance.

*Delay Drift.* Generally, a variable delay is introduced during the signal modification. This contributes to the overall system delay, and it may drift. Typically, a constraint on the maximum delay drift of about 3 ms is enforced during RCELP. One exception is VMR-WB where the pitch contour is constrained to provide perfect time synchrony between the original speech and the modified speech at frame boundaries. This means that there is no additional delay from RCELP in this implementation.

*Pitch Doubling and Halving.* Pitch doubling and halving can present a significant challenge if RCELP is used continuously. For multimode coders an alternative mode with more frequent pitch period updates without pitch period interpolation could be used in such cases. However, solutions to continuously use RCELP also for pitch doubling and halving are discussed in [17.70].

*Complexity.* As reported in [17.70] the complexity of initial versions was very high. Some initial experimental versions were reported to be 75 million operations per second (MOPS), but projections indicated solutions with



Amazon EX1010 Page 405

a complexity of about 15 MOPS would be achievable. Such solutions have since materialized in the form of standards such as EVRC, SMV, and VMR-WB which are all around 40 weighted MOPS (WMOPS) or less for full duplex encoding and decoding.

### 17.13.2 RCELP in Standards

RCELP is utilized in a number of standards to efficiently represent the pitch period: TIA-127 EVRC [17.26], TIA/3GPP2 SMV [17.64], 3GPP2 VMR-WB [17.28], and it was part of a proposal for the ITU-T 4 kb/s standardization effort [17.8] before that effort was abandoned by the ITU. Although, those standards implement RCELP in different ways and variations, they all achieve the same goal of reducing the bit rate required to represent the pitch period. Some apply RCELP continuously while others apply RCELP only for specific modes. Some of the RCELP features of these standards are summarized in Table 17.5.

## 17.14 eX-CELP

The eXtended CELP (eX-CELP) technique [17.71] is more a collection of techniques or a general approach than a single specific technique. The general idea is to emphasize the perceptual important features during encoding within the context of analysis-by-synthesis. Basically, the closed-loop waveform matching of analysis-by-synthesis is relaxed by combining open-loop and closed-loop control. The necessity to relax the strict waveform matching originates from the observation that at low bit rate it is not possible to achieve toll quality through strict waveform matching, and instead emphasis on the perceptually important features is necessary. This is achieved by incorporating signal classification into weighting, using RCELP-like techniques, combining open-loop and closed-loop, designing the fixed codebook with multiple sub-codebooks addressing different signal characteristics, and utilizing multimode encoding.

*Signal Classification.* eX-CELP uses elaborate signal classification. As many as six classes are used [17.27]:

- silence/background noise
- stationary unvoiced
- nonstationary unvoiced
- onset
- nonstationary voiced
- stationary voiced

The classification takes place in multiple stages and is refined as information becomes available in the algorithm.

*Signal Modification.* The signal modification algorithm not only modifies the pitch contour to allow a lower bit rate for the encoding of the pitch period like RCELP, it also modifies the speech signal to increase the pitch contribution of the adaptive codebook. According to [17.71], it uses waveform interpolation or harmonic smoothing to pre-smooth voiced transition areas in an open-loop manner. This results in faster buildup of the adaptive codebook. The underlying objective is to increase/enhance the pitch contribution as much as possible without introducing noticeable distortion to the signal.

*Combination of Open-Loop and Closed-Loop.* The algorithm combines open-loop and closed-loop extensively. One example is the gain quantization. It is well known that for background noise, accurate waveform matching is not necessary for a perceptual accurate reproduction. However, features such as a natural (smooth) energy contour and a dense excitation is more important. Hence, in the presence of background noise the gain quantization favors open-loop over close-loop, and it aims at maintaining a smooth energy contour. Similarly, it would favor selection of excitation from a relatively denser codebook.

*Fixed Codebook.* The fixed codebook is made up of multiple sub-codebooks of varying pulse density and even a sub-codebook of dense random-like noise. Each sub-codebook is designed and tuned with a certain type of signal or speech in mind. Each codebook is generally searched according to the analysis-by-synthesis principle, but the selection between the best outputs of the various sub-codebooks is influenced by the classification information, and other signal parameters such as estimate of background noise level and *peakiness* of the speech.

*Multimode.* eX-CELP distinguishes between two fundamentally different encoding modes, types 0 and 1.



Amazon EX1010 Page 406

**Table 17.6** Key general features of type 0 and type 1 coding modes in eX-CELP

| Entity | Type 0 | Type 1 |
|---|---|---|
| Signal character | All other | Staitonary voiced |
| Subframes | Fewer | More |
| Pitch period (adaptive codebook) | Subframe based | Frame based |
| Fixed codebook | Pulse & Gaussian sub-codebooks | Pulse sub-codebooks |
| Adaptive codebook gains | Joint 2-D VQ (subframe based) | A priori VQ (frame based) |
| Fixed codebook gains | Joint 2-D VQ (subframe based) | A posteriori VQ (frame based) |

**Table 17.7** MOS of SMV at various operating points [17.71]

| Coder | Average bit rate | MOS (clean speech) | MOS (noisy speech) |
|---|---|---|---|
| EVRC | $x$ | 3.58 | 3.35 |
| SMV Mode 0 | $1.0 \cdot x$ | 3.90 | 3.57 |
| SMV Mode 1 | $0.71 \cdot x$ | 3.64 | 3.53 |
| SMV Mode 2 | $0.56/0.60 \cdot x$ | 3.46 | 3.53 |

**Table 17.8** MOS of 4 kbit/s eX-CELP [17.8]

| Input | Coder | MOS |
|---|---|---|
| MIRS (modified intermediate reference system) | G.726 | 3.14 |
| | G.729 | 3.43 |
| | 4 kb/s eX-CELP | 3.36 |
| Flat | G.726 | 3.12 |
| | G.729 | 3.27 |
| | 4 kb/s eX-CELP | 3.32 |

The bit allocation, the quantization methods, and even the subframe structure for the two modes can be different. In type 1, targeting stationary voiced speech, the subframe adaptive codebook gains are prequantized open-loop using VQ prior to subframe processing, while fixed codebook gains are left unquantized during subframe processing and quantized jointly using VQ after subframe processing is complete. For type 0, a more-conventional joint two-dimensional VQ of the subframe based adaptive and fixed codebook gains is used. However, the gain estimation and quantization is still not strictly analysis-by-synthesis for type 0, as focus remains on producing a perceptually faithful output, mixing open-loop and closed-loop. Key general features of the two types are listed in Table 17.6.

### 17.14.1 eX-CELP in Standards

The 3GPP2 SMV standard [17.64] is based on eX-CELP. Table 17.7 is reproduced from [17.71] and compares the MOS scores of SMV at various operating points of quality versus average bit rate with EVRC. The third column contains the MOS scores for clean speech, and the fourth column contains the MOS scores of speech in background noise. Also one of the promising ITU-T 4 kbit/s candidates [17.8] was based on eX-CELP. Some MOS scores comparing the performance to 32 kbit/s G.726 and 8 kbit/s G.729 are reproduced in Table 17.8.

## 17.15 iLBC

The iLBC coder [17.31, 32] is quite different from other analysis-by-synthesis speech coders. Most other analysis-by-synthesis speech coders use a long-term predictor or an adaptive codebook continuously and across the frame boundaries. Due to the relatively large bulk delay in repeating the previous waveform stored in the long-term predictor memory or the adaptive codebook, if a frame is lost, the degrading effect tends to propagate a while. The iLBC coder attempts to address this issue and improve the robustness against frame loss. It achieves this by employing block-independent coding of the adaptive codebook. In other words, the adaptive codebook of each speech frame is encoded independent of the previous frames. Therefore, if a frame is lost, the quality degrading effect due to a mismatched adaptive codebook is limited to the lost frame and will not propagate to the future frames.

Of course, the improved robustness does not come for free. By not allowing the adaptive codebook to be used across the frame boundaries, the iLBC coder gives up the opportunity to exploit the corresponding long-term redundancy in the speech signal across the frame boundaries. Therefore, for clear channel it inevitably sacrifices the output speech quality to some extent when compared with more-conventional CELP coders. In effect, the iLBC coder makes a trade-off between the



Amazon EX1010 Page 407

speech quality in clear-channel and the speech quality under frame loss conditions. Given that the iLBC coder was developed for voice over internet protocol (VoIP) over the general Internet that can have a fairly high packet loss rate, it is reasonable for iLBC to make such a trade-off.

The iLBC coder was standardized as an IETF experimental standard [17.32]. It has two versions: a 13.3 kb/s version with a 30 ms frame size and 10 ms look-ahead, and a 15.2 kb/s version with a 20 ms frame size and 5 ms look-ahead. The two versions divide each speech frame into 5 ms subframes. The short-term prediction residual signal is first calculated. The two consecutive subframe of residual having the largest weighted energy are identified. Within these two subframes, the *start state (segment)* is selected as either the first $S$ samples or the last $S$ samples of the two consecutive subframes, depending on which segment has a higher energy. The integer $S$ is 57 or 58 samples, depending on whether it is the 20 ms version of iLBC or the 30 ms version. The adaptively selected start state is encoded with scalar quantization without using information in the previous frames.

A dynamic codebook encoding procedure first encodes the remaining samples in the two subframes containing the start state. Next, it encodes the remaining subframes forward in time, and then it encodes the remaining subframes backward in time. The maximum-energy selection criterion allows iLBC to capture the pitch epoch of a pitch cycle waveform or the onset of voiced segments as the start state. The forward–backward approach mentioned above then allows an adaptive codebook to be constructed and used in the entire current frame without using the adaptive codebook in the previous frames.

There are further details in the codebook search method and how iLBC encodes the codebook gains and re-scale the gain for power matching after encoding. Interested readers are referred to [17.32].

It should be noted that even though iLBC performs frame-independent coding of the adaptive codebook, the iLBC coder is not completely frame independent. At least the short-term synthesis filter memory from the previous frame is still used when starting the encoding and decoding operation in the current frame. This means that iLBC still has a slight error propagation from one frame to the next, although the degree of error propagation is significantly smaller than that of the other conventional CELP coders.



## 17.16 TSNFC

Two-stage noise feedback coding (TSNFC) [17.72] is a relatively new class of analysis-by-synthesis coders although it builds on the decade-old technique of noise feedback coding (NFC) [17.16, 73]. The ANSI Amer-

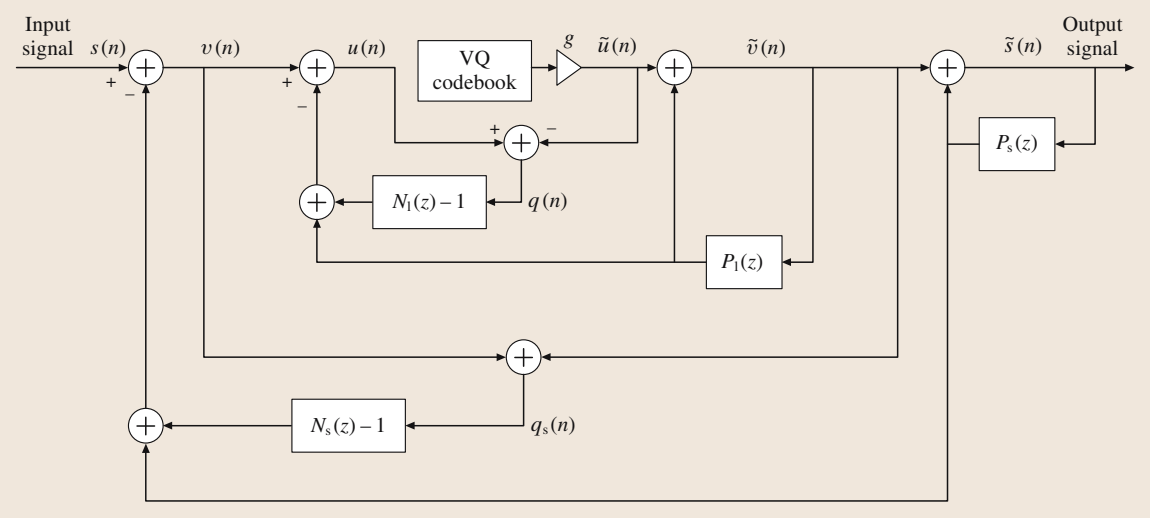



**Fig. 17.12** TSNFC encoder structure

ican national standard BV16 coder [17.33] and the similar wide-band version BV32 are based on TSNFC. While the original NFC uses sample based quantization of the excitation signal and has less in common with analysis-by-synthesis, the recent extension of NFC to TSNFC and the development of CELP-like VQ techniques [17.72, 74] for vector quantization of the excitation signal have created a commonality between TSNFC and CELP, making a brief overview in the context of analysis-by-synthesis relevant.

One structure for TSNFC is shown in Fig. 17.12. It clearly looks nothing like a CELP coder, compare to Fig. 17.5. The short-term predictor, $P_s(z)$, is equivalent to the short-term predictor of CELP coders and results in a short-term prediction error filter given by

$$A(z) = 1 - P_s(z)\,. \tag{17.36}$$

Similarly, the long-term predictor $P_l(z)$ corresponds to the pitch predictor of CELP (or the adaptive codebook). Before TSNFC is described in more detail another interesting observation in relation to analysis-by-synthesis can be made. Studying the TSNFC encoder structure in Fig. 17.12 one can observe the decoder output speech, $\tilde{s}(n)$, being generated as part of the quantization process of the excitation. Conceptually, every codevector of the VQ codebook is fed through the structure in Fig. 17.12 and the candidate minimizing the MSE of $q(n)$ is selected as the output of the VQ. With a short-term noise feedback filter given by $N_s(z) - 1$, minimizing the MSE of $q(n)$ will result in a spectral envelope of the quantization noise, $e(n) = \tilde{s}(n) - s(n)$ given by $N_s(z)$. Hence, the spectral envelope of the coding noise can be controlled directly by $N_s(z)$. This has a parallel to the CELP coder where the spectral envelope of the coding noise is controlled by the inverse of the perceptual weighting filter. Hence, a CELP coder with a perceptual weighting filter of

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}\,, \quad 0 < \gamma_2 < \gamma_1 < 1\,, \tag{17.37}$$

and a TSNFC coder with a short-term noise feedback filter of

$$N_s(z) - 1 = \frac{A(z/\gamma_2)}{A(z/\gamma_1)} - 1\,, \quad 0 < \gamma_2 < \gamma_1 < 1 \tag{17.38}$$

will both result in a coding noise with a spectral envelope shaped according to

$$N_s(z) = \frac{A(z/\gamma_2)}{A(z/\gamma_1)}\,. \tag{17.39}$$

Similarly, the TSNFC structure in Fig. 17.12 will result in a coding noise with spectral fine structure shaped according to $N_l(z)$. A suitable choice according to [17.33] is

$$N_l(z) = 1 + \beta z^{-T}\,, \tag{17.40}$$

where $\beta$ is the long-term noise feedback filter coefficient and $T$ is the pitch period. Typically, $\beta$ is adaptively controlled by the pitch analysis, and $0 \le \beta \le 1$.

Derivation of short-term predictor coefficients and quantization thereof can be carried out with appropriate techniques from the literature, similarly with the estimation and quantization of the long-term predictor parameters. The methods and structures for VQ of the excitation and the similarities to analysis-by-synthesis in CELP coders will be discussed in more detail in the following.

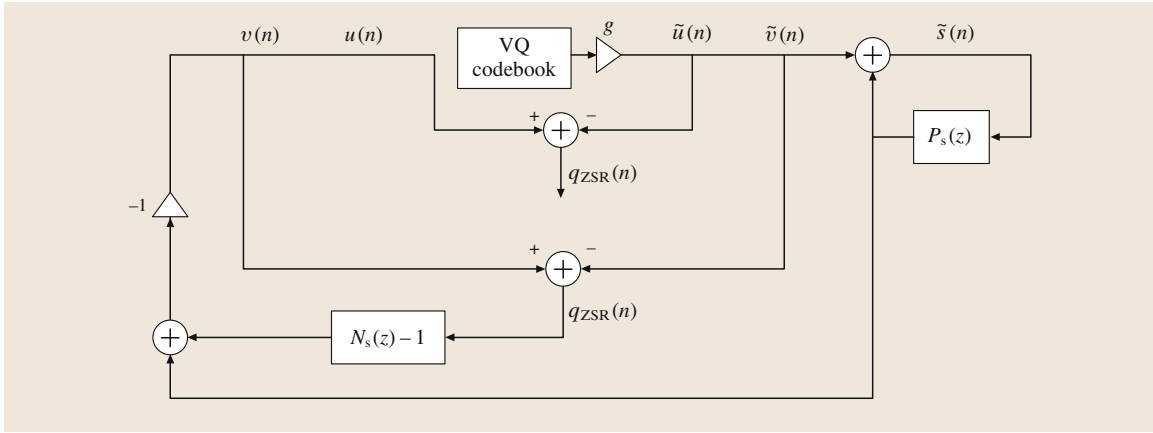



**Fig. 17.13** TSNFC ZSR structure



Amazon EX1010 Page 409

### 17.16.1 Excitation VQ in TSNFC

Similar to CELP it is advantageous to apply the superposition principle when calculating the error signal, $q(n)$, for the excitation VQ. It is calculated as the superposition of the zero state response (ZSR), $q_{\mathrm{ZSR}}(n)$, and the zero-input response (ZIR), $q_{\mathrm{ZIR}}(n)$ [17.72]. With a minimum pitch lag being greater than the dimension of the excitation VQ, the structure in Fig. 17.12 reduces to the ZSR structure in Fig. 17.13. This structure is simplified as shown in Fig. 17.14 according to [17.74], where $H(z)$ with the short-term noise feedback filter of (17.38) is given by

$$H(z) = \frac{Q_{\mathrm{ZSR}}(z)}{\tilde{U}(z)} = -\frac{1}{N_{\mathrm{s}}(z)A(z)} = -\frac{A(z/\gamma_1)}{A(z/\gamma_2)A(z)} \,. \tag{17.41}$$

The ZIR structure is derived from Fig. 17.12 as shown in Fig. 17.15. Typically in TSNFC, a relatively short dimension of the excitation VQ is used. In BV16 [17.33] a 4-dimensional excitation VQ is used. The frame size is 40 samples (5 ms) during which $A(z)$ remains constant. Hence, the codebook needs only to be filtered with $H(z)$ according to Fig. 17.14 once per frame, or equivalently, once per 10 codebook searches. In other words, the filtered codebook is generated only once per 10 codebook searches. The filtered codebook vectors are denoted $q_{\mathrm{ZSR}}(k, n)$, where $k$ indicates the codebook index, $k = 0, 1, \ldots, K-1$. On the other hand, the ZIR, $q_{\mathrm{ZIR}}(n)$ needs to be generated for every four-sample excitation vector. However, it is independent of the codebook vectors. Hence, on a frame basis:

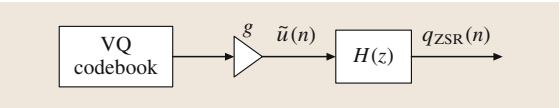



**Fig. 17.14** Simplified TSNFC ZSR structure

- the filtered codebook vectors need to be generated once,
- the 10 ZIR vectors needs to be generated, once for each four sample excitation vector,
- and after each of the 10 codebook searches and before the next ZIR calculation the filter memories need to be updated.

Efficient methods for updating the filter memory is presented in [17.74]. The codebook search is carried out efficiently according to

$$k_{\mathrm{opt}} = \arg\min_k \left\{ \sum_{n=0}^{N-1} [q_{\mathrm{ZIR}}(n) + q_{\mathrm{ZSR}}(k, n)]^2 \right\} \,, \tag{17.42}$$

where $N = 4$ in the example above. With a signed codebook as in BV16, and using vector form notation, this reduces to

$$k_{\mathrm{opt}} = \arg\min_k \left\{ E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right] \overset{s}{\pm} R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right] \right\} \,, \tag{17.43}$$

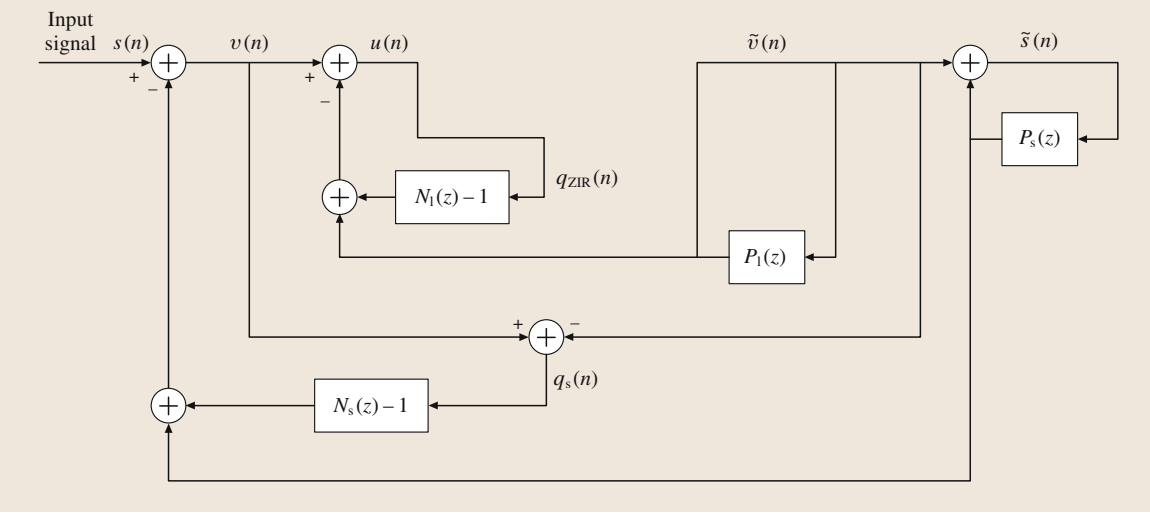



**Fig. 17.15** TSNFC ZIR structure

**Table 17.9** MOS of BV16

| Coder | MOS |
|---|---|
| G.711 | 3.91 |
| G.726 | 3.56 |
| G.728 | 3.54 |
| G.729 | 3.56 |
| BV16 | 3.76 |

**Table 17.10** MOS of BV32

| Coder | MOS |
|---|---|
| G.722 @ 48 kb/s | 3.60 |
| G.722 @ 56 kb/s | 3.88 |
| G.722 @ 64 kb/s | 3.96 |
| BV32 | 4.11 |

where $s$ indicates the sign, and

$$E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right] = \sum_{n=0}^{N-1} q_{\mathrm{ZSR}}^2(k,n)$$

$$R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right] = \sum_{n=0}^{N-1} q_{\mathrm{ZSR}}(k,n) \cdot q_{\mathrm{ZIR}}(n)\,.$$

Note that the optimal sign is given as the opposite sign of the correlation, and hence, effectively, only half the codevectors need to be searched. Similarly, with a signed codebook only half the codevectors need to be filtered with $H(z)$ as the other half of filtered codevectors are given by simple negation. In (17.43), $E\left[\boldsymbol{q}_{\mathrm{ZSR}}(k)\right]$ can be pre-computed before the first of the 10 excitation vectors is quantized. Hence, the only search-loop computation ends up being the calculation of $R\left[\boldsymbol{q}_{\mathrm{ZSR}}(k), \boldsymbol{q}_{\mathrm{ZIR}}\right]$. These techniques are very similar to techniques used in CELP coding, but now applied to the TSNFC structure.

### 17.16.2 TSNFC in Standards

The PacketCable, SCTE, and ANSI BV16 standard for voice over cable is based on TSNFC. BV16 is a narrow-band coder and has an algorithmic delay of 5 ms, a bit rate of 16 kbit/s, a complexity comparable to G.729A (significantly lower than G.729 and G.728). Results from a formal subjective test of BV16 are summarized in Table 17.9. CableLabs has also included BV32 [17.30], a wide-band version of BV16, in PacketCable 2.0 codec and media specification [17.75]. BV32 also has an algorithmic delay of 5 ms, a bit-rate of 32 kbit/s, a complexity comparable to G.729 (significantly lower than G.722.2/AMR-WB), and a subjective quality better than 64 kbit/s G.722. Formal subjective test results of BV32 are summarized in Table 17.10.

## 17.17 Embedded CELP

The recently standardized ITU-T G.729.1 coder [17.76] is a bit-rate- and bandwidth-scalable embedded coder based on G.729. The 16 kHz sampled wide-band input speech is split into equal-bandwidth high and low bands with a quadrature mirror filterbank (QMF). The low-band (narrow-band) signal is encoded with the narrow-band core (also referred as Layer 1) layer. Although the frame size of G.729.1 is 20 ms as opposed to the 10 ms frame size of G.729, the core layer is bitstream compatible with G.729. However, G.729.1 has a significantly longer algorithmic delay. The algorithmic delay of G.729 is 15 ms while the algorithmic delay of G.729.1 is 48.9375 ms. Layer 2 is a 4 kb/s narrow-band embedded enhancement layer to G.729. Layers 3–12 provide wide-band (50–7000 Hz) capability at a total bit rate increasing from 14 kbit/s to a maximum of 32 kb/s at increments of 2 kb/s per layer. An overview of the encoder is presented in Fig. 17.16. Layer 3 encodes the high-band signal using time-domain bandwidth extension (TDBWE) [17.76]. Layers 4–12 encode the weighted error signal from the narrow-band layer 1 and 2 embedded CELP coder, 50–4000 Hz, jointly with the high-band signal, 4000–7000 Hz. A transform predictive coder (TPC) with time-domain aliasing cancelation (TDAC) is utilized for layers 4–12. The coder sends redundant information to the decoder in order to mitigate

**Table 17.11** Overview of G.729.1 embedded CELP

| Layer | Technology | Analysis by synthesis | Bandwidth | Cummulative bit rate |
|---|---|---|---|---|
| 1 | ACELP | Yes | 50–4000 Hz | 8 kb/s |
| 2 | ACELP | Yes | 50–4000 Hz | 12 kb/s |
| 3 | TDBWE | No | 4000–7000 Hz | 14 kb/s |
| 4–12 | TPC | No | 50–7000 Hz | 16–32 kb/s |



Amazon EX1010 Page 411

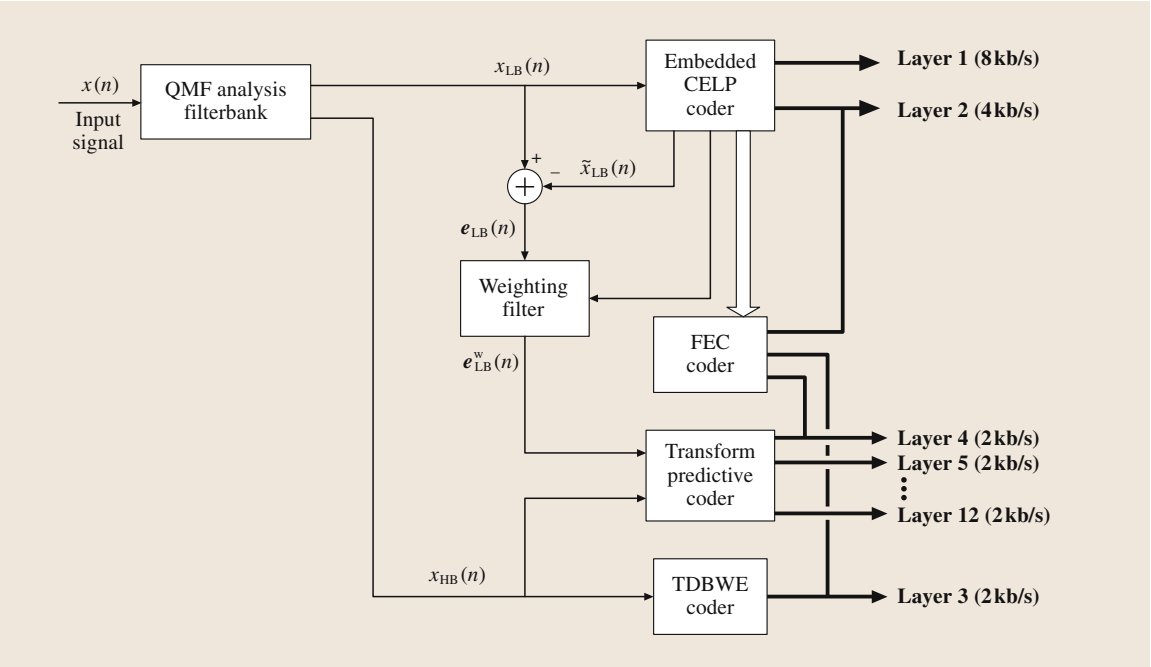



**Fig. 17.16** High-level block diagram of G.729.1

the effect of frame loss. This information is calculated with the frame erasure concealment (FEC) encoder and comprises information pertaining to the narrow-band embedded CELP coder in order to speed up recovery of the CELP decoder after frame loss. The information includes

- signal classification information
- estimated position of the last glottal pulse
- energy calculated as either maximum or average sample energy (dependent on signal classification)

Table 17.12 elaborates the FEC encoded information with bit-allocation and information regarding which layer includes the additional information. It should be noted that the FEC information is only available if the layers listed in Table 17.12 are actually received by the decoder.

**Table 17.12** Redundant information included in G.729.1 bitstream to help embedded CELP coder recover from frame loss

| Information | Location | Bit allocation |
|---|---|---|
| Classification | Layer 2 | 2 bits |
| Glottal pulse location | Layer 3 | 7 bits |
| Energy | Layer 4 | 5 bits |

Since the TDBWE and transform predictive coders are not analysis-by-synthesis coders, they will not be treated in any greater details here. Instead, the reader is referred to [17.76] for further information.

The embedded CELP coder is, however, of interest as not only the core layer, but also the enhancement layer, is derived according to analysis-by-synthesis and presents a particular problem. In the following, the terms encoder and decoder are used to refer to the embedded CELP only encoder and decoder portions of G.729.1. The particular problem pertains to the issue of ensuring that the encoder and decoder remain synchronized regardless of the number of decoded layers. Consider the fundamental meaning of the term analysis-by-synthesis. Regardless of the number of decoded layers the coder must remain an analysis-by-synthesis coder:

*At the encoder, without knowing how many layers the decoder will decode, how is analysis-by-synthesis maintained?*

This appears to be an oxymoron, but the key is to realize that the encoder has multiple layers of encoding, and then structure the algorithm such that the various levels of encoding remain synchronized with the various levels of decoding. This may sound simple. However, looking at an example, it quickly becomes clear that it is not quite that simple. Let us assume that



Amazon EX1010 Page 412

a CELP enhancement layer includes a fixed codebook. The straightforward way would be to update the adaptive codebook at the encoder with a short-term excitation signal including this enhancement layer contribution. This would be optimal in the sense of the output speech quality including the enhancement layer. However, at the decoder this would present a problem if the enhancement layer is not received as the adaptive codebook would then no longer be synchronized with the encoder. The analysis-by-synthesis property would be violated: in retrospect the encoder did not do the same synthesis part in the analysis-by-synthesis process as the decoder. Once the analysis-by-synthesis property is violated the speech quality is somewhat up in the air, and the waveform matching during analysis-by-synthesis is somewhat senseless as the decoder will not repeat the same synthesis.

Not only does the example illustrate the particular issue of embedded analysis-by-synthesis, it also demonstrates one reason why an embedded analysis-by-synthesis coder would be of inferior speech quality to a nonembedded analysis-by-synthesis coder: the obvious optimal approach, in terms of optimizing the speech quality of the output including the enhancement layer, had to be disregarded in order not to violate the analysis-by-synthesis property.

To maintain the analysis-by-synthesis property for embedded analysis-by-synthesis, the memory of one layer cannot be updated with any information of a higher layer. In the example above, with an enhancement layer containing an additional fixed codebook, it means that the adaptive codebook of the lower layer (the core layer) can only be updated with the short-term synthesis filter excitation without the fixed codebook contribution of the enhancement layer. Furthermore, at the encoder, separate short-term synthesis filter memory may need to be maintained. Another example, albeit perhaps not truly analysis-by-synthesis, is the ADPCM encoder of G.726 [17.77] and the low-band ADPCM encoder of G.722 [17.78]. Both are embedded backward adaptive predictive coders. In G.722 the low-band prediction error signal is quantized sample by sample with six bits. However, the low-band ADPCM decoder can decode based on four, five, or six bits per sample. In order for the backward adaptive entities of the low-band ADPCM to stay synchronized between encoder and decoder at all times, the encoder and decoder only use the four-bit information for updating the backward adaptive entities of the low-band ADPCM.



## 17.18 Summary of Analysis-by-Synthesis Speech Coders

Table 17.13 lists and compares the properties of the various analysis-by-synthesis techniques and coders. Each of the Sects. 17.4 through 17.17, with detailed discussions of the various analysis-by-synthesis techniques, summarize which standards utilize the techniques. This section will view it from a different angle, and for individual speech coding standards list the applicable techniques and features. Note that multiple techniques apply to some of the standards. Furthermore, key characteristics of the coders are provided as well in order to put the techniques into context and provide the reader with a general overview of properties of speech coding standards. It should be noted that the scope was limited to analysis-by-synthesis coders, and that space limitations prevent inclusion of every standard speech coder. Note that ACB for 'pitch type' indicates the use of the adaptive codebook approach to implement the pitch prediction, while 'filter' indicates implementation as a regular filter. The term 'frac. lag' indicates usage of fractional pitch lag. Furthermore, 'excitation' refers to the fixed codebook (or innovation) excitation, and hence, excludes the pitch related part of the excitation. The bit rate for the 'excitation' in Table 17.13 also excludes the gain for the fixed codebook excitation. The column 'year standardized' should only be considered approximate, as there is some variation in the years listed in the literature due to a common delay between the initial selection of a speech coding algorithm, and the date it is officially standardized. Furthermore, the 'complexity' should only be considered a rough guideline as great variations for a given algorithm exist. Such variation can be due to many factors, e.g., capability of target processor, level of optimization, etc. Also, it should be noted that the unit for the complexity is either WMOPS or MIPS. The WMOPS number is typically obtained from a simulation software in C code with operators with weights emulating the complexity of each operator on a typical digital signal processor (DSP). On the other hand, the MIPS number is typically obtained from an actual implementation on a specific DSP. Although the purpose of the WMOPS number is to estimate the complexity on a DSP (the MIPS number), it does not always provide an exact number.

Amazon EX1010 Page 413

**Table 17.13** Overview of analysis-by-synthesis speech coding standards

| Coder | Year standardized | Standards organization | Bit rate | Audio bandwidth | Algorithmic delay | Complexity | Excitation type | Excitation dimension | Excitation bit rate | Pitch type | Other AbS techniques |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GSM-FR | 1986 | ETSI | 13 kb/s | 0–4 kHz | 20 ms | 4.5 MIPS | RPE | 40 | 8.2 kb/s | Filter | |
| FS-1016 | 1989 | DoD | 4.8 kb/s | 0–4 kHz | 30 ms | 25 MIPS | Overlap sparse | 60 | 1.33 kb/s | ACB | |
| IS-54 | 1989 | TIA | 8 kb/s | 0–4 kHz | 28.125 ms | 25 MIPS | VSELP | 40 | 2.8 kb/s | ACB | |
| G.728 | 1992 | ITU-T | 16 kb/s | 0–4 kHz | 0.625 ms | 36 MIPS | Trained | 5 | 11.2 kb/s | None | LD-CELP |
| GSM-HR | 1993 | ETSI | 5.6 kb/s | 0–4 kHz | 25 ms | 25 MIPS | VSELP | 40 | 1.8 kb/s<br>2.8 kb/s | ACB fraction lag | |
| PDC-HR | 1993 | ARIB | 3.45 kb/s | 0–4 kHz | 45 ms | 18.7 MOPS | PSI | 80 | | ACB fraction lag | |
| EVRC | 1994 | TIA | 8.55 kb/s<br>4.0 kb/s | 0–4 kHz | 33 ms | 25 MIPS | ACELP | 53/43 | 5.25 kb/s<br>1.5 kb/s | ACB | RCELP |
| G.723.1 | 1995 | ITU-T | 6.3 kb/s<br>5.3 kb/s | 0–4 kHz | 37.5 ms | 19 MIPS | MP ACELP | 60 | 3.3 kb/s<br>2.27 kb/s | ACB 5-tap | |
| G.729 | 1996 | ITU-T | 8 kb/s | 0–4 kHz | 15 ms | 22 MIPS | ACELP | 40 | 3.4 kb/s | ACB fraction lag | CS-ACELP |
| TIA-EFR | 1996 | TIA | 7.4 kb/s | 0–4 kHz | 25 ms | 14 WMOPS | ACELP | 40 | 3.4 kb/s | ACB fraction lag | |
| PDC-EFR | 1996 | ARIB | 6.7 kb/s | 0–4 kHz | | | ACELP | 40 | | | |
| GSM-EFR | 1997 | ETSI | 12.2 kb/s | 0–4 kHz | 20 ms | 18 MIPS | ACELP | 40 | 7 kb/s | ACB fraction lag | |
| GSM-AMR | 1999 | ETSI | 4.75 kb/s to 12.2 kb/s | 0–4 kHz | 25 ms | 20 MIPS | ACELP | 40 | 1.8 kb/s to 7 kb/s | ACB fraction lag | |
| SMV | 2001 | 3GPP2 | 8.55 kb/s<br>4.0 kb/s | 0–4 kHz | 30.5–35.5 ms<br>27.5–32.5 ms | 40 WMOPS | Sub ACELP | 40<br>80, 53/54 | 4.4, 6 kb/s<br>1.5, 1.95 kb/s | ACB | eX-CELP<br>RCELP |
| AMR-WB<br>G.722.2 | 2001 | 3GPP<br>ITU-T | 6.6 kb/s to 23.85 kb/s | 0–8 kHz | 26 ms | 38 WMOPS | ACELP | 64 | 2.4–17.6 kb/s | ACB fraction lag | |
| VMR-WB | 2002 | 3GPP2 | 6.6 kb/s<br>8.85 kb/s<br>12.65 kb/s | 0–8 kHz | 33.75 ms | 38 WMOPS | ACELP | 64 | 2.4–7.2 kb/s | ACB fraction lag | RCELP |
| iLBC | 2002 | IETF<br>PacketCable | 13.3 kb/s<br>15.2 kb/s | 0–4 kHz | 40 ms<br>25 ms | 18 MIPS<br>15 MIPS | FB-LPC | 40 | 12.0 kb/s<br>14.2 kb/s | FB-ACB | |
| BV16 | 2003 | SCTE<br>ANSI | 16 kb/s | 0–4 kHz | 5 ms | 12 MIPS | Trained | 4 | 10 kb/s | Filter 3-tap | TSNFC |
| G.729.1 | 2006 | ITU-T | 8 kb/s to 32 kb/s | 0–4 kHz<br>0–8 kHz | 48.9375 ms | 36 WMOPS | ACELP | 40 | 3.4 kb/s+<br>3.4 kb/s | ACB fraction lag | Embedded CS-CELP |



Amazon EX1010 Page 414

## 17.19 Conclusion

The analysis-by-synthesis method for coding the excitation signal of linear predictive coders has been the most important driving force behind the relentless reduction of the bit-rate for high-quality speech coding in the last two decades. Almost all speech coding standards established after 1989 are based on it. This chapter gives a tutorial on analysis-by-synthesis speech coding techniques in general and describes many variations of the analysis-by-synthesis excitation coding paradigm in particular. Due to space limitation, the description concentrates on dominant types of analysis-by-synthesis excitation coding techniques as exemplified by various speech coding standards, with the relationship between them discussed in the context of a family tree. It is hoped that, after reading this chapter, the reader will have a good understanding of the dominant types of analysis-by-synthesis speech coding techniques.



### References


17.1 R.V. Cox: Speech coding standards. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier Science, Amsterdam 1995)

17.2 B.S. Atal, J.R. Remde: A new model of LPC excitation for producing natural-sounding speech at low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1982) pp. 614–617

17.3 B.S. Atal, M.R. Schroeder: Stochastic coding of speech signals at very low bit rates, Proc. IEEE Int. Conf. Commun. (1984) p. 48.1

17.4 M.R. Schroeder, B.S. Atal: Code-excited linear prediction (CELP): high quality speech at very low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 937–940

17.5 J.-H. Chen: A robust low-delay CELP speech coder at 16 kbit/s, Proc. IEEE Global Commun. Conf. (1989) pp. 1237–1241

17.6 J.-H. Chen, R.V. Cox, Y.-C. Lin, N.S. Jayant, M.J. Melchner: A low-delay CELP coder for the CCITT 16 kb/s speech coding standard, IEEE J. Sel. Areas Commun. **10**(5), 830–849 (1992)

17.7 R. Salami, C. Laflamme, J.-P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, Y. Shoham: Design and description of CS-ACELP: a toll quality 8 kb/s speech coder, IEEE Trans. Speech Audio Process. **6**(2), 116–130 (1998)

17.8 J. Thyssen, Y. Gao, A. Benyassine, E. Shlomot, C. Murgia, H.-Y. Su, K. Mano, Y. Hiwasaki, H. Ehara, K. Yasunaga, C. Lamblin, B. Kovesi, J. Stegmann, H.-G. Kang: A candidate for the ITU-T 4 kbit/s speech coding standard, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 681–684

17.9 A. McCree, J. Stachurski, T. Unno, E. Ertan, E. Paksoy, R. Viswanathan, A. Heikkinen, A. Ramo, S. Himanen, P. Blocher, O. Dressler: A 4 kb/s hybrid MELP/CELP speech coding candidate for ITU standardization, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2002) pp. 629–632

17.10 P. Kroon, W.B. Kleijn: Linear-prediction based analysis-by-synthesis coding. In: *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier Science, Amsterdam 1995)

17.11 M. Halle, K.N. Stevens: Analysis by synthesis, Proc. Sem. Speech Compression and Process., Vol. II, ed. by W. Wathen-Dunn, L.E. Woods (1959), AFCRC-TR-59-198, Paper D7

17.12 C.G. Bell, H. Fujisaki, J.M. Heinz, K.N. Stevens, A.S. House: Reduction of speech spectra by analysis-by-synthesis techniques, J. Acoust. Soc. Am. **33**(12), 1725–1736 (1961)

17.13 N.S. Jayant, P. Noll: *Digital Coding of Waveforms* (Prentice Hall, Englewood Cliffs 1984)

17.14 L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals* (Prentice Hall, Englewood Cliffs 1978)

17.15 W.B. Kleijn, R.P. Ramachandran, P. Kroon: Generalized analysis-by-synthesis coding and its application to pitch prediction, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1992) pp. I–337–I–340

17.16 B.S. Atal, M.R. Schroeder: Predictive coding of speech signals and subjective error criteria, IEEE Trans. Acoust. Speech Signal Process. **3**, 247–254 (1979)

17.17 I.A. Gerson, M.A. Jasiuk: Vector sum excited linear prediction (VSELP) speech coding at 8 kbps, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 461–464

17.18 P. Vary, K. Hellwig, R. Hofmann, R.J. Sluyter, C. Galand, M. Rosso: Speech codec for the European mobile radio system, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1988) pp. 227–230

17.19 S. Singhal, B. Atal: Improving performance of multi-pulse LPC coders at low bit rates, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1984) pp. 9–12

17.20 J.P. Campbell Jr., V.C. Welch, T.E. Tremain: An expandable error-protected 4800 bps CELP coder


Amazon EX1010 Page 415

(U.S. Federal standard 4800 bps voice coder), Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 735–738
17.21 I.A. Gerson, M.A. Jasiuk: A 5600 bps VSELP speech coder candidate for half-rate GSM, Proc. 1993 IEEE Workshop Speech Coding for Telecommunications (1993) pp. 43–44
17.22 T. Ohya, H. Suda, T. Miki: 5.6 kbits/s PSI-CELP of the half-rate PDC speech coding standard, Proc. 1994 IEEE Vehicular Technol. Conf. (1994) pp. 1680–1684
17.23 J.-P. Adoul, P. Mabilleau, M. Delprat, S. Morisette: Fast CELP coding based on algebraic codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1957–1960
17.24 C. Laflamme, J.-P. Adoul, H.Y. Su, S. Morisette: On reducing computational complexity of codebook search in CELP coder through the use of algebraic codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 177–180
17.25 ITU-T: G.723.1: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s (1996)
17.26 ANSI/TIA-127-A-2004: Enhanced variable rate codec speech service option 3 for wideband spread spectrum digital systems (2004) (ANSI/TIA-127-A-2004)
17.27 Y. Gao, E. Shlomot, A. Benyassine, J. Thyssen, H. Su: The SMV algorithm selected by TIA and 3GPP2 for CDMA applications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 709–712
17.28 3GPP2 C.S0052-0 V1.0: Source-controlled variable-rate multimode wideband speech codec (VMR-WB) service option 62 for spread spectrum systems, (June 11 2004)
17.29 J.-H. Chen, J. Thyssen: BroadVoice16: A PacketCable speech coding standard for cable telephony, Proc. 40th Asilomar Conf. Signals Systems and Computers (2006)
17.30 J.-H. Chen, J. Thyssen: The BroadVoice speech coding algorithm, Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Vol. 4 (2007) p. IV-549-IV-552
17.31 S.V. Andersen, W.B. Kleijn, R. Hagen, J. Linden, M.N. Murthi, J. Skoglund: iLBC – a linear predictive coder with robustness to packet losses, Proc. IEEE Workshop Speech Coding (2002) pp. 23–25
17.32 S.V. Andersen, A. Duric, H. Astrom, R. Hagen, W.B. Kleijn, J. Linden: Internet low bit rate codec (iLBC), IETF RFC **3951** (2004)
17.33 American National Standard: BV16 speech codec specification for voice over ip applications in cable telephony, ANSI/SCTE 24-21 2006 (2006)
17.34 B.S. Atal, S.L. Hanauer: Speech analysis and synthesis by linear prediction, J. Acoust. Soc. Am. **50**, 637–655 (1971)
17.35 E.F. Deprettere, P. Kroon: Regular excitation reduction for effective and efficient LP-coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 965–968
17.36 W.B. Kleijn, D.J. Krasinski, R.H. Ketchum: Improved speech quality and efficient vector quantization in SELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1988) pp. 155–158
17.37 I.M. Trancoso, B.S. Atal: Efficient procedures for finding the optimum innovation in stochastic coders, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1986) pp. 2375–2378
17.38 G. Davidson, A. Gersho: Complexity reduction methods for vector excitaiton coding, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1986) pp. 3055–3058
17.39 D. Lin: Speech coding using pseudo-stochastic block codes, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1354–1357
17.40 I.A. Gerson, M.A. Jasiuk: Vector sum excited linear prediction (VSELP), Proc. 1989 IEEE Workshop Speech Coding for Telecommunications (1989) pp. 66–68
17.41 P. Kroon, B.S. Atal: Pitch predictors with high temporal resolution, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 661–664
17.42 J.S. Marques, I.M. Trancoso, J.M. Tribolet, L.B. Almeida: Improved pitch prediction with fractional delays in CELP coding, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1990) pp. 665–668
17.43 J.H. Chen, M.S. Rauchwerk: An 8 kb/s low-delay CELP speech coder, Proc. IEEE Global Commun. Conf. (1991) pp. 1894–1898
17.44 S. Miki, K. Mano, H. Ohmuro, T. Moriya: Pitch synchronous innovation CELP (PSI-CELP), Proc. 1993 Eurospeech Conf. (1993) pp. 261–264
17.45 T. Moriya: Two-channel conjugate vector quantization for noisy channel speech coding, IEEE J. Sel. Areas Commun. **10**(5), 866–874 (1992)
17.46 ITU-T: G.729: coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP) (1996)
17.47 J.-P. Adoul, C. Lamblin: A comparison of some algebraic structures for CELP coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1987) pp. 1953–1956
17.48 R.A. Salami: Binary code excited linear prediction (BCELP): a new approach to CELP coding of speech without the codebooks, IEEE Electron. Lett. **25**(6), 401–403 (1989)
17.49 A. Le Guyader, D. Massaloux, J.P. Petit: Robust and fast code-excited linear predictive coding of speech signals, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 120–122
17.50 J.-P. Adoul, C. Lamblin, A. Leguyader: Baseband speech coding at 2400 BPS using spherical vector quantization, IEEE Int. Conf. Commun. (1984) pp. 1.12.1–1.12.4
17.51 C. Laflamme, J.-P. Adoul, S. Morisette: A real time 4.8 Kbits/sec CELP on a single DSP chip (TMS320C25), IEEE Workshop Speech Coding for Telecommun. (1989) pp. 35–36



Amazon EX1010 Page 416

17.52 R.A. Salami, D.G. Appleby: A new approach to low bit rate speech coding with low complexity using binary pulse excitation (BPE), IEEE Workshop Speech Coding for Telecommun. (1989) pp. 63–65
17.53 C. Laflamme, J.-P. Adoul, R. Salami, S. Morisette, P. Mabilleau: 16 kbps wideband speech coding technique based on algebraic CELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1991) pp. 13–16
17.54 R. Salami, C. Laflamme, J.-P. Adoul, D. Massaloux: A toll quality 8 kb/s speech codec for the personal communications system (PCS), IEEE Trans. Vehicular Technol. **43**(3), 808–816 (1994)
17.55 K. Järvinen, J. Vaino, P. Kapanen, T. Honkanen, P. Haavisto, R. Salami, C. Laflamme, J.-P. Adoul: GSM enhanced full rate speech codec, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1997) pp. 771–774
17.56 ETSI EN 300 726 V8.0.1: Digital cellular telecommunications systems (Phase 2+); enhanced full rate (EFR) speech transcoding; (GSM 06.60 version 8.0.1 Release 1999) (2000)
17.57 3GPP TS 26.190 V6.1.1: 3rd generation partnership project; technical specification group services and system aspects; speech codec speech processing functions; adaptive multi-rate – wideband (AMR-WB) speech codec; transcoding functions (release 6) (2005)
17.58 R. Salami, C. Laflamme, J.P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, Y. Shoham: Design and description of CS-ACELP: a toll quality 8 kb/s speech coder, IEEE Trans. Speech Audio Process. **6**(2), 116–130 (1998)
17.59 ITU-T: G.729 Annex A: reduced complexity 8 kbit/s CS-ACELP speech codec (1996)
17.60 R. Salami, C. Laflamme, B. Bessette, J.P. Adoul: ITU-T G.729 Annex A: reduced complexity 8 kb/s CS-ACELP Codec for simultaneous vocie and data, IEEE Commun. Mag. **35**, 56–63 (1997)
17.61 ITU-T: G.722.2: wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB) (2002)
17.62 3GPP TS 26.090 V6.0.0: 3rd generation partnership project; technical specification group services and system aspects; mandatory speech codec speech processing functions; adaptive multi-rate (AMR) speech codec; transcoding functions (release 6) (2004)
17.63 ANSI/TIA/EIA-136-410-99: TDMA cellular PCS – radio interface enhanced full-rate voice codec (ANSI/TIA/EIA-136-410-99) (R2003) (1999)
17.64 3GPP2 C.S0030-0 V1.0: Selectable mode vocoder service option for wideband spread spectrum communication systems, (June 15 2001)
17.65 ISO/IEC 14496-3 FCD, ISO/JTC 1/SC 29 N2203CELP: Information technology – coding of audiovisual objects, Part 3: audio, subpart 3: CELP, (May 13 1998)
17.66 A. Kataoka, T. Moriya, S. Hayashi: An 8-kbit/s speech coder based on conjugate structure CELP, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1993) pp. II–592–II–595
17.67 T. Moriya, H. Suda: An 8 kbit/s transform coder for noisy channels, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) pp. 196–199
17.68 A. Kataoka, T. Moriya, S. Hayashi: An 8-kb/s conjugate structure CELP (CS-CELP) speech coder, IEEE Trans. Speech Audio Process. **4**(6), 401–411 (1996)
17.69 W.B. Kleijn, P. Kroon, L. Cellario, D. Sereno: A 5.85 kb/s CELP algorithm for cellular applications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1993) pp. II–596–II–599
17.70 W.B. Kleijn, R.P. Ramachandran, P. Kroon: Interpolation of the pitch-predictor parameters in analysis-by-synthesis speech coders, IEEE Trans. Speech Audio Process. **2**(1), 42–53 (1994)
17.71 Y. Gao, A. Benyassine, J. Thyssen, H. Su, E. Shlomot: A speech coding paradigm, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2001) pp. 689–692
17.72 J.-H. Chen: Novel codec structures for noise feedback coding of speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (2006) pp. I.681–I.684
17.73 J.D. Makhoul, M. Berouti: Adaptive noise spectral shaping and entropy coding in predictive coding of speech, IEEE Trans. Acoust. Speech Signal Process. **27**, 63–73 (1979)
17.74 J. Thyssen, J.-H. Chen: Efficient VQ techniques and general noise shaping for noise feedback coding, Proc. Interspeech 2006 ICSLP (2006) pp. 221–224
17.75 PacketCable 2.0 codec and media specification, PKT-SP-CODEC-MEDIA-I02-061013 (2006)
17.76 ITU-T: G.729.1: G.729 based embedded variable bitrate coder: An 8-32 kbit/s scalable wideband coder bitstrean interoperable with G.729 (2006)
17.77 ITU-T: G.726: 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM) (1990)
17.78 ITU-T: G.722: 7 kHz audio-coding within 64 kbit/s (1988)



Amazon EX1010 Page 417

# 18. Perceptual Audio Coding of Speech Signals

J. Herre, M. Lutzky

Traditionally algorithms for speech coding exploit the features of speech signals by employing algorithmic models of the human vocal tract. More recently, the use of generic audio coders for coding of speech signals has gained increasing importance. Based on the properties of human hearing, such perceptual audio coders offer attractive properties including full-bandwidth audio output, increased naturalness, and good handling of any type of non-speech material. The chapter discusses the principles of perceptual audio coding, some relevant standards, and a number of perceptual audio coders that find application in speech and audio transmission and storage.

18.1 **History of Audio Coding** ..... 393

18.2 **Fundamentals of Perceptual Audio Coding** ..... 394
18.2.1 General Background ..... 394
18.2.2 Coder Structure ..... 394
18.2.3 Perceptual Audio Coding Versus Speech Coding ..... 395

18.3 **Some Successful Standardized Audio Coders** ..... 396
18.3.1 MPEG-1 ..... 396
18.3.2 MPEG-2 ..... 396
18.3.3 MPEG-2 Advanced Audio Coding ..... 397
18.3.4 MPEG-4 Advanced Audio Coding ..... 397
18.3.5 Progress in Coding Performance ..... 397

18.4 **Perceptual Audio Coding for Real-Time Communication** ..... 398
18.4.1 Delay Sources in Perceptual Audio Coding ..... 398
18.4.2 MPEG-4 Low-Delay AAC ..... 399
18.4.3 ITU-T G.722.1-C ..... 401
18.4.4 Ultra-Low-Delay Perceptual Audio Coding ..... 402

18.5 **Hybrid/Crossover Coders** ..... 403
18.5.1 MPEG-4 Scalable Speech/Audio Coding ..... 404
18.5.2 ITU-T G.729.1 ..... 405
18.5.3 AMR-WB+ ..... 406
18.5.4 ARDOR ..... 408

18.6 **Summary** ..... 409

**References** ..... 409



## 18.1 History of Audio Coding

Historically, source coding of speech was motivated by the human desire for communication with other individuals over distance – commercially speaking: how to get as many speech connections across a given telephone network as possible while maintaining high intelligibility and good subjective voice quality. As a result, the field of speech coding evolved very much with a focus on real-time two-way communication and a sound fidelity which is adapted to the needs of this application area. Accordingly, the technical approach adopted by typical speech coders reflects and exploits the specific process of speech production in the human vocal tract (see Part A of this Handbook) to a large extent.

Much later (around 1980), people started thinking about the efficient transmission and storage of *music* and *generic audio* signals. The fact that music constitutes a much broader class of signals than speech, and does not adhere to a common model of generation, led to very different technical solutions to the audio coding problem as compared to speech coding. Typically, audio coding is thus strongly motivated by the quest to exploit perceptual properties of the final receiver, i. e., the human auditory system, in order to provide maximum (ideally: indistinguishable) subjective quality across a broad range of signals. These ideas are reflected by the concept of *perceptual audio coding* for *general audio signals*. Naturally, perceptual audio coders are also able to code speech signals, and thus serve for a range of speech-related applications. Moreover, as more bandwidth becomes gradually available, using perceptual audio coders for speech coding offers several clear benefits over traditional speech coders,

Amazon EX1010 Page 418

such as the ability to convey the whole audible signal bandwidth, the associated increase in naturalness and intelligibility, the capability to represent also speech with arbitrary background sounds (including music), handling of stereo/multichannel signals etc.

This chapter provides some background about the field of perceptual audio coding and touches upon several relevant coding standards. Subsequently, a number of perceptual audio coders are discussed which find application in speech and audio transmission and storage.

## 18.2 Fundamentals of Perceptual Audio Coding

### 18.2.1 General Background

The basic task of an audio coder can be summarized briefly: the (uncompressed) source audio data should be represented as compactly as possible while maintaining its original sound quality. Ideally, the decompressed audio signal should sound identical to the original one under all circumstances (*perceptual transparency*). To this end, the paradigm of *perceptual audio coding* turns out to be an extremely powerful approach which aims at optimizing the subjectively perceived audio quality rather than other objective distortion metrics, such as mean-squared-error (MSE) and signal-to-noise ratio (SNR). This is achieved by employing knowledge about human perception of sound as represented by the field of psychoacoustics. In particular, the use of this knowledge allows to exploit the limits of perceptibility for coding distortion and achieve transparent coding (if the incurred coding distortion is below the threshold of perceptibility) or at least minimize the subjective disturbance caused by the coding distortion at low bit rates.

As the range of input signals for an audio coder is potentially unlimited (just think of the range of possible acoustic and electronic musical instruments, environmental sounds etc.), no universal source model is available as a basis for the coding process. This situation is very different from, e.g., speech coding, where a well-grounded model of the human vocal tract can be used as a source (production) model of the input signal, and thus facilitates the exploitation of its inherent redundancy. As a consequence, perceptual audio coders have to be extremely flexible so as to cover the wide range of potential input signals. Moreover, they have to rely significantly on the properties of the final signal receiver, i. e., the human auditory system, exploiting signal irrelevance. In particular, the so-called masking effect describes the phenomenon that louder signal components conceal the presence of other, weaker components which are close in frequency or time (see, e.g., [18.1, 2]).

### 18.2.2 Coder Structure

Although there may be many differences in technical details among current audio compression schemes, most common coders lean on the basic paradigm of *filterbank-based audio coding*. This may be explained by the fact that a filterbank-based coder framework combines both the ability of reducing redundancy and exploiting the potential behind the removal of irrelevancy. On the one hand, coding of a spectral representation is an efficient means to take advantage of linear correlation between subsequent samples of an input signal. Redundancy can be further reduced by entropy coding of the spectral coefficients. On the other hand, a spectral representation of the input signal opens the door to a good first-order modeling of the limits of human auditory perception. Once the threshold of perceptibility (audibility) is estimated, the precision of transmission can be easily adapted to match the threshold by adjusting the quantization distortion of the corresponding spectral coefficients.

The upper part of Fig. 18.1 depicts the well-known basic block diagram of a generic monophonic perceptual audio coder which comprises the following components:

- *Analysis Filterbank:* The input signal is mapped to a subsampled spectral representation using various types of analysis filterbanks. For reasons of coding efficiency, modern coding schemes typically employ filterbanks with critical sampling (i. e., same number of input samples and spectral coefficients) and overlapping analysis windows between subsequent analysis frames. Examples include the modified discrete cosine transform (MDCT) [18.3], polyphase filterbanks [18.4], or hybrid structures [18.5]. An account of common filterbanks for perceptual audio coding can be found in [18.6].
- *Perceptual model:* the signal's time- and frequency-dependent threshold of perceptibility (masking threshold) is estimated by a perceptual model. This value describes the maximum quantization error that



Amazon EX1010 Page 419

can be introduced into the audio signal while still maintaining perceptually unimpaired signal quality. Typically, perceptual models for audio coding consider psychoacoustic effects like masking in frequency domain (inter- and intraband masking), temporal masking effects (postmasking) and the asymmetry of masking between tonal and noise-like stimuli [18.7]. In order to match closely the properties of the peripheral human auditory system, masking calculations are usually performed on a nonuniform frequency scale, such as the BARK or ERB scale [18.1, 2].

- *Quantization and coding:* the spectral values are quantized and coded with a precision corresponding to the masking threshold estimate. Both uniform and nonuniform scalar quantization and vector quantization have been used in perceptual audio coding. Additional reduction of redundancy can be achieved by employing entropy coding techniques, such as Huffman coding. Usually, the quantization/coding kernel of an encoder operates under two constraints. While the estimated masking threshold defines a target for the minimum precision of the coded signal representation, coders frequently operate under the limitations of a fixed bit rate selected by the user. As it may not be possible to satisfy both constraints at the same time, a good encoding algorithm has to produce an encoding solution representing an acceptable compromise with respect to perceived audio quality in such cases. Besides the perceptual modeling aspect, these strategies for *bit allocation/noise allocation* are essential parts of an optimized audio coder [18.8].
- *Bitstream encoding:* finally, all relevant information (i. e. the coded spectral values and additional side information) is packed into a bitstream and transmitted to the decoder.

Correspondingly, the processing steps appear in reversed order in the decoder (see the lower part of Fig. 18.1). The bitstream is decoded and parsed into coded spectral data and side information. The inverse quantization of the quantized spectral coefficients is then carried out. Finally, the spectral values are mapped back to a time-domain representation using a synthesis filterbank.

While all common filterbank-based perceptual audio coders are based on this generic structure, practical coders are extended by a number of optional building blocks (*coding tools*) which provide further improvements in coding performance for specific coding constellations. These include provisions for joint coding of stereo signals, enhanced exploitation of redundancy (e.g., by means of predictors) or irrelevance (by enhancing the coder's noise-shaping abilities).

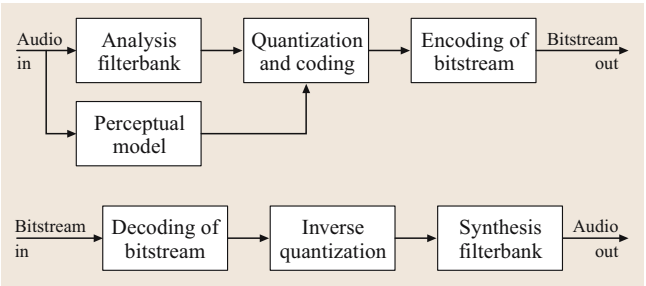



**Fig. 18.1** Generic structure of a monophonic perceptual audio coder/decoder

### 18.2.3 Perceptual Audio Coding Versus Speech Coding

When comparing the concepts behind speech coding and perceptual audio coding, several aspects are to be considered. While efficient speech coding is very much focused on exploiting a model of the signal source (i. e., the human vocal tract), perceptual audio coders can not rely on such an assumption but have to resort to generic means to take advantage of source characteristics (i. e., spectral decomposition and entropy coding). On the other hand, the perceptual audio coding approach allows to benefit substantially from our knowledge about the signal receiver (i. e., the human auditory system) by shaping the introduced coding noise such that its audibility is minimized by auditory masking in frequency and time. Compared to this, the ability to exploit masking effects in a traditional speech coding framework [e.g., by perceptual weighting of the coding error in code-excited linear prediction (CELP) schemes] appears much less pronounced.

Another traditional difference between speech and audio coding lies in their supported audio bandwidth. While music signals contain perceptually significant components up to at least 16 kHz and thus require a correspondingly high bandwidth for an unimpaired listening impression from audio coders, the foremost goal of speech coding for voice communication has been *intelligibility*. As a consequence, communication standards often are based on *narrow-band* audio channels (300 Hz–3.4 kHz). Unfortunately, some essential information of the human voice is located beyond this frequency range. As an example, the consonants *f* and *s* have the same spectral composition in a narrow-band



Amazon EX1010 Page 420

representation, which leads to an equal sounding of the words *failing* and *sailing* when transmitted by a narrow-band speech coder. A differentiation is only possible for a listener from the semantic message context. In order to make long conversations less exhausting, *wide-band* speech coders (50 Hz–7 kHz) gain importance. Even this extended frequency range, however, does not reproduce the entire audible spectrum, and therefore discards relevant auditory information. Due to their focus on music, perceptual audio coders traditionally provide a full bandwidth transmission at high enough bit rates, and thus offer optimal quality also in very difficult communication situations (e.g., simultaneous interpretation where at least 12 kHz bandwidth is demanded).

## 18.3 Some Successful Standardized Audio Coders

This section provides a sketch of several successful standardized perceptual audio coders that have evolved over the recent two decades rather than a comprehensive technical account. In this context, the moving pictures expert group (MPEG), formally International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC1/SC29/WG11, standardization group plays a major role. On the one hand, the first widely available commercial audio codecs were developed by this group and found broad application. On the other hand, subsequent generations of MPEG audio coding schemes have continued to define the state of the art until today. Thus, the evolution of MPEG audio coding will be taken as an illustration of the general development in this field, acknowledging the presence of other proprietary coders which have been successful in the marketplace for considerable time.



### 18.3.1 MPEG-1

In 1992, MPEG finalized its first international standard on coding of moving pictures and associated audio, called MPEG-1 [18.9]. The audio part of this standard (ISO/IEC 11172-3) specifies a generic standard for the coding of general audio signals (as opposed to speech signals as used in telecommunications / speech applications) and became the driving technology for many important applications relying on audio compression. The standard defines three operating modes (*layers*) in order to meet the requirements of a broad range of applications of audio coding concerning target bit rate and complexity:

The layer 2 coding scheme defines a codec with medium compression performance and computational complexity which had its first applications in the broadcasting environment, including use for the European digital audio broadcasting (DAB) system. The codec achieves broadcast quality at a bit rate of 128 kbit/s per channel.

Relative to this, the layer 1 coding scheme represents a simplified version of the layer 2 coder with both lower computational complexity and compression. The intended major application for this scheme was digital recording of audio for consumers [the digital compact cassette (DCC)]. Layer 1 reaches broadcast quality at a bit rate of 192 kbit/s per channel.

In contrast to this, the layer 3 coding scheme was designed to provide the best possible compression performance by more-refined and computationally demanding coding techniques. Audio transmission via integrated services digital network (ISDN) channels and satellite radio were the first major applications of this codec. Layer 3 has become very popular in the context of Internet and personal audio under the name MP3 . The codec offers broadcast quality at a bit rate of 96 kbit/s per channel.

### 18.3.2 MPEG-2

While the coding schemes provided by MPEG-1 Audio were developed to meet the needs of most high-quality applications, more capabilities were added to the existing three Layer coder family by the MPEG-2 audio standard [18.10], which was finalized in 1994. Most notably, MPEG-2 provides the specification for coding at low sampling rates (LSR), expanding the set of standard sampling rates supported by MPEG-1 audio (i. e., 32 kHz, 44.1 kHz, and 48 kHz) by their half-rate counterparts (i. e., 16 kHz, 22.05 kHz, 24 kHz). Though this leads to clear limitations in the obtainable audio bandwidth, it results in an increase in coding efficiency for low bit rates and a reduction of computational demands. Typically, MPEG-2 low-sampling-rate coding is used in multimedia applications, e.g., in desktop personal computers (PCs= where the best possible quality may not be of utmost concern.

As a second less well-known extension, the MPEG-2 specification defines backward compatible multichan-

Amazon EX1010 Page 421

nel counterparts for the MPEG-1 audio coder family. MPEG-2 multichannel audio bitstreams can be decoded by MPEG-1 audio decoders.

### 18.3.3 MPEG-2 Advanced Audio Coding

After the completion of the MPEG-2 multichannel audio specification in 1994, the MPEG audio standardization group initiated another effort to define an enhanced multichannel coding standard which should not be constrained by the requirement of MPEG-1 backward compatibility. The standardization process was successfully completed in 1997 and its result is known as MPEG-2 advanced audio coding (MPEG-2 AAC), which became an addendum to the MPEG-2 standard (ISO/IEC 13818-7) [18.11]. While the new scheme had been initially developed as a multichannel audio coder, further tests confirmed its coding performance also for stereo and monophonic signals with broadcast quality according to ITU radiocommunication sector (ITU-R) requirements at a bit rate of only 64 kbit/s per channel. Furthermore, the coder can be used within a wide range of sampling rates (8–96 kHz) and data rates (from below 16 kbit/s to 128 kbit/s per audio channel).

### 18.3.4 MPEG-4 Advanced Audio Coding

While the MPEG-4 audio specification provides a rich set of different specialized (e.g., speech) and generic coders for all conceivable types of applications [18.12, 13], the MPEG-2 AAC technology was adopted as the core part of MPEG-4 *general audio coding* (i. e., the filterbank-based MPEG-4 perceptual audio coder) and augmented with a number of extensions and functionalities. In the context of coding speech signals, several of these extensions are of particular interest:

- *MPEG-4 scalable audio coding:* one of the core functionalities provided by MPEG-4 Audio is called *scalability*, i. e., the ability of decoding only part of the full bitstream into a meaningful result, albeit with a lower perceptual quality. MPEG-4 scalable audio bitstreams consist of several embedded *layers* of information each of which originates from a partial coder. Interestingly, MPEG-4 Audio defines also coder configurations which include both speech coders and perceptual audio coders, and thus combine both concepts into a single system. Such systems will be described in some more detail in Sect. 18.5.1.
- *MPEG-4 low-delay AAC:* another requirement for MPEG-4 Audio coding was the ability to provide full-quality general audio coding with an encoding/decoding delay that is comparable to usual speech codecs and, thus is low enough to enable high-quality two-way communication (e.g., teleconferencing). This goal is achieved by a derivative of AAC, called MPEG-4 low-delay AAC. The concepts behind this codec will be described in Sect. 18.4.2.
- *Bandwidth extension:* the concept of bandwidth extension [18.14, 15] is one of the most attractive recent additions to the MPEG-4 audio coding since it provides a notable increase in coding efficiency at low bit rates/intermediate audio quality and, at the same time, removes the traditional problem of band-limited audio output at low bit rates. This technique allows to reconstruct a good perceptual approximation of the signal's original high-frequency content (e.g., above 5–6 kHz) by transmitting its low-frequency part together with a small amount of side information representing the high-frequency spectral envelope. The combination of an AAC coder with bandwidth extension, also known as *high-efficiency AAC (HE-AAC)*, achieves full bandwidth output with attractive sound quality at 24 kbit/s per audio channel and below.
- *Parametric Stereo:* Another step towards even higher coding efficiency for MPEG-4 AAC-based coders is the combination with the *parametric stereo (PS)* coding tool that has been developed as part of the MPEG-4 high-quality parametric audio coding technology [18.16, 17]. The tool allows to efficiently represent a stereo signal by transmitting a monophonic downmix and a compact parametric side information which carries a perceptual characterization of the stereophonic sound image. Based on this description, the parametric stereo decoder expands the transmitted mono downmix signal into a stereophonic output. Combined with a HE-AAC coder, good sound quality can be achieved at 32 kbit/s per stereo channel and lower.

### 18.3.5 Progress in Coding Performance

As the original focus of audio coding has been on achieving highest possible compression while preserving subjective audio quality, it is quite natural to ask what improvements have been made in compression in the recent decades. Figure 18.2 attempts to provide an estimate of the bit rate (in kbit/s) needed to code a typical (not: critical) stereo signal in *good* quality, i. e.,



Amazon EX1010 Page 422

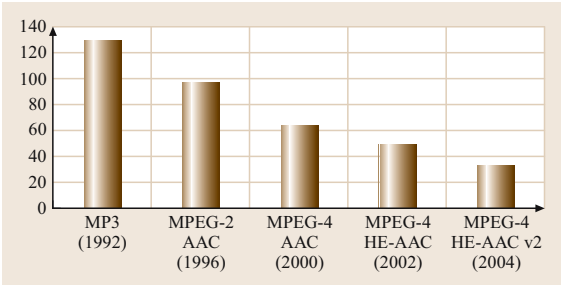



**Fig. 18.2** Bit rates for *good-quality* stereo coding (in kbit/s)

in a sonic quality that is acceptable to most users for everyday consumption. Though this is admittedly a very loose definition, the *learning curve* shown in the figure is still instructive, showing a clear reduction in bit rate from 128 kbit/s (MP3) to 32 kbit/s (HE-AAC v2 = AAC plus bandwidth extension plus parametric stereo), i. e., by 75%. Evidently, there has been significant progress in the area of nontransparent coding, and it is interesting to see how much further the frontiers can be pushed in the future.

With respect to *transparent* coding of audio signals (including critical test signals), however, no comparable increase in compression efficiency has been encountered and coding at 128 kbit/s per stereo signal using MPEG-2 AAC can still be considered state of the art at the time of writing of this text.

## 18.4 Perceptual Audio Coding for Real-Time Communication

The preceding sections discussed technology that can be employed for perceptual coding of speech signals in general storage or transmission systems (e.g., radio or Internet radio systems, digital media, voice storage in announcement systems) for which overall delay does not play a major role. For application in two-way communication (e.g., telephony, teleconferencing, telepresence), however, a low overall delay is of vital importance to the quality of the service. For such applications the encoder/decoder (*codec* ) must typically not introduce more than 20–30 ms of algorithmic delay, i. e., latency that is inherent in the algorithm disregarding speed of calculation and bitstream transmission. This limitation in codec delay is necessary in order to achieve an end-to-end transmission delay of typically no more than 150 ms in order to be acceptable for interactive speech communication. The actually observed end-to-end delay, however, also includes a number of additional delay sources, such as the delay introduced by the transmission network (e.g., by buffering within network switches or transcoding operations by network gateways) or audio postprocessing. This section analyzes the typical sources of codec delay in perceptual audio coding and discusses a number of low delay perceptual audio coding technologies that are suitable for real-time two-way speech communication.

### 18.4.1 Delay Sources in Perceptual Audio Coding

Generally, the goal of good compression efficiency can only be achieved at the expense of a certain encoding/decoding delay. The overall delay incurred by a typical perceptual audio coder results from several contributions related to the following codec parameters:

- Filterbank and framing delay
- Look-ahead time for block switching
- Algorithmic transmission delay

This section will discuss the background of these different delay contributions. As an example, the delay of an MPEG-2 AAC codec will be calculated in order to give some impression about the order of magnitude of delay for a modern high-performance perceptual codec. All calculations are based on the so-called *algorithmic delay* which describes the theoretical minimum delay allowed by an algorithm assuming negligible delay contributions due to speed of calculation, bitstream transmission or other implementation or application specific circumstances.

#### Filterbank and Framing Delay

In order to exploit the spectral masking properties of the human auditory system, perceptual audio coding schemes employ an analysis/synthesis filter bank pair. While numerous types of filter banks have been used for audio coding, the modified discrete cosine transform (MDCT) [18.3] has been adopted extensively for modern audio codecs, like MPEG-2 AAC and MPEG-4, and has shown its merits for compression at very low bit rates. For an MDCT with an overlap of 50% between subsequent symmetric windows, the filterbank introduces a total delay equal to its window size, i. e., twice the



Amazon EX1010 Page 423

frame size.

$$N_{\mathrm{MDCT}} = 2 \cdot \text{frame-size} \,. \tag{18.1}$$

#### Look-Ahead Delay for Block Switching Decision

Most modern audio codecs that are designed to operate at very low bit rates use an MDCT filterbank with a high spectral resolution. In particular, high efficiency can be reached with long frames (about 20 ms and more) for stationary signals. In the case of transient signals, like percussive sounds with sharp signal onsets (*attacks*), this would lead to the well-known pre-echo phenomenon [18.8]. This artifact can be avoided by using dynamic block switching [18.18], i. e., by dynamically switching between different filterbank window sizes, and thus reducing the blockwise noise spread in time. Due to restrictions in the permissible sequence of window types, *instantaneous* switching between long and short windows is not possible but an intermediate transition window type (*start block*) has to be inserted between long and short windows. Therefore, the detection of the optimum window type requires a look-ahead, and thus a further delay in the encoder. In general, an encoder using block switching incurs an additional delay of

$$N_{\mathrm{ahead}} = \text{frame-size} \cdot \frac{\text{numShortWindows} + 1}{2 \cdot \text{numShortWindows}} \,, \tag{18.2}$$

where numShortWindows is the number of short windows that fit into a frame (e.g., eight for AAC).

#### Algorithmic Transmission Delay

Since not all segments of an audio signal are equally demanding to code, the number of bits needed to code a specific frame varies. To obtain a constant bit rate, the bit reservoir mechanism has proven to be useful [18.8]. Since the use of the bit reservoir is equivalent to a local variation in bit rate, the size of the input buffer of the decoder must be adapted to the maximum local bit rate (i. e., the maximum number of bits which can be allocated for a single frame per channel). For the practically important case of a transmission on a channel with a rate equal to the encoding bit rate, use of the bit reservoir introduces an additional delay since the decoder has to wait at least until its input buffer is read before audio output can be started. Thus, increasing the size of the bit reservoir will also increase the overall codec delay. In fact, the overall delay of the audio coder may be dominated entirely by the size of the bit reservoir.

The delay expressed in terms of samples caused by the bit reservoir is

$$N_{\mathrm{bitres}} = \frac{\text{bitres-size}}{\text{bitrate}} F_{\mathrm{s}} \,, \tag{18.3}$$

where bitres-size is the bit reservoir size expressed in bits and $F_{\mathrm{s}}$ is the sampling rate in Hz. In addition, the time to transmit an average size frame has to be taken into account for delay considerations:

$$N_{\mathrm{frame}} = \frac{\text{average-framelength}}{\text{bitrate}} F_{\mathrm{s}} \,. \tag{18.4}$$

Note that in systems using packet-based transmission (one complete encoder frame per packet) with a channel bit rate much higher than the encoding bit rate, the transmission (and the use of the bit reservoir) does not cause the additional delay discussed above.

#### Overall Delay

From the discussion above, the overall delay of a typical coder can be calculated as follows:

$$t_{\mathrm{delay}} = \frac{N_{\mathrm{MDCT}} + N_{\mathrm{ahead}} + N_{\mathrm{bitres}} + N_{\mathrm{frame}}}{F_{\mathrm{s}}} \,, \tag{18.5}$$

with $F_{\mathrm{s}}$: the coder sampling rate (in Hz), $N_{\mathrm{MDCT}}$: the filterbank and framing delay, $N_{\mathrm{ahead}}$: look-ahead delay for block switching, $N_{\mathrm{bitres}}$: delay due to bit reservoir use, and $N_{\mathrm{frame}}$: frame transmission delay.

Note that the overall delay scales inversely with the sampling frequency.

For an AAC coder running at 24 kbit/s per channel and a sampling rate of 24 kHz, the resulting overall codec delay for a non packet based transmission is about 109.3 ms without the use of the bit reservoir. Assuming the nominal size of the input buffer as indicated in the MPEG-2 AAC standard (6144 bit/channel), a maximum additional delay of 256 ms is incurred, leading to a total delay of 365.3 ms. From this example it becomes obvious that the delay of common perceptual audio coders substantially exceeds the requirements for real-time two-way communication applications.

### 18.4.2 MPEG-4 Low-Delay AAC

The MPEG-4 error resilient low-delay advanced audio (ER AAC-LD) coder [18.19] was derived from the architecture of MPEG-2/4 AAC to achieve the desired low delay required for two-way real-time communications applications with a minimum number of changes. More specifically, the low-delay codec is derived from the MPEG-4 general audio coder, which comprises the regular MPEG-2 AAC algorithm plus additions provid-



Amazon EX1010 Page 424

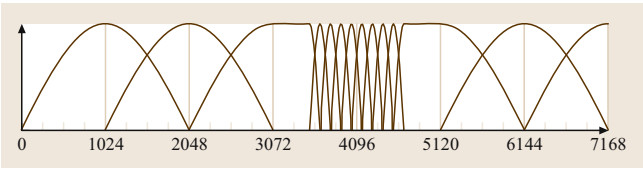



**Fig. 18.3** Typical window sequence for AAC block switching (after [18.19], with permission by the AES)

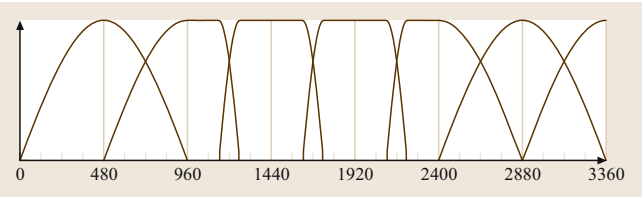



**Fig. 18.4** Typical window sequence for AAC-LD window shape adaptation (after [18.19], with permission by the AES)

ing parametric coding of noise components (perceptual noise substitution [18.20]) and long term prediction (LTP). Naturally, all capabilities for handling stereo signals and flexible sampling rates were inherited from MPEG-2 AAC to AAC-LD.

The following modifications were applied to the standard algorithm to achieve low-delay operation:

### Frame Length and Filterbank Delay

The frame length has been reduced by a factor of approximately two to 512 or 480 samples, the latter providing a frame length of 10 ms at a sampling rate of 48 kHz, which is commensurate with the granularity of many speech transmission systems. Accordingly, the length of the MDCT analysis window has been reduced to 1024 or 960 time domain samples (corresponding to 512 and 480 spectral values). This leads to an overall framing and filterbank delay of 1024 or 960 samples.

### Block Switching

Due to the considerable contribution of the look-ahead time to the encoding delay, block switching is not used. Instead, the temporal spread of quantization noise (*pre-echo*) is handled by the temporal noise shaping (TNS) [18.8] tool.

### Window Shape

Besides the standard sine window shape, AAC-LD uses a second window shape which exhibits a lower overlap between subsequent frames. Selection of this window shape for transient signal portions allows the TNS module to provide better protection against pre-echo effects by minimizing the temporal aliasing [18.19] which is inherent in the MDCT's time-domain aliasing cancelation (TDAC) concept [18.3]. Figures 18.3 and 18.4 show a comparison of the typical window sequences of regular AAC and AAC-LD. Note that, in contrast to block switching, this dynamic adaptation of the window shape does not imply any additional encoding delay.

### Algorithmic Transmission Delay

Use of the bit reservoir is minimized in order to reach the desired target delay. As an extreme case, no bit reservoir is used at all.

### Overall Delay

Taking into account the delay optimizations discussed previously, the AAC-LD overall delay reduces to the following expression:

$$t_{\text{delay}} = \frac{N_{\text{MDCT}} + N_{\text{reduced-bitres}} + N_{\text{frame}}}{F_{\text{s}}} \qquad (18.6)$$

or for packet-based transmission (no bit reservoir):

$$t_{\text{delay}} = \frac{N_{\text{MDCT}}}{F_{\text{s}}} \,. \qquad (18.7)$$

Table 18.1 contrasts this with the delay incurred by a regular AAC encoder/decoder chain. For a window length of 960 samples, a sampling frequency of 48 kHz and packet based transmission (without use of a bit reservoir), an overall algorithmic delay of 20 ms is achieved comparable with common speech codecs (e.g., the GSM full-rate codec).

**Table 18.1** Comparison of delay contributions between MPEG-2 AAC and MPEG-4 AAC-LD (expressed in samples)

| Delay Source | MPEG-2 AAC | MPEG-4 AAC-LD |
|---|---|---|
| Filterbank/framing delay $N_{\text{MDCT}}$ | 2048 | 960/1024 |
| Look-ahead delay $N_{\text{ahead}}$ | $1024 \cdot \frac{9}{16}$ | 0 |
| Bit reservoir delay $N_{\text{bitres}}$ (assuming a constant rate channel) | up to $6144 \frac{F_{\text{s}}}{\text{bitrate}} - 1024$ | As low as 0 |
| Frame transmission delay $N_{\text{frame}}$ (assuming a constant rate channel) | 1024 | 480/512 |



Amazon EX1010 Page 425

Note that it would be difficult to achieve similar low algorithmic delay values by using MPEG-1 layer 3, mainly due to two reasons: firstly, the composite hybrid filterbank used in layer 3 exhibits a higher filterbank related delay than a plain MDCT with the same spectral resolution. Furthermore, since no TNS tool is available with layer 3, it is necessary to resort to block switching techniques (and accept the associated look-ahead delay) in order to avoid pre-echo problems for transient signals.

A comparison between regular AAC and AAC-LD for two operating points (24 kbit/s AAC versus 32 kbit/s AAC-LD and 54 kbit/s AAC versus 64 kbit/s AAC-LD) showed that the significant decrease in coding delay is achieved at a moderate decrease in coding efficiency by 8 kbit/s per channel [18.19] and that the efficiency of AAC-LD compares favorably with that of an MPEG-1 layer 3 coder.

### 18.4.3 ITU-T G.722.1-C

The G.722.1 annex C codec, also known under its nickname *Siren 14*, is the 14 kHz audio bandwidth extension of the 7 kHz codec G.722.1 annex A [18.21]. The codec can operate at three possible fixed bit rates of 24, 32, or 48 kbit/s at a fixed sampling rate of 32 kHz. It features an algorithmic delay of 40 ms and is mainly used in video conferencing systems. Figure 18.5 provides an overview of the Siren 14 encoder. While the codec is in general similar to MPEG-4 ER AAC-LD in its basic structure, both codecs differ substantial in their philosophy. An informative synopsis of both schemes is instructive and will be provided in the following.

Both codecs use a cosine modulated filter bank with 50% overlap between subsequent frames which is called modulated lapped transform (MLT) for the Siren codec rather than MDCT.

Both codecs split the spectrum into frequency bands to allow a different quantization step size for spectral components in each band. G.722.1 features frequency bands of constant width (500 Hz), called *regions,* in contrast to AAC-LD (and AAC in general) that groups spectral lines into nonuniform bands that relate to the perceptual frequency scales (e.g., BARK, ERB) of the human auditory system. Effectively, the non-uniform grouping allows more-precise shaping of the introduced quantization noise at lower frequencies (and a coarser control at higher frequencies).

Both Siren 14 and AAC-LD also make use of similar Huffman coding techniques for the entropy coding of the quantized spectral coefficients. The selection procedure of the most suitable Huffman codebook is, however, based on different approaches: whereas the Siren codebook choice is strictly derived from the energy of the audio signal (which is transmitted as side information

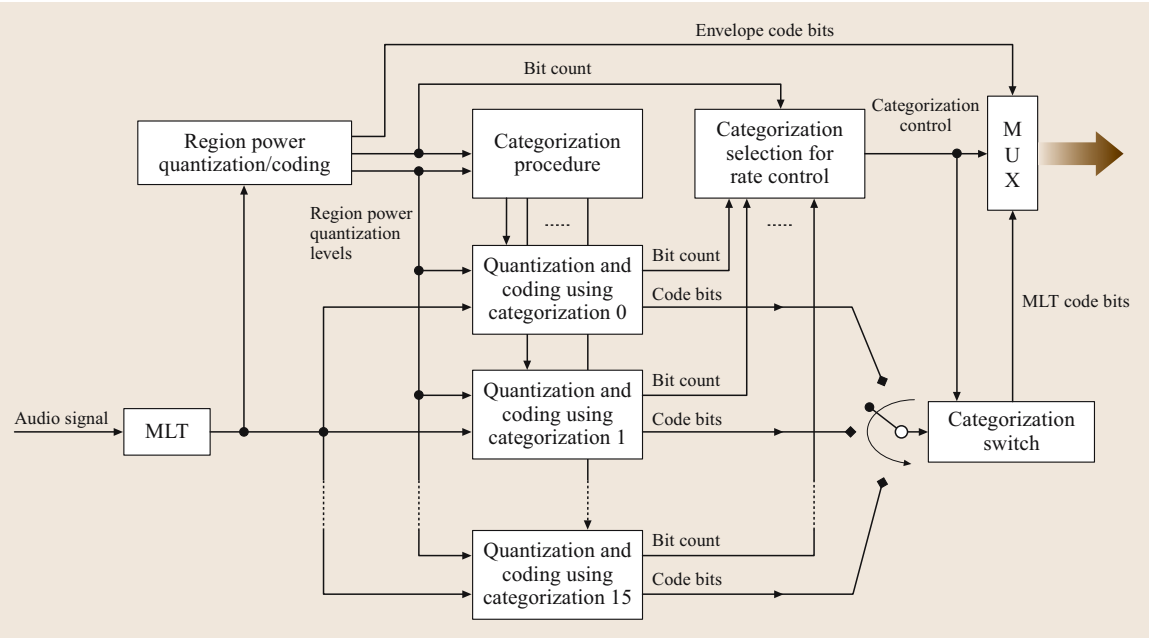



**Fig. 18.5** Structure of a G.722.1 encoder (after [18.21])

to the decoder), the AAC encoder can freely select the proper codebooks (usually according to the minimum bit demand) and explicitly conveys its choice to the decoder.

The most obvious difference between both systems is the absence of an explicit psychoacoustic model in the Siren 14 codec as compared to the common model of perceptual coding (see Fig. 18.1). Instead, the decision of the quantization step size (here called *categorization*), choice of Huffman codebook and other parameters are completely derived from the region power information which is transmitted to the decoder as side information. As the only further encoding choice, the Siren 14 encoder selects one out of 16 possible categorizations (*categorization selection for rate control*) and transmits this selection as four *categorization control bits* to the decoder to accommodate a constant transmission bit rate.

Many of the design differences between AAC-LD and Siren 14 originate from the different philosophies of the associated standardization bodies. While ITU-T standardizes bit-exact audio encoders and decoders to guarantee a known and reliable signal quality, MPEG restricts standardization to the bitstream format and decoder behavior to ensure interoperability and, at the same time, allow further improvements in encoder implementation and audio quality over time. As a consequence, MPEG codecs traditionally leave many decisions that influence the resulting audio quality to the encoder algorithm and transmit these decisions explicitly in the bitstream. In contrast, an ITU-T codec implements a fixed overall algorithm in a maximally efficient way with regard to transmission of side information, frequently deriving decisions implicitly from other known parameters such as band energies. The advantage of this approach is a low side information rate which comes at the price of restricted algorithmic flexibility. In summary, Siren 14 is designed as a simpler system than AAC-LD in terms of algorithmic structure (including coding tools) and computational complexity. Due to its compact side information structure it provides good audio quality for average music and speech signals at low bit rates. On the other side, it exhibits weakness with critical audio material such as transient signals, and does not benefit from a higher bit rate to the same extent known from AAC codecs [18.22].

### 18.4.4 Ultra-Low-Delay Perceptual Audio Coding

Traditionally, speech coding is based on a predictive structure which provides a very low encoding/decoding delay. In contrast, perceptual audio coding requires the presence of a psychoacoustic model plus the ability to control the spectral shape of the quantization distortion to meet the masking requirements. Both requirements call for a subband spectral decomposition of the audio signal. Hence, traditional perceptual modeling and coding can not be applied directly to predictive coding of a time domain signal. To solve this problem, the ultra-low-delay (ULD) codec [18.23, 24] makes use of a *pre-/postfiltering* approach [18.25] that separates the application of psychoacoustics (i. e., the irrelevance reduction) from the redundancy reduction, so that two separate processing steps are applied.

Figure 18.6 shows the general structure of the ULD encoder and decoder.

Very much like in traditional filterbank-based perceptual audio coding, the input of the ULD perceptual model computes subband coefficients from an analysis filterbank. For the purpose of this coder, however, a rather small number of subbands are chosen providing both sufficient frequency and time resolution to model

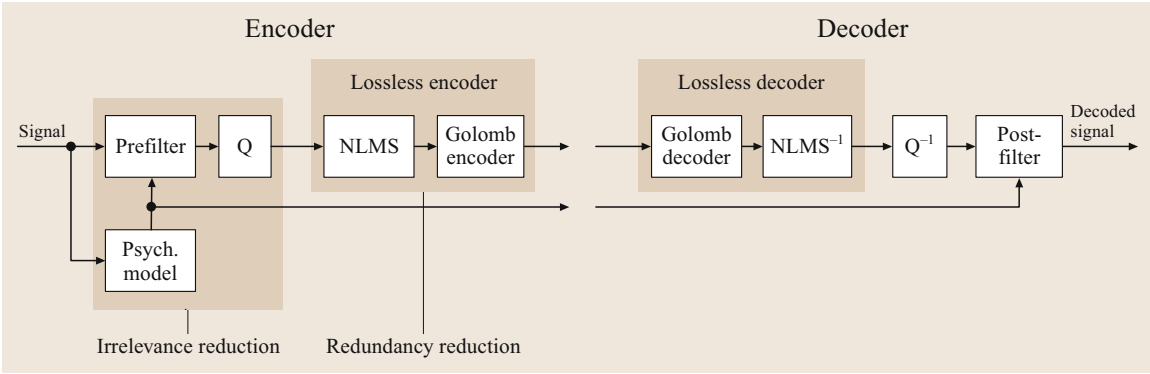



**Fig. 18.6** Structure of ultra-low-delay (ULD) audio encoder/decoder



Amazon EX1010 Page 427

spectral and temporal masking effects. The output of the perceptual model is the masking threshold for each subband. The input signal passes through the prefilter, i. e., a time-varying linear filter the coefficients of which are continuously updated such that its magnitude response is the inverse of the masking threshold. As a consequence, it *normalizes* the signal with regard to its masking threshold. The filter coefficients for the postfilter are transmitted as side information to the decoder. Since the prefiltered signal contains all audio spectral components weighted according to their perceptual relevance, it is subsequently quantized by a simple time-domain signal quantizer (e.g., a scalar uniform quantizer), producing a quantization error that is flat over frequency. In this way, the combination of perceptual model, prefilter and quantizer exploits the *irrelevance* of the input signal. The delay of this stage is determined by the perceptual model and its filterbank. For an implementation using a filterbank with 128 uniformly spaced subbands and a window length of 256, this leads to a delay of 128 samples. Additionally, the interpolation of the masking threshold values between subsequent blocks introduces another 128 samples, totalling to 256 samples. The quantized time domain values produced by the irrelevance reduction stage are then further compressed by a subsequent *redundancy reduction* module which aims at lossless encoding of its input signal with a minimum number of bits using both predictive coding techniques and entropy coding.

Common lossless audio codecs are typically based on blockwise forward adaptive prediction (i. e., the prediction coefficients for a block are transmitted as side information), implying a delay of at least one block size. To avoid this additional delay, backward adaptive predictive coding can be used as it is known from low-delay speech [18.26] or audio coding [18.27]. While these coders employ backward adaptive prediction to update LPC filters, the ULD coder makes use of this technique to obtain a lossless compression of quantized (integer) values. The backward adaptation is implemented by using the popular normalized least-mean-square (NLMS) algorithm [18.28]. The NLMS predictor is followed by an entropy coding with low delay, such as arithmetic coding, adaptive Huffman coding or Golomb coding. The latter has the advantage of a low computational complexity and no additional delay. Constant bit rates can be achieved by an encoder rate loop including quantizer, prediction module, and entropy encoder [18.29].

On the decoder side, the lossless decoder performs the inverse entropy and predictive coding steps, and the quantized values are then mapped back to their original scaling. Finally, the audio output signal is produced by applying a postfilter which operates as the inverse counterpart of the prefilter. Hence, this filter undoes the perceptual normalization of the input signal introduced by the prefilter. By imposing a magnitude response resembling the masking threshold curve, it effectively also shapes the spectrally flat quantization noise to match this curve.

The overall algorithmic encoding/decoding delay of a typical ULD codec is about 256 samples corresponding to about 8 ms at 32 kHz sampling rate or below 6 ms at 48 kHz, possible bit rates ranging from around 32 to 96 kbit/s per channel, where high audio quality can be achieved at 80 kbit/s per channel. The flexibility in block length and psychoacoustic look-ahead can be used to derive even more delay optimized versions. As an example, the delay can be reduced to 1.5 ms at a cost of around 1/3 in bit rate. The robustness of the ULD decoder against transmission errors can be significantly improved by applying an adaptive reshaping of the prediction error [18.30]. A ULD encoder consumes less than 100 MHz on a 16 bit fixed-point digital signal processor (DSP, Analog Devices BF533) in a configuration with 48 kHz sampling rate, constant bit rate, 64-tap NLMS predictor and Golomb encoding.

## 18.5 Hybrid/Crossover Coders

Perceptual audio coding has made significant advances in performance over time. Nonetheless, the ability of using a source model, as it is used by speech coders, provides a distinct advantage for coding speech signals over generic perceptual audio coding at very low bit rates (e.g., below 20 kbit/s per channel). Typically, when running at very low bit rates, filterbank-based perceptual audio coders tend to smear the temporal structure of individual glottal excitation pulses as a consequence of coarsely quantizing spectral coefficients. This often leads to a *reverberant* quality of the decoded signal.

Conversely, attempting to use traditional speech coders for sound that is not a pure speech signal (e.g., speech with background noise or multiple talkers) re-



Amazon EX1010 Page 428

sults in significant distortion because the input signal does not conform to the speech production model embedded into the coder. Moreover, coding of complex signals, such as music, with speech coders usually produces extremely undesirable artifacts due to the coder's inability to accurately reproduce tonal signal components with a frequency outside the pitch range of human voice.

These complementary characteristics of speech coders and perceptual audio coders have repeatedly motivated researchers to look into combining both schemes into a single architecture such that the strengths of both approaches are preserved. The following sections will discuss a number of such approaches as they have emerged over time.

### 18.5.1 MPEG-4 Scalable Speech/Audio Coding

One of the first combinations between speech coding technology and perceptual audio coding was conceived as part of the scalable MPEG-4 audio coder [18.13]. It encodes audio and speech material in a way that allows to adapt the bit rate in response to the time-varying channel capacity of transmission networks without the need for reencoding with the instantaneous target bit rate. Among other scalable codec combinations, MPEG-4 supports a hybrid scheme including both an MPEG-4 speech coder and (or several) AAC audio coders.

In this configuration, a scalable bitstream consists of a CELP core layer bitstream and one or more AAC enhancement layers. The core bitstream can be decoded on its own, resulting in a meaningful decoder output with normal speech coding quality and restrictions. Decoding of both the core bitstream and the AAC enhancement layers will increase output quality and bandwidth substantially over the core layer. Thus, transmission or decoding of a subset of the full bitstream will lead to a valid decoded signal though at a lower quality. The structure of such a coder is depicted by Fig. 18.7. The CELP base layer coder operates at a lower sampling rate than the subsequent enhancement layers. The scalable combination of these component coder works as follows:

- The audio input is downsampled and encoded by the CELP coder. The produced bitstream constitutes the base layer portion of the scalable bitstream. It is decoded locally and upsampled to match the sampling rate of the T/F-based enhancement layers and passed through an MDCT analysis filterbank. Since the ratio of core to enhancement layer sampling rate is normally an integer value, the upsampling operation can be implemented simply, e.g., by inserting zeros values and band limiting of the spectral coefficients after the filterbank.
- In a parallel signal path, the delay compensated input signal is passed through the MDCT analysis filterbank, and the residual coding error signal is computed.
- The residual signal is passed through a frequency selective switch (FSS) which permits to fall back to the original signal on a scale-factor band basis if this can be coded more efficiently.
- The resulting spectral coefficients are quantized/coded by an AAC coding kernel, leading to an enhancement layer bitstream.

Figure 18.8 shows the corresponding decoder. The core layer is decoded and upsampled to match the sampling rate of the enhancement layer and converted into an MDCT representation. This spectral representation is then refined by combining it with the spectral data decoded from the enhancement layer using an inverse frequency selective switch (IFSS). Finally, an inverse MDCT (IMDCT) converts the signal back into a time-domain output.

Further refinement stages (enhancement layers) could be added by continued recoding of the residual coding error signal, and can be computed efficiently in the spectral domain. Since each enhancement layer car-

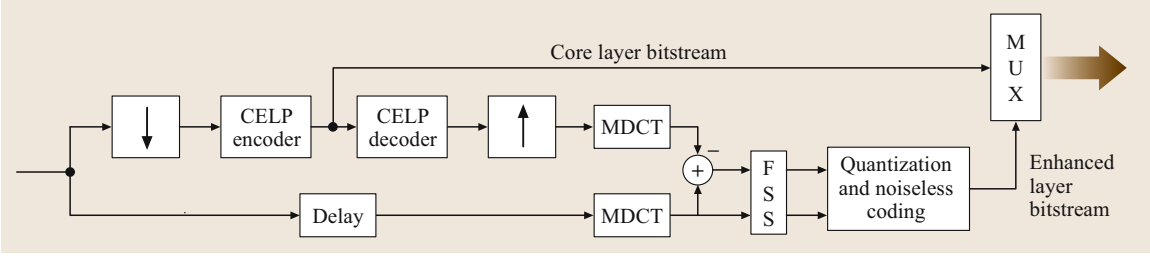



**Fig. 18.7** Scalable MPEG-4 coder with speech coding core and general audio enhancement



Amazon EX1010 Page 429

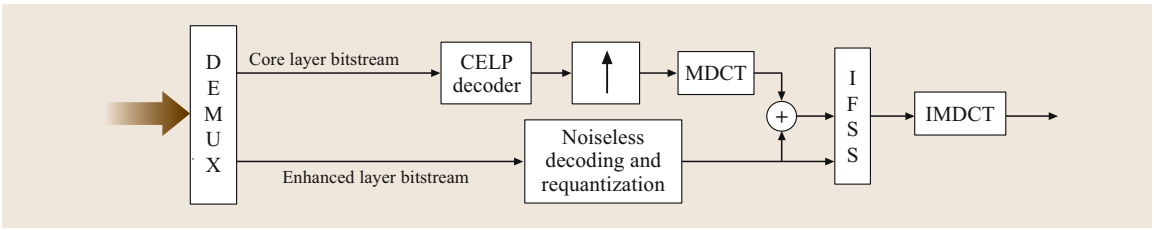



**Fig. 18.8** Scalable MPEG-4 decoder with speech coding core and general audio enhancement

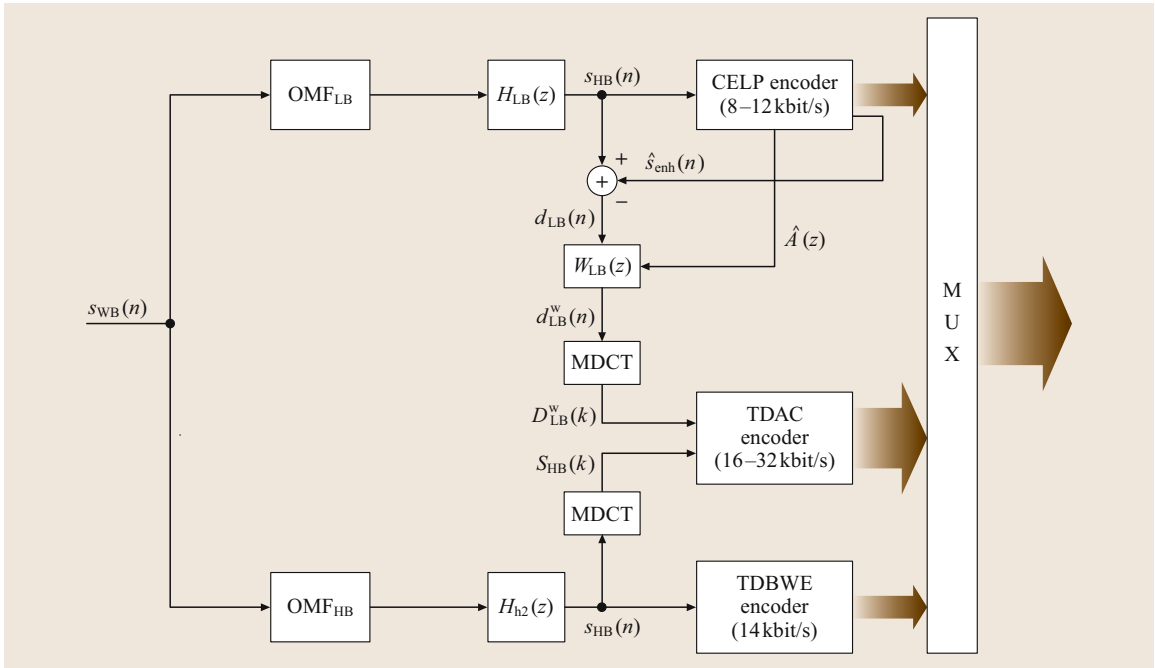



**Fig. 18.9** Structure of a G.729.1 encoder (after [18.31])

ries some associated side information overhead (e.g., scale-factor data), this type of scalability is efficient for sufficiently large enhancement steps (typically at least 8 kbit/s), and is therefore referred to as *large step scalability* in MPEG-4 Audio. A more-extensive discussion MPEG-4 audio scalability features is provided in [18.13].

### 18.5.2 ITU-T G.729.1

Another example of a coder that combines CELP speech coding with a filterbank-based enhancement stage is the recent ITU-T G.729.1 scheme. It offers a 12-layer scalable (*embedded*) bitstream structure with bit rates between 8 kbit/s and 32 kbit/s, and is interoperable with widely deployed G.729 systems. For 8 or 12 kbit/s, the coder provides traditional narrow-band coding capability, and at higher bit rates it is scalable in steps of 2 kbit/s, transmitting wide-band signals (50 Hz–7000 Hz).

Figure 18.9 illustrates the G.729.1 encoder architecture. In order to accommodate both narrow- and wide-band operation, the 16 kHz sampling rate input signal $s_{WB}(n)$ is split into decimated lower-frequency half-band $s_{LB}(n)$ and decimated upper-frequency half-band $s_{HB}(n)$ signals by a quadrature mirror filter (QMF). These signals are then coded by three stages:

- The backward-compatible base codec consists of a narrow-band CELP core operating at 8 kbit/s (layer 1 only) or 12 kbit/s (layers 1 and 2) on a 50 Hz high-pass-filtered version of the lower-frequency band signal $s_{LB}(n)$.



Amazon EX1010 Page 430

- The CELP coding error (residual signal) $d_{LB}(n)$ is perceptually filtered by $W_{LB}(z)$ (whose parameters are derived from the LPC module of the CELP encoder) and transformed into the MDCT domain. This perceptually weighted difference signal $D^{w}_{LB}$ is combined with the MDCT-domain high-band signal $S_{HB}(k)$ in the time-domain aliasing cancelation codec and transmitted within layers 4–12, resulting in bit rates from 16 kbit/s up to 32 kbit/s.
- In order to enable a full wide-band representation of the audio signal for bit rates at or above 14 kbit/s, layer 3 adds information for a parametric time-domain bandwidth extension (TDBWE) tool in the decoder.

Figure 18.10 shows the structure of the corresponding decoder. Layers 1 and, if present, 2 are decoded by the embedded CELP decoder, postfiltered and post-processed by a high-pass filter (HPF) to obtain the low-frequency input signal for the QMF synthesis filterbank (IQMF). If layer 3 is present (i. e., for bit rates larger or equal to 14 kbit/s), the TDBWE decoder produces a high-frequency synthesis which is further processed in the MDCT domain, and transformed back to provide the high-frequency band signal for the QMF synthesis filterbank. For all further layers, the TDAC decoder produces MDCT coefficients which describe the (weighted) residual signal for the lower-frequency QMF band or the reconstructed signal in the high-frequency QMF band. In the latter case, the transmitted MDCT coefficients replace the TDBWE signal as they become available. Both the low- and high-frequency TDAC decoder contributions are passed through an inverse MDCT. In order to reduce pre/post-echo artifacts (i. e., temporal smearing of the signal resulting from processing with a high frequency resolution), such conditions are detected and the corresponding signals are modified based on the time envelope information from the CELP and TDBWE parts.

The overall algorithmic delay of the codec is 48.94 ms, and its complexity is around 35 wmops ([18.31, Table 5]).

### 18.5.3 AMR-WB+

Another interesting and recent example of a hybrid speech/audio coding scheme is the extended wide-band adaptive multirate coder (AMR-WB+) [18.32] which extends the wide-band AMR speech coder (AMR-WB) towards the ability of handling general audio signals. As an important difference from the coders discussed in the preceding two sections (MPEG-4 Scalable Audio Coding in Sect. 18.5.1 and G.729.1 in Sect. 18.5.2), this

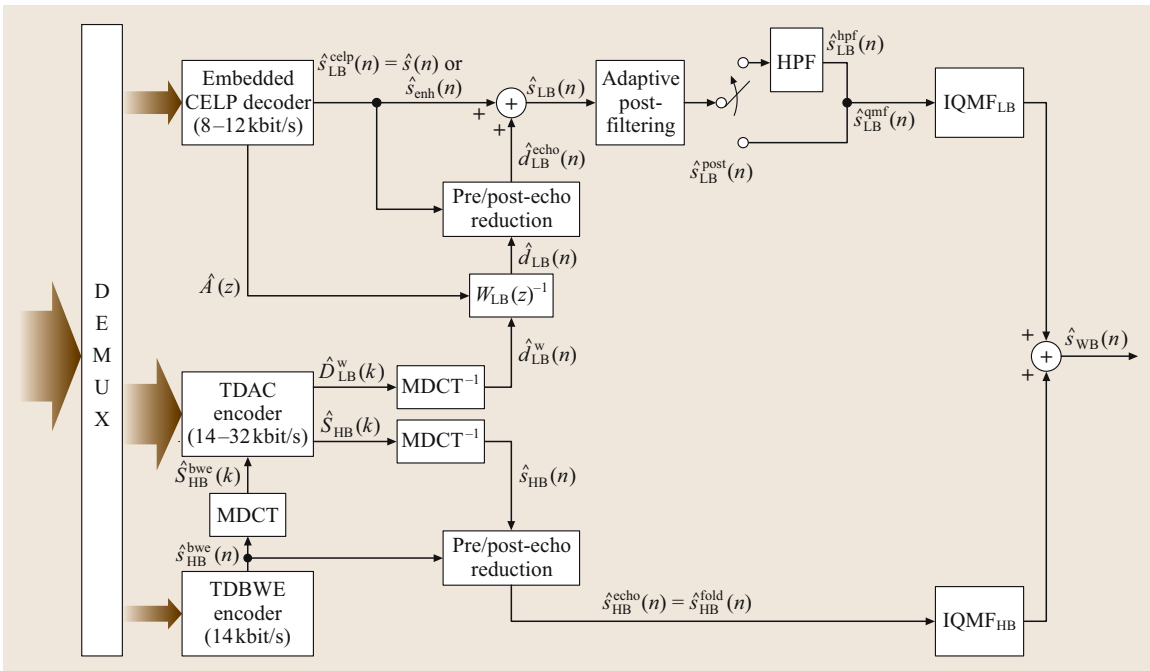



**Fig. 18.10** Structure of a G.729.1 decoder (after [18.31])



Amazon EX1010 Page 431

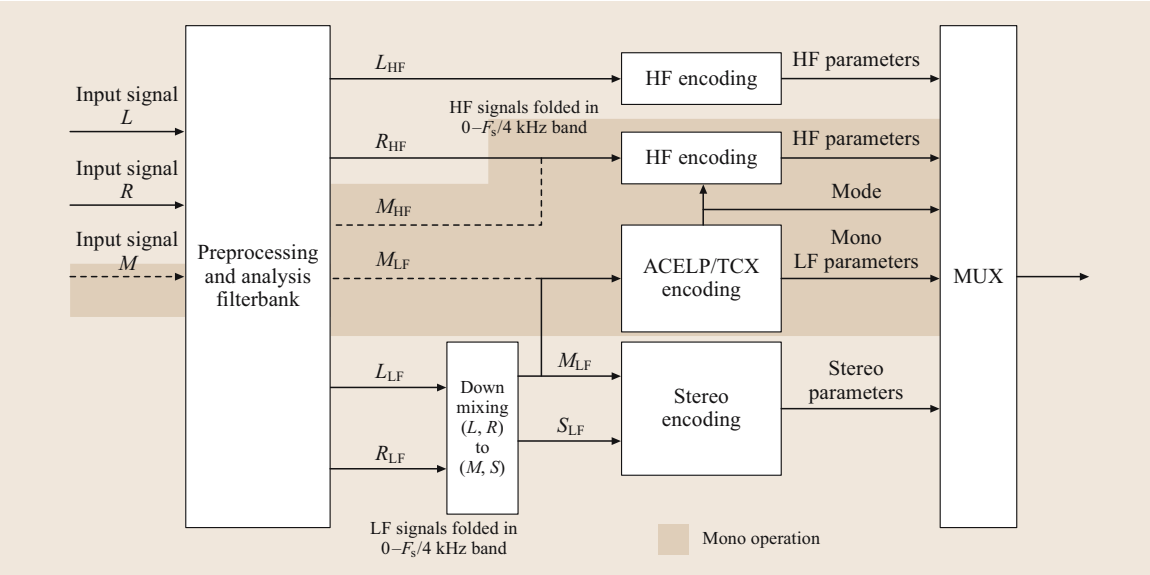



**Fig. 18.11** High-level structure of the extended AMR-WB encoder (after [18.32])

flexibility is not achieved by running a filterbank-based audio coding on top of (i. e., as an enhancement layer to) an underlying speech coder, but by allowing a *dynamic selection* between two alternate coding modes. While the algebraic code excited linear prediction (ACELP) mode handles speech-like signals in an efficient way, a filterbank-based audio coding mode (transform coded excitation, TCX) provides good quality for non-speech signal types. As a further extension, the AMR-WB+ coder includes a parametric stereo processing for efficient handling of stereo audio signals.

Figure 18.11 shows the high-level structure of the encoder. The input signal is first converted to the coder's internal sampling frequency (nominally 25.6 kHz), and then split into two equal-sized subbands (denoted with the subscripts *HF* and *LF* for high- and low-frequency band sin the figure, respectively). For operation with a mono input signal (denoted *M* in the figure), only the parts with the grey background are used. Each block of 1024 subband samples (called a superframe) is processed independently. The lower subband signal (nominally between 0 and 6.4 kHz) is encoded using the hybrid ACELP/TCX coder while the high band (nominally between 6.4 and 12.8 kHz) is encoded using a bandwidth extension method.

If a stereo signal is encoded (signals *L* and *R* in the figure), its high frequency subband signals are encoded separately using the bandwidth extension mechanism. The low-frequency band of the stereo signal is converted into *mid* (sum) and *side* (difference) signals, *M* and *S*, respectively. While the mid signal is fed into the ACELP/TCX coder, the side signal is further processed by the stereo encoding stage (i. e., parametrically coded above 1 kHz nominal, and ACELP/TCX coded below this frequency). All quantized and coded parameters are multiplexed into a bitstream.

At the heart of the coding scheme, the ACELP/TCX core divides each superframe of 1024 samples into four frames of 256 samples, each of which can be encoded in four possible modes:

- as a 256-sample ACELP frame
- as a 256-sample TCX frame
- as a part of a larger 512-sample TCX frame that is constructed by concatenating two subsequent frames
- as a part of a long 1024-sample TXC frame that encompasses the whole superframe

The selection of the optimum coding mode is carried out on a per-frame basis and can either be done in a closed- or open-loop fashion. The latter is computationally less demanding but results in some quality degradation for non-speech signals. A more-detailed description of the hybrid ACELP/TCX coding approach can be found in [18.33].

The AMR-WB+ decoder (Fig. 18.12) demultiplexes the received bitstream, recovers the quantized parameters and performs the corresponding synthesis operations to reconstruct the output signal.



Amazon EX1010 Page 432

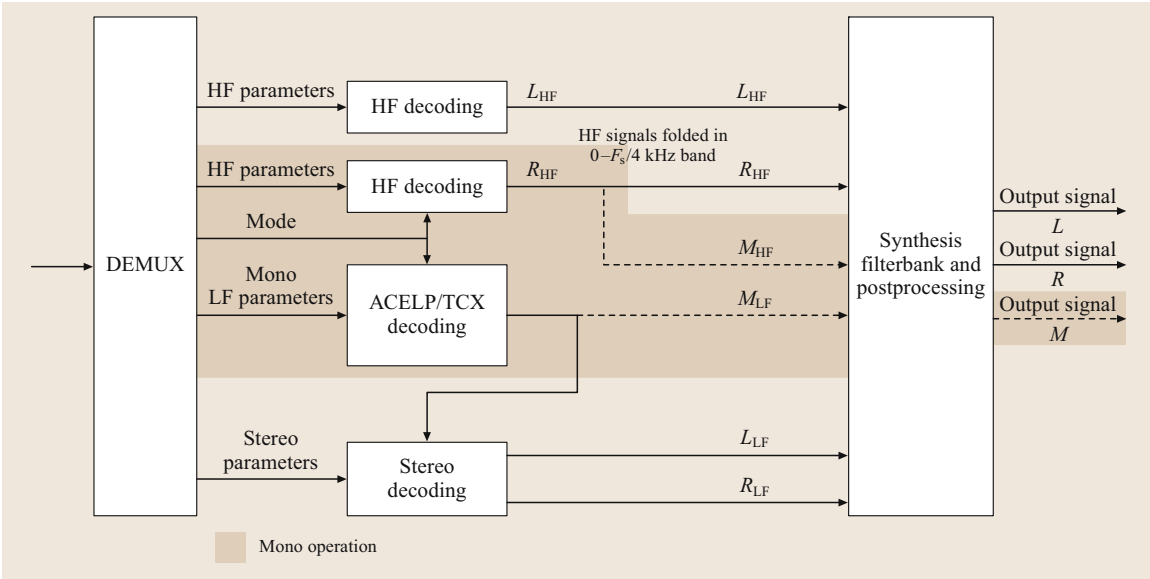



**Fig. 18.12** High-level structure of the extended AMR-WB decoder (after [18.32])

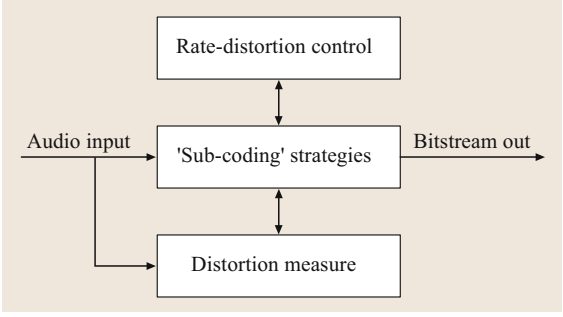



**Fig. 18.13** High-level structure of the ARDOR coder (after [18.36])

The AMR-WB+ codec supports various sampling frequencies between 8 and 48 kHz and bit rates of 6–36 kbit/s (mono operation)/8–48 kbit/s (stereo operation). When running at the nominal internal sampling frequency, the codec features an overall algorithmic delay of about 113.8 ms [18.34]. For a more-detailed overview of AMR-WB+ see [18.35].

### 18.5.4 ARDOR

While all of the combined speech/audio coders discussed previously are the results of intense development work within standardization bodies, there is also considerable activity in the research community on the topic of hybrid coding. One example is the coder that was developed under the *adaptive rate-distortion optimized sound coder* (ARDOR) European research project [18.36] and aims at providing a unified universal audio coder that shows optimal behavior for a wide range of bit rates, input signal types and other requirements (such as delay). The general architecture of the coder is shown in Fig. 18.13. The key idea is to process the input signal using a number of *subcoding* strategies (i. e., coding modules such as CELP-based speech coding, filterbank-based audio coding and sinusoidal audio coding). This may happen either individually (i. e., one subcoder at a time) or, more generally, in combination (i. e., each subcoder carries a part of the signal to be coded). In order to make optimum use of the pool of subcoders, a central element of the concept lies in the *rate-distortion (R/D) optimization* approach which helps select the subcoder that provides the best performance in a rate-distortion sense for a given input signal (or parts thereof). This gives the framework for an overall optimization of the whole system. As a third important aspect of the overall system, the rate-distortion optimization process employs a *perceptual distortion measure* in order to optimize subjective quality rather than an inappropriate simple metric like SNR. Thus, accurate modeling of perceptual quality metrics is paramount to guiding the overall system in a meaningful way. An example of R/D optimization in the ARDOR context can be found in [18.37].



Amazon EX1010 Page 433

Naturally, a coder resulting from such a research project cannot compete with highly optimized standardized schemes in the marketplace in terms of its implementation and computational complexity. Instead, it delivers a proof of feasibility for a comprehensive approach to the universal coding problem and provides valuable insights into the underlying problems. Finally, it demonstrates the possible gain attainable from such a concept, and in this way further pushes the borders. More work in this area is needed to arrive at a practically attractive solution.

## 18.6 Summary

Perceptual audio coding has made significant progress during the last two decades and is becoming increasingly attractive not only for music signals but also for carrying high-quality speech signals. A number of perceptual audio coding schemes have been developed fulfilling the delay requirements imposed by real-time two-way voice communication and thus enabling telephone and teleconferencing applications with high sound reproduction fidelity. While perceptual audio coders cannot offer the same speech compression efficiency as their source-model-based speech coding counterparts at very low bit rates, they traditionally excel in terms of quality that scales continuously with bit rate without saturating and their tolerance to mixed and complex signals.

It is interesting to see the first signs of successful convergence between the speech and the audio coding worlds in a quest for hybrid schemes that combine the advantages of both coding paradigms into a single system on the way towards the *universal coder*.

### References


18.1 B.C.J. Moore: *Introduction to the Psychology of Hearing*, 3rd edn. (Academic, New York 1989)

18.2 E. Zwicker, H. Fastl: *Psychoacoustics, Facts and Models* (Springer, Berlin, Heidelberg 1990)

18.3 J. Princen, A. Johnson, A. Bradley: Subband/Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation, IEEE ICASSP, 2161–2164 (1987)

18.4 J.H. Rothweiler: Polyphase Quadrature Filters – a new Subband Coding Technique, IEEE ICASSP, 1280–1283 (1983)

18.5 K. Brandenburg, E. Eberlein, J. Herre, B. Edler: Comparison of Filterbanks for High Quality Audio Coding, IEEE ISCAS (1992)

18.6 M. Bosi: Filter Banks in Perceptual Audio Coding, Proc. of the 17th International AES Conference on High Quality Audio Coding (1999)

18.7 R.P. Hellman: Asymmetry of Masking between Noise and Tone, Percept. Psychophys. **11**, 241–246 (1972)

18.8 J. Herre: Temporal Noise Shaping, Quantization and Coding Methods in Perceptual Audio Coding: A Tutorial Introduction, Proc. of the 17th International AES Conference on High Quality Audio Coding (1999)

18.9 ISO/IEC: JTC1/SC29/WG11 MPEG International Standard ISO/IEC 11172, Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s (ISO, Geneva 1993)

18.10 ISO/IEC: JTC1/SC29/WG11 MPEG International Standard ISO/IEC 13818-3, Generic Coding of Moving Pictures and Associated Audio: Audio (1994)

18.11 ISO/IEC: JTC1/SC29/WG11 MPEG International Standard ISO/IEC 13818-7, Generic Coding of Moving Pictures and Associated Audio: Advanced Audio Coding (1997)

18.12 ISO/IEC: JTC1/SC29/WG11 MPEG International Standard ISO/IEC 14496-3:2001, Coding of Audio-Visual Objects, Part 3 Audio (2001)

18.13 F. Pereira, T. Ebrahimi (Eds.): *The MPEG-4 Book, IMSC Multimedia Series* (Prentice Hall, Englewood Cliffs 2002)

18.14 ISO/IEC: JTC1/SC29/WG11 MPEG 14496-3:2001/Amd.1:2003, Coding of Audio-Visual Objects – Part 3: Audio, Amendment 1: Bandwidth extension (2003)

18.15 M. Dietz, L. Liljeryd, K. Kjoerling, O. Kunz: *Spectral Band Replication, a Novel Approach in Audio Coding* (112th AES Convention, Munich 2002), Preprint 5553

18.16 ISO/IEC: JTC1/SC29/WG11 MPEG 14496-3:2001/Amd.1:2003, Coding of Audio-Visual Objects – Part 3: Audio, Amendment 2: Parametric coding for high quality audio (2004)

18.17 W. Oomen, E. Schuijers, B. den Brinker, J. Breebaart: *Advances in Parametric Coding for High-Quality Audio* (114th AES Convention, Amsterdam 2002), Preprint 5852




Amazon EX1010 Page 434

18.18 B. Edler: Codierung von Audiosignalen mit überlappender Transformation und adaptiven Fensterfunktionen, Frequenz **43**, 252–256 (1989), in German

18.19 E. Allamanche, R. Geiger, J. Herre, T. Sporer: *MPEG-4 Low Delay Audio Coding based on the AAC Codec* (106th AES Convention, Munich 1999), Preprint 4929

18.20 J. Herre, D. Schulz: *Extending the MPEG-4 AAC Codec by Perceptual Noise Substitution* (104th AES Convention, Amsterdam 1998), Preprint 4720

18.21 ITU-T Recommendation G.722.1 (5/2005): Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss

18.22 R. Geiger, M. Lutzky, M. Schmidt, M. Schnell: *Structural Analysis of Low Latency Audio Coding Scheme* (119th AES Convention, New York 2005), Preprint 6601

18.23 G. Schuller, A. Härmä: Low Delay Audio Compression using Predictive Coding, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando (2002)

18.24 G. Schuller, B. Yu, D. Huang, and B. Edler: Perceptual Audio Coding using Adaptive Pre and Post-Filters and Lossless Compression, IEEE Transactions on Speech and Audio Processing (2002) pp. 379–390

18.25 B. Edler, G. Schuller: Audio Coding Using a Psychoacoustic Pre- and Post-Filter, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul (2000)

18.26 J.-H. Chen, R.V. Cox, Y.-C. Lin, N. Jayant, M.J. Melchner: A low-delay CELP coder for the CCITT 16 kb/s speech coding standard, IEEE J. Sel Areas in Comm **10**, 830–849 (1992)

18.27 A. Härmä, U. K. Laine, and M. Karjalainen: Backward adaptive warped lattice for wideband stereo coding in Proc. of EUSIPCO '98, Greece (1998)

18.28 S.S. Haykin: *Adaptive Filter Theory* (Prentice Hall, Englewood Cliffs 1999)

18.29 U. Krämer, G. Schuller, S. Wabnik, J. Klier, J. Hirschfeld: Ultra Low Delay audio coding with constant bit rate, 117th AES Convention, San Francisco, Preprint 6197

18.30 S. Wabnik, G. Schuller, J. Hirschfeld, U. Kraemer: Packet Loss Concealment in Predictive Audio Coding, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, (2005) New Paltz

18.31 ITU-T Recommendation G.729.1 (5/2006): G.729 based Embedded Variable bit-rate coder: An 8–32 kbit/s scalable wideband coder bitstream interoperable with G.729

18.32 GSM 3rd Generation Partnership Project (3GPP), 3GPP TS 26.290: Audio codec processing functions; Extended AMR Wideband codec; Transcoding functions

18.33 B. Bessette, R. Lefebvre, and R. Salami: Universal Speech/Audio Coding Using Hybrid ACELP/TCX Techniques, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia (2005)

18.34 ETSI TR 126 936 V6.1.0 (2006-03): Universal Mobile Telecommunications System (UMTS), Performance characterization of 3GPP audio codecs

18.35 R. Salami, R. Lefebvre, K. Kontola, S. Bruhn, A. Taleb: Extended AMR-WB for high-quality audio on mobile devices, IEEE Commun. Mag. **44**(5), 90–97 (2006)

18.36 N.H. van Schijndel, S. van de Par: Rate-distortion optimized hybrid sound coding. In:, Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2005) (2005) pp. 235–238

18.37 R. Vafin, W. B. Kleijn: Rate-Distortion Optimized Quantization in Multistage Audio Coding, IEEE Transactions on Speech and Audio Processing 14:311–320 (2006)




Amazon EX1010 Page 435

# Part D Text-to-Speech Synthesis

Ed. by S. Narayanan


19 Basic Principles of Speech Synthesis
J. Schroeter, Florham Park, USA

20 Rule-Based Speech Synthesis
R. Carlson, Stockholm, Sweden
B. Granström, Stockholm, Sweden

21 Corpus-Based Speech Synthesis
T. Dutoit, Mons, Belgium

22 Linguistic Processing for Speech Synthesis
R. Sproat, Urbana, USA

23 Prosodic Processing
J. van Santen, Beaverton, USA
T. Mishra, Beaverton, USA
E. Klabbers, Beaverton, USA

24 Voice Transformation
Y. Stylianou, Heraklion, Greece

25 Expressive/Affective Speech Synthesis
N. Campbell, Keihanna Science City, Japan


Amazon EX1010 Page 436

**S. Narayanan**



University of Southern California Street
Department of Electrical Engineering
Los Angeles, CA, USA
*shri@sipi.usc.edu*

Shrikanth Narayanan is Andrew J. Viterbi Professor of Engineering at the University of Southern California. Previously, he was with AT&T Bell Labs and AT&T Research. He is an Editor for Computer Speech and Language Journal and an Associate Editor for the IEEE Signal Processing Magazine. He has published over 240 papers and holds 5 patents. He is a recepient of several best paper awards.

Amazon EX1010 Page 437

# 19. Basic Principles of Speech Synthesis

J. Schroeter

Speech synthesis enables voice output by machines or devices. Text-to-speech (TTS) synthesis does so by using text as input. Ever since the talking machine by von Kempelen in 1791 [19.1], researchers and technologists have endeavored to make machines talk. The first electronic synthesis, Homer Dudley's *Voder* (*Voice Coder*), was demonstrated at the 1939 World Fair in New York City [19.2]. Today, TTS systems enjoy wide use in assistive technologies, telecommunications, entertainment, and education. In this chapter we will review the basic principles of this technology, which serves as an introduction to later chapters that provide the reader with more-detailed information.

19.1 **The Basic Components of a TTS System** .... 413
19.1.1 TTS Frontend ..... 413
19.1.2 TTS Backend ..... 415

19.2 **Speech Representations and Signal Processing for Concatenative Synthesis** ..... 421
19.2.1 Time-Domain Pitch Synchronous Overlap Add (TD-PSOLA) ..... 422
19.2.2 LPC-Based Synthesis ..... 423
19.2.3 Sinusoidal Synthesis ..... 423

19.3 **Speech Signal Transformation Principles** . 423
19.3.1 Prosody Transformation Principles .. 424
19.3.2 Principle Methods for Changing Speaker Characteristics and Speaking Style ..... 424

19.4 **Speech Synthesis Evaluation** ..... 425

19.5 **Conclusions** ..... 426

**References** ..... 426

## 19.1 The Basic Components of a TTS System

A simple block diagram of a text-to-speech (TTS) system is depicted in Fig. 19.1. The frontend on the left is logically separated from the synthesizer backend on the right. The frontend works on a symbolic level, taking text as input and creating control information as its output. The synthesis backend uses the control information and renders speech output from it. In this chapter, we will use English as an example language. We also point out that, for many aspects of TTS, there is a choice of rule-based versus data-driven methods that can be applied. Finding the right mix or compromise between both extremes is one topic of ongoing research.

### 19.1.1 TTS Frontend

Figure 19.2 depicts the principle tasks of a TTS frontend. It provides document structure detection and text normalization, interprets text markup, and performs a linguistic analysis to produce what amounts to tagged text that then undergoes phonetic analysis to create phone-based information, and prosodic analysis to determine pitch, durations, as well as assigning stresses and pauses. Although presented here in a simple linear sequence of steps, many of the tasks of a TTS frontend actually rely on specific information created in *later* steps. Therefore, the sequence of steps presented here is only the most commonly used with different interdependencies handled as exceptions and/or resulting in delayed decisions. Much more detailed information on text processing in TTS frontends can be found in Chap. 22 by Sproat.

Document structure detection can be as general as interpreting punctuation marks, may include *filtering* out email headers (e.g., in a unified messaging application), or may even take paragraph formatting into account. Document structure detection is simplified in case the document follows the standard generalized markup language (SGML) standard. SGML is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form [19.3].

Text normalization handles abbreviation and acronyms with the goal of matching how an educated human reader would render the input text. 'St.' could be

Amazon EX1010 Page 438

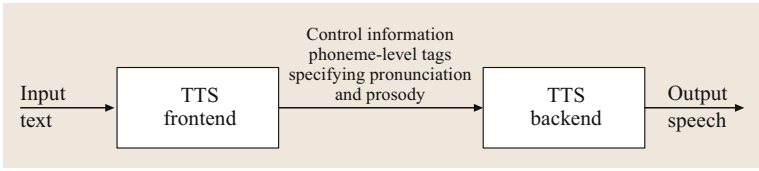



**Fig. 19.1** Block diagram of a text-to-speech (TTS) system

rendered as *Street* or as *Saint*, 'Dr.' as *Drive* or *Doctor*. 'IBM' is spelled out, while 'NASDAQ' is not.

Text markup can be used to control how the TTS engine renders its output. Proper markup may make the TTS output sound *intelligent*, such as letting the frontend know that it should use address mode for reading a street address. It may also tell the TTS engine to render a sentence with emotions like *angry*, *sad*, *happy*, or *neutral*. One widely used markup standard for synthesis is speech synthesis markup language (SSML) [19.4].

In addition to text preprocessing, linguistic–syntactic analysis tasks include a morphological analysis for proper word pronunciation and a syntactic analysis to facilitate accenting and phrasing and to handle ambiguities in the written text. Most TTS systems forego fully parsing the input text to limit computational complexity and because input consists often of incomplete sentence fragments.

Phonetic analysis focuses on the phone level within each word, tagging each phone with information about what sound to produce and how to produce it (speaking style, emphasis). As part of this process, it performs a grapheme-to-phoneme conversion (i. e., determining the exact pronunciation of each word of the input sentence) and homograph disambiguation (e.g., figuring out whether an input sentence uses the present tense or the past-tense version of the word *read*). For these tasks, it relies on dictionaries to look up word pronunciations, usually backed up by general letter-to-sound rules as a fallback or, for some languages, as the primary source for pronunciations. An etymological analysis helps provide clues about the meaning of words and explain irregularities in pronunciation, for example, in names. All of this information then drives the prosodic analysis.

Prosodic analysis determines the progression of intonation, speaking rate, and loudness across an utterance, which are ultimately represented at the phoneme level as pitch (fundamental frequency), duration, and amplitude. The intonation and timing parts largely determine the rhythm of a synthetic sentence in addition to syllabic stress (expressing the prominence of a syllable relative to its surrounding syllables). Repetition of rhythm and stress patterns across sentences contributes to listener fatigue and low perceptual scores. Therefore, proper prosody facilitates listener focus, highlights

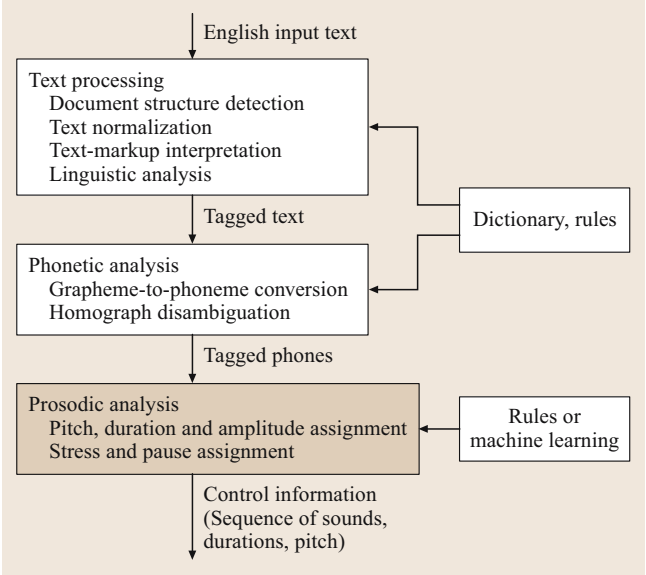



**Fig. 19.2** Tasks and processing in a TTS frontend

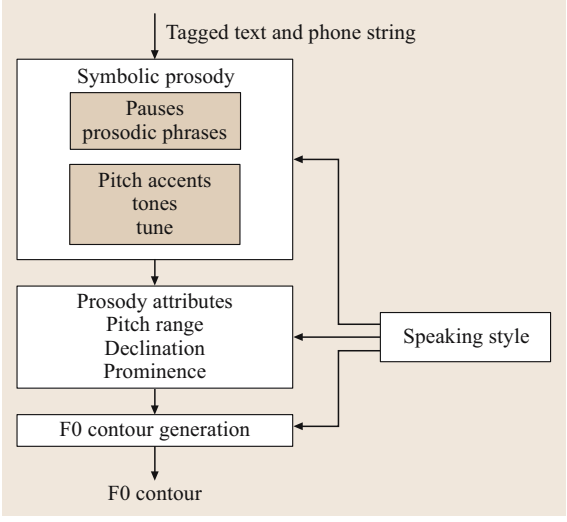



**Fig. 19.3** Block diagram of determining the pitch contour of a speech utterance

important or new content, or even conveys nuances of meaning.

Timing is an important part of prosody assignment, including durations of individual phones. There are several approaches to duration modeling, but two guiding principle approaches exist. The more common rule-based approach is to start with segment intrinsic durations and moderate these durations based on rules that combine a relatively small set of different types of data, extracted by prior analyses either additively or multiplicatively [19.5, 6]. A more-recent data-driven approach is to use large speech corpora and to apply machine learning methods involving a much larger set of control factors [19.7].

Intonation refers to the tonal/melodic parts of prosody. Given a certain speaking style, Fig. 19.3 summarizes the process of intonation/pitch assignment in terms of creating an F0 contour for a given utterance. Starting from a symbolic representation, for example in terms of tone and break indices (ToBI) [19.8], pauses and prosodic phrases are determined, followed by setting pitch accents and/or tones. Suprasegmental prosodic attributes like pitch range (the difference between the highs and lows in the voice of a speaker), declination (the gradual lowering of pitch over the course of a sentence), and the expression of prominence are certainly affected by the particular speaker's speaking style.

The final step towards arriving at an F0 contour uses the available symbolic prosody data to create a time series for F0. Stylized or actually observed pitch templates in a corpus might be used. A simpler approach is to use a generative model such as Fujisaki's [19.9] that expands pulse or step-like accent and phrase commands into continuous pitch contours, employing a source–filter model. More information on prosodic processing can be found in Chap. 23.

Finally, all of the frontend information is then forwarded to the TTS backend for synthesis. Note that TTS frontends have to be somewhat language specific. Pronunciation dictionaries and letter-to-sound rules used in the frontend may even be tailored to match the specific speaker's language accent and speech production peculiarities of the human voice that TTS tries to mimic.

### 19.1.2 TTS Backend

TTS backends are largely independent of TTS frontends. This gives TTS system designers the option to pick from different frontends and backends, provided the interface between them can be instantiated.

A TTS backend uses the information provided by the frontend to synthesize speech using a specific synthesis method. Traditionally, we distinguished *rule-based* versus *corpus-based* synthesis methods. The former included articulatory synthesis and formant synthesis, while the latter was almost synonymous with concatenative synthesis. Note, however, that one should keep the paradigm of knowledge representation (rule versus corpus) separate from the synthesis method itself. The reason for this is that the clear-cut distinction no longer exists. We now have examples of articulatory and/or formant synthesizers that incorporate data-driven methods and other examples of using rules in concatenative synthesize backends. However, it is still true that every given synthesis method is somewhat dependent on the speech representation it uses. We will present details on speech representations in Sect. 19.2 below. First, however, we will review the principle differences between articulatory, formant, and concatenative synthesis methods.

#### Articulatory Synthesis

Articulatory synthesis uses mechanical and acoustic models of speech production to synthesize speech. These models can be as intricate as solving the Navier–Stokes partial differential equation for the vocal tract and the glottis [19.10].

The process of computing vocal tract acoustics from its geometry is sketched in Fig. 19.4. The dynamic movements of the articulators (e.g., jaw, tongue, lips, velum, etc.), together with the vocal-tract excitation (not shown

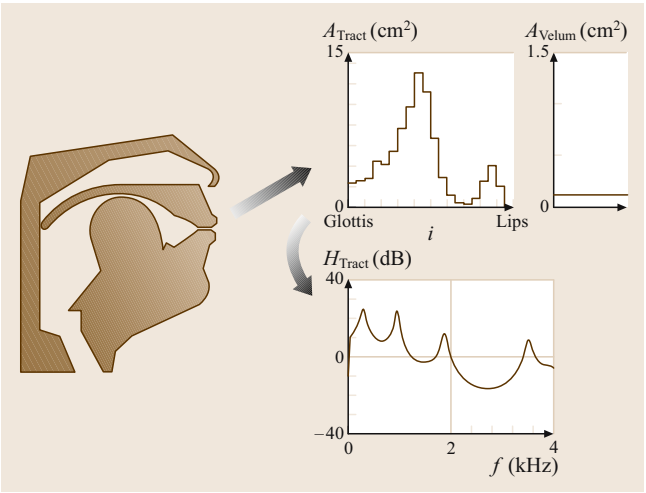



**Fig. 19.4** The human speech-production system with glottis, vocal tract, and nasal tract

in Fig. 19.4; vibrations of the vocal folds of the glottis for voiced sounds, aspiration at the glottis and fricative noise generated at vocal tract constrictions for unvoiced sounds) allow for speech synthesis using articulatory parameters as controls. A complete articulatory synthesizer would use information such as the forces and timings of all relevant groups of articulatory muscles to drive a realistic mathematical model of glottis and vocal tract.

Clearly, any measurements of the vocal tract geometry are helpful for articulatory synthesis research. Such measurements are obtained by one of several techniques. For example, ultrasound pictures of the tongue shapes [19.11] can reveal important details of tongue movements. Nuclear magnetic resonance [19.12] is now fast enough to track vocal tract dynamics in real time.

From these kinds of data, so-called articulatory models (static models, e.g. [19.13], or dynamic models, e.g. [19.14]) then convert positional coordinates of the articulators to a series $A_i^{\mathrm{TRACT}}$, $i = 1, \ldots, p$ of piecewise-constant vocal-tract cross-sectional areas that can be used to compute the vocal-tract acoustics. An example is shown in the top right of Fig. 19.4.

There is a multitude of ways to characterize the acoustics of the vocal tract. Here we will focus on a frequency-domain method (see [19.1] for others). Assuming plane wave propagation in the vocal tract (an assumption that is valid for frequencies under 4 kHz), the input/output characteristics of the vocal tract can be characterized by its chain matrix (also known as its $ABCD$ matrix) that links sound pressure at the lips $P_{\mathrm{L}}$ and volume flow at the lips $U_{\mathrm{L}}$to their counterparts at the glottis, $P_{\mathrm{G}}$ and $U_{\mathrm{G}}$, respectively:

$$\begin{pmatrix} P_{\mathrm{L}} \\ U_{\mathrm{L}} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} P_{\mathrm{G}} \\ U_{\mathrm{G}} \end{pmatrix} , \tag{19.1}$$

where the tract ABCD matrix is the product of individual section matrices as follows, each section with a different cross-sectional area $A_i^{\mathrm{TRACT}}$ and, optionally, length:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \prod_{k=1}^{p} \begin{pmatrix} A_k & B_k \\ C_k & D_k \end{pmatrix} . \tag{19.2}$$

Here the ABCD matrix elements are

$$\begin{aligned} A_i &= \cosh(\Gamma_i \Delta) , & B_i &= -Z_i \sinh(\Gamma_i \Delta) \\ C_i &= -Z_i^{-1} \sinh(\Gamma_i \Delta) , & D_i &= \cosh(\Gamma_i \Delta) , \end{aligned} \tag{19.3}$$

with characteristic impedances $Z_i$, lossy propagation constants $\Gamma_i$, and vocal-tract section length $\Delta$ (assumed to be the same for all sections). $Z_i$ and $\Gamma_i$ depend on the cross-sectional area $A_i^{\mathrm{TRACT}}$. Both of these quantities are determined by the (per-unit-length) inductance $L_i$, capacitance $C_i$, and the loss elements resistance $R_i$ and conductance $G_i$ [19.1, p. 34], and are given by:

$$Z_i = \sqrt{\frac{(R_i + \mathrm{i}\omega L_i)}{(G_i + \mathrm{i}\omega C_i)}} \tag{19.4}$$

and

$$\Gamma_i = \sqrt{(R_i + \mathrm{i}\omega L_i)(G_i + \mathrm{i}\omega C_i)} \tag{19.5}$$

with $\mathrm{i} = \sqrt{-1}$ and $\omega$ the radian frequency. There are several meaningful quantities that are easily computed from (19.1). One that we will need in Sect. 19.2 below is the *vocal-tract transfer function* that relates volume velocity at the lips to glottal volume velocity (glottal flow):

$$H = \frac{U_{\mathrm{L}}}{U_{\mathrm{G}}} = \frac{1}{A - C Z_{\mathrm{L}}} . \tag{19.6}$$

Here $Z_{\mathrm{L}}$ is the radiation impedance at the lips. From (19.6), we compute the sound pressure at the lips as $P_{\mathrm{L}} = H_{\mathrm{TRACT}} U_{\mathrm{G}}$, where $H_{\mathrm{TRACT}} = Z_{\mathrm{L}} H$. An example for $H_{\mathrm{TRACT}}$ is depicted in the lower right of Fig. 19.4.

The inverse Fourier transform of $H_{\mathrm{TRACT}}$ is the impulse response of the vocal tract that, convolved with the actual glottal flow, results in synthetic speech. Many more details including information on how to model the glottal and noise excitations can be found, for example, in [19.15].

In general, articulatory synthesis produces intelligible synthetic speech, but today its output is still far from natural sounding. The reason for this fact is that each of the several models that are employed in the process has to be extremely accurate in reproducing the characteristics of a given *single* speaker. Also note that most of these models largely depend on expert guesses (*rules*) and not enough on observed data. While the promise of high-quality articulatory synthesis is the possibility of easily modifying any model with the objective of changing underlying speaker characteristics, the complexities and computational requirements of this method still hamper its broad adaptation in practical speech synthesizers.

### Formant Synthesis

Formant synthesis encapsulates the vocal tract as a black box, aiming to reproduce its input/output characteristics, namely (19.6), in form of formant filters. Further approximations are made for the radiation impedance $Z_{\mathrm{L}}$ and for the excitation mechanisms. More generally, formant synthesizers instantiate the so-called source–filter theory of speech production in a terminal analog fashion. The theory postulates that there is no influence of the

filter back on the source (contrary to articulatory synthesizers). Instead of vocal tract areas, formant frequencies and bandwidths are used as control factors. The filters are excited either with a pulse train or with noise.

Assuming $Z_L = 0$ in (19.6) and neglecting any losses in the vocal tract, the transfer function (19.6) simplifies to $H = A^{-1}$. For the $i$-th tract section, the $A$ element of its chain matrix becomes (19.3):

$$A_i = \cos\left(\frac{\omega\Delta}{c}\right) \quad \forall i = 1, \ldots, p\,. \tag{19.7}$$

Here $c$ is the speed of sound. Representing the vocal tract using $p$ sections, each of length $\Delta$, the round-trip wave propagation delay through a section is $2\Delta/c$. This delay corresponds to the delay operator $z^{-1}$ in a digital implementation. Consequently, (19.7) may be written as

$$\cos\left(\frac{\omega\Delta}{c}\right) = \frac{z^{-1/2} + z^{+1/2}}{2} = \frac{z^{1/2}}{2}(1 + z^{-1})\,. \tag{19.8}$$

By chaining the individual terms for the entire vocal tract, we get a polynomial of order $p$, of the form

$$\frac{U_L}{U_G} = \frac{z^{-p/2}}{\sum_{i=0}^{p} a_i z^{-i}} \Rightarrow \sum_{i=0}^{p} a_i z^{-i} U_L = z^{-p/2} U_G\,. \tag{19.9}$$

In the time domain, the right-hand form of (19.9) expands to

$$\sum_{i=0}^{p} a_i u_L(n-i) = u_G\left(n - \frac{p}{2}\right). \tag{19.10}$$

Making the simple substitutions $s(n) \equiv u_G$ and $u_G(n - p/2) \equiv e(n)$, we arrive at the standard linear prediction coding (LPC) equation [19.16]:

$$s(n) = e(n) - \sum_{i=1}^{p} a_i s(n-i) \quad \text{with } a_0 = 1 \tag{19.11}$$

showing that we can represent a lossless non-nasal vocal tract using a standard LPC polynomial.

*Serial Formant Synthesizer.* Assuming $p$ even and only conjugate complex roots, the LPC polynomial (19.9) can be factored into its roots, resulting in

$$\begin{aligned}\frac{U_L}{U_G} &= \prod_{i=1}^{p/2} \frac{(1 - z_i)(1 - z_i^*)}{(1 - z_i z^{-1})(1 - z_i^* z^{-1})} \\ &= \prod_{i=1}^{p/2} \frac{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) + \mathrm{e}^{-2\sigma_i T}}{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) z^{-1} + \mathrm{e}^{-2\sigma_i T} z^{-2}}\,,\end{aligned} \tag{19.12}$$

where the $i$-th root $z_i = \mathrm{e}^{-\sigma_i T}\mathrm{e}^{-\mathrm{i}\omega_i T}$, the asterisk (*) denotes the conjugate complex, and $T$ is the sampling interval. Note that the frequency-independent numerator of each factor (i. e., the filter element) normalizes the gain of that factor to be unity at zero frequency ($z = 1$).

Figure 19.5 shows the frequency characteristic of a single filter element. Note that the peak gain is given by its quality factor $Q_i = \omega_i/2|\sigma_i|$, and the bandwidth of the filter in Hertz is $B_i = F_i/Q_i = \omega_i/2\pi Q_i = |\sigma_i|/\pi$. Cascading $p/2$ second-order filter sections instantiates (19.12). Note that specifying the formant frequencies $F_i$ and bandwidths $B_i$ fully determines the transfer function of the chained filters.

*Parallel Formant Synthesizer.* A serial synthesizer is a good approach for open, non-nasal vocal tract configurations (e.g., for vowels and liquids). In order to reproduce the speech spectra of obstruents and nasals, however, we need to control individual filter section amplitudes in addition to formant frequencies and bandwidths. This is possible if individual filter sections are connected in parallel, each with its own gain factor. The price we pay for the additional flexibility in matching spectrum levels is that spurious zeros are introduced between the formant frequencies.

In order to arrive at a parallel filter representation, we start from (19.12) and expand the product form into a summation form through partial fractions:

$$\begin{aligned}\frac{U_L}{U_G} &= \prod_{i=1}^{p/2} \frac{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) + \mathrm{e}^{-2\sigma_i T}}{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) z^{-1} + \mathrm{e}^{-2\sigma_i T} z^{-2}}\,, \\ &= \sum_{i=1}^{p/2} \frac{A_i - B_i z^{-1}}{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) z^{-1} + \mathrm{e}^{-2\sigma_i T} z^{-2}}\,, \\ &\approx \sum_{i=1}^{p/2} \frac{A_i}{1 - 2\mathrm{e}^{-\sigma_i T}\cos(\omega_i T) z^{-1} + \mathrm{e}^{-2\sigma_i T} z^{-2}}\,.\end{aligned} \tag{19.13}$$

Here, the residual terms $A_i$ (the *gain*) and $B_i$ (the phasing of the resonance, neglected in most practical realizations) allow for the additional control that we desire.

*Combined Serial/Parallel Formant Synthesis.* Including a serial filter realization for vowels and liquids and a parallel filter realization for nasals and fricatives is clearly advantageous. Figure 19.6 shows a block diagram of Klatt's synthesizer [19.17]. Also shown is a parametric voicing source that can be tuned for dif-

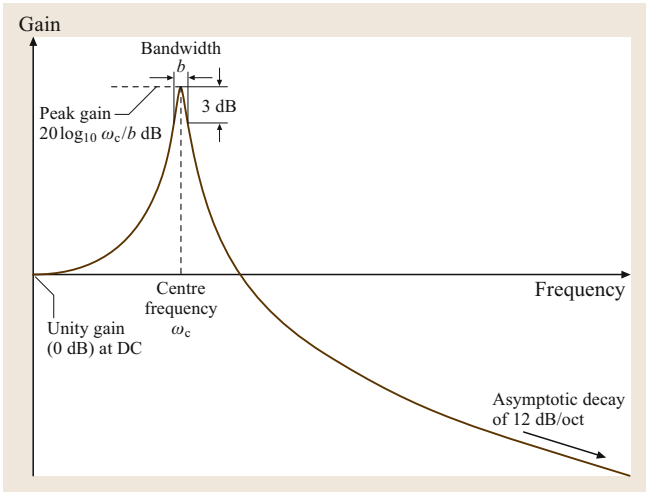



**Fig. 19.5** The frequency response of a single second-order filter section of a formant synthesizer with angular center frequency $\omega_c$ and bandwidth $b = 2\pi B$

ferent voices or speaking styles. The synthesizer also includes an aspiration/breathiness component added to the glottal source that assists in the production of voiced fricatives. A commercial system that makes use of articulatory and acoustic-phonetic knowledge and drives the Klatt synthesizer is HLSyn [19.18]. Readers interested in rule-based and formant synthesis can find more information in Chap. 20.

Formant synthesis is the preferred method for creating speech stimuli for research in speech perception, given the high level of control such experiments require. Another advantage of formant synthesizers is their moderate computational requirements. For example, memory requirements can be as little as 1 MB. Therefore, formant synthesis is ideal for applications such as in handheld devices, for example, talking dictionaries, calendars, etc., or even in cell phones (e.g., for reading back names and key presses). Intelligibility is high, but naturalness is generally lacking. Matching a target speaker is usually impossible.

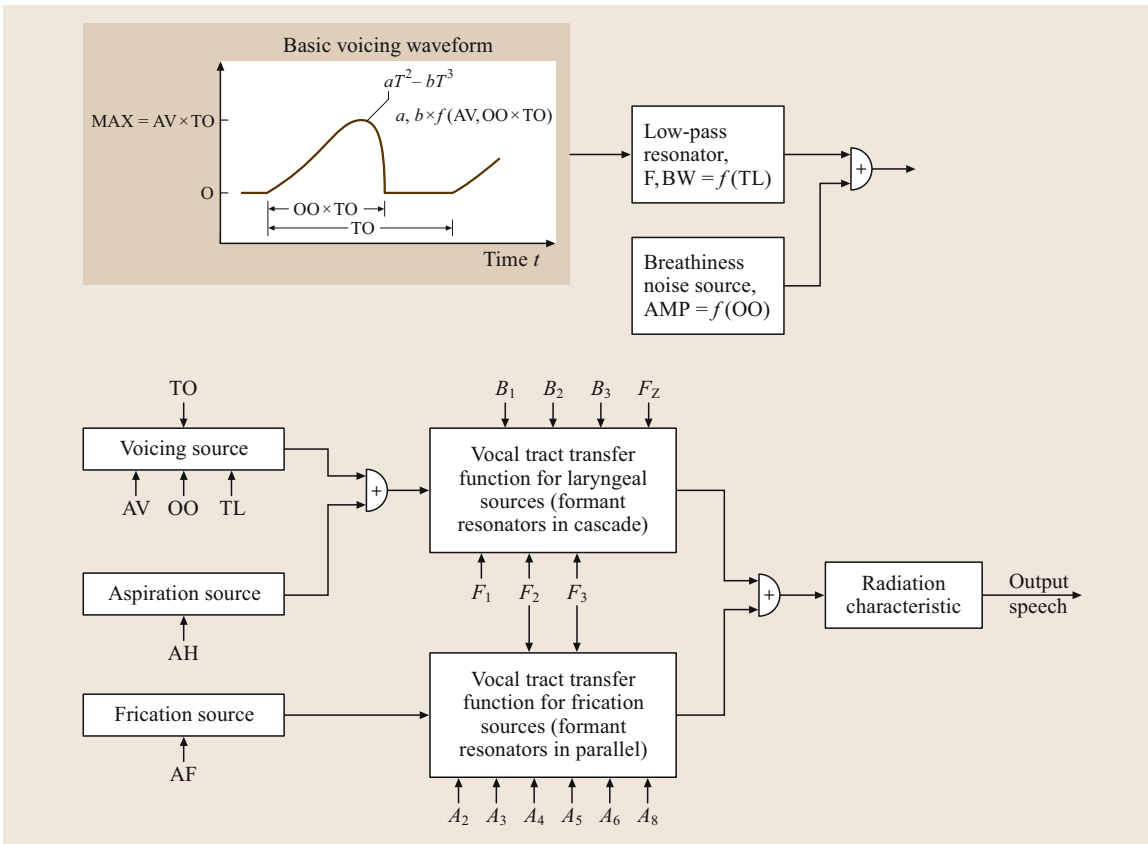



**Fig. 19.6** Hybrid serial/parallel synthesizer of Klatt

### Concatenative Synthesis

Concatenative synthesis employs segments of recorded speech from a prerecorded inventory (voice database). A block diagram of a typical concatenative TTS system is shown in Fig. 19.7. The phonetic and prosody targets provided by the frontend on the left act as a *fuzzy* query into the voice database that stores all sound units. The sequence of units that optimally match the input query for a to-be-synthesized sentence are retrieved, assembled, and handed off to the speech waveform modification and synthesis module on the right. Unit assembly and waveform synthesis are both backend tasks. Note that changing the language requires changing the frontend, but usually also requires changing the store of sound units. Changing the target speaker (e.g., from male to female) also usually requires changing the store of sound units.

The sound store may contain one or more of different types of speech units. For example, using phones (e.g., about 50 for English) leads to unsatisfactory joints between units because of the large coarticulatory effects between adjacent phones. Intuitively, longer units are more likely to result in higher-quality synthesis, given that the rate of concatenations (how many unit-to-unit transitions occur per second of speech) is lower than in the case of shorter units. For general TTS applications, however, longer units are impractical because of the tremendous multiplicity of possible unit variants [19.19]. Therefore, using whole word units is impossible because of the huge demands on a voice talent that would have to read a few hundreds of thousands of words in a consistent voice and manner but also in different prosodic contexts. Given these contradictory requirements, most TTS implementations until the mid-1990s used one of two types of inventory units: the diphone or the demisyllable.

A diphone is the segment of speech that starts at the middle of one phone and extends to the middle of the next phone. It brackets exactly one phone-to-phone transition. The cut points at the middle of the phones are in the acoustically most stable region. Note that the average length of a diphone is identical to that of a phone. Because each diphone has *two* phone identities, the theoretical size of a diphone inventory is quadratic in the number of the phones a language has. However, because of the fact that a language might not use all of these combinations, the actual number of diphones that needs to be in the inventory is significantly smaller. For example, English requires an inventory of at least 1000 diphones [19.20].

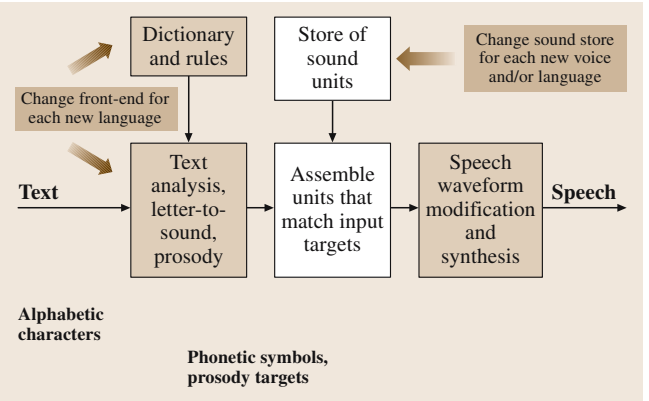



**Fig. 19.7** Block diagram of a concatenative synthesizer

Demisyllables are alternative units for concatenative synthesis [19.21]. A demisyllable encompasses half a syllable; that is, either the syllable-initial portion up to the first half of the syllable nucleus, or the syllable-final portion starting from the second half of the syllable nucleus. The number of demisyllables in English is roughly the same as the number of diphones. Because demisyllable units are usually longer than diphones and allow for better capture of longer-term coarticulation effects compared to diphones, they should pose fewer concatenation problems.

With a minimum-size inventory, units must be modified by signal processing to match the frontend targets and to smooth over the concatenation points. After recording the voice talent, special care must be spent on selecting the most appropriate instances of each unit to be included in the inventory. Semi-automatic tools can help in finding those instances of units that minimize concatenation problems when used in later synthesis [19.22, pp. 199-222].

The offline (not carried out at synthesis time) process of selecting optimum units can be improved further by incorporating it in the online synthesis process. The quality of the synthesized speech should increase given that full context information is available at synthesis time, but only general statistical information exists when done offline. This idea originated at advanced telecommunications research (ATR) in Japan in the late 1980s [19.23]. The resulting new subcategory of concatenative synthesis is now called *unit-selection synthesis* [19.24]. It was largely enabled by the ever-increasing power of computers. Unit-selection synthesis adds the problem of how to do fast searches of potentially millions of options to the set of tasks in speech synthesis.

*Unit-Selection Synthesis.* Different from earlier concatenative synthesizers, unit-selection synthesis automatically picks the optimal synthesis units (on the fly) from an inventory that can contain thousands of examples of a specific unit (such as a diphone), and concatenates the selected sequence of units to produce the synthetic speech. This process is outlined in Fig. 19.8, which shows how the method must dynamically find the best path through the unit-selection network corresponding to the sounds for the word 'two'. The optimal selection of the sequence of units depends on factors such as spectral similarity at unit boundaries and on matching prosodic targets set by the front-end. In the following, we will highlight the essential steps, as used in the CHATR speech synthesis system [19.25]. Many more aspects and details are covered in Chap. 21.

There are two principal kinds of cost measures that guide unit selection. One is unit segmental distortion (USD); another name for it is *target cost*. USD captures the spectral differences between candidate units in the inventory and a hypothetical target, noting that the real target is not available at synthesis time. Therefore, USDs have to be estimated from available features for the target, given the sentence to be synthesized. An example would be to represent the cost of selecting an /ah/ from the word *want* to synthesize the word *cart*. The USD in this case is the penalty for violating the desired /ah-r/ context by selecting /ah/ in an /ah-n/ context. USDs are represented by the nodes in Fig. 19.8.

The other kind of cost is unit concatenative distortion (UCD), also called *join cost*. It measures the spectral discontinuity across the boundaries of candidate units. Since the candidates exist in the inventory, UCDs can be computed directly from the candidates. UCDs are represented by the arcs (arrows) in Fig. 19.8.

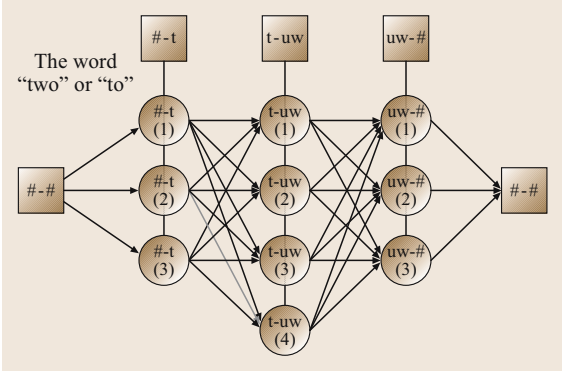



**Fig. 19.8** Viterbi search of a diphone inventory for the words 'two' or 'to'

We formalize the unit selection process as follows. We start with denoting $t_j$ to be the $j$-th (idealized) target unit in the string $T = \{t_1, \ldots, t_j, \ldots, t_N\}$ that spans the sentence to be synthesized. Similarly, we form a set of $N$ corresponding units selected from the inventory to be $\Theta = \{\theta_1, \ldots, \theta_j, \theta_{j+1}, \ldots, \theta_N\}$. Then the total unit selection cost for any path through the network depicted in Fig. 19.8 is given by

$$d(\Theta, T) = \sum_{j=1}^{N} d_{\mathrm{u}}(\theta_j, t_j) + \sum_{j=1}^{N-1} d_{\mathrm{t}}(\theta_j, \theta_{j+1}) \quad (19.14)$$

where $d_{\mathrm{u}}(\theta_j, t_j)$ is the USD or target cost between target unit $t_j$ and candidate unit $\theta_j$ and $d_{\mathrm{t}}(\theta_j, \theta_{j+1})$ denotes the UCD or join cost between the two neighboring units $\theta_j$ and $\theta_{j+1}$. The optimal path (the optimal sequence of units) can be found efficiently using the Viterbi method that scales linearly with the grid size (the number of time steps multiplied by the number of candidate units in a column of the network) [19.26].

The join cost has to be zero if both units are natural neighbors in the inventory, that is, if they were recorded in sequence (e.g., came from the same word) because, by definition, there is no spectral mismatch in this case. The exact type of join costs to use is an area of active research that tries to gauge a listener's perceptual *annoyance* of spectral mismatches [19.27].

USD or target costs are conceptually more difficult to understand. In practice, they are also more difficult to instantiate. Formally, we start with

$$d_{\mathrm{u}}(\theta_j, t_j) = \sum_{i=1}^{q} w_i^{\mathrm{t}} \phi_i[\mathrm{T}_i(f_i^{\theta_j}), \mathrm{T}_i(f_i^{t_j})] \,, \quad (19.15)$$

where the $w_i^{\mathrm{t}}$ are trained weights (see later), and the $\mathrm{T}_i(\cdot)$ can either be continuous functions for a set of features $f_i$ such as segmental pitch, power, or duration, or a set of integers in case of categorical features $f_i$ such as unit identity, phonetic class, etc. In the latter case, $\phi_i$ is looked up in a distance table. Otherwise, a simple quadratic distance may suffice:

$$\phi_i[\mathrm{T}_i(f_i^{\theta_j}), \mathrm{T}_i(f_i^{t_j})] = [\mathrm{T}_i(f_i^{\theta_j}) - \mathrm{T}_i(f_i^{t_j})]^2 \,. \quad (19.16)$$

The training of the weights $w_i^{\mathrm{t}}$ is done offline. For each phoneme in each phonetic class in the training speech database (which might be the whole recorded inventory), we treat each exemplar of a unit as a target and all others as candidate units. Let $L$ be the number of phones in the phone set and $M$ the number of lowest target-cost

candidates that we wish to consider. Then, a $(L \times M) \times q$ rectangular matrix $\boldsymbol{\Phi}$ for this set is given by

$$\boldsymbol{\Phi}_{(L\times M)\times q} = [\phi_1, \ldots, \phi_{\mathrm{L}}]^{\mathrm{T}} , \tag{19.17}$$

where $T$ denotes matrix transposition. Similarly,

$$\boldsymbol{w}^{\mathrm{t}}_{qx1} = [w^{\mathrm{t}}_1, \ldots, w^{\mathrm{t}}_q]^{\mathrm{T}} \tag{19.18}$$

is the desired vector of weights. Furthermore,

$$\boldsymbol{a}_{(L\times M)\times 1} = [d_{\mathrm{a}}(\theta_1, t_1) \ldots d_{\mathrm{a}}(\theta_M, t_1)| \ldots |d_{\mathrm{a}}(\theta_1, t_L) \ldots d_{\mathrm{a}}(\theta_M, t_L)]^{\mathrm{T}} \tag{19.19}$$

is a vector of acoustic distances $d_{\mathrm{a}}$, similar to the UCD join cost, but now applied over all corresponding frames of a pair of candidates and targets units using, for example, dynamic time warping (DTW). The optimal weight set $\boldsymbol{w}^{\mathrm{t}}$ then solves the least-squares system of linear equations

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w}^{\mathrm{t}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a} . \tag{19.20}$$

Because of coverage issues, unit selection appears to be well suited for limited-domain applications such as synthesizing telephone numbers to be embedded within a fixed carrier sentence. Even for more open-domain applications, such as email reading, advanced unit selection can reduce the number of unit-to-unit transitions per sentence synthesized and, consequently, increase the segmental quality of the synthetic output. Furthermore, the use of multiple instantiations of a unit in the inventory, taken from different linguistic and prosodic contexts, reduces the need for prosody modifications that potentially degrade naturalness. Consequently, today's most natural sounding unit-selection systems employ inventories that comprise many dozens of hours of speech, in particular when more than one speaking style (e.g., newsreader) or emotion (e.g., neutral) is to be covered [19.28] (Chap. 25).

*HMM-Based Synthesis.* An alternative approach to instantiating the USD/target costs in unit selection is based on clustering contexts offline and selecting the best unit at runtime from the optimal cluster, given the context information for the sentence to be synthesized [19.29]. This task is well known in acoustic modeling in speech recognition. Therefore, a similar approach using hidden Markov models (HMMs) can also be used for this step in synthesis. Using HMM states as units and adding a speech generation method from HMM state data (spectra and excitation) leads to an HMM-based synthesis method [19.30].

In the next section, we will look into some of the signal representations typically used in concatenative TTS systems. The idea is that limited signal processing may alleviate the combinatorial problems [19.19] that potentially degrade even the intelligibility of a TTS system because of the lack of appropriate units in the inventory. Signal processing may also allow speaker transformations, thus lowering the cost of generating new voice inventories. We will look into speaker and speech transformations in Sect. 19.3.

## 19.2 Speech Representations and Signal Processing for Concatenative Synthesis

A speech representation suitable for concatenative synthesis satisfies the following requirements.

1. In order to minimize disk and/or memory requirements, voice inventories are stored in compressed form at high quality. The encoding/decoding process should be of low computational complexity.
2. Encoding/decoding has to be perceptually transparent for maximum naturalness. Any perceptually detectable distortions should be avoided. Early examples of concatenative synthesizers violated this requirement, sacrificing naturalness while maximizing intelligibility.
3. Algorithms used for encoding and decoding need to allow for random access to individual units of speech (e.g., diphones).
4. Ideally, the selected speech representation must allow for natural-sounding modifications of prosody. Note, however, that for most signal processing algorithms, modifying pitch more than a few percent may destroy naturalness.
5. Advanced signal processing might avoid extensive and impractically long recording sessions. Advanced voice conversion might be used to approximate emotional speaking styles [19.28] (Chap. 25). However, many people believe that today's voice conversion algorithms still do not produce the necessary quality.

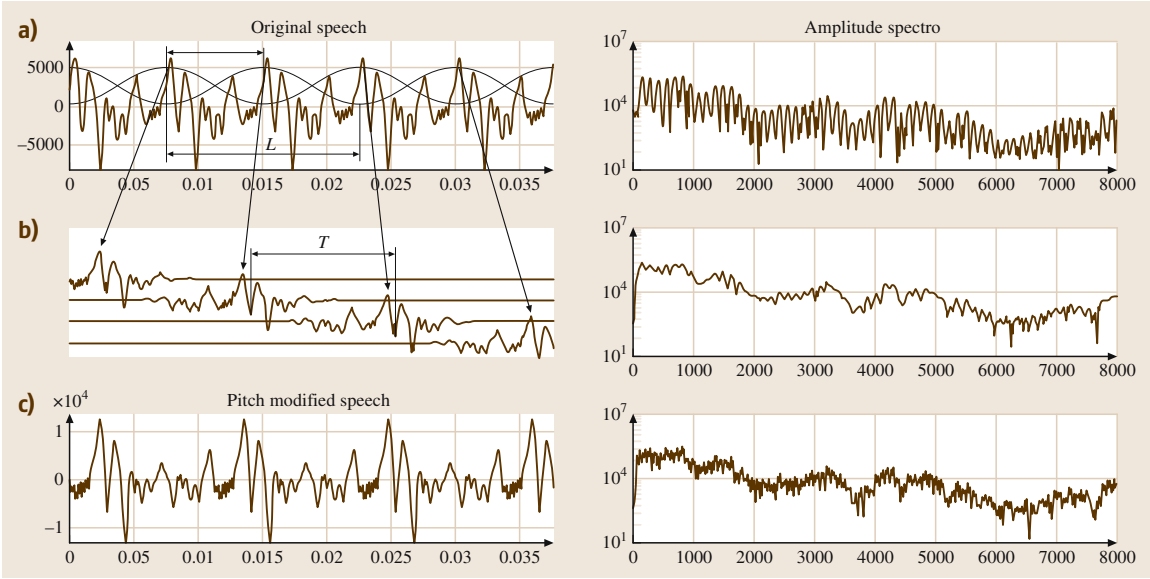



**Fig. 19.9a–c** Time-domain pitch synchronous overlap add (TD-PSOLA, after [19.31, Fig. 10.1, p. 252])

Given this list of requirements, in the following, we will now outline three classes of speech signal processing algorithms with their (native) speech representations: low-complexity time-domain algorithms such as TD-PSOLA (using the waveform of the speech signal directly), LPC-based representations, and frequency domain-based speech representations. Common to all of them is the fact that changes in segment durations are almost trivially tied to modifications of parameter update rates. They are different, however, in the manner in which they facilitate pitch changes and allow or disallow changes in spectral envelopes and spectral fine structure of the speech signal.

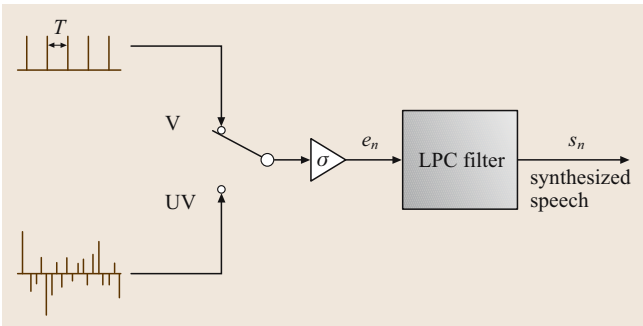



**Fig. 19.10** Block diagram of LPC synthesis. Voiced excitation is facilitated by periodic pulses with pitch period $T$. Gaussian noise is used for unvoiced excitation. The excitation is amplified by a gain factor $\sigma$

## 19.2.1 Time-Domain Pitch Synchronous Overlap Add (TD-PSOLA)

In its simplest form, time-domain pitch-synchronous overlap add (TD-PSOLA; see, e.g., [19.32]) consists of extracting two pitch periods from a voiced speech signal, windowing each segment with a Hanning window centered on one glottal closure (maximum excitation) point. This positioning of the window weighs down the signal in the vicinity of the previous and next glottal closure events as shown in Fig. 19.9. Note that the figure depicts time-domain signals on the left and spectra on the right. Panel (a) shows the original signal. The extracted and windowed signal traces (b) are recombined in (c), after shortening (for increased pitch) or padding with zero amplitude signal samples (for lowering pitch). The resulting reconstructed signal of a different pitch value is shown in panel (c). A specific variant of TD-PSOLA is to apply the method on the LPC filter excitation (pitch-synchronous residual excited linear prediction (PSRELP), see [19.33]) instead of on the speech waveform. As an option, PSRELP allows for smoothing of the spectral envelope at concatenation points by modifying the LPC filter, alleviating concatenation problems somewhat. Both methods, however, can

Amazon EX1010 Page 447

lead to audible glitches since there is no obvious way of smoothing the waveforms that TD-PSOLA is applied to. While the PSRELP variant allows for changing the spectral envelope, the spectral fine structure cannot be controlled by either method.

### 19.2.2 LPC-Based Synthesis

We introduced linear prediction (LPC) in (19.11) in Sect. 19.1.2, as a stepping stone towards formant synthesis. In the simplest version of LPC synthesis, depicted in Fig. 19.10, the excitation signal $e(n)$ is either a train of pulses (for voiced speech) or white Gaussian noise (for unvoiced speech). The LPC filter gives the synthetic speech the desired spectral envelope, matching the formants without explicit formant identification. This is enough to create intelligible speech but fails to produce natural-sounding speech because of the simplistic excitation model. This kind of LPC synthesis clearly is not natural sounding. However, because of its relationship to vocal tract geometry, LPC lends itself to spectral modifications that are based on geometric differences of vocal tracts (e.g., male-to-female speech conversion). In addition, note that some TTS systems employ advanced LPC-based encoders and decoders such as code-excited linear prediction (CELP) or multipulse [19.34], and/or coders that employ glottal model excitation pulses. While using CELP or multipulse excitation raises the quality dramatically, only glottal models have the advantage of meaningful interpretation in terms of human speech production and, consequently, lend themselves to sensible smoothing and interpolation. It is difficult, however, to match an individual speaker's voice characteristic with a glottal model-based synthesizer. Both methods allow for changes in spectral envelope, but fall somewhat short when we would like fine control of the spectral details.

### 19.2.3 Sinusoidal Synthesis

The main disadvantage of the LPC synthesis of Fig. 19.10 is its lack of faithful reproduction of a speaker's voice characteristics and the lack of fine control of the speech spectrum. It can be shown that the amplitude and phase relationships of the first few harmonics contain crucial information on speaker identity (independent of the super-segmental prosodic and phonetic cues). Therefore, modeling speech harmonics directly using a sinusoidal speech representation seems to be a more appropriate approach towards meeting the transparency requirement. As with TD-PSOLA, in sinusoidal synthesis we have the choice of including LPC filtering to allow for smoothing at concatenation points and for other modifications of the spectral envelope. In contrast to TD-PSOLA, here pitch modifications are done in the frequency domain by stretching or compressing the spectrum.

Let $s(t)$ be a signal of interest, either speech or LPC excitation, then [19.35]

$$\begin{aligned} s(t) &= \sum_{l=1}^{L(t)} A_l(t)\cos[\boldsymbol{\Omega}_l(t)] \text{ , with } \boldsymbol{\Omega}_l(t) \\ &= \int_{t_l}^{t} \omega_l(\sigma)\,\mathrm{d}\sigma + \phi_l \text{ .} \end{aligned} \tag{19.21}$$

Here $A_l(t)$ is the slowly time-varying amplitude of the $l$-th sine wave, $\boldsymbol{\Omega}_l(t)$ its instantaneous phase, $\omega_l$ the radian frequency of the $l$-th sine wave, and $\phi_l$ its phase. During analysis of a speech signal, we have to estimate the harmonic frequencies and phases. Potential problems are the *birth* and *death* of specific sinusoids and the representation of unvoiced speech.

Several variants of the sinusoidal approach exist. One is the harmonic-plus-noise model representation (HNM) [19.36]. It splits the speech spectrum into a low-frequency voiced part that is modeled by the sinusoidal approach, and a high-frequency unvoiced part that represents the noise-like characteristic of speech spectra at high frequencies. Another approach is Straight [19.37], which focuses on highly accurate estimation of the harmonic amplitudes.

## 19.3 Speech Signal Transformation Principles

The goal of speech signal transformation is to create output speech with characteristics that are different from the speech from which it was derived. This may include changing speaker identity (making the output speech sound like a different speaker), or may convert speech recorded in a neutral newsreader style so that it sounds angry or happy (change its emotional tone). Therefore, one important reason to transform speech is to regenerate its variability that limited recordings cannot capture [19.28]. There are many kinds of variability

in speech, including speaker variability, linguistic inter and intra-speaker variability, and task variability. The acoustic correlates of these dimensions are variations in pitch, duration, amplitude, and spectrum. We already discussed how to change these low-level attributes, with the exception of spectral characteristics, which we will cover in this section.

We will discuss voice transformation to change speaker identity, voice alteration targeted at changing gender, individual differences, or even dialect, and modifications to speaking style and emotions for a given speaker. Although the methods outlined below may serve all of these applications, there are important differences between them. While changing the speaker identity can multiply the return on the effort that went into creating a single high-quality TTS voice by enabling offering several derived voices instead of just the one originally recorded, modifying intra-speaker characteristics such as speaking style or emotion avoids recording voice databases in that specific style or emotion.

What are the speech and speaker characteristics that might be changed by signal processing? One component of linguistic variability of speech is caused by variations in intonation. In addition, phonemes exhibit different spectral properties depending on context (coarticulation), speaking style/emotion, and speaker identity. Average pitch and formant frequencies are higher for females than for males. Different dialects cause the use of different phonemes for a word and/or the use of different allophones for the same phoneme in a given context. In the following, we will highlight methods for modifying spectral characteristics. However, we will also look at *individualized* prosody models that capture a specific speaker and/or emotional/speaking style. Details on more-expressive speech synthesis can be found in Chap. 25.

### 19.3.1 Prosody Transformation Principles

Changing segmental signal characteristics such as duration, amplitude, and pitch enables us to change supersegmental aspects of prosody as well with the goal of conveying a different emotion or a different speech act. This assumes, however, that we know what longer-term changes are required. Again, a data-driven approach seems most promising for capturing the way a specific speaker uses prosody in a specific context (e.g., in a dialog scenario) and for a specific speech act (e.g., a greeting). Prosody models trained on this information then allow a TTS system to request the right prosody for the given context. For a unit-selection synthesizer, this allows the unit selection module to do a good job of retrieving the correct speech segments that might minimize prosodic changes that might be necessary.

Any of the parametric intonation/prosody models summarized previously in Sect. 19.1.1 and detailed in Chap. 23 may be included in the signal transformation framework introduced below, provided that it includes model parameters of the chosen prosody model or representation.

### 19.3.2 Principle Methods for Changing Speaker Characteristics and Speaking Style

Here we discuss ways to transform the speech of one speaker to sound like speech from another speaker. The same methods can be used to change the speaking style or emotional tone for a given speaker. The simplest approach for mapping source speech (what we have) to target speech (what we intend to make it sound like) with another set of characteristics is to use two codebooks of vector-quantized parameter vectors, each trained on one of the two sets. Then we train a mapping function between corresponding codebook entries using a parallel speech corpus where both speakers uttered the same sentences. This technique was pioneered by Abe at Nippon Telephone & Telegraph (NTT) in Japan [19.38]. Time alignment of vectors from one set with the corresponding vectors from the other is not trivial because of differences in vowel reductions, phone omissions or additions, or other individual pronunciation differences. Dynamic time warping (DTW) may be used to establish this correspondence and histograms of these correspondences serve as weights for the mapping function:

$$\boldsymbol{S}_i^{(A\to B)} = \frac{\sum_{j=1}^{J} w_{ij}\boldsymbol{S}_j^{(B)}}{\sum_{j=1}^{J} w_{ij}} \,. \tag{19.22}$$

Here $A$ and $B$ represent source and target sets, respectively, $\boldsymbol{S}_i^{(A\to B)}$ is the $i$-th speech vector of the mapped source codebook, and $w_{ij}$ are the histogram counts of how often vector $i$ of codebook $A$ was found to correspond to vector $j$ in codebook $B$, itself containing $J$ vectors $\boldsymbol{S}_j^{(B)}$. We can interpret (19.22) as an interpolation between different target spectral vectors. Therefore, it is important to choose a representation $\boldsymbol{S}$ where linear interpolation makes sense and does not lead to artifacts.

An improved method of mapping features from a source corpus to those of a target corpus is to esti-

mate an appropriate transformation function. The most popular method for doing this is based on Gaussian mixture models (GMMs). One of the early adopters of GMM-based methods for voice conversion was Stylianou [19.39] (Chap. 24). Let $\omega_m$ denote the $m$-th acoustic class out of a total of $M$ classes. In a Gaussian mixture density model, the probability density functions of any of the $K$ observation vectors $\boldsymbol{x}_k (k = 1, \ldots, K)$ is given by:

$$p(\boldsymbol{x}_k) = \sum_{m=1}^{M} P(\omega_m) p(\boldsymbol{x}_k|\omega_m) = \sum_{m=1}^{M} c_m N(\boldsymbol{x}_k, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) , \quad (19.23)$$

where $\boldsymbol{x}_k$ is an $L$-dimensional random vector of speech features. The conditional probability density for each of the $M$ classes $\omega_m$ is assumed to be a Gaussian component density $N(\boldsymbol{x}_k, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, and the $c_m$ (the probabilities for classes $\omega_m$) are the related mixture weights normalized to have unity sum. Each $L$-variate Gaussian component density is of the form:

$$N(\boldsymbol{x}_k, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) = \frac{1}{(2\pi)^{L/2} |\boldsymbol{\Sigma}_m|^{1/2}} \times \exp\left[ -\frac{1}{2} (\boldsymbol{x}_k - \boldsymbol{\mu}_m)' \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{x}_k - \boldsymbol{\mu}_m) \right] . \quad (19.24)$$

The $L$-dimensional mean vector $\boldsymbol{\mu}_m$ and the covariance matrix $\boldsymbol{\Sigma}_m$ determine the center location and the spread, respectively, of the $m$-th mixture. Equation (19.24) expresses arbitrary distributions as a linear combination of Gaussians. If we associate the $m$-th component with a specific acoustic class (e.g., liquids, fricatives), it should be intuitively clear that mapping techniques based on (19.23) tend to be more robust than those based on (19.22), because the simpler approach does not distinguish between different acoustic classes. In addition, the simpler codebook mapping approach lacks a mechanism for generalizing to unseen data that the Gaussian representation provides.

How do we determine the mapping function $\boldsymbol{F}_x(\boldsymbol{x})$ that converts estimates of the source speech $\boldsymbol{x}_k$ into estimates of the target speech $\tilde{\boldsymbol{y}}_k$? We make the sensible assumption that this function should minimize the mean-squared error between a transformed set $\tilde{\boldsymbol{Y}} = \left\{ \tilde{\boldsymbol{y}}_k = \boldsymbol{F}_x(\boldsymbol{x}_k), k = 1, \ldots, K \right\}$ of time-aligned source data and the actual training set $\boldsymbol{Y} = \{\boldsymbol{y}_k, k = 1, \ldots, K\}$ of target data. To start, let $h_m(\boldsymbol{x}) = P(\omega_m|\boldsymbol{x})$ be the conditional probability of $\boldsymbol{x}$ belonging to class $\omega_m$. With (19.23) and Bayes' rule, we find

$$h_m(\boldsymbol{x}) = P(\omega_m|\boldsymbol{x}) = \frac{P(\omega_m) p(\boldsymbol{x}|\omega_m)}{p(\boldsymbol{x})} = \frac{c_m N(\boldsymbol{x}, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{i=1}^{M} c_i N(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} . \quad (19.25)$$

Then the desired mapping function $\boldsymbol{F}_x(\boldsymbol{x})$ is chosen to be of the form:

$$\boldsymbol{F}_x(\boldsymbol{x}) = \sum_{m=1}^{M} h_m(\boldsymbol{x}) \left[ \boldsymbol{v}_m + \boldsymbol{\Gamma}_m \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_m) \right] , \quad (19.26)$$

where the unknown $L$-dimensional vectors $\boldsymbol{v}_m$ and the unknown $L \times L$ matrices $\boldsymbol{\Gamma}_m$ are determined by solving normal equations for the least-squares problem between $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$. Note that this method may be improved further by basing the conversion function on joint source and target densities [19.40] instead of basing them solely on source densities as highlighted here. More on voice transformation can be found in Chap. 24 by Stylianou.

## 19.4 Speech Synthesis Evaluation

Evaluation of TTS systems is critical for many reasons. One reason is to gauge the progress in the field; another is to pick the best system for a specific application. Because so many specific interests might come to bear, it is impossible to list recipes for proper evaluation procedures that would fit any scenario. Short of that, we can only give broad guidelines here.

There are three quality criteria that one might be interested in: accuracy, intelligibility, and naturalness. They might overlap with each other with respect to which part of a TTS system impacts on which criteria. We define the accuracy of a TTS system as the ability to read a given input text the way a knowledgeable human reader would. For obvious reasons, accuracy is the one criterion solely homed in the TTS frontend. Contrary to this, unsatisfactory intelligibility or naturalness is much more difficult to tie to any single component of a TTS system. Between the latter two, designers of formant synthesizers primarily aim at maximizing intelligibility because higher naturalness can be achieved with unit-

selection methods at the cost of higher computational complexity. Vice versa, some of the newer concatenative synthesis systems may overemphasize naturalness with the result of lower intelligibility.

Accuracy may be evaluated by running a text corpus of task-relevant acronyms and abbreviations within appropriate carrier sentences through the TTS frontend and judging the generated output *text*. Moreover, evaluating intelligibility and naturalness requires conducting elaborate listening tests. In intelligibility tests, long known from testing speech coders, word or sentence lists are presented and subjects write down the words they hear [19.41]. However, evaluating the intelligibility of a TTS system clearly exposes more issues and aspects than are covered by standard intelligibility tests that rely on word lists. On a sentence level, for example, incorrect prosody can destroy intelligibility/comprehension. So far, a generally accepted standard intelligibility test for TTS systems is lacking, although some effort has begun to address this problem [19.42]. For overall quality evaluation, the International Telecommunication Union (ITU) recommends a specific method [19.43] that seems appropriate for testing naturalness. A five-point (or more) rating scale is employed for testing characteristics such as *overall impression*, *listening effort*, *comprehension*, etc. Another option is to ask listeners to state their preference of one option/system over another (A/B tests). Finally, selecting appropriate test material is also relevant. For example, unit-selection systems may sound very good for short sentences spoken in isolation, but show weaknesses when longer paragraphs of text are being rendered. When having a bake-off of different TTS systems, selecting test material appropriate for the intended application seems optimal. For more details on quality testing of TTS systems, see, for example [19.44].

## 19.5 Conclusions

This chapter summarized the basic principles used in speech synthesis as an introduction to more-detailed chapters later in this book. It is obvious that text-to-speech systems have come a long way towards delivering high-quality output to listeners.

We divided speech synthesis systems into two distinct parts, the text-processing frontend and the speech signal-processing backend. We identified document structure detection, text normalization, interpretation of text markup, and linguistic analysis as frontend tasks. Creating synthetic speech from a symbolic representation is the task of the backend. Here, we touched on articulatory synthesis, formant synthesis, and concatenative synthesis, the latter including unit-selection synthesis. In terms of speech representations and related speech signal-processing algorithms, we highlighted TD-PSOLA, LPC-based synthesis, and sinusoidal synthesis. We then explored transformation techniques for modifying the speech signal either in prosody, or in speaker identity or different speaking styles/emotions. Finally, we touched on evaluation issues for accuracy, intelligibility, and naturalness.

It should be noted that we are still far from delivering perfect synthesis for all possible applications. However, one can note that some of today's systems are now so good that listeners may be fooled into believing that they are listening to recordings of human speakers, in particular, when the system is fine-tuned for a specific application with a limited task domain. For best results, the frontend and the backend of a text-to-speech system can be optimized for a given application. For example, including and maintaining a pronunciation dictionary of names of all family names of telephone subscribers could be essential for using speech synthesis in an automated directory assistance application. Capturing desired voice characteristics (*persona*) from a voice talent that is being recorded for a unit-selection synthesis voice database could be essential for customers accepting an automated dialog system that speaks with a synthetic voice. Much current and future research will focus on improving speech synthesis further with a special focus on automating and streamlining most of the tedious manual processes involved in creating high-quality systems.

### References

19.1 J.L. Flanagan: *Speech Analysis, Synthesis and Perception* (Springer, Berlin, Heidelberg 1972) pp. 204–210, http://www.haskins.yale.edu/featured/heads/SIMULACRA/kempelen.html

19.2 H. Dudley, R.R. Riesz, S.A. Watkins: A synthetic speaker, J. Franklin Inst. **227**, 739–764 (1939), http://www.bell-labs.com/org/1133/Heritage/Vocoder/

19.3 W3C Standard Generalized Markup Language: http://www.w3.org/MarkUp/SGML/

19.4 W3C Speech Synthesis Markup Language Version 1.0: http://www.w3.org/TR/2003/CR-speech-synthesis-20031218/ (http://www.xml.com/pub/a/2004/10/20/ssml.html)

19.5 J.P. van Santen: Timing. In: *Multilingual Text-to-Speech Synthesis – The Bell Labs Approach*, ed. by R. Sproat (Springer, New York 1998) pp. 115–139

19.6 D.H. Klatt: Linguistic use of segmental duration in English: Acoustic and perceptual evidence, J. Acoust. Soc. Am. **59**, 1208–1221 (1976)

19.7 A. Schweitzer, B. Moebius: Exemplar-based production of prosody: Evidence from segment and syllable durations, Proc. Speech Prosody 2004 (Nara), ed. by B. Bel, I. Marlien (ISCA, Grenoble 2004)

19.8 K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, J. Hirschberg: TOBI: A standard for labeling English prosody, Proc. ICSLP'92 Banff (1992) pp. 867–870

19.9 H. Fujisaki: The role of quantitative modeling in the study of intonation, Proc. Int. Symp. Japanese Prosody (1992) pp. 163–174

19.10 G. Richard, M. Liu, D. Sinder, H. Duncan, Q. Lin, J. Flanagan, S. Levinson, D. Davis, S. Slimon: Numerical simulations of fluid flow in the vocal tract, Proc. of Eurospeech Madrid (1995) pp. 18–21

19.11 M. Stone: A three-dimensional model of tongue movement based on ultrasound and x-ray microbeam data, J. Acoust. Soc. Am. **87**, 2207–2217 (1990)

19.12 T. Baer, J.C. Gore, L.C. Gracco, P.W. Nye: Analysis of vocal tract shape and dimensions using magnetic resonance imaging: Vowels, J. Acoust. Soc. Am. **90**, 799–828 (1991)

19.13 C.H. Coker: A model of articulatory dynamics and control, Proc. IEEE **64**, 452–459 (1976)

19.14 E.L. Saltzman, K.G. Munhall: A dynamical approach to gestural patterning in speech production, Ecol. Psychol. **1**(4), 333–382 (1989)

19.15 J. Schroeter, M.M. Sondhi: Speech coding based on physiological models of speech production. In: *Advances in Speech Signal Processing*, ed. by S. Furui, M.M. Sondhi (Marcel Dekker, New York 1991) pp. 231–268

19.16 J.D. Markel, A.H. Gray: *Linear Prediction of Speech* (Springer, New York 1976)

19.17 D.H. Klatt: Software for a cascade/parallel formant synthesizer, J. Acoust. Soc. Am. **67**, 971–995 (1980)

19.18 H.M. Hanson, K.N. Stevens: A quasiarticulatory approach to controlling acoustic source parameters in a Klatt-type formant synthesizer using HLsyn, J. Acoust. Soc. Am. **112**, 1158–1182 (2002)

19.19 J.P.H. van Santen: Combinatorial issues in text-to-speech synthesis, EuroSpeech '97 5th European Conference on Speech Communication and Technology **5**, 2511–2514 (1997)

19.20 J.P. Olive: Rule synthesis of speech from diadic units, Proc. ICASSP **77**, 568–570 (1977)

19.21 O. Fujimura, J. Lovins: Syllables as concatenative phonetic elements. In: *Syllables and Segments*, ed. by A. Bell, J.B. Hooper (North-Holland, New York 1978) pp. 107–120

19.22 J. Olive, J. van Santen, B. Möbius, C. Shih: Synthesis. In: *Multilingual Text-to-Speech Synthesis – The Bell Labs Approach*, ed. by R. Sproat (Kluwer Academic, Dordrecht 1998), Chap. 7

19.23 Y. Sagisaka: Speech synthesis by rule using an optimal selection of non-uniform synthesis units, Proc. ICASSP **88**, 679–682 (1988)

19.24 A. Hunt, A.W. Black: Unit selection in a concatenative speech synthesis system using a large speech database, Proc. ICASSP **96**, 373–376 (1996)

19.25 A.W. Black, N. Campbell: Optimising selection of units from speech databases for concatenative synthesis, ESCA Eurospeech **95**, 581–584 (1995)

19.26 L. Rabiner, B.H. Juang: *Fundamentals of Speech Recognition* (Prentice-Hall, Englewood Cliffs 1993) pp. 339–341

19.27 J. Vepa, S. King: Join cost for unit selection speech synthesis. In: *Text-to-Speech Synthesis – New Paradigms and Advances*, Professional Technical Reference, ed. by S. Narayanan, A. Alwan (Prentice-Hall, Upper Saddle River 2004) pp. 35–62, Chap. 3

19.28 E. Eide, R. Bakis, W. Hamza, J.F. Petrelli: Toward expressive synthetic speech. In: *Text-to-Speech Synthesis – New Paradigms and Advances*, Professional Technical Reference, ed. by S. Narayanan, A. Alwan (Prentice-Hall, Upper Saddle River 2004) pp. 219–248, Chap. 11

19.29 A.W. Black, P. Taylor: Automatically clustering similar units for unit selection in speech synthesis, Proc. Eurospeech **97**, 601–604 (1997)

19.30 K. Tokuda, H. Zen, A.W. Black: An HMM-based approach to multilingual speech synthesis. In: *Text-to-Speech Synthesis – New Paradigms and Advances*, Professional Technical Reference, ed. by S. Narayanan, A. Alwan (Prentice-Hall, Upper Saddle River 2004) pp. 135–153, Chap. 7

19.31 T. Dutoit: *An Introduction to Text-to-Speech Synthesis* (Kluwer Academic, Dordrecht 1997)

19.32 E. Moulines, F. Charpentier: Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones, Speech Commun. **9**(5–6), 453–467 (1990)

19.33 M. Macchi, M.J. Altom, D. Kahn, S. Singhal, M. Spiegel: Intelligibility as a function of speech coding method for template-based speech synthesis, Proc. Eurospeech **93**, 893–896 (1993)


Amazon EX1010 Page 452

19.34 W. Kleijn, K. Paliwal (Eds.): *Speech Coding and Synthesis* (Elsevier, Amsterdam 1995)

19.35 T.F. Quartieri, R.J. McAulay: Shape invariant time-scale and pitch modification of speech, IEEE Trans. Signal Process. **40**(3), 497–510 (1992)

19.36 Y. Stylianou: Applying the harmonic plus noise model in concatenative speech synthesis, IEEE Trans. Speech Audio Process. **9**(1), 21–29 (2001)

19.37 H. Kawahara, I. Masuda-Katsuse, A. de Cheveigne: Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds, Speech Commun. **27**(3-4), 187–207 (1999)

19.38 M. Abe, S. Nakamura, K. Shikano, H. Kuwahara: Voice conversion through vector quantization, Proc. IEEE ICASSP **88**, 655–658 (1990), S14.1

19.39 I. Stylianou: *Modèles Harmoniques plus Bruit combines avec des Méthodes Statistiques, pour la Modication de la Parole et du Locuteur*, Doctoral Thesis (Ecole Nationole Supérieure des Télécommunications, Paris 1996), in French

19.40 A. Kain, M. Macon: Spectral voice conversion for text-to-speech synthesis, Proc. IEEE ICASPP **98**, 285–288 (1998)

19.41 M.F. Spiegel, M.J. Altom, M.J. Macchi: Comprehensive assessment of the telephone intelligibility of synthesized and natural speech, Speech Commun. **9**, 279–291 (1990)

19.42 A. Syrdal: Development of a standard for the evaluation of intelligibility of text-to-speech synthesis systems by ANSI Accredited Standards Committee S3, Bioacoustics, working group S3/WG 91, Text-to-Speech Synthesis Systems, Personal communication (2007)

19.43 ITU-T: *A Method for Subjective Performance Assessment of the Quality of Speech Output Devices* (International Telecommunications Union, Geneva 1994), Recommendation P.85

19.44 Y.V. Alvarez, M. Huckvale: The reliability of the ITU-T P.85 standard for the evaluation of text-to-speech systems, Proc. ICSLP **2002**, 329–332 (2002)


Amazon EX1010 Page 453

# 20. Rule-Based Speech Synthesis

R. Carlson, B. Granström

In this chapter, we review some of the issues in rule-based synthesis and specifically discuss formant synthesis. Formant synthesis and the theory behind have played an important role in both the scientific progress in understanding how humans talk and also the development of the first speech technology applications. Its flexibility and small *footprint* makes the approach still of interest and a valuable complement to the current dominant methods based on concatenative data-driven synthesis. As already mentioned in the overview by *Schroeter* (Chap. 19) we also see a new trend to combine the rule-based and data-driven approaches. Formant features from a database that can be used both to optimize a rule-based formant synthesis system and to optimize the search for good units in a concatenative system.

20.1 **Background** ..... 429
20.2 **Terminal Analog** ..... 429
20.2.1 Formant Synthesizers ..... 429
20.2.2 Higher-Level Parameters ..... 430
20.2.3 Voice Source Models ..... 431
20.3 **Controlling the Synthesizer** ..... 432
20.3.1 Rule Compilers for Speech Synthesis 432
20.3.2 Data-Driven Parametric Synthesis .. 433
20.4 **Special Applications of Rule-Based Parametric Synthesis** ..... 434
20.5 **Concluding Remarks** ..... 434
**References** ..... 434



## 20.1 Background

Rule synthesis and formant synthesis are historically highly related. The theory of formant synthesis goes back to the first attempts to model the acoustics of the speech waveform. In the publications by *Fant* [20.1, 2] and *Stevens* [20.3] the foundations of the acoustics of speech and the relation between articulatory models, acoustic manifestations, and spoken communication can be found. The synthesis models were initially evaluated by carefully copying recorded speech. After these initial experiments new methods were developed to control the synthesizer. The classic publications by *Holmes*, *Mattingly*, and *Shearme* [20.4], *Mattingly* [20.5], *Flanagan* [20.6], *Klatt* [20.7], and *Allen*, *Hunnicutt*, and *Klatt* [20.8] present the basic approaches for speech synthesis. An early effort to synthesize speech based on explicit rules can be found in *Carlson* and *Granström* [20.9]. Several review publications on speech synthesis have been published since then: *Sagisaka* [20.10], *Dutoit* [20.11], *Carlson* and *Granström* [20.12]. The paper by *Klatt* [20.13] gives an extensive review of the developments of the speech synthesis technique before the general introduction of concatenative synthesis.

## 20.2 Terminal Analog

### 20.2.1 Formant Synthesizers

The traditional text-to-speech systems use a terminal analog as a sound-generating device, i. e., the aim with this kind of synthesizer is only that it should be able to produce the sounds (speech spectra) that are found in natural speech. The internal structure is not a model of acoustic speech production in the vocal tract. The basic concept is the combination of sound sources and filters, describing the transfer function, see *Lawrence* [20.14] and *Fant* [20.15]. The source–filter model exists in several versions; the classical configuration by *Klatt* [20.16], is exemplified in Fig. 20.1. The vocal-tract transfer function is simulated by a sequence of second-order filters in a cascaded arrangement, while a parallel structure is used mostly

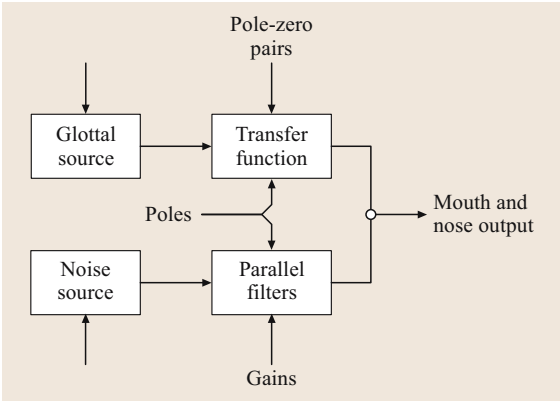



**Fig. 20.1** Block diagram of the main components of the Klatt terminal-analog speech synthesizer (after [20.18])

for the synthesis of consonants. One important advantage of a cascade synthesizer is the automatic setting of formant amplitudes. The disadvantage is that it sometimes can be difficult to do detailed spectral matching between natural and synthesized spectra because of the simplified model. Parallel synthesizers such as the one by *Holmes* [20.17] do not have this limitation.

The Klatt model is widely used in research both for general synthesis purposes and for perceptual experiments. A simplified version of this system is used in commercial products that builds on the research at Massachusetts Institute of Technology (MIT): MITalk [20.8], Klattalk [20.20], and DECtalk [20.21].

The formant terminal analog, GLOVE [20.19], based on the OVE synthesizer [20.22], was developed at KTH (Royal Institute of Technology) and has been explored in speech research and several practical applications [20.23, 24]. The structure of the GLOVE synthesizer is shown in Fig. 20.2. The controllable parameters are indicated by two-letter symbols. To the left, the two sound sources can be seen. For mixed excitation the sources are connected in two ways. The parameter $N_M$ flow-modulates the noise source, typical for voiced fricatives. The parameter $N_A$ adds noise to the glottal source, as in breathy or whispered voices. The five parameters above the voice source are the glottal parameters referred to in the voice source section below. The sound source signals are fed into the three parallel branches with poles and zeroes. All are controlled by amplitude parameters ($A_N$, $A_0$, $A_H$ and $A_C$). The upper branch is primarily used for introducing an extra pole and zero in nasals and nasalized sounds. The middle branch is the main branch for sounds produced with glottal excitation and the lowest branch models sounds with supraglottal excitation, such as stops and fricatives. This basic configuration can be augmented in several ways. The interaction between the source and the vocal tract, which can be substantial, is in this case only modeled by the $B_M$ parameter that modulates the bandwidth of the first formant, dependent on the glottal opening, or more precisely the glottal flow. The main difference between the MIT and KTH traditions can be found in how the consonants are modeled. In the OVE case, a fricative is filtered by a zero–pole–pole configuration rather than the parallel branch in the Klatt synthesizer.

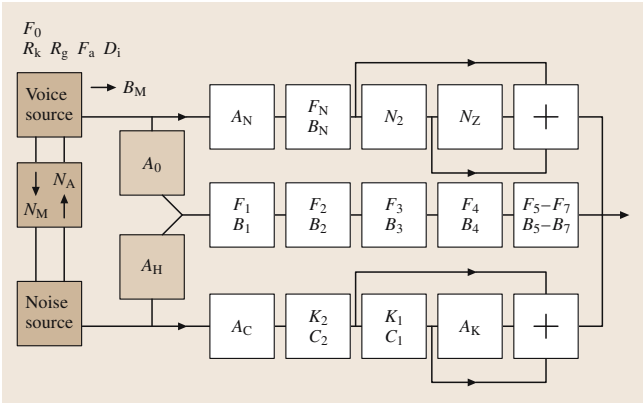



**Fig. 20.2** Block diagram of the main components of the terminal-analog speech synthesizer GLOVE (after [20.19])

With terminal analog synthesizers, it is possible to simulate most human voices, and to replicate an utterance without noticeable quality reduction. However, it is interesting to note that some voices are easier to model than others. Despite the progress, speech quality is not good enough in all applications of text to speech. The main reasons for the limited success in formant-based synthesis can be explained by the incomplete knowledge needed for automatic control of the parameters. It should be noted that the transfer of knowledge from phonetics to speech technology and explicit rule-based description has not been an easy process.

### 20.2.2 Higher-Level Parameters

Since the control of a formant synthesizer can be a very complex task, some efforts have been made to help the developer. The *higher-level parameters* described by *Stevens* and *Bickley* [20.18], *Hanson* and *Stevens* [20.25] and *Stevens* [20.26], for example, explore an intermediate level that is more understand-

able from the developer's point of view compared to the detailed synthesizer specifications. The goal with this approach is to find a synthesis framework to simplify the process and to incorporate the constraints that are known to exist within the process. A formant frequency should not have to be adjusted specifically by the rule developer depending on nasality or glottal opening. This type of adjustment might be better handled automatically according to a well-specified model. The same process should occur with other parameters such as bandwidths and glottal settings. This approach requires detailed understanding of the relation between acoustic and articulatory phonetics. The Prosynth project *Ogden* et al. [20.27] has made use of the multilevel view and included phonetic and prosodic aspects in the approach; see *Heid* and *Hawkins* [20.28].

### 20.2.3 Voice Source Models

The traditional voice source model has been a simple or double impulse. This is one reason why the voices produced by early text-to-speech systems lack naturalness to a great extent. While the male voice has sometimes been regarded as generally acceptable, an improved glottal source will open the way to more-realistic synthesis of child and female voices and also to greater naturalness and variation in male voices.

Most source models work in the time domain with various controls to manipulate the pulse shape [20.30–40]. One influential voice source model is the LF model [20.41]. It has a truncated exponential sinusoid followed by a variable-cut-off −6 dB/octave low-pass filter modeling the effect of the return phase, i. e., the time from maximum excitation of the vocal tract to complete closure of the vocal folds. Figure 20.3 shows the function of the control parameters. In addition to the amplitude and fundamental frequency control, two parameters largely influence the amplitudes of the two to three lowest harmonics, and one parameter, the high-frequency content of the spectrum. Another vocal source parameter is the diplophonia parameter (inspired by *Klatt* and *Klatt* [20.33]) with which creak, laryngalization or diplophonia can be modelled. This parameter influences the function of the voiced source in such a way that every second pulse is lowered in amplitude and shifted in time.

The acoustic interactions between the glottal source and the vocal tract also have to be considered [20.42]. One of the major factors in this respect is the varying bandwidth of the formants. This is especially true for

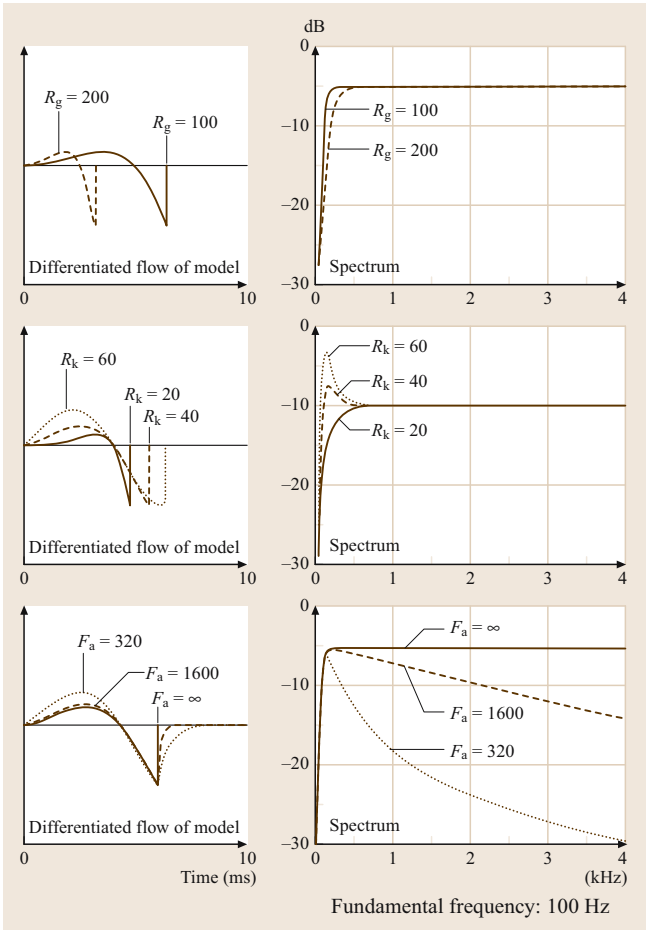



**Fig. 20.3** The influence of the parameters $R_G$, $R_K$ and $F_A$ on the differentiated glottal flow pulse shape and spectrum. The spectra are pre-emphasized by 6 dB/oct (after [20.29])

the first formant, which can be heavily damped during the open phase of the glottal source. However it is not clear that such a variation can be perceived by a listener. Listeners tend to be rather insensitive to bandwidth variation [20.6]. When more-complex models are included, the output from the model has to change from a glottal flow model to a model of the glottal opening. The subglottal cavities can then be included in an articulatory model.

Noise sources have attracted much less research effort than the voiced source. However, some aspects are discussed by *Stevens* [20.3, 43], *Shadle* [20.44], and *Badin* and *Fant* [20.45]. Typically, simple white noise is filtered by resonances that are stationary between each parameter frame. Some formant synthesizers do include

some interaction between the voice source and the noise source, but the interaction is rather primitive. The modeling of transient sounds and aspiration dependent on vocal-cord opening is still a challenge.

## 20.3 Controlling the Synthesizer



As already mentioned initial work on formant synthesis was related to the development of theories and models of the acoustics of speech. Synthesis experiments were mostly devoted to testing the models and initial speech perception experiments, for example the classic experiments by *Delattre*, *Liberman*, and *Cooper* [20.47]. The work by *Holmes*, *Mattingly* and *Shearme* [20.4] was a major step towards controlling the synthesizer based on symbolic input such as phones or phonemes. Each phoneme was described by a simplified parametric pattern. It was clear that quality was limited if the settings were not adjusted, and rule systems to modify these settings were developed. A number of models were created to take coarticulation into account such as smoothed step functions *Liljencrants* [20.48] or explicit rule descriptions *Klatt* [20.9, 49–51].

Unfortunately rule development for formant synthesis has some problems due to the parametric domain itself. The context-dependent acoustic realizations in terms of formant settings reflect the underlying articulatory gestures rather than the formant pattern in the context (e.g., [20.52]). In a similar way parameters such as variation along the *hyper/hypo-articulation* dimension have a strong influence on the acoustic realization and talking speed [20.53, 54].

A challenging problem in rule development is the locus equations. For example velar stop coarticulation is dependent on many factors and cannot easily be summarized in a single formula [20.55].

These are some of the reasons why formant synthesis has not been competitive in relation to concatenative synthesis. However, the need to synthesize different voices and voice characteristics and to model emotive speech is a strong motivation to keep research on parametric synthesis active. The driving force is that rule-based formant synthesis has the necessary flexibility to model both linguistic and extralinguistic processes. It is, for example, evident that large databases for corpus-based approaches cannot be recorded consistently in a range of desired speaking styles. We will in the following describe some tools that have been used to develop formant synthesis rules and alternative data-driven methods. Rules for prosodic realization will be discussed further in the contribution by Van Santen et al. (Chap. 20).

### 20.3.1 Rule Compilers for Speech Synthesis

Development tools for text-to-speech systems have received considerable attention. Often the basis of such tools has followed the development of phonological theory. The work on generative phonology and especially the publication of *The Sound Pattern of English* by *Chomsky* and *Halle* [20.56] led to a special kind of synthesis system based on context-sensitive rewrite rules. The linguistically oriented notation inspired speech researchers to create special rule compilers for text-to-speech developments in the 1970s (e.g. [20.36]). The implementations vary depending on the developer's inclination. It is important to note that crucial decisions were often hidden in the systems. The rules may operate rule-by-rule or segment-by-segment. With greater emphasis on prosodic modeling and the related development of nonlinear phonology [20.57], synthesis procedures inspired by such theories were created, as in the systems described by *Hertz* [20.58], *Hertz*, *Kadin*, and *Karplus* [20.59], *Lazzaretto* and *Nebbia* [20.60], *Ceder* and *Lyberg* [20.61], and *Leeuwen* and *Lindert* [20.46, 62]. The common feature of these notations was that they keep information on different linguis-

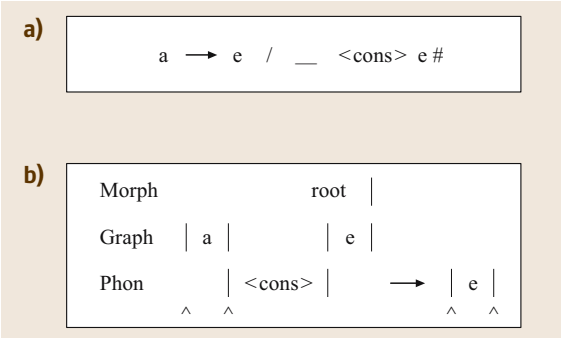



**Fig. 20.4** **(a)** Notation according to the KTH *RULSYS* system. (after [20.9]) **(b)** Notation according to the *Speech Maker* system developed at IPO (after [20.46])

tic levels (tiers) separate in a more-explicit way than the essentially linear representation based on generative phonology. This gave potentially higher flexibility, but also more-complex notations. The rules now looked more like two-dimensional schemes rather than one-line representations. This posed greater demands on the developer and his/her tools. Rather than conventional text editors, editors should preferably be graphic editors, where relations between tiers can easily be specified, as in the Speech Maker system developed at IPO (Institute for Preception Research) [20.46]. In Fig. 20.4, this representation is compared to the one used in the KTH system. The rules describe how a letter "a" is pronounced as [e] if it occurs before a single consonant sound and a root final "e". In the KTH notation, the root boundary is introduced as the symbol "#" by earlier rules. After the rule application, the letter origin of the [e] is lost, which places greater demands on the rule order.

## 20.3.2 Data-Driven Parametric Synthesis

The flexibility in formant synthesis can also be a problem, since articulatory constraints, for example, are not directly included in the formant-based model. The underlying articulatory gestures are not easily transformed to the acoustic domain described by the formant model. When increasing our ambitions to multilingual, multispeaker and multistyle synthesis it is obvious that we want to find at least semiautomatic methods to collect the necessary information, using speech and language databases. Traditionally, speech synthesis has been based on very labor-intensive optimization work. The notion *analysis by synthesis* has not been explored except by manual comparisons between hand-tuned spectral slices and a reference spectrum. The work by *Holmes* and *Pearce* [20.64] is a good example of how to speed up this process. With the help of a synthesis model, the spectra are automatically matched against analyzed speech. Automatic techniques such as this will also probably play an important role in making speaker-dependent adjustments. One advantage of these methods is that the optimization is done in the same framework as that to be used in the production. The synthesizer constraints are thus already imposed in the initial state.

As early as in the 1950s *Peterson* et al. [20.65] suggested that unit concatenation might be a possible solution for speech synthesis. *Dixon* and *Maxey* [20.66] made a special effort to create a unit library for diphone synthesis. Early synthesis research at AT&T based on *diadic units* [20.67] demonstrated an alternative to rule-based formant synthesis.

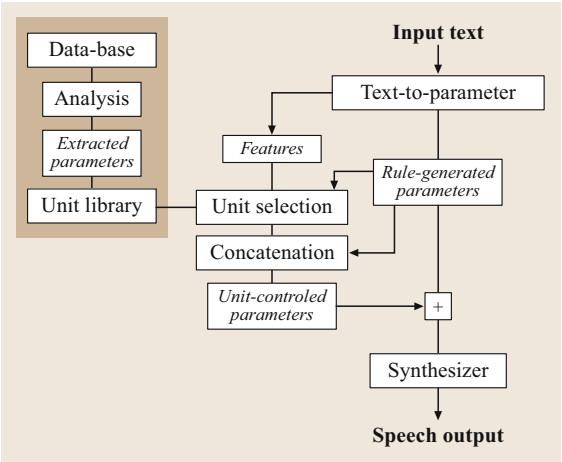



**Fig. 20.5** Rule-based synthesis system using a data-driven unit library (after [20.63])

These alternative approaches, which reduce the need for detailed formant synthesis rules but still keep the flexibility of the formant model, extract formant synthesis parameters directly from a labeled corpus. *Mannell* [20.68] reported a promising effort to create a diphone library for formant synthesis. The procedure included a speaker-specific extraction of formant frequencies from a labeled database. In a sequence of papers from Utsunomiya University, Japan automatic formant tracking was used to generate speech synthesis of high quality using formant synthesis and an elaborate voice source (e.g., *Mori* et al. [20.69]). Research efforts to combine data-driven and rule-based methods has recently been reported by *Hertz* [20.70], and *Carlson* and *Granström* [20.63]. The approaches take advantage of the fact that a unit library can better model detailed gestures than general rules. Figure 20.5 illustrates the approach in the KTH text-to-speech system. A database is used to create a unit library and the library information is mixed with the rule-driven parameters. Automatic methods of formant extraction are of course preferred when creating a unit library, due to the amount of data that has to be processed [20.71–73]. However, available methods do not always perform adequately and manual editing is often necessary.



Amazon EX1010 Page 458

## 20.4 Special Applications of Rule-Based Parametric Synthesis

In some cases where a speaker model is not readily available database-driven concatenative synthesis is not so well suited. One such example that we have been involved in is the experimental synthesis of varieties of deaf speech in the context of speech training [20.74, 75]. The aim in this project was to synthesize some prototypical versions of the deviations found in the speech of several deaf children, like consonantal realizations, reduced and distorted vowel inventory, and deviant prosody. The rule-based method made it easy to synthesize and evaluate different versions with fewer deviations, thus simulating the effect of (successful) speech training. The often-systematic differences that make intelligibility difficult for occasional listeners, tend to affect the understanding of persons like family less. This points to another use of speech synthesis, where the deviant speech could be *corrected*, but still retain the voice of the deaf or speech-impaired person. One recent example of work aimed at dysarthric speakers is described in *Kain* et al. [20.76], where some improvement was demonstrated both in intelligibility and quality by formant analysis, modification, and resynthesis. The experiment concerned only vowels in nonsense CVC (consonant-vowel-consonant) words uttered by one dysarthric speaker, but points to an interesting possibility.

One early application of speech synthesis was as speech prosthesis for nonvocal individuals. Ideally such a device should be able to use a variety of speaking styles, displaying different attitudes, and emotions in a voice selected by the user. This makes conventional unit-selection methods less suitable, due to the difficulty of recording an appropriate database. Several attempts have made to construct such a prosthesis, e.g., *Murray* et al. [20.77], and *Cudd* et al. [20.78], or the components necessary for, e.g., emotion control [20.79]. However, presently commercially available speech prostheses still lack most of the desired functions.

## 20.5 Concluding Remarks

Recently we have also seen renewed commercial interest in speech synthesis using the formant model [e.g., the Aurix text-to-speech (TTS) system from 20/20 Speech]. One motivation is the need to generate speech using a very small *footprint*. Thus, one can predict that formant synthesis will again be an important research subject because of its flexibility and also because of how the formant synthesis approach can be compressed into a limited application environment.

### References


20.1 G. Fant: *Acoustic Theory of Speech Production* (Mouton, The Hague 1960)

20.2 G. Fant: *Speech Acoustics and Phonetics, Selected Writings Series: Text, Speech and Language Technology*, Vol. 24 (Springer, Berlin, Heidelberg 2006)

20.3 K. Stevens: *Acoustic Phonetics* (MIT Press, Cambridge 1999)

20.4 J. Holmes, I.G. Mattingly, J.N. Shearme: Speech synthesis by rule, Lang. Speech **7**, 127–143 (1964)

20.5 I.G. Mattingly: Synthesis by rule as a tool for phonological research, Lang. Speech **14**(1), 47–56 (1971)

20.6 J.L. Flanagan: *Speech Analysis, Synthesis and Perception* (Springer, Berlin, Heidelberg 1972)

20.7 D. K. Klatt: Structure of a phonological rule component for a synthesis-by-rule program, IEEE Trans. ASSP-**24** (1976)

20.8 J. Allen, M.S. Hunnicutt, D. Klatt: *From Text to Speech. The MITalk System* (Cambridge University Press, Cambridge 1987)

20.9 R. Carlson, B. Granström: A text-to-speech system based entirely on rules, Proc. ICASSP 76, Philadelphia (1976)

20.10 Y. Sagisaka: Speech synthesis from text, IEEE Commun. Mag. **28**(1), 35–41 (1990)

20.11 T. Dutoit: *An Introduction to Text-to-Speech Synthesis* (Kluwer Academic, Dordrecht 1997)

20.12 R. Carlson, B. Granström: *Speech synthesis. In: Hardcastle WJ and Laver J. The Handbook of Phonetic Science* (Blackwell, Oxford 1997) pp. 768–788

20.13 D.K. Klatt: Review of text-to-speech conversion for English, J. Acoust. Soc. Am. **82**(3), 737–793 (1987)

20.14 W. Lawrence: The synthesis of speech from signals which have a low information rate. In: *Commu-*


Amazon EX1010 Page 459

*nication Theory*, ed. by W. Jackson (Butterworths, London 1953) pp. 460–469
20.15 G. Fant: Speech Communication Research, Ing. Vetenskaps Akad. Stockholm **24**, 331–337 (1953)
20.16 D.K. Klatt: Software for a cascade/parallel formant synthesizer, J. Acoust. Soc. Am. **67**, 971 (1980)
20.17 J. Holmes: Formant synthesizers, cascade or parallel, Speech Commun. **2**, 251–273 (1983)
20.18 K. Stevens, C. Bickley: Constraints among parameters simplify control of Klatt formant synthesizer, J. Phonetics **19**(1) (1991)
20.19 R. Carlson, B. Granström, I. Karlsson: Experiments with voice modelling in speech synthesis, Speech Commun. **10**, 481–489 (1991)
20.20 D. Klatt: The Klattalk text-to-speech conversion system, Proc. ICASSP **82**, 1589–1592 (1982)
20.21 D. Klatt: DecTalk user's manual, Digital Equipment Corporation (1990)
20.22 J. Liljencrants: The OVE III speech synthesizer, IEEE Trans.Audio Electroac. **16**(1), 137–140 (1968)
20.23 R. Carlson, B. Granström, S. Hunnicutt: A multilanguage text-to-speech module, Proc. ICASSP 82 **82**(3), 1604–1607 (1982)
20.24 R. Carlson, B. Granström, S. Hunnicutt: Multilingual text-to-speech development and applications. In: *Advances in speech, hearing and language processing*, ed. by A.W. Ainsworth (JAI, London 1991)
20.25 H.M. Hanson, K.N. Stevens: A quasiarticulatory approach to controlling acoustic source parameters in a Klatt-type formant synthesizer using HLsyn, J. Acoust. Soc. Am. **112**, 1158–1182 (2002)
20.26 K. Stevens: Toward Formant Synthesis with Articulatory Controls, Proceedings of IEEE Workshop on Speech Synthesis (2002)
20.27 R. Ogden, S. Hawkins, J. House, M. Huckvale, J. Local, P. Carter, J. Dankovicová, S. Heid: ProSynth: An integrated prosodic approach to device-independent natural-sounding speech synthesis, Comput. Speech Lang. **14**, 177–210 (2000)
20.28 S. Heid, S. Hawkins: Synthesizing systematic variation at boundaries between vowels and obstruents. In: *Proceedings of the XIVth International Congress of Phonetic Sciences*, Vol.1, ed. by J.J. Ohala, Y. Hasegawa, M. Ohala, D. Granville, A.C. Bailey (University of California, Berkeley 1999) pp. 511–514
20.29 C. Gobl, J. Karlsson: Male and female voice source dynamics. In: *Vocal Fold Physiology: Acoustic, Perceptual, and Physiological Aspects of Voice Mechanisms*, ed. by J. Gauffin, B. Hammarberg (Singular, San Diego 1991)
20.30 T. V. Ananthapadmanabha: Acoustic analysis of voice source dynamics, STL-QPSR **2**(3) 1–24 (1984)
20.31 P. Hedelin: A glottal LPC-vocoder, Proc. IEEE, San Diego, 1.6.1–1.6.4 (1984)
20.32 J. Holmes: Influence of the glottal waveform on the naturalness of speech from a parallel formant synthesizer, IEEE Trans. Audio Electroac. **AU-21**, 298–305 (1973)
20.33 D.K. Klatt, L. Klatt: Analysis, synthesis, and perception of voice quality variations among female and male talkers, J. Acoust. Soc. Am. **87**, 820–857 (1990)
20.34 A.E. Rosenberg: Effect of glottal pulse shape on the quality of natural vowels, J. Acoust. Soc. Am. **53**, 1632–1645 (1971)
20.35 M. Rothenberg, R. Carlson, B. Granström, J. Lindqvist-Gauffin: *A three-parameter voice source for speech synthesis", Proc. of Speech Communication Seminar, Stockholm 1974; in Speech Communication*, Vol. 2 (Almqvist and Wiksell, Stockholm 1975) pp. 235–243
20.36 R. Carlson, B. Granström: A Phonetically Oriented Programming Language for Rule Description of Speech. In: *Speech Communication*, Vol. 2, ed. by G. Fant (Almqvist Wiksell, Uppsala 1975) pp. 245–253
20.37 P. Alku: *An automatic inverse filtering method for the analysis of glottal waveforms, Dissertation* (Helsinki University of Technology, Helsinki 1992)
20.38 C. Gobl, A. Ní Chasaide: Acoustic characteristics of voice quality, Speech Commun. **11**, 481–490 (1992)
20.39 C. Gobl, A. Ní Chasaide: The role of voice quality in communicating emotion, mood and attitude, Speech Commun. **40**, 189–212 (2003)
20.40 I. Karlsson: Modelling speaking styles in female speech synthesis, Speech Commun. **11**, 491–497 (1992)
20.41 G. Fant, J. Liljencrants, Q. Lin: A four parameter model of glottal flow, Speech Transmission Laboratory Quarterly and Status Report STL-QPSR No 4 (1985)
20.42 C. Bickley, K. Stevens: Effects of the vocal tract constriction on the glottal source: Experimental and modelling studies, J. Phon. **14**, 373–382 (1986)
20.43 K.N. Stevens: Airflow and turbulence noise for fricative and stop consonants: Static considerations, J. Acoust. Soc. Am. **50**(4), 1180–1192 (1971)
20.44 C.H. Shadle: The Aerodynamics of Speech. Handbook of Phonetics, ed. by W.J. Hardcastle, J. Laver, (Blackwell, Oxford 1997) pp. 33–64
20.45 P. Badin, G. Fant: Fricative modeling: some essentials, Proc. Europ. Conf. Speech Technol. Paris (1989)
20.46 H.C. van Leeuwen, E. te Lindert: Speech Maker: A flexible and general framework for text-to-speech synthesis, and its application to Dutch, Comput. Speech Lang. **7**(2), 149–168 (1993)
20.47 P.C. Delattre, A.M. Liberman, F.S. Cooper: Acoustic loci and transitional cues for consonants, J. Acoust. Soc. Am. **27**, 769–773 (1955)
20.48 J. Liljencrants: Speech synthesizer control by smoothed step functions, STL-QPSR **1969**(4), 43–50 (1969)



Amazon EX1010 Page 460

20.49 D.H. Klatt: Synthesis of stop consonants in initial position, J. Acoust. Soc. Am. Suppl. **147**, S93 (1970)

20.50 N. Umeda: Linguistic rules for text-to-speech synthesis, Proc. IEEE **64**(4), 443–451 (1976)

20.51 D.K. Klatt: Synthesis by rule of segmental durations in English sentences. In: *Frontiers in Speech Communication Research*, ed. by B. Lindblom, S. öhman (Academic, New York 1979)

20.52 G. Bailly, R. Laboissière, J. L. Schwartz: Formant trajectories as audible gestures: an alternative for speech synthesis, J. Phon. **19**(1), 9–23 (1991)

20.53 B. Lindblom: Explaining phonetic variation: A sketch of the H and H theory. In: *Speech Production Modeling*, ed. by Hardcastle, Marchal (Kluwer Academic, Dordrecht 1990)

20.54 R. J. J. H. van Son, L. Pols: Comparing formant movements in fast and normal rate speech, Proc. Europ. Conf. on Speech Commun. Technol. **89** (1989)

20.55 A. Slater, S. Hawkins: Effects of stress and vowel context on velar stops in British English, ICSLP 92 (Proc. 1992 Int. Conf. Spoken Language Processing) **1**, 57–60 (1992)

20.56 N. Chomsky, M. Halle: *Sound pattern of English* (Harper and Row, New York 1968)

20.57 J.B. Pierrehumbert: *The Phonetics of English Intonation* (IULC, Bloomington 1987)

20.58 S. R. Hertz: Streams, phones, and transitions: toward a new phonological and phonetic model of formant timing, J. Phon. **19**(1) (1991)

20.59 S.R. Hertz, J. Kadin, K.J. Karplus: The Delta rule development system for speech synthesis from text, Proc. IEEE **73**(11), 1589–1601 (1985)

20.60 S. Lazzaretto, L. Nebbia: SCYLA: Speech compiler for your language, Proc. European Conf on Speech Comm and Technology, Edinburgh **1**, 381–384 (1987)

20.61 K. Ceder, B. Lyberg: Yet another rule compiler for text-to-speech conversion? Proc. ICSLP92, Banff, Canada, pp. 1151–1154 (1992)

20.62 H. C. van Leeuwen, E. te Lindert: Speechmaker, text-to-speech synthesis based on a multilevel, synchronized data structure, Proc. ICASSP–**91** (1991)

20.63 R. Carlson, B. Granström: Data-driven multimodal synthesis, Issues Speech Commun. **47**(1–2), 182–193 (2005)

20.64 W. J. Holmes, D. J. B. Pearce: Automatic derivation of segment models for synthesis-by-rule. Proc ESCA Workshop on Speech Synthesis, Autrans, France (1990)

20.65 G. Peterson, W. Wang, E. Sivertsen: Segmentation techniques in speech synthesis, J. Acoust. Soc. Am. **32**, 639–703 (1958)

20.66 N.R. Dixon, H.D. Maxey: Terminal Analog Synthesis of Continuous Speech Using the Diphone Method of Segment Assembly, IEEE Trans. Audio Electroac. **AU-16**, 40–50 (1968)

20.67 J.P. Olive: Rule synthesis of speech from dyadic units, Proc. ICASSP **77**, 568–570 (1977)

20.68 R. H. Mannell: Formant diphone parameter extraction utilising a labeled single speaker database. In: Proc. ICSLP-98 (1998)

20.69 H. Mori, T. Ohtsuka, H. Kasuya: A data-driven approach to source-formant type text-to-speech system, ICSLP **2002**, 2365–2368 (2002)

20.70 S. Hertz: Integration of Rule-Based Formant Synthesis and Waveform Concatenation: A Hybrid Approach to Text-to-Speech Synthesis, In: Proc. IEEE 2002 Workshop on Speech Synthesis, 11–13, Santa Monica (2002)

20.71 D. Talkin: Looking at Speech. In: Speech Technology, 74–77 (1989)

20.72 A. Acero: Formant analysis and synthesis using hidden Markov models, Proc. Eurospeech **99**, 1047–1050 (1999)

20.73 M. Lee, J. van Santen, B. Möbius, J. Olive: Formant tracking using context-dependent phonemic information, IEEE TSAP **13**(5), 741–750 (2005)

20.74 A.-M. Öster: The use of a synthesis-by-rule system in a study of deaf speech, STL-QPSR 1/ **1985**, 95–107 (1985)

20.75 B. Granström, A.-M. Öster: Speech synthesis for hearing impaired persons – in research, training and communication, STL/QPSR 2-3/ **94**, 93–111 (1994)

20.76 A. Kain, X. Niu, J. Hosom, J. Miao, J. van Santen: Formant Re-synthesis of Dysarthric Speech. Proceedings of IEEE Workshop on Speech Synthesis (2004)

20.77 I. Murray, J. Arnott, N. Alm, A. Newell: A communication system for the disabled with emotional synthetic speech produced by rule, Procs. Eurospeech **91**(1), 311–314 (1991)

20.78 P. A. Cudd, S. Hunnicutt, J. Arthur, B. Granström, S. Aguilera, B. Waernulf, P. Dalsgaard, G. Wilson: Voices, attitudes and emotions in speech synthesis. In Placencia Porrero, I., and Puig de la Bellacasa, P. (Eds.), *Proc of 2nd TIDE Congress on The European Context for Assistive Technology* (pp. 344-347). Paris, Amsterdam: IOS Press Ohmsha (1995)

20.79 J. Cahn: The generation of affect in synthesized speech, J. Am. Voice I/O Soc. **8** (1990)


Amazon EX1010 Page 461

# 21. Corpus-Based Speech Synthesis

T. Dutoit

In this chapter, we present the main trends in corpus-based speech synthesis, assuming a stream of phonemes and prosodic target as input. From the early diphone-based speech synthesizers to the state-of-the art unit-selection-based synthesizers, to the promising statistical parametric techniques, we emphasize the engineering trade-offs that arise when designing such systems.

In particular, we examine the mathematical foundations of available methods for modifying the fundamental frequency and the duration of speech units for concatenative synthesis, as well as for smoothing discontinuities at concatenation points. For each of these problems, we analyze time- and frequency-domain processing, using algorithms such as time-domain pitch-synchronous overlap-add (TD-PSOLA), multiband resynthesis overlap-add (MBROLA), and the harmonic-plus-noise model (HNM).

We then provide a comprehensive description of how and why concatenative speech synthesis has progressively adopted large speech corpora, using the principle of context-oriented clustering as a smooth transition from fixed inventory synthesis to unit selection and statistical parametric synthesis.

Our description of unit selection emphasizes important issues related to the definition of optimal target and concatenation costs, as well as to the design of the speech corpus (including memory cost issues) and the reduction of computational costs.

We conclude the chapter with the mathematical framework underlying HMM-based speech synthesis and an outline of its main perspectives.

21.1 **Basics** ..... 437

21.2 **Concatenative Synthesis with a Fixed Inventory** ..... 438
21.2.1 Diphone-Based Synthesis ..... 438
21.2.2 Modifying Prosody ..... 440
21.2.3 Smoothing Joints ..... 444
21.2.4 Up from Diphones ..... 445

21.3 **Unit-Selection-Based Synthesis** ..... 447
21.3.1 Selecting Units ..... 447
21.3.2 Target Cost ..... 448
21.3.3 Concatenation Cost ..... 448
21.3.4 Speech Corpus ..... 449
21.3.5 Computational Cost ..... 450

21.4 **Statistical Parametric Synthesis** ..... 450
21.4.1 HMM-Based Synthesis Framework ..... 451
21.4.2 The State of the Art and Perspectives ..... 453

21.5 **Conclusion** ..... 453

**References** ..... 453



## 21.1 Basics

Text-to-speech (TTS) synthesis is often seen by engineers as an easy task compared to speech recognition. In an international conference on speech processing, a famous scientist once brought up a tube of toothpaste (whose brand was in fact Signal) and, pressing it in front of the audience, he coined the words: *This is speech synthesis; speech recognition is the art of pushing the toothpaste back into the tube* ... It is true, indeed, that it is easier to create a bad, first trial TTS system than to design a rudimentary speech recognizer. After all, recording numbers up to 60 and a few words (*it is now*, *am*, *pm*) and being able to play them back in a given order provides the basis of a working talking clock, while trying to recognize such simple words as *yes* or *no* immediately implies some more-elaborate signal processing. If speech synthesis was really *that* simple, however, one could only blame the text-to-speech (TTS) research and development (R&D) community for not having been able to massively produce a series of talking consumer products as early as the 1980s.

Amazon EX1010 Page 462

The major point is that users are generally much more tolerant to automatic speech recognition (ASR) errors than they are willing to listen to unnatural speech. There is magic in a speech recognizer that transcribes continuous radio speech into text with a word accuracy as low as 50%; in contrast, even a perfectly intelligible speech synthesizer is only moderately tolerated by users if it delivers nothing other than *robot voices*.

This importance of *naturalness* versus *intelligibility* is actually very typical of the synthesis of natural signals (as opposed to their recognition). One could thus advantageously compare speech synthesis to the synthesis of human faces: while it is quite easy to sketch a cartoon-like drawing that will be unanimously recognized as a human face, it is much harder to paint a face that will be mistaken for a photograph of a real human being. To a large extent you can modify the size, position, and orientation of most of the elements of a hand drawing without breaking the intelligibility barrier (just think of the cubists, etc.), but even a slight change to a photorealistic painting will immediately make the complete work look like what it actually is: a painting of a face, not a real photograph of it.

Delivering intelligibility *and* naturalness has thus been the holy grail of speech synthesis research for the past 30 years. Speech expressivity is now increasingly seen as an additional objective to reach (Chap. 25). Add to it that engineering costs (computational cost, memory cost, design cost for having another synthetic voice or another language) have always had to be taken into account, and you will start to have an approximate picture of the challenges underlying text-to-speech synthesis.

This chapter outlines the specific problems encountered when trying to reach these goals, and shows how today's most-advanced solutions benefit from *corpus-based* speech synthesis techniques. We basically distinguish three approaches: *concatenative synthesis based on a fixed inventory* (Sect. 21.2), in which speech is obtained by gluing speech chunks taken from a limited-sized, carefully prepared, unit inventory; *concatenative synthesis based on unit selection* (Sect. 21.3), in which the units inventory is several orders of magnitude larger; and *statistical parametric synthesis* (Sect. 21.4), in which speech synthesis implies the adequate concatenation of statistical models of speech units.



## 21.2 Concatenative Synthesis with a Fixed Inventory

Producing speech samples automatically does not merely reduce to the playback of a sequence of prerecorded words or phonemes, due to *coarticulation*. Coarticulation results from the fact that each articulator moves continuously from the realization of one phoneme to the next. It appears even in the most carefully uttered speech. In fact, it *is* speech. Thus, producing intelligible speech requires the ability to produce continuous, coarticulated speech.

More generally, transients in speech are more important for intelligibility than stable segments [21.1, 2], while modifying stable segments (e.g., the center of vowels) can very easily affect naturalness. Speech, indeed, is never really periodic, nor stable. Even sustained vowels exhibit small frequency and amplitude variations (respectively termed *jitter* and *shimmer*), and have substantial inharmonic components due to non-complete closure of the vocal folds after the so-called *glottal closure instant*; the presence of these noise-like components is correlated with specific events within the pitch period. As a result, the intuitive concept of *adding a bit of noise* to intonation curves, to amplitude curves, or to voiced speech waveforms in order to *make them sound more natural* merely leads to more-noisy synthetic speech: inharmonicity in voiced speech is not pure randomness.

Concatenative synthesis techniques try to deliver the expected intelligibility and naturalness of synthetic speech by gluing together speech chunks that embody natural coarticulation, jitter, shimmer, and inharmonicity.

### 21.2.1 Diphone-Based Synthesis

Constraining speech chunks to embody coarticulatory effects can be achieved to some extent by using *diphones* (or *dyads*) as basic units. A diphone is a speech segment that starts in the middle of the stable part (if any) of a phoneme and ends in the middle of the stable part of the next phoneme [21.3]. Diphones therefore have the same average duration as phonemes (about 100 ms), but if a language has $N$ phonemes, it typically has about $N^2$ diphones, or slightly fewer in practice since not all diphones are encountered in natural languages. This leads to a typical diphone database size of 1500 diphones

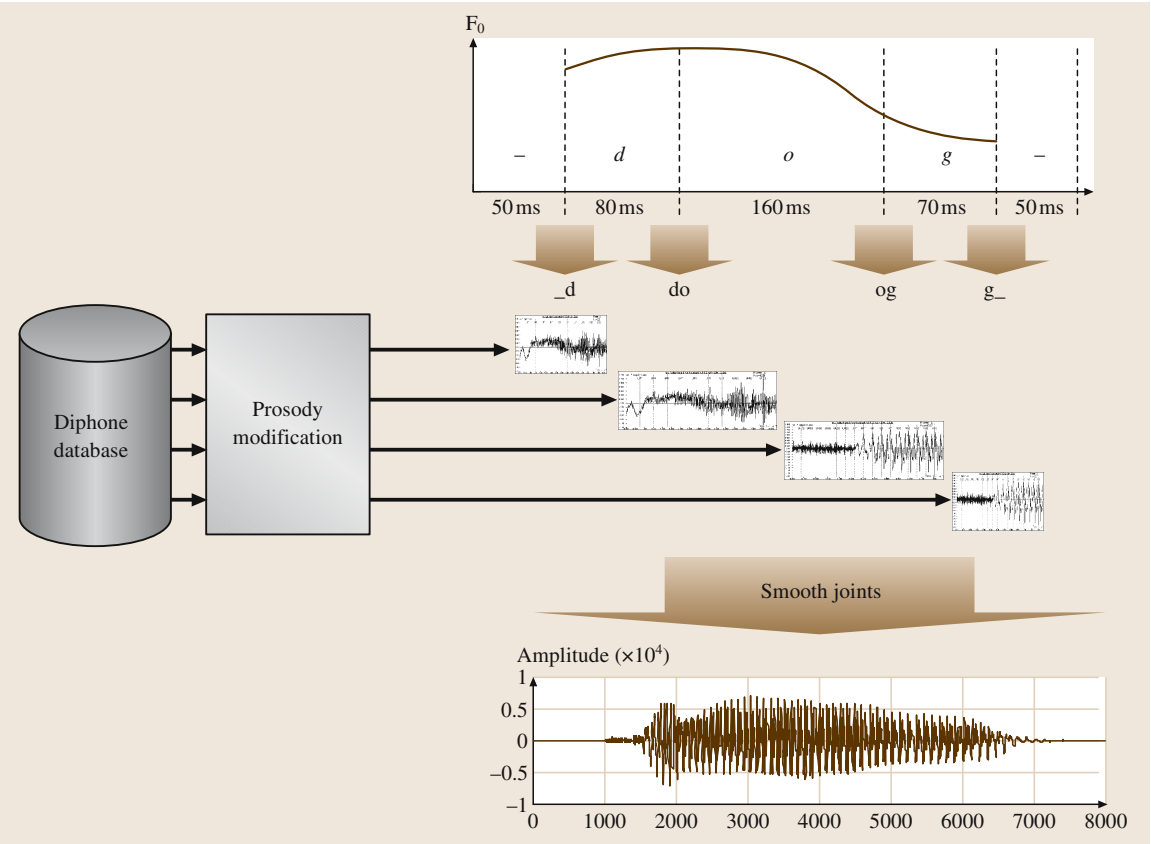



**Fig. 21.1** A schematic view of a diphone-based speech synthesizer. In order to produce the word 'dog' with given phoneme durations and fundamental frequency, the diphone inventory (or database) is queried for the corresponding sequence of diphones. The duration and pitch of each diphone is modified, and some smoothing is applied on joints

(about 3 min of speech, i. e., about 5 MB for speech sampled at 16 kHz/16 bits).

In order to create such a synthesizer, one needs to set up a list of the required diphones; a corresponding list of words is carefully completed in such a way that each segment appears at least once (twice is better, for security). Unfavorable positions, such as those inside strongly stressed syllables or in strongly reduced (over-coarticulated) contexts, are usually excluded. A corpus is then read by a professional speaker (avoiding speaking-style variations as much as possible, and even possibly without large pitch variations, so as to facilitate speech analysis) and digitally recorded and stored. The elected segments are spotted, either manually with the help of signal visualization tools, or automatically thanks to segmentation algorithms, the decisions of which are checked and corrected interactively. Segment waveforms are then collected into a diphone inventory (or database). This operation is performed only once.

At run time, once the synthesizer (Fig. 21.1) receives some phonetic input (phonemes, phoneme duration, pitch) from the natural language processing (NLP) module, it sets up a list of required diphones, together with their required duration and fundamental frequency contour. The available diphones would only match these prosodic requests by chance, since they have generally been extracted from words and sentences which may be completely different from the target synthetic sentence. Some *prosody modification* is therefore necessary. How this is achieved is examined in more detail in Sect. 21.2.2. Additionally, since the diphones to be concatenated have generally been extracted from different words – that is, in different phonetic contexts – the end of one diphone and the beginning of the next often do not fully match in terms of amplitude and/or spectral

envelope. Part of this problem can be solved by *smoothing* individual pairs of successive segments. Smoothing will be discussed in Sect. 21.2.3. In Sect. 21.2.4, we will see what solutions are available for expanding the unit inventory.

## 21.2.2 Modifying Prosody

*Prosody* refers to the properties of a speech signal that are related to audible changes in pitch, syllable length, and loudness (Chap. 23).

Of the three parameters, pitch is clearly the most sensitive. Even slightly modifying the shape of the pitch curve readily breaks the naturalness of a speech segment. This effect is still more audible when pitch modification ratios on both sides of concatenated units differ too much. Applying a constant pitch-modification ratio on a complete segment (as in [21.5]) does less harm, provided that the ratio is not too high (typically $\leq 2$) or too low (typically $\geq 0.5$). Applying duration modification to diphones to match the required phoneme length is less sensitive, provided the ratio is reasonable ($\leq 2$ and $\geq 0.5$). Modifying the loudness of diphones on the fly is trivial, but generally not much used in diphone-based synthesis. An offline equalization step is generally preferred, in which related endings of diphones are given similar amplitude spectra, the difference being distributed over their neighborhood

Neither pitch nor duration modification is a straightforward operation. Simply resampling speech data (creating additional speech samples by interpolation of original ones while keeping the same sampling frequency for playing them) does modify duration, but also the spectral envelope. All formants become compressed/dilated along the frequency axis, which has exactly the same audible effect as that of changing the *pitch* wheel of a tape recorder: human voices turn into doggy or mousy voices.

Two main classes of algorithms have been proposed for prosody modification, namely *time-domain* and *frequency-domain* algorithms.

### Time-Domain Prosody Modification

The distinctive feature of the time-domain pitch-synchronous overlap-add (TD-PSOLA) algorithm [21.6] is that it makes it possible to perform both the required pitch and duration modifications directly on continuous waveforms, without any parametric speech modeling.

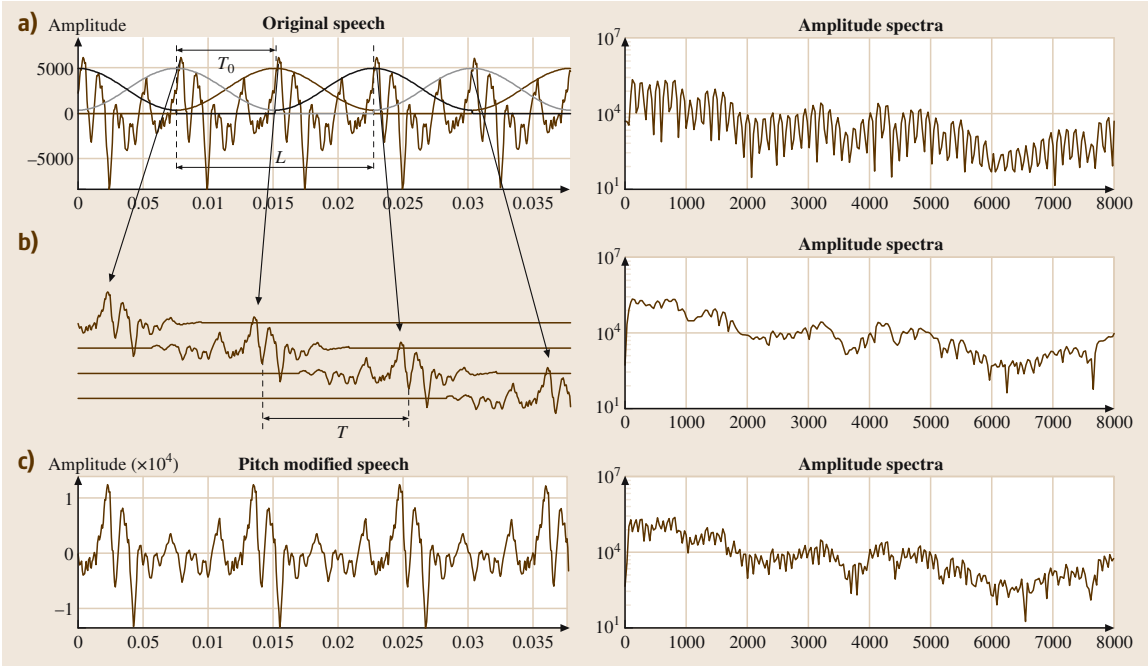



**Fig. 21.2a–c** The TD-PSOLA reharmonization process. The pitch-modified waveform **(c)** has the same spectral envelope as the original waveform **(a)**. The center plot **(b)** shows the OLA frames and the amplitude of their Fourier transform, which is approximately the spectral envelope of the initial signal (after [21.4])

As a matter of fact, if we restrict ourselves to a purely periodic signal $s(n)$, it is possible to obtain a perfect pitch-modified version of $s(n)$ by summing overlap-add (OLA) frames $s_i(n)$, extracted pitch-synchronously from $s(n)$ with a weighting window $w(n)$, and changing the time shift between frames from the original pitch period $T_0$ to the desired period $T$:

$$s_i(n) = s(n)w(n - iT_0)\,, \tag{21.1}$$

$$\tilde{s}(n) = \sum_{i=-\infty}^{\infty} s_i[n - i(T - T_0)]\,. \tag{21.2}$$

If $T \neq T_0$, the operation results, according to the Poisson formula, in a reharmonization of the spectrum of $s_i(n)$ (which, if we assume perfect periodicity, does not depend on $i$ when the original voiced sound is purely periodic) with fundamental frequency $1/T$:

$$\text{if } s_i(n) \overset{F}{\longleftrightarrow} S_i(\omega)$$

$$\text{then } \tilde{s}(n) \overset{F}{\longleftrightarrow} \frac{2\pi}{T} \sum_{i=-\infty}^{\infty} S_i\left(i\frac{2\pi}{T}\right)\delta\left(w - i\frac{2\pi}{T}\right)\,. \tag{21.3}$$

Consequently, provided $w(n)$ can be chosen so that the spectrum of $s_i(n)$ closely matches the spectral envelope of $s(n)$, (21.2) provides a simple and efficient way to change the pitch of a periodic signal (Fig. 21.2).

Obtaining an estimate of the spectral envelope of $s(n)$ through $s_i(n)$ is obtained when the length of the weighting window applied to each OLA frame is approximately twice the local pitch period (Fig. 21.3), which implies that this length is signal dependent. Notice also that $w(n)$ needs to be smoothly zero-ended, so as to avoid clicks at each OLA operation. Typically, Hanning windows are used. Another practical constraint is that $F_0$ modification produces its best quality when OLA frames are centered on the glottal closure instant, or at least on an instant of important speech excitation (as is the case in Fig. 21.2). As a result, TD-PSOLA requires the prior computation (or the semiautomatic setting) of *pitch marks* on all speech units. However, computing glottis closure instants (the only reference instant available in voiced speech) has two major drawbacks: finding glottal closure instants is not very robust if performed automatically, and it is time consuming if performed manually. A possible alternative to this problem is to measure glottal activity directly, with the help of an electroglottograph (EGG), instead of trying to deduce it from speech. Another solution is to use the local center of gravity of speech as the center of the OLA frames [21.7].

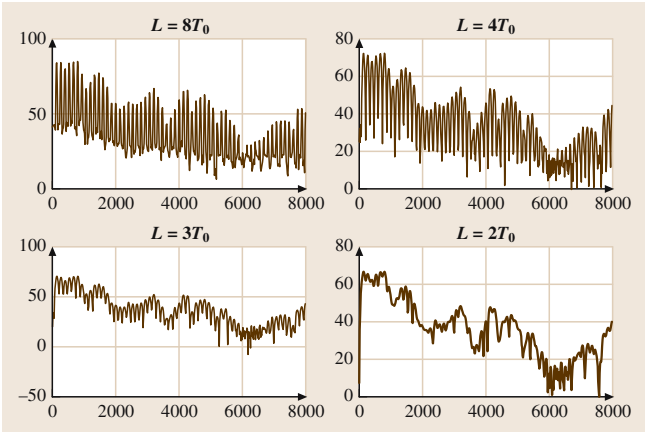



**Fig. 21.3** Amplitude spectra of the OLA frames extracted from the vowel [a] for several values of the weighting window length $L$ ($F_s$: 16 kHz; window: Hamming). Choosing $L = 2T_0$ provides an approximation of the spectral envelope

For simultaneous duration and pitch modification, pitch-synchronous frames must be used. Let us first assume that, prior to any processing, we associate with each analysis pitch mark $\eta^i$ the value of its local pitch period $T_0$, so that a pair $(\eta^i, T_0^i)$ suffices to define the corresponding analysis OLA frame $s_i(n)$. What we want to do is a mapping between pitch marks on the analysis time axis $t$ and corresponding pitch marks on the synthesis time axis $t'$, where the relationship between $t$ and $t'$ is defined by a function $t'(t)$, so as to impose pitch on the synthesis time axis to be equal to some function

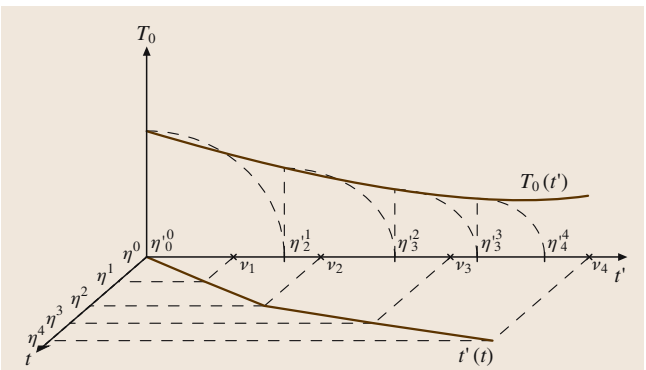



**Fig. 21.4** Pitch and timing modifications with TD-PSOLA. Starting from five analysis OLA frames on the analysis time axis ($t$), four synthesis pitch marks are positioned on the synthesis time axis ($t'$), and each is respectively associated to an analysis frame. In the example shown, frame 1 has been eliminated, while frame 2 has been duplicated (after [21.4])



Amazon EX1010 Page 466

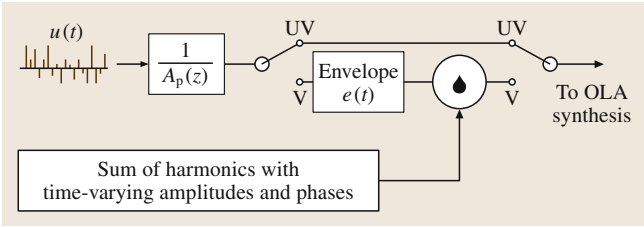



**Fig. 21.5** Speech synthesis in the HNM model

$T_0(t')$ (Fig. 21.4). It is easy to derive the position of synthesis pitch marks $\eta'^j$: the shift between two successive synthesis pitch marks $\eta'^j$ and $\eta'^{j+1}$ is adjusted to the value of the synthesis pitch period at time $\eta'^j$: $T_0(\eta'^j)$. By applying the time alignment function $t'(t)$ to each initial pitch mark position $\eta^i$, we also obtain the corresponding *virtual pitch marks*, denoted by $\eta^i = t'(\eta^i)$. It is then easy to associate an analysis pitch mark $\eta^i$ with each synthetic pitch mark $\eta'^j$ by looking for the closest virtual pitch mark on the synthesis time axis (Fig. 21.4). Given the time distortion introduced by $t'(t)$, some analysis pitch marks are not associated with synthesis pitch marks, while others are duplicated. A list of synthesis parameters is finally created, defined by triplets $(\eta'^j, \eta^i, T_0^i)$ in which $\eta'^j$ gives the position of each synthesis OLA frame and $\eta^i$, $T_0^i$ refers to the analysis OLA frame associated to the current synthesis frame. These triplets are indicated as ${\eta'^j}_i$ in Fig. 21.4.

Notice that duration-only modification can be achieved in the time domain by simply replicating or eliminating some OLA frames. In this case, OLA frames need not be pitch-synchronous (as in the waveform similarity overlap add (WSOLA) method [21.8], and the pointer interval controlled overlap and add (PICOLA) algorithm [21.9], implemented in the ISO MPEG-4 audio standard). When two noncontiguous frames are overlapped, a cross-correlation analysis is performed to find the optimal lag or lead between the frame centers of the source waveforms. In order to prevent the algorithm from introducing artificial short-term autocorrelation in the synthesis signal when unvoiced OLA frames are *n*-plicated (i. e., when speech is very much slowed down), every other duplication of an unvoiced frame can be reversed in time (which does not change its spectral content). With such an artifice, speech can reasonably be slowed down by a factor of four, even though some tonal effect is encountered in voiced fricatives, which typically combine voiced and unvoiced frequency regions and therefore cannot be inaudibly time reversed.

TD-PSOLA has been widely used in diphone-based synthesis, because of its high synthesis quality and extraordinarily low computational load (typically seven operations per sample). Synthesized speech, however, is not perfect: spectral mismatches cannot easily be eliminated at segmental boundaries and tonal quality appears when large prosodic modifications are applied on the concatenated units.

### Frequency-Domain Prosody Modification

Sinusoidal approaches and hybrid harmonic/stochastic representations flourished in speech processing as early as the early 1980s [21.10, 11], and have been proposed for speech synthesis [21.7, 12]. In particular, the

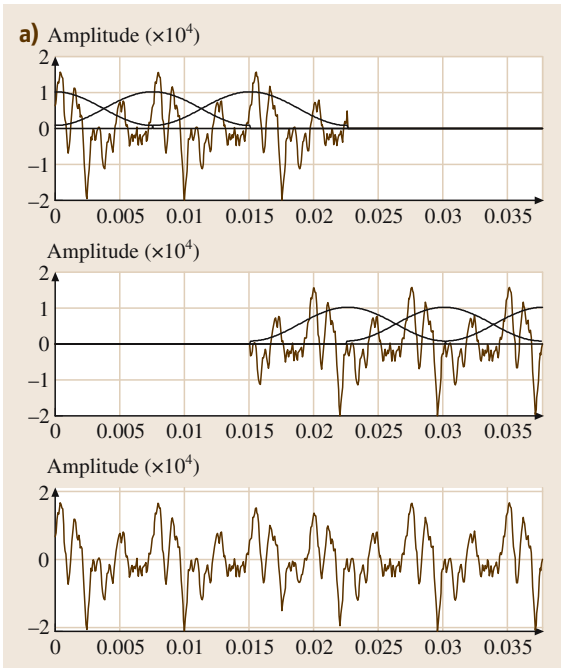



**Fig. 21.6a–c** *Top*: last frames of the first segment to be concatenated; *center*: first frames of the second one; *bottom*: after OLA; **(a)** Phase mismatch: waveforms are identical but the OLA windows are not centered on the same relative positions within the period. **(b)** Pitch mismatch: both segments have exactly the same spectral envelope, but were pronounced with different pitches. OLA windows are positioned coherently. The synthetic signal is obtained after changing the interval between the frames of the right segment in order to impose a constant pitch. **(c)** Spectral envelope mismatch: the left segment is the diphone 'ma'; the right one is 'am'. The pitch is constant, and the windows are again positioned coherently. The spectral discontinuity is concentrated in one period (after [21.4]) ▶⇒▶▶



Amazon EX1010 Page 467

harmonic-plus-noise model (HNM) [21.7]) is widely used because of its additional smoothing (Sect. 21.2.3) and speech compression capabilities. In this model (Fig. 21.5), speech is assumed to be composed of two additive components, a harmonic component $h(t)$, and a noise component $n(t)$, corresponding to a separation of the speech spectrum into two bands:

$$\tilde{s}(t) = h(t) + n(t) \,. \tag{21.4}$$

A time-varying maximum voiced frequency determines the limit between the two bands. In the lower band, the signal $h(t)$ is represented as a sum of $K(t)$ harmonically related sine waves with slowly varying amplitudes and frequencies:

$$h(t) = \sum_{k=1}^{K(t)} A_k(t) \cos[k\theta(t) + \phi_k(t)] \tag{21.5}$$

$$\text{with } \theta(t) = \int_{-\infty}^{t} \omega_0(l)\,\mathrm{d}l \,. \tag{21.6}$$

The amplitudes $A_k(t)$ and phases $\phi_k(t)$, as well as the fundamental frequency $\omega_0(t)$ of these sine waves, are estimated from real speech signal by minimizing a weighted time-domain least-squares criterion [21.7]. The estimation is repeated for successive speech frames, extracted from the input speech with constant frame shift $S$ (i. e., non-pitch-synchronously). In other words, the centers of the analysis frames $\eta^i$ are obtained by

$$\eta^{i+1} = \eta^i + S \,. \tag{21.7}$$

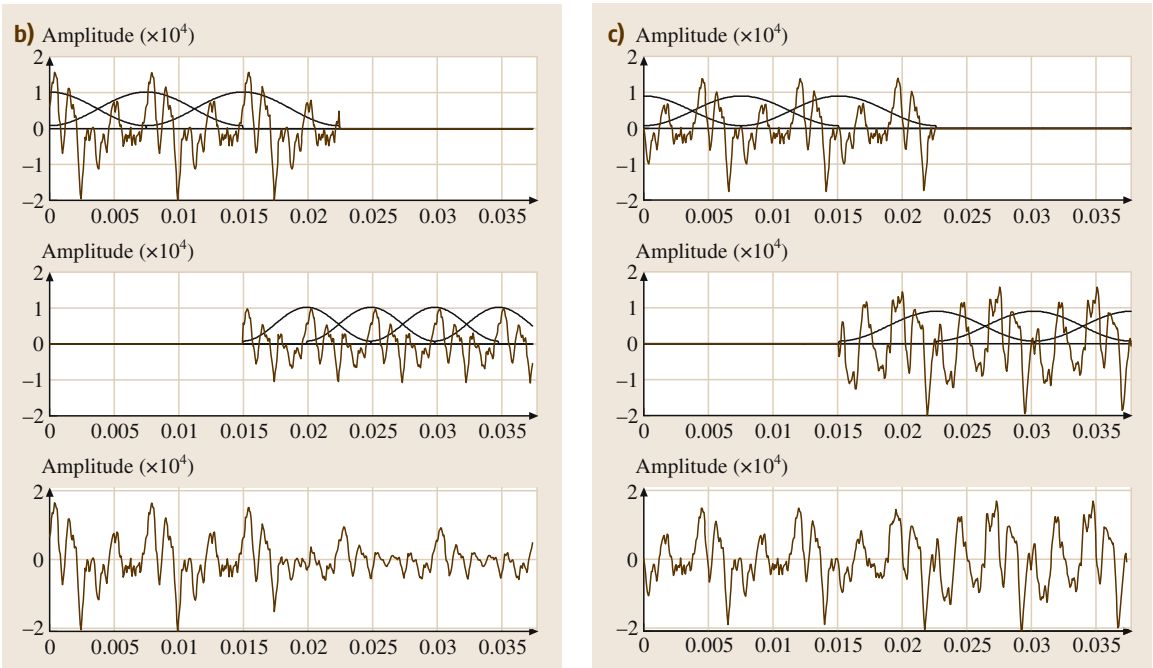
The upper band component, i. e., the noise part $n(t)$, is modeled by filtering a white Gaussian noise $u(t)$ by a time-varying, normalized all-pole filter $h(\tau, t)$ and multiplying the result by a time-domain amplitude envelope function $e(t)$:

$$n(t) = e(t)[h(\tau, t) * u(t)] \,. \tag{21.8}$$

This makes it possible to account for the nonperiodic components of speech, which include frication noise and period-to-period variations of the glottal excitation. In particular, the envelope function is synchronized with the harmonic part. If this is not the case, then the noise part is not perceptually integrated with the harmonic part but is rather perceived as additive, separate noise.

In order to modify the pitch and duration of speech units modeled by HNM, a set of synthesis pitch marks $\eta'^j$ are first obtained from the $\eta^i$ through the same operation as in Fig. 21.4. Then, for each synthesis pitch mark found, pitch modification implies estimating amplitudes and phases of harmonics at the new harmonic frequencies by resampling the complex speech spectrum.

The last step is the generation of the synthetic signal itself using the stream of modified



HNM parameters. Synthesis is performed pitch-synchronously using an overlap and add (OLA) process. OLA is performed on synthesis OLA frames $\tilde{s}_i(t)$ computed from (21.3), using the aforementioned pitch-modified amplitudes and phases. Synthesis OLA frames $\tilde{s}(t) = h(t) + n(t) = \sum_i a_i(t)\cos(\phi_i(t)) + \tilde{s}_r(t)$ are made pitch-synchronous by adjusting the phases of their harmonics, so as to impose their center of gravity. In practice, the noise part is also filtered by a high-pass filter with a cut-off frequency equal to the maximum voiced frequency associated with the frame.

### 21.2.3 Smoothing Joints

Concatenating speech units possibly extracted from different words or phonetic contexts is not straightforward. Done without care, it produces audible clicks, which cannot always be removed by simple OLA between the end of the left unit and the beginning of the right unit. This is due to at least three types of mismatches between successive units, respectively related to phase, pitch, and spectral envelope mismatches [21.13], and illustrated in Fig. 21.6 for the concatenation of vowels [a] sampled at 16 kHz.

#### Phase Mismatch

*Phase mismatch* originates in overlap-add frames that are not centered at the same place within the period (Fig. 21.6a); even if the wave shapes to be concatenated are identical, the OLA operation results in an unexpected discontinuity in this case.

With time-domain methods, this can be avoided by adequate pitch-marking of boundary frames (as done in TD-PSOLA for all speech segments). Another solution is to adjust the position of OLA frames on the left and right units so as to maximize the cross-correlation between windowed frames (as in WSOLA), thereby maximizing the chance that OLA frames are synchronous [21.8]. This solution, which corresponds to finding a common relative synchronization point, is easier than absolute synchronization on glottal closure instants, but it is more time-consuming (it is usually performed online, since synchronization points cannot be precomputed and stored for all possible pairs of speech segments). It is also possible to use the aforementioned center of gravity of the OLA frames as an absolute reference point for centering frames [21.7], which can be done offline.

With frequency-domain techniques, phase mismatch can be avoided by making sure that all synthesis OLA frames have the same initial harmonic phases. This is done in the *multiband resynthesis OLA* (MBROLA) technique, a time-domain technique based on speech samples that have preliminarily (once and for all) been edited using frequency-domain modification [21.13]. As a result of this modification, MBROLA frames have identical pitch and harmonic phases. A more-realistic option, used with HNM, is to measure phase mismatches before synthesizing OLA frames, and making sure that the center of gravity of the synthesis OLA frames are synchronized by correcting their phases (this corresponds to imposing the phase of the fundamental on all synthesis OLA frames).

#### Pitch Mismatch

*Pitch mismatch* arises when overlapped frames have very different fundamental periods, even if they have identical spectral envelopes and are windowed on similar relative positions within the pitch period. This situation cannot easily be avoided in a database of several thousand segments (Fig. 21.6b).

It is nevertheless possible to minimize this effect by recruiting professional speakers to record the segments database and training them to read the corpus with the most constant pitch possible (TD-PSOLA) or by editing the recorded speech so as to impose constant pitch (MBROLA).

Another solution is to apply frequency-domain pitch modification on the fly before the OLA operation, as in HNM. First, the difference between the pitch value of the last frame on the left and of the first frame on the right of a concatenation point is measured. Then this difference is distributed on several frames on the left and on the right of the concatenation point. This is typically achieved by resampling the complex speech spectrum of these frames and computing new harmonic amplitudes and phases for the new pitch value.

#### Spectral Envelope Mismatch

Last but not least, *spectral envelope mismatch* is due to coarticulation and speaker variability and appears whenever the speech units to be concatenated have been extracted from rather different phonetic contexts (Fig. 21.6c).

It can be minimized somewhat, by using adjustable unit boundaries [21.14] (i. e., by changing the left and right borders of the diphones on the fly).

The standard implementation of TD-PSOLA, however, cannot smooth out spectral envelope mismatches. As a result, synthesizing highly fluent speech with TD-PSOLA often requires a painstaking trial-and-error process in which the most probable diphone sequences



Amazon EX1010 Page 469

are systematically tested and diphones are rejected when an important spectral envelope mismatch is encountered. Other versions of the rejected diphones are then extracted from words with a different phonetic context than the initial one; tests are run again, and so on.

A partial solution consists of resynthesizing the speech segment database so as to provide segments with interesting properties which can be taken into account at synthesis time, as in MBROLA. Since all MBROLA OLA voiced frames have the same $F_0$ and the same initial harmonic phases, it is possible to perform period-based smoothing in the time domain. This feature somehow compensates for the loss of segmental quality due to the resynthesis operation.

It is also possible to achieve spectral envelope smoothing in the frequency domain. Harmonic/stochastic models such as HNM proceed by linearly interpolating harmonic amplitudes (perceptually, discontinuities in the parameters of the noise part are not important and are thus ignored). First, the differences between the amplitudes of each harmonic (between frames on the left and frames on the right of a concatenation point) are measured. Then, these differences are weighted and propagated left and right from the concatenation point. The number of frames used in the interpolation process depends on the variance of the number of harmonics and the size, in frames, of the concatenated speech units.

Such a simple linear interpolation of the spectral envelopes makes formant discontinuities less perceptible. However, if formant frequencies are very different at the left and right of the concatenation point, the problem is not completely solved. A partial solution to this was proposed in [21.15], where smoothing between two units (the unit on the left $l$ and the unit on the right $r$) is performed with reference to a template speech segment $lr$ (i. e., a natural example of $lr$ taken from continuous speech). The template segment, termed the *fusion unit*, is merged with the end of $l$ and the beginning of $r$, in the frequency domain.

### 21.2.4 Up from Diphones

The acoustic units chosen as building bricks for a concatenative synthesizer should obviously exhibit some basic properties: they should account for as many coarticulatory effects as possible; they should be available with all sorts of durations, pitch contours, and voice qualities (so as to alleviate the task of the prosody modification block); they should be easily connectable (without requiring complex rule-based smoothing techniques of formant synthesizers for instance; Chap. 20). Their number should also be as small as possible (to avoid having to prepare, record, segment, and store a large unit inventory), although on the other hand longer units (hence, larger unit inventories) decrease the density of concatenation points, therefore providing better speech quality.

Diphones are a good compromise when the number of units has to be kept really small (i. e., typically for cheap embedded applications). They imply, however, a high density of concatenation points (one per phoneme), which reinforces the importance of an efficient concatenation algorithm. Besides, they can only partially account for the many coarticulatory effects of a spoken language, since these often affect a whole phoneme rather than just its right or left halves independently: formant targets themselves tend to be changed, or a formant starts rising (or falling) during the realization of one phoneme and continues that movement smoothly throughout the ensuing one without any steady state. Such effects are especially obvious when somewhat transient phones, such as liquids and (worst of all) semivowels, are concatenated.

Assimilations (i. e., very strong forms of coarticulation, in which a phonetic trait of a phoneme is completely modifies as a result of the preceding or following phonemes) are still a bigger source of concern. Partial assimilation cases are approximately covered by diphones, while total cases would require the introduction of several realizations per phoneme in the database (as many as there are allophones) and a corresponding number of diphones. Such a strategy is sometimes called *allo-diphone-based synthesis*.

Triphones (which embody a complete phoneme) constitute a variant to the use of allo-diphonic units as a complement to diphones. They can embody the strong dynamic effects quoted above, providing these only affect one phoneme. To a larger extent, associating diphones with tri- and tetraphones to cope with a limited number of very specific contextual effects is often called a *polyphone* approach [21.16]. France Telecom [21.17], for instance, has reported on an enhanced French polyphone database composed of 1290 diphones, 751 triphones, and 288 quadriphones.

Another way of keeping the database size small while taking syllables into account to a degree is to split them into two at their vocalic nucleus, which acts as a coarticulation barrier in most cases. Such demi-syllables (of the form $C$–$V$ or $V$–$C$) have been fully adopted for the synthesis of German, in which long consonant clusters are manifold. Such a combination of

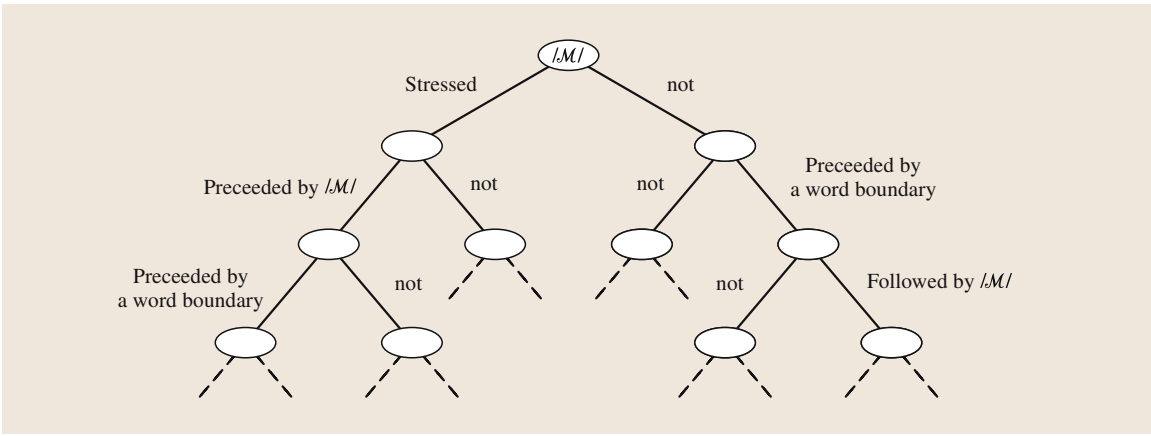



**Fig. 21.7** An example of the COC cluster splitting process for occurrences of the phoneme /ɱ/ (after [21.19]). As a result of this process, allophones of /ɱ/ are detected, and explained



HAlf syllables, DIphones, and sufFIXes is the basis of the HADIFIX system [21.18].

Notice, finally, that all the types of speech units proposed above are based on some pre-established phonetic knowledge of coarticulatory or assimilation effects and where they mostly occur. In contrast, nothing truly impedes them from being directly automatically derived on the basis of some automatic analysis of the distribution of their spectral envelopes. This has led to the proposal of a corpus-based, automatic unit set design algorithm termed *context-oriented clustering* (COC) [21.19].

COC builds a decision tree for each phoneme, which automatically *explains* the variability in the acoustic realization of this phoneme in terms of contextual factors. Starting from a speech database with phonetic labels, it constitutes initial clusters composed of all the speech units (allophones in this case) with the same phonetic label (Fig. 21.7). The context tree is then automatically derived by a greedy algorithm. At each step, the cluster with the highest variance (and with more than a minimum number of segments) is split into two classes on the basis of a partition of a *context factor*, and provided the resulting split leads to an average subcluster variance that is significantly lower than the initial one. Intracluster variances are computed on segments after a normalization of their duration, based on a linear warping technique. Notice that COC is very similar to the tree-based decomposition of classification and regression tree (CART) techniques. The major difference is that CARTs are trained in a *supervised* way (they account for pre-established decisions), while COC methods are *unsupervised*. Hence the use of an acoustic distance to obtain the best contextual factor, in place of the maximum mutual information principle of CARTs. Possible context factors used (i. e., the factors that are submitted to the tree-building algorithm to partition clusters) are the phonetic labels of neighboring phones up to a given distance from the central one, as well as stress and syntactic boundaries. Broad phonetic classes can also be used as contextual factors, so as to reduce the size of the decision tree and to cope with the problem of data sparsity.

It is important to understand that the output of this COC method, when used for concatenative synthesis using fixed inventories, is basically the resulting tree. Once the tree has been built, the content of its leaves (the speech units taken from the speech database and associated to leaves) is reduced to a single representative speech unit (even possibly rerecorded on purpose). This technique can thus be seen as an automatic method for building a list of allophones for each phoneme (i. e., variants of a given phoneme due to a specific phonetic/syntactic/stress context).

This idea was further extended to subphonetic units by first segmenting some large speech corpus using hidden Markov model (HMM)-based forced alignment, and then applying a similar context clustering method to the contents of the HMM states [21.20].

While speech synthesis based on the concatenation of units taken from a fixed inventory is clearly intelligible (and has been so from its very beginning [21.21]), it cannot really be termed natural, whichever set of units are chosen. In the next section, we examine how naturalness can be approached, although sometimes at the expense of intelligibility.

Amazon EX1010 Page 471

## 21.3 Unit-Selection-Based Synthesis

The lack of naturalness of speech synthesized from a fixed inventory (i. e., with the constraint of having only one instance of each possible concatenative unit stored in the unit database) has two main causes. First, this strategy unavoidably biases the unit recording step towards the choice of a somewhat overarticulated instance, which will fit in most contexts and lead to the best intelligibility score. Additionally, as we have seen in the previous section, signal-processing tricks are needed to adapt the pitch and duration of each unit (if only to be able to synthesize stressed and unstressed versions of the same syllable). Some signal degradation is the price to pay for this processing.

It will thus come as no surprise that speech synthesis made a huge step forward when researchers started keeping several instances of each unit, and provided means of *selecting*, at run time, which instance to use for synthesizing a given sentence.

### 21.3.1 Selecting Units

After first trials in this direction in the *ν*-Talk TTS system [21.22], with mixed results, the foundations of unit-selection-based speech synthesis was laid in [21.23, 24], which quickly led to ATR's CHATR TTS system, and to AT&T's NextGen TTS system [21.25]. At



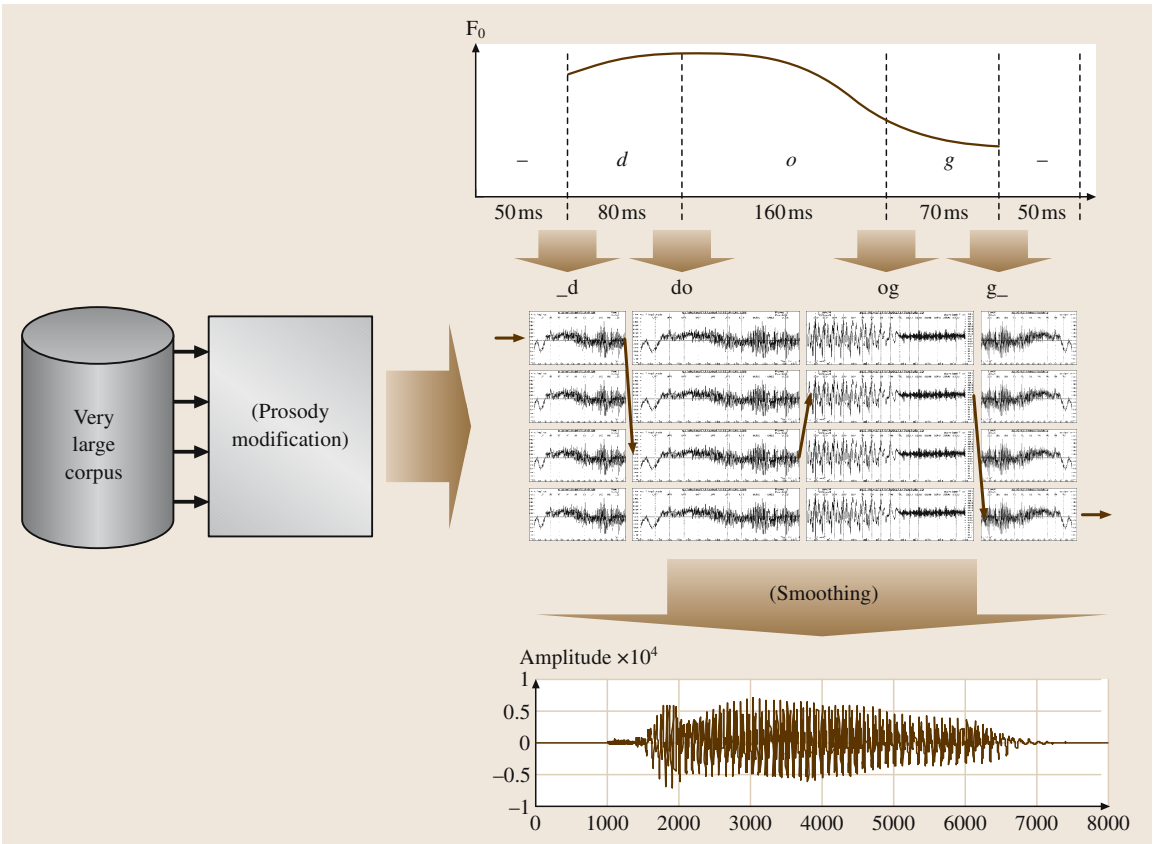



**Fig. 21.8** A schematic view of a unit-selection-based speech synthesizer. The prosody modification and smoothing modules have been included between parentheses, since they are not always implemented. As a matter of fact, since this approach uses very large speech corpora, it is often possible to find speech units that naturally join smoothly while exhibiting prosodic features close to what is expected. Notice that, unlike suggested in Fig. 21.8, unit-selection-based synthesis systems do not systematically use diphone units. Diphones have been shown here for the sake of easy comparison with Fig. 21.1

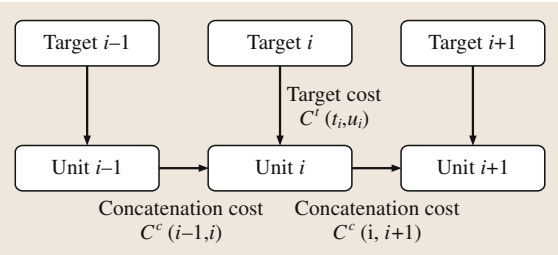



**Fig. 21.9** Target and concatenation costs

run time, given a phoneme stream and target prosody for an utterance, the unit selection algorithm selects from a speech corpus an optimum set of acoustic units, i. e., the one that *best* matches the target specifications (Fig. 21.8, to be compared to Fig. 21.1). Units may still be diphones, although phonemes and subphonetic units such as half-phonemes can be used as alternatives when complete diphones are not available with the required pitch and duration. For every *target unit* $t_i$ required (for example, for every diphone to be synthesized), the selection algorithm first proposes a list of *candidate units* from the speech database, each in a different context (and in general not exactly in the same context as the target unit). The final choice among candidates is based on minimizing the sum of two cost functions: a *target cost* $C^t(t_i, u_i)$, which is an estimate of the difference between a candidate unit $u_i$ and the target unit $t_i$ it is supposed to represent, and a *concatenation cost* $C^c(u_{i-1}, u_i)$, which is an estimate of the quality of the joint between candidate units $u_{i-1}$ and $u_i$ for two consecutive targets (Fig. 21.9). The best sequence of $n$ units $u_1^n$ for a given sequence of $n$ targets $t_1^n$ is the one that minimizes the total cost:

$$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i) + C^c(S, u_1) + C^c(u_n, S) , \quad (21.9)$$

where $S$ denotes the target for a silence. This best sequence is found using a Viterbi search.

When good candidate units cannot be found with the correct pitch and/or duration, prosody modification can be applied. Also, as candidate units usually do not concatenate smoothly (unless a sequence of such candidate units can be found in the speech corpus, perfectly matching the target requirement), some smoothing can be applied (with the methods exposed in Subsects. 21.2.2 and 21.2.3). The latest synthesizers, however, tend to avoid prosodic modifications and smoothing, assuming that the speech unit inventory is rich enough for the requested units to be available with approximately correct pitch and duration [21.26]. This assumption is sometimes referred to as *take the best to modify the least*.

The challenges in unit-selection synthesis are related to the development of efficient target and concatenation costs for finding the *best* path in the candidate unit lattice, to the definition of optimal speech corpus, and to the engineering of efficient search algorithms. These issues are addressed in the next subsections.

### 21.3.2 Target Cost

The optimal target cost should estimate the *perceptual* distance from a specific inventory unit to the desired target unit. Candidate units are characterized by a feature vector. For each target unit in the utterance to be synthesized, appropriate feature values are predicted. The target cost $C^t(t_i, u_i)$ for a given candidate $u_i$ to a given target $t_i$ is calculated as the sum of weighted feature vector differences between the candidate and target. These differences are $p$ target subcosts $C_j^t(t_i, u_i)(j = 1, \ldots, p)$:

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i) . \quad (21.10)$$

Target feature weights $w_j^t$ are trained during the creation of a TTS voice, to optimize the mapping from the feature vector differences between units to the perceptual differences between these units.

Initial experiments on unit selection (such as [21.23]) proposed to use a combination of symbolic features such as phonetic context or stress, and numerical features such as duration and $F_0$. The list of features was later biased towards symbolic (or *phonological*) features, which leave more freedom for the ultimate choice of units while keeping most of the intention of the virtual speaker [21.27]. This is especially useful for intonation, whose numerical prediction is particularly difficult. Instead of trying to predict $F_0$ values from text, most TTS systems currently predict a sequence of abstract *tonal* units, which are only *qualitatively* related to the intended intonative movements. As a result, intonation is obtained as a by-product of unit selection itself, provided that the correct linguistic features are used for the estimation of the target cost.

### 21.3.3 Concatenation Cost

The ideal concatenation cost should reflect the perceived discontinuity between successive units, so as to favor the



Amazon EX1010 Page 473

selection of units such that *cut-and-paste* concatenation will not be noticed.

Many proposals have been made for the best definition of the concatenation cost $C^{c}(u_{i-1}, u_i)$, based on a parameterization (i. e., using some speech model) of the initial and final speech frames of concatenated units, combined with subcosts associated with the values of these $q$ parameters:

$$C^{c}(u_{i-1}, u_i) = \sum_{j=1}^{q} w_j^{c} C_j^{c}(u_{i-1}, u_i) \, . \qquad (21.11)$$

Among the parameters used for this purpose, the most prominent are: formant values, linear prediction (LP) spectra, line spectrum frequencies (LSFs), mel-frequency cepstrum coefficients (MFCCs), and various other frequency-modified spectral representations. Distances between the values of these multi-dimensional parameters vary from the Euclidian distance to the Kullback–Leibler divergence, through weighted Euclidian distances or the Mahalanobis distance. Although it is not clear which one leads to optimal results in all cases (i. e., for all recording conditions and all languages, if possible), several studies have compared combinations of these options on specific corpora, by computing correlations between the values of these parameter/distance combinations and the impression of discontinuity reported by human listeners [21.28]. Clearly, this question is still open; its solution obviously requires (and will lead to) better modeling of perceived discontinuities in speech.

A particular case arises if the complete target sentence, or a significant part of it, is available in the corpus. In this case, the corresponding consecutive units should obviously be selected. This is easily achieved by setting the concatenation cost to zero for originally consecutive units in the speech corpus. As a result, the remarkable naturalness of speech produced by unit selection therefore comes from the fact that they tend to avoid as many concatenation points as possible, by selecting the largest sequences of originally consecutive units [hence the name, nonuniform units (NUU), that is sometimes used for this technique]. The challenge is to make sure that this does not happen at the expense of intelligibility: using units with low concatenation cost but high target cost can lead to a high degree of naturalness (which at first impresses the naïve listener) but low intelligibility.

### 21.3.4 Speech Corpus

Like any other expression of a natural language, however, speech obeys some form of Zipf's law: *In a corpus of natural language utterances, the frequency of any word is roughly inversely proportional to its rank in the frequency table.* The resulting power-law probability distribution implies that speech is composed of a *large number of rare events* (LNRE) [21.29]. It was estimated, for instance, that in order for a diphone database to cover a randomly selected sentence in English with a probability of 0.75 (meaning that the probability is 0.75 that all the diphones required to produce the sentence are available in the database), 150,000 units are required. This corresponds to a speech corpus of about 5 h. Yet the estimation was based on a limited number of contextual features [21.30]. Most industrial systems today use 1–10 h of speech, i. e., 150–1500 Mb, and reach very high quality synthesis *most of the time*.

Unit selection is unquestionably the best option for *limited-domain* speech synthesis [21.31], in which the data sparsity problem is avoided by constraining the vocabulary and the syntactic and prosodic complexity of the sentences to be synthesized. For the synthesis of unrestricted text, a standard interim solution to data sparsity consists of enriching the initial large speech database with a list of *safety units* (typically, a complete set of diphones), aimed at completely covering the phonetics (here meaning the coarticulation) of the language, regardless of prosody. This provides the TTS system with a fallback strategy in case of data unavailability, and ensures a lower-bound naturalness similar to that of fixed-inventory systems in the worst case (which is fortunately only exceptionally encountered). Another option is to try to obtain increasingly large speech databases. The latest TTS system developed at ATR, XIMERA, is based on a 110 h corpus of a Japanese male and a 60 h corpus of a Japanese female [21.32].

Another consequence of the LNRE effect is that designing the corpus (from which the unit database is extracted) to ensure maximum coverage is of primary importance. Several greedy algorithms that progressively build a list of sentences taken from a very large text corpus so as to reach maximum coverage with a minimum number of sentences have been proposed [21.26].

Speech segmentation also plays an important part in the creation of a speech corpus. Given the size of the speech corpus, only automatic speech segmentation methods can reasonably be used, based on forced alignment of speech using HMM models. The HMM models of phonemes used in this case differ from those used in speech recognition in that they are trained from the speaker-dependent data of the target speaker. While such an automatic method inevitably produces segmentation errors (most of them due to a bad pho-

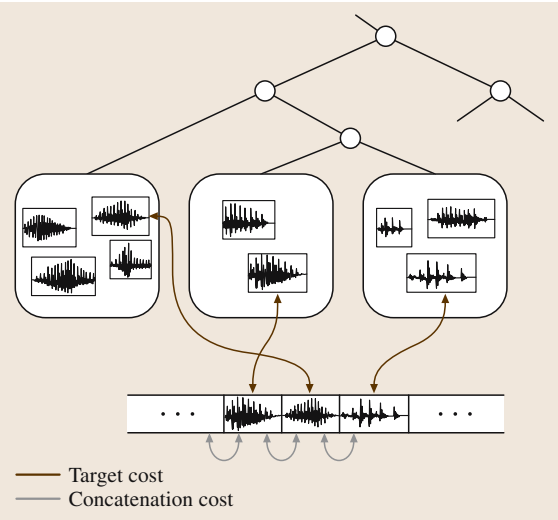



**Fig. 21.10** Context-oriented clustering as a means of selecting candidate units for a given target (after [21.34])

netic transcription of the corpus), it is found that good context coverage and consistent segmentation by HMMs typically overcomes the drawback of an imperfect automatic segmentation when compared to manual segmentation [21.33].

It should be noted, however, that the development of larger segmented speech corpora has a high cost. Hence, it is becoming increasingly difficult for companies to deliver tailored voices on demand. It is also worth noticing that large unit-selection speech corpora have recently been made available for research and assessment purposes, in the framework of the Blizzard challenge [21.32]. Similarly, the voice quality offered by today's TTS systems is somehow uniform. Given the increasing quest for *expressive* speech synthesis, unit selection typically leads to parallel recordings of several speech corpora, each with an identifiable expression; this adds to the cost of a system. Chapters 24 and 25 cover these areas in detail.

Last but not least, given the size of the speech database required to implement a speech synthesizer based on unit selection, many efforts have been reported towards *pruning* the corpus, and using efficient speech *coding* algorithms. Pruning the corpus down to a given size can be done by ordering units in the original corpus as a function of their frequency of use (estimated by running the TTS on a large input text corpus), and then retaining all units up to the required corpus size. It is also useful to spot and remove outlier units in the database [21.35]. Speech coding implies the use of time-domain coders for moderate data-compression ratios (typically 1:5) or the parameterization of speech using models, which leads to much higher compression ratios at the expense of some signal degradation and additional computational cost [21.36].

### 21.3.5 Computational Cost

Computing target costs from each target in a large speech database to each candidate unit for that target is found to be time consuming. By recycling the COC principle mentioned in Sect. 21.2.4 and extending it to contextual *and* prosodic features (pitch, duration, and amplitude), it has been possible to drastically reduce the computational load for unit selection [21.37].

The main novelty here is that, once a COC tree is built for a given phoneme, not only is the tree retained, but also its leaves, i. e., the units that have been clustered based on their feature vectors. At run time, for each target in the sentence to synthesize, the related COC is browsed, so that only the units in the appropriate leaves are considered as candidate units (Fig. 21.10). The same idea has been used in the IBM and Microsoft TTS systems, based on [21.21]: after some preliminary segmentation of a speech corpus using HMM models, COC is applied to units associated with HMM states [21.33, 38].

The computation of concatenation costs is by far the most time-consuming operation in unit selection. Precomputing all possible concatenation costs is clearly prohibitive given the large number of units in the database. However, it was found that a limited number (typically one million) of unit pair concatenation costs provides a coverage of 99% of all concatenation costs actually computed. It is thus possible to store this subset in memory, which increases the speed of unit selection by a factor of four [21.39].

## 21.4 Statistical Parametric Synthesis

One of the major problems encountered in unit selection is data sparsity (see the discussion in Sect. 21.3.4). Synthetic utterances can be perfectly natural if the target sequence of units happens to be available in a continu-

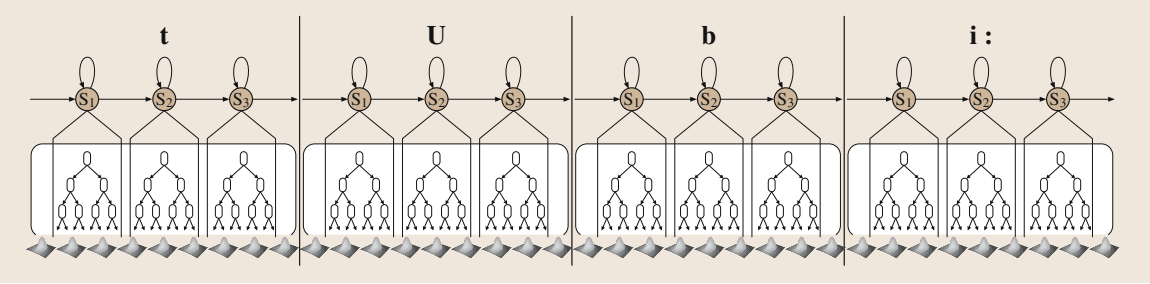



**Fig. 21.11** A statistical parametric model for the spectral dynamics of the words 'to be' (after [21.34])

ous sentence in the database, but they can also embody disturbing discontinuities (which can hamper intelligibility) when the required targets are not available. In a word, unit selection lacks the capacity to generalize to unseen data.

In contrast, ASR techniques based on HMM/GMM models have developed such a generalization property, by making use of context-oriented distribution clustering for parameter tying (Chap. 27). This has motivated several research teams to make a very promising step, by considering the speech synthesis problem as a statistical analysis/sampling problem. Actually, this idea arose as early as 1989 [21.41], although the early papers on the subject only reported poor synthetic speech quality (which only adds to the merit of the researchers who pursued in this area).

In the so-called *HMM-based speech synthesis* approach, implemented in [21.34], speech is described as the acoustic realization of a sequence of phonemes. Each phoneme is modeled by a three-state HMM. Each state emits spectral feature vectors (typically MFCCs) with emission probabilities given by multivariate Gaussians associated with the leaves of a context clustering tree (Fig. 21.11). This model is very much inspired by similar work in ASR. Additionally, the pitch and duration of phonemes are modeled by separate clustering trees and Gaussians (Fig. 21.11).

The new idea in HMM-based synthesis, whose reach is certainly not limited to speech synthesis, is to use this model to generate a stream of parameters (related to a given parametric model of speech), which can then be converted into speech (Fig. 21.12).

### 21.4.1 HMM-Based Synthesis Framework

In the following paragraphs, we will assume that the models of Fig. 21.13 have previously been trained [21.34] so as to maximize the probability of the speech corpus given the model. The training part is similar to that of an ASR system, except the contextual factors that are used when building the context clustering trees incorporate phonetic, stress, and syntactic information (which is *produced* by the TTS system, while it is not accessible to an ASR system).

From the target sequence of phonemes (augmented with stress marks and syntactic information), a sentence HMM $\lambda$ is constructed (again, as in Fig. 21.11) by concatenating context-dependent phoneme HMMs. The state durations of the phoneme HMMs are determined so as to maximize the output probability of state durations given their phonetic and syntactic context. A sequence of spectral parameters $\boldsymbol{o} = [\boldsymbol{o}_1^{\mathrm{T}}, \ldots, \boldsymbol{o}_t^{\mathrm{T}}, \ldots, \boldsymbol{o}_T^{\mathrm{T}}]^{\mathrm{T}}$ is then determined in such a way that the output probability of this sequence given the HMM model is maximized, where $T$ is the number of frames in the observation vector sequence (known from the previously estimated state durations).

More precisely, since the duration of each HMM state is known, the sequence of spectral vectors $\boldsymbol{o}_t^{\mathrm{T}}$ to be produced is associated with a given sentence HMM state sequence $\boldsymbol{q} = \{q_1, \ldots, q_t, \ldots, q_T\}$. For each such HMM state, the contextual information of the related phoneme is used to find which leaf of the COC tree (associated with this phoneme) will model the production of the state-dependent spectral parameters. Since the

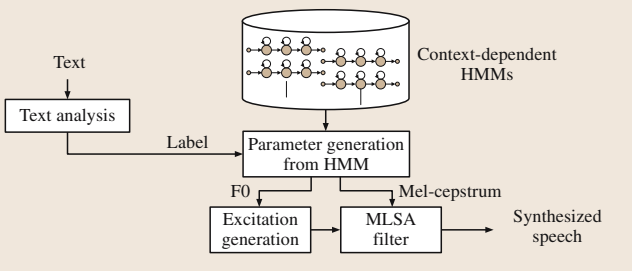



**Fig. 21.12** Speech synthesis using a statistical parametric model (after [21.40])

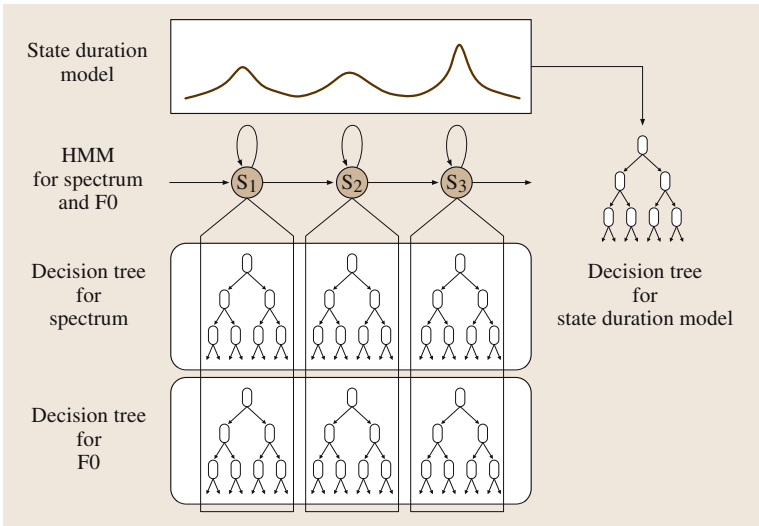



**Fig. 21.13** A statistical parametric model for the spectrum F0 and duration of a phoneme (after [21.34])



spectral features of the speech segments in each leaf are described by a single multivariate Gaussian distribution $N(\boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t})$, the $P(\boldsymbol{o}|\boldsymbol{q}, \lambda)$ is given by

$$P(\boldsymbol{o}|\boldsymbol{q}, \lambda) = \prod_{t=1}^{\mathrm{T}} N(\boldsymbol{o}_t|\boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t}) \,. \tag{21.12}$$

Stated as such, however, the sequence of spectral parameters produced by each state of the model will obviously be composed of a sequence of identical spectral parameters: the mean of the underlying multivariate Gaussian (which by nature maximizes the Gaussian distribution). The main feature of HMM-based synthesis is thus the use of *dynamic* features, through the inclusion of dynamic coefficients in the feature vector.

This too, is typical of HMM-based ASR techniques (which make intensive use of the so-called delta and delta-delta features; Chap. 27), but poses a specific problem in HMM-based speech synthesis: how to find a sequence of (static *and* dynamic) spectral features which at the same time has maximum likelihood given the HMM model *and* the dynamic constraints. This is explored in the next paragraphs, widely inspired from [21.40].

It is assumed that the speech parameter vector $\boldsymbol{o}_t$ consists of the $M$-dimensional static feature vector

$$\boldsymbol{c}_t = [c_t(1), c_t(2), \ldots, c_t(M)]^{\mathrm{T}} \tag{21.13}$$

*and* of the $1, \ldots, (D-1)$-th order dynamic feature vectors

$$\boldsymbol{o}_t = \left[\boldsymbol{c}_t^{\mathrm{T}}, \boldsymbol{\Delta}^{(1)}\boldsymbol{c}_t^{\mathrm{T}}, \ldots, \boldsymbol{\Delta}^{(D-1)}\boldsymbol{c}_t^{\mathrm{T}}\right]^{\mathrm{T}} , \tag{21.14}$$

where $\boldsymbol{\Delta}^{(d)}\boldsymbol{c}_t$ is the $d$-th dynamic feature vector given by

$$\boldsymbol{\Delta}^{(d)}\boldsymbol{c}_t = \sum_{\tau=-L_-^{(d)}}^{L_+^{(d)}} w^{(d)}(\tau)\boldsymbol{c}_{t+\tau} \tag{21.15}$$

and $w^{(d)}(\tau)$ is the window coefficient used for calculating the $d$-th dynamic feature.

Equation (21.12) can be rewritten in matrix form as

$$\boldsymbol{o} = \boldsymbol{W}\boldsymbol{c} \,, \tag{21.16}$$

where

$$\begin{aligned}
\boldsymbol{c} &= \left[\boldsymbol{c}_1^{\mathrm{T}}, \boldsymbol{c}_2^{\mathrm{T}}, \ldots, \boldsymbol{c}_T^{\mathrm{T}}\right]^{\mathrm{T}} , \\
\boldsymbol{W} &= [\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_T]^{\mathrm{T}} \otimes \boldsymbol{I}_{M\times M} , \\
\boldsymbol{W}_t &= \left[\boldsymbol{w}_t^{(0)}, \boldsymbol{w}_t^{(1)}, \ldots, \boldsymbol{w}_t^{(D-1)}\right] , \\
\boldsymbol{w}_t^{(d)} &= \left[0, \ldots, 0, w^{(d)}(-L_-^{(d)}), \ldots, w^{(d)}(L_+^{(d)}), \right. \\
&\qquad \left. 0, \ldots, 0\right]^{\mathrm{T}}
\end{aligned} \tag{21.17}$$

in which $L_-^{(0)} = L_+^{(0)} = 0$ and $w^{(0)}(0) = 1$.

Maximizing (21.12) can thus be rewritten to take the dynamic constraints into account, i. e., by maximizing $P(\boldsymbol{W}\boldsymbol{c}|\boldsymbol{q}, \lambda)$ or its logarithm:

$$\frac{\partial \log P(\boldsymbol{W}\boldsymbol{c}|q, \lambda)}{\partial \boldsymbol{c}} = 0 \,. \tag{21.18}$$

Notice that (21.12) can also be rewritten as

$$P(\boldsymbol{o}|q, \lambda) = N(\boldsymbol{o}|\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \tag{21.19}$$

Amazon EX1010 Page 477

by considering each multivariate Gaussian $N(\boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t})$ as a component of a (yet higher dimension) multivariate Gaussian $N(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$.

Equation (21.18) then produces a closed-form solution

$$\boldsymbol{R}_q \boldsymbol{c} = \boldsymbol{r}_q \,, \tag{21.20}$$

with:

$$\begin{aligned} \boldsymbol{R}_q &= \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma}_q^{-1} \boldsymbol{W} \,, \\ \boldsymbol{r}_q &= \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Sigma}_q^{-1} \boldsymbol{\mu}_q \,. \end{aligned} \tag{21.21}$$

This remarkable solution (21.20) produces the (static) vector sequence $\boldsymbol{c}$ maximizing (21.19) under the constraints (21.16).

### 21.4.2 The State of the Art and Perspectives

HMM-based speech synthesis currently produces speech with remarkable fluidity (smoothness), but rather poor voice quality. The resulting buzziness is still clearly due to the use of a source–filter model based on linear prediction.

It has, however, several important potential advantages over unit selection. First, its use of context clustering is far more flexible than that of unit selection, since it allows for the creation of separate trees for spectral parameters $F_0$ and duration. Second, its coverage of the acoustic space is better, given the generative capability of the HMM/COC/Gaussian models. Even more importantly, it embodies a complete model of natural speech with very limited footprint (1 MB). Last but not least, it provides a natural framework for voice modification and conversion.

Recently, statistical parametric synthesis has been adapted to the HMM-based synthesis of streams of articulatory parameters, instead of vocal-tract spectrum parameters [21.42]. In this case, a mapping function is required from the articulatory to acoustic domains. This attempt to unify statistical and articulatory approaches is very promising.

Another possible development of HMM-based synthesis is to use its output as target acoustic features for unit selection, as in XIMERA [21.43].



## 21.5 Conclusion

Clearly, the winning paradigm today in the industry is unit selection, while more-rudimentary diphone-based systems are still used for low-cost consumer products. Statistical parametric synthesis offers very promising results, and allows a degree of flexibility in speech control that is not possible with unit selection.

Some speech synthesizers are also available for free for research purposes. See for instance the MBROLA (http://tcts.fpms.ac.be/synthesis/mbrola.html), FESTIVAL (http://www.cstr.ed.ac.uk/projects/festival/), FestVox (http://festvox.org/) or FreeTTS (http://freetts.sourceforge.net/docs/) projects. See also TTSBOX, the Matlab tutorial toolbox for text-to-speech synthesis (http://tcts.fpms.ac.be/projects/ttsbox/).

### References


21.1 C.M. Harris: A study of the building blocks in speech, J. Acoust. Soc. Am. **25**, 962–969 (1953)

21.2 R.V. Shannon, F.G. Zeng, V. Kamath, J. Wygonski, M. Ekelid: Speech recognition with primarily temporal cues, Science **13**(5234), 270 (1995)

21.3 N.R. Dixon, H.D. Maxey: Terminal analog synthesis of continuous speech using the diphone method of segment assembly, IEEE Trans. ASSP **AU-16**(1), 40–50 (1968)

21.4 T. Dutoit: *An Introduction to Text-To-Speech Synthesis* (Kluwer Academic, Dordrecht 1997)

21.5 B. Bozkurt, T. Dutoit, R. Prudon, C. d'Alessandro, V. Pagel: Improving the quality of MBROLA synthesis for non-uniform units synthesis. In: *Text to Speech Synthesis: New Paradigms and Advances*, ed. by S. Narayanan, A. Alwan (Prentice-Hall, Upper Saddle River 2004)

21.6 E. Moulines, F. Charpentier: Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones, Speech Commun. **9**, 5–6 (1990)

21.7 Y. Stylianou: Applying the harmonic plus noise model in concatenative synthesis, IEEE Trans. Speech Audio Process. **9**(1), 21–29 (2001)

21.8 W. Verhelst, M. Roelands: An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech, Proc. ICASSP 93, Vol. II (1993) pp. 554–557


Amazon EX1010 Page 478

21.9 N. Morita, F. Itakura: Time-scale modification algorithm for speech by use of pointer interval control overlap and add (PICOLA) and its evaluation, Proc. Annu. Meeting of Acoust. Soc. Jpn., Vol. 86 (1986) pp. 9–16

21.10 R.J. Mac Aulay, T.F. Quatieri: Speech analysis/synthesis based on a sinusoidal representation, IEEE Trans. Acoust. Speech Signal Process. **34**, 744–754 (1986)

21.11 J. Marques, L. Almeida: Frequency-varying sinusoidal modeling of speech, IEEE Trans. Acoust. Speech Signal Process. **37**(5), 763–765 (1989)

21.12 M.W. Macon: *Speech Synthesis Based on Sinusoidal Modeling*, Ph.D. Dissertation (Georgia Institute of Technology, Atlanta 1996)

21.13 T. Dutoit, H. Leich: MBR-PSOLA: Text-to-speech synthesis based on an MBE resynthesis of the segments database, Speech Commun. **13**, 435–440 (1993)

21.14 A. Conkie, S. Isard: Optimal coupling of diphones, Proc. 2nd ESCA/IEEE Workshop On Speech Synthesis Mohonk, ed. by J. Olive (1994)

21.15 J. Wouters, M.W. Macon: Control of spectral dynamics in concatenative speech synthesis, IEEE Trans. Speech Audio Process. **9**(1), 30–38 (2001)

21.16 V. Aubergé: *La synthèse de la parole: des règles aux lexiques*, Ph.D. Dissertation (Institut de la Communication Parlée, Grenoble 1991), in French

21.17 D. Bigorne, O. Boeffard, B. Cherbonnel, F. Emerard, D. Larreur, J.L. Le Saint-Milon, I. Metayer, C. Sorin, S. White: Multilingual PSOLA text-to-speech system, Proc. Int. Conf. Acoust. Speech Signal Process., Vol. 2 (1993) pp. 187–190

21.18 T. Portele, W. Sendlemeier, W. Hess: HADIFIX, a system for German speech synthesis based on demisyllables, diphones, and suffixes, Proc. First ESCA Workshop on Speech Synthesis (1990) pp. 161–164

21.19 S. Nakajima: Automatic synthesis unit generation for English speech synthesis based on multi-layered context oriented clustering, Speech Commun. **14**, 313–324 (1994)

21.20 R. Donovan, P. Woodland: Improvements in an HMM-based speech synthesizer, Proc. Eurospeech 95, Vol. 1 (1995) pp. 573–576

21.21 J.P. Olive: Rule synthesis of speech from diadic units, Proc. ICASSP, Vol. 77 (1977) pp. 568–570

21.22 Y. Sagisaka, N. Kaiki, N. Iwahashi, K. Mimura: ATR $\nu$-TALK speech synthesis system, Proc. ICSLP 92, Vol. 1 (1992) pp. 483–486

21.23 A.J. Hunt, A.W. Black: Unit selection in a concatenative speech synthesis system using a large speech database, Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP '96), Vol. 1 (1996) pp. 373–376

21.24 N. Campbell, A. Black: Prosody and the selection of source units for concatenative synthesis. In: *Progress in Speech Synthesis*, ed. by J. van Santen, R. Sproat, J. Olive, J. Hirshberg (Springer, Berlin, Heidelberg 1995)

21.25 M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, A. Syrdal: The AT&T next-gen TTS system, Proc. Joint Meeting of ASA (1999)

21.26 M. Balestri, A. Paechiotti, S. Quazza, P.L. Salza, S. Sandri: Choose the best to modify the least: a new generation concatenative synthesis system, Proc. Eurospeech, Vol. 99 (1999) pp. 2291–2294

21.27 P. Taylor, A.W. Black: Speech synthesis by phonological structure matching, Proc. Eurospeech, Vol. 99 (1999) pp. 623–626

21.28 J. Vepa, S. King: Join cost for unit selection speech synthesis. In: *Speech Synthesis*, ed. by A. Alwan, S. Narayanan (Prentice-Hall, Upper Saddle River 2004)

21.29 B. Möbius: Rare events and closed domains: Two delicate concepts in speech synthesis, Int. J. Speech Technol. **6**(1), 57–71 (2003)

21.30 J.P.H. van Santen: Combinatorial issues in text-to-speech synthesis, Proc. Euro. Conf. Speech Commun. Technol., Vol. 5 (1997) pp. 2511–2514

21.31 A. Black, K. Lenzo: Limited domain synthesis, Proc. ICSLP (2000) pp. 411–414

21.32 C. Bennett, A. Black: The Blizzard Challenge 2006, Proc. Blizzard Challenge 2006 (2006)

21.33 X. Huang, A. Acero, J. Adcock, H. Hon, J. Goldsmith, J. Liu, M. Plumpe: Whistler: A trainable text-to-speech system, Proc. ICSLP, Vol. 96 (1996) pp. 659–662

21.34 K. Tokuda, H. Zen, A. Black: An HMM-based approach to multilingual speech synthesis. In: *Text to Speech Synthesis: New Paradigms and Advances*, ed. by S. Narayanan, A. Alwan (Prentice Hall, Upper Saddle River 2004) pp. 135–153

21.35 Y. Zhao, M. Chu, H. Peng, E. Chang: Custom-tailoring TTS voice font – keeping the naturalness when reducing database size, Proc. Eurospeech, Vol. 2003 (2003) pp. 2957–2960

21.36 D. Chazan, R. Hoory, Z. Kons, A. Sagi, S. Shechtman, A. Sorin: Small footprint concatenative text-to-speech synthesis system using complex spectral envelope modeling, Proc. Interspeech, Vol. 2005 (2005) pp. 2569–2572

21.37 A.W. Black, P. Taylor: Automatically clustering similar units for unit selection in speech synthesis, Proc. Eurospeech, Vol. 2 (1997)

21.38 R.E. Donovan, E.M. Eide: The IBM trainable speech synthesis system, Proc. Int. Conf. Spoken Lang. Process., Vol. 5 (1998) pp. 1703–1706

21.39 M. Beutnagel, M. Mohri, M. Riley: Rapid unit selection from a large speech corpus for concatenative speech synthesis, Proc. Eurospeech '99, Vol. 2 (1999) pp. 607–610

21.40 H. Zen, K. Tokuda, T. Kitamura: An introduction of trajectory model into hmm-based speech synthesis, Proc. Speech Synthesis Workshop (2005)




Amazon EX1010 Page 479

21.41 A. Falaschi, M. Giustiniani, M. Verola: A hidden Markov model approach to speech synthesis, Proc. Eurospeech, Vol. 1989 (1989) pp. 2187–2190

21.42 K. Nakamura, T. Toda, Y. Nankaku, K. Tokuda: On the use of phonetic information for mapping from articulatory movements to vocal tract spectrum, Proc. ICASSP, Vol. 06 (2006) pp. 93–96

21.43 H. Kawai, T. Toda, J. Ni, M. Tsuzaki, K. Tokuda: XIMERA: A new TTS from ATR based on corpus-based technologies, Proc. 5th ISCA Speech Synthesis Workshop (2004) pp. 179–184



Amazon EX1010 Page 480

# 22. Linguistic Processing for Speech Synthesis

R. Sproat

An important part of any text-to-speech synthesis system is the linguistic processing component that takes input text and converts it into a feature representation from which actual synthesis can proceed. Linguistic analysis is hard, in a large measure because written language massively underspecifies linguistic information. This chapter reviews several issues in linguistic analysis starting from low-level text normalization issues, and ending with higher-level problems such as accent prediction and document-level analysis. We end with some prognosis of the future prospects for improvements over current technology.

22.1 **Why Linguistic Processing is Hard** ..... 457
22.2 **Fundamentals: Writing Systems and the Graphical Representation of Language** ... 457
22.3 **Problems to be Solved and Methods to Solve Them** ..... 458
22.3.1 Text Preprocessing ..... 458
22.3.2 Morphological Analysis and Word Pronunciation ..... 461
22.3.3 Syntactic Analysis, Accenting, and Phrasing ..... 462
22.3.4 Sense Disambiguation: Dealing with Ambiguity in Written Language ..... 464
22.4 **Architectures for Multilingual Linguistic Processing** ..... 465
22.5 **Document-Level Processing** ..... 465
22.6 **Future Prospects** ..... 466
**References** ..... 467



## 22.1 Why Linguistic Processing is Hard

If one compares the problems faced by a text-to-speech (TTS) system to those faced by an automatic speech recognizer (ASR) system, one's first impression may well be that, compared with the situation with ASR, the language analysis portion of TTS has it rather easy. In an ASR system one has to reconstruct what was said from rather noisy input, so the language modeling problem can be formidable. For TTS, in contrast, one already knows what the words are since they, along with other linguistic information, are represented in the text. This view is a misconception.

While written language represents speech insofar as a literate speaker of a language can generally read a text aloud, writing is still a highly imperfect and incomplete representation. Writing is not transcription. Rather, writing presumes that the reader knows the language in question, and can fill in the details when the information provided by the written form is incomplete or misleading.

There are many such instances of incomplete or misleading information, ranging from abbreviations for which one must figure out the full word form, through homographs where one must choose among a set of differently pronounced words that happen to be spelled the same way, to the fact that no writing system has any systematic representation of phrasal prosody. In what follows we shall discuss each of these issues in turn, show why they are hard, and present some approaches that have been taken to solving them. We start with a brief reminder of how writing encodes linguistic information.

## 22.2 Fundamentals: Writing Systems and the Graphical Representation of Language

The basic input to a text-to-speech system is text, typically text that is encoded in the standard orthography of the language to be synthesized. Thus, the starting point of any discussion of linguistic processing in text-to-

speech synthesis should be an overview of how language is encoded in writing – or in other words how writing systems work. In this section we give a brief synopsis of writing systems, focusing on the issues that are of importance to the TTS problem.

As far as we know, writing was developed independently by exactly four civilizations: by the Egyptians around 3000 BC; by the Sumerians at around the same time; by the Chinese around 1500 BC; and by the Mayans around 1000 AD. Of these, only Chinese writing survives in an unbroken tradition into modern times in that a highly modified version of the original Shang dynasty script is used to encode modern descendants of the Chinese language spoken 3500 years ago. All writing systems encode pronunciation, but they do so to different degrees, and a number of writing systems also encode other, mostly semantic information. All ancient scripts – Mayan, Egyptian, Sumerian, and Chinese – encoded both sound and meaning; in modern Chinese, meaning is loosely encoded for most characters by means of *semantic radicals*, and sound is very roughly encoded in *phonetic components*. Among writing systems that putatively only encode sound, such as those based on the Greek-derived alphabets, there is a wide difference in the transparency of the encoding; for languages that use the Latin alphabet, Finnish is among the most straightforward in terms of letter–sound correspondence, and English is almost certainly the least straightforward.

However, the important point to bear in mind is that, while all writing systems encode sound and possibly other linguistic information, they are under no obligation to do so in a way that makes the recovery of that information easy. Writing is designed for people who know the language and can fill in the details. While there are certainly psycholinguistic consequences of different designs (there is a vast literature on this topic), as a practical matter it is relatively unimportant that Japanese *kanji* (Chinese characters) routinely have upwards of five distinct pronunciations; or that Arabic and Hebrew fail to represent most short vowels; or that distinctive stress placement information is not marked in Russian orthography. Each of these bits of missing information can generally be provided by someone who knows the language. Of course for a TTS system this is a challenge since it requires the system to model the language that native speakers all know.

TTS systems typically take their input in the form of electronic text, which of course obviates the need to deal with optical character recognition. But despite much progress on standardization by the unicode consortium, there are still areas where coding variants are allowed, leading to complications. For example, many Indian scripts have complex vowel symbols consisting of two or more separate components. The unicode standard permits more than one way of specifying these components, so that a system that deals with text in these languages must know about the legal variants. Furthermore, there are other variants which, though *illegal* are nevertheless found; Buckwalter [22.1] gives a nice overview of the complexities of Arabic text processing that are caused by the many variant encodings.



## 22.3 Problems to be Solved and Methods to Solve Them

### 22.3.1 Text Preprocessing

In Sect. 22.2 we introduced some of the ways in which writing encodes language but we tacitly assumed that we were dealing with ordinary words of the language spelled in the ordinary way.

But consider the following perfectly ordinary-looking paragraph taken from a story in the *New York Times*, April 11, 2006:

> *What made the* ***$12.08*** *transaction remarkable was that the customer was not just outside* ***Ms.*** *Vargas's workplace here on California's central coast. She was at a McDonald's in Honolulu. And within a two-minute span* ***Ms.*** *Vargas had also taken orders from drive-through windows in Gulfport,* ***Miss.****, and Gillette,* ***Wyo.***

In addition to the ordinary words and names such as one might expect to find in a dictionary (see Sect. 22.3.2), there are five items, marked in boldface, which are not ordinary words. The first is a numerical expression, more specifically a currency expression, which in this instance is to be read as *twelve dollars and eight cent*. The next four are abbreviations, namely two instances of *Ms.* for *Miz* (though note that this word has no conventional full-form spelling), one of 'Miss.' for *Mississippi* and one of *Wyo.* for *Wyoming*. Note that in this latter instance the period that marks the expression as an abbreviation also serves the double duty as the end-of-sentence marker.

A *text preprocessor* is the component of a TTS system that deals with analysis of *nonstandard words* such as the examples above, along with other low-level text-processing issues such as sentence segmentation and, in some languages, word segmentation. In this section we will review these issues, starting with the problem of end-of-sentence detection and word segmentation, broadly construable under the rubric of *tokenization*.

### Tokenization

The first thing one needs to do in any text-preprocessing system is to divide the text into smaller units or *tokens* for further processing. In principle a token can correspond to any unit of linguistic structure, such as a word, a sentence or a paragraph. Most discussions of tokenization presume that the tokens are words, but this choice is somewhat arbitrary. We will therefore assume the broader definition and discuss the problem of dividing a text into sentences, and thence into words. These levels of tokenization will typically be further subdivided – words into morphemes, or sentences into phrases – and these issues will be discussed later in the chapter.

For sentence tokenization an ideal situation would be if there were a delimiter or set of delimiters that are used to delimit sentences and nothing else. A situation close to that obtains in Chinese, where declarative sentences are delimited by a small circle '。' that is used only for that purpose. In many languages, including English, the situation is less straightforward since the most common sentence delimiter, the period, has other functions including, most problematically, marking abbreviations; it can even serve both functions at once, as in the example of *Wyo.* above. Proper sentence tokenization is therefore intimately tied up with other aspects of preprocessing, such as abbreviation expansion. TTS preprocessors will therefore typically invoke abbreviation expansion before or concomitantly with sentence tokenization.

Tokenization into words is either relatively easy or relatively hard, depending upon which language one is dealing with. In English and many other languages, words are typically delimited by spaces or punctuation symbols and, while things are not always completely trivial, these orthographic conventions make the problem of word segmentation in such languages fairly easy. At the other end of the scale are languages like Chinese, Japanese, and Thai, where words are never delimited by spaces, and tokenization is therefore in principle fairly hard; in such cases punctuation usually *does* delimit words, but punctuation is of course relatively sparse and therefore does not provide much of a guide.

It is worth noting that the use or nonuse of spaces is mostly not *language* dependent, but *script* dependent. Languages that use alphabets such as Greek, Latin, Cyrillic will always mark word boundaries. This is also true of most scripts used for Indian languages. On the other hand, languages that use the Chinese writing system or derivatives including Chinese, Japanese, and traditionally Korean and Vietnamese do not use spaces. Only in the 20th century, with the wider use of Hangul did Korean introduce spacing to separate words (more accurately, accentual phrases or *eojeol*). On the other hand, even after abandoning the *chữ'-nôm* Chinese-derived writing system in favor of romanization in the 19th century, Vietnamese continues to this day not to mark word boundaries with spaces: spaces are used in Vietnamese not to mark word boundaries, but syllable boundaries [22.2]. A similar point can be made about the use of *capitalization*. Again, capitalization is not language dependent but script dependent. The Greek, Latin, Cyrillic, and Armenian scripts support capitalization, and languages using those scripts use capitalization to mark certain classes of words, typically names, words in sentence-initial position, words in titles, and so forth. Other scripts do not support capitalization.

Word segmentation has been studied extensively in Chinese and Japanese, less extensively in Thai, and even less in Vietnamese. Although there are important linguistic differences between these languages that make details of the problem different, for reasons of space we will restrict discussion here to Chinese.

If one can assume that one has a dictionary that lists all the words that one will find in a given text, the problem of segmentation is relatively straightforward. In *most* cases, simple algorithms like *maximum matching* work reasonably well: starting with a pointer at the left (or right) edge of the sentence, find the longest match in the dictionary, set the pointer after (before) that match, and iterate this process until you reach the end (beginning) of the sentence. However, such simple algorithms run afoul of cases where there is ambiguity. To handle such cases, people have typically used statistical language-modeling techniques, taking into account the probabilities of the words involved, and the transition probabilities between the words; class-based language modeling techniques, such as part-of-speech tag language models have also been used. It is important to realize though that, even with these techniques there are still cases that cannot be disambiguated. Such cases typically depend upon discourse or even real-world knowledge. *Gan* [22.3] discusses such cases. For example the sentence 马路上生病了 *mǎ lù shàng shēng bìng*



Amazon EX1010 Page 483

*le* can either mean "the horse became sick on the road" if the segmentation is *[mǎ] [lù] [shàng] [shēng bìng] le*, or "(someone) became sick on the road" if the segmentation is *[mǎ lù] [shàng] [shēng bìng] le*. Since both of these are valid interpretations and there is no way without further information to tell which one is more appropriate, it follows that ordinary language-modeling techniques cannot handle such cases.

One might wonder why, for Chinese TTS applications, one even cares about word segmentation. The reasons are that knowing what word one is dealing with and knowing where word boundaries fall are just as important in Chinese as they are in English. Many characters in Chinese (and orders of magnitude more in Japanese) have multiple pronunciations and in many cases the pronunciation depends upon what word the character occurs in. For example, in Mandarin the character 行 can be pronounced either as *xíng* as in 行人 *xíngrén* 'passer-by', or as *háng* as in 银行 *yínháng* 'bank'. Character pronunciations are also dependent upon word class. In traditional Chinese characters, as used in Taiwan, the character 乾 is pronounced as *gān* if it means *dry*, but *qián* if it means *heaven*. The former reading would be appropriate for the nonce compound 羊肉乾 *yángròu gān* 'mutton jerky', whereas the former would be appropriate in a personal name such as 林魁乾 *lín kuìqián*. This implies that for unknown words one needs in general to know more than just the sequence of characters is a word, but also what kind of word it is.

Another obvious motivation for word segmentation is prosodic phrasing (Sect. 22.3.3): it is usually desirable to avoid putting a prosodic break in the middle of a word.

### Abbreviation Expansion

Abbreviation expansion might seem to be a rather easy problem. Faced with an abbreviation such as *kg.*, an English speaker will automatically think of *kilogram*. It is just a matter of table lookup. A moment's reflection, however, will reveal that things are not quite this simple. There are several issues that make the problem complicated.

- First, one must decide that one has an abbreviation as opposed to an ordinary word. Abbreviations are not always marked with a period, and even when they are, the period may be doing double duty as an end-of-sentence marker as in the example above. Deciding that *No* should be read as *North* in *No Arlington*, requires contextual information. Even if you are sure you have an abbreviation, there is the issue of how to read it. In [22.4] we carefully distinguished between three classes of abbreviation:
  - EXPN: terms that are to be *expanded* into full words, like *kg*
  - LSEQ: terms that are read as letter sequences, like *CIA*
  - ASWD: terms that are read as if they were an ordinary word, like *NATO*
- Any given abbreviation may be ambiguous, a standard example in English being *Dr.*, which can be *Doctor* or *Drive*.
- Even if you are sure which word is intended, you often need to predict the form of the word. Thus 1 kg is probably *one* ***kilogram***, but 2 kg is probably *two* ***kilograms***, whereas 2 kg *watermelon* is again *two* ***kilogram***. In a highly inflected language like Russian the situation is even worse. Thus the Cyrillic equivalent of *kg* can be expanded as *kilogram*, *kilograma*, *kilogramy*, *kilogramov*, *kilogramami* . . . , depending upon the context.
- People frequently make up new abbreviations, so you cannot count on being able to find all abbreviations in the dictionary. Novel abbreviations are especially rich in domains where there is a strong incentive to abbreviate, as in printed classified advertisements (where you are paying by the inch) or in the notes taken by customer service representatives (who have very little time to write down a summary of their conversations with customers). For example, in a database of AT&T customer care call summaries, we found no fewer than 18 distinct ways of abbreviating the word *customer*, only a handful of which could reasonably be considered standard.

Most systems use some version of table lookup. Sense disambiguation techniques such as the ones we discuss in Sect. 22.3.4 can be used for disambiguation.

Abbreviation expansion is an interesting and difficult problem and it is unfortunate that it has not received more systematic attention in the literature. The only such systematic study that we are aware of is work that we did during the 1999 Johns Hopkins Workshop on Speech and Language processing, and reported in [22.4]. In that paper, we used language modeling coupled with a channel model that predicted possible abbreviation expansions. An explicit goal was to deal with nonce abbreviations, such as the example of *customer* cited above. The most interesting examples in that work came from the domain of classified advertisements, where heavy abbreviation is typical, but there were sufficient examples of the full forms of words. For example, in addition to *brk*



Amazon EX1010 Page 484

*apts* one also found *brick apartments*; with a reasonable model of possible abbreviations, one could infer from the existence of *brick apartments* that *brk apts* might be a reasonable abbreviation of that, and so propose *brick apartments* as a possible expansion. An example of automatic abbreviation from a different domain, namely AT&T customer service transcriptions is shown below:

```
xplnd chrgs .. cust stated he w/
pay 26.45 & then w/ cancel his
srvc w/ att

explained charges .. customer
stated he will pay 26.45 & then
will cancel his service with att
```

Here full words such as *explained* and *charges* occur elsewhere in the corpus and are reasonable expansions for the abbreviated words *xplnd* and *chrgs*. Note that *w/* is used here both in its normal usage of *with*, as well as *will*, and that the expander gets these both correct.

#### Number Expression Expansion

Unlike other aspects of preprocessing, number expression expansion is fairly algorithmic – once you know which of the particular *uses* of the number expression is intended. To illustrate this point consider the following examples:

1. He saw **123** goats.
2. I live at **123** King Avenue.
3. I have never used Lotus **123**.

In case 1 the expression is being used to represent a normal number name *one hundred (and) twenty three*. In case 2, it represents a numerical *label*; in English such labels would typically be parsed into small chunks, so that a sequence *1234* would often be read as *twelve twenty-four*; in this case the sequence would be read as *one twenty-three*. In case 3, another kind of label, read as a sequence of digits – *one two three* – is intended. Deciding which of these is appropriate is a *sense disambiguation* problem, and techniques such as those discussed in Sect. 22.3.4 work reasonably well.

However, once one knows which type of number expression is intended, expansion is straightforward. For labels such as cases 2 and 3, one needs to parse the sequence into small chunks that are read as number names (as in case 1). The typical method for expanding number names in TTS systems is to write special-purpose rules, or to hard-code the expansion in a programming language such as C++. However, there is a theoretically more satisfactory method, namely to view the problem as a problem of *regular transduction*, implementable using weighted finite-state transducers (WFSTs); see Sect. 22.4 and [22.5]. Briefly, one can factor the problem into two components: the transduction of a digit sequence into a sum-of-products of powers of ten; and another transduction from such a factorization and the number name itself. The relatively few languages with productive vigesimal systems can also be handled straightforwardly by such methods.

What we have just described works reasonably well for languages such as English or Chinese where numbers are largely invariant in form. Languages such as Spanish where numbers inflect for grammatical categories such as gender add an additional layer of complexity: the WFST-based approach still works, but one must perform additional contextual analysis. A particularly complex case can be found in Russian, where number names inflect for gender and case. The form of nouns modified by number names is affected by both the case assigned to the noun phrase containing the number-name/noun combination, and by the value of the number itself; for details see [22.5].

### 22.3.2 Morphological Analysis and Word Pronunciation

Morphological analyzers have long been used in TTS systems as an aid to word pronunciation. One of the earliest morphological analyzers was DECOMP, part of the MITalk English TTS system [22.6]. DECOMP consisted of a finite-state model of morpheme combination, augmented with spelling change rules. DECOMP relied heavily on generative phonology in its implementation of morphology-dependent phonological alternations.

Word pronunciation frequently depends upon understanding the morphological composition of a word. If one blindly applies a general letter-to-sound transduction to the word *hothead*, one will likely pronounce the <th> sequence as /θ/; clearly one needs to know that the word is actually a compound out of two words, and that <th> should not be interpreted as a digraph.

In Russian, vowel pronunciation is heavily dependent upon stress placement in the word, and stress placement is in turn dependent upon morphological information. In the nominal declension system, for instance, placement of stress for a given form depends upon the particular declension class to which the noun in question belongs. Such information is not generally predictable from the phonological shape of the noun and seems to be an arbitrary property of the noun in question.

Morphology is not only relevant to the pronunciation of ordinary words, but also proper names; many per-

sonal names, for instance, have morphological structure that can be exploited in predicting pronunciation [22.7]. Thus, for instance, the pronunciation of a personal name beginning in *Mc* can often be predicted by separating off the *Mc*, looking up or otherwise predicting the pronunciation of the remainder, and reattaching the pronunciation of *Mc*. Morphological analysis of sorts is also needed for the correct pronunciation of many brand names, particularly in cases where capitalization or punctuation gives no hints as to decomposition. Recognizing that *espeech* in a URL is actually *e+speech* allows one to avoid a pronunciation such as /ɛspič/. Part-of-speech tagging (see Sect. 22.3.3) is another motivation for morphological analysis. Knowing that a form like *splurged* is probably a past tense or past participial verb form – and less likely to be a simplex noun – is useful information if one wants to assign part-of-speech information to a sentence containing this word.

Computational morphologists often distinguish between *lemmatization* (or *stemming*) and full-blown morphological analysis. A lemmatizer typically returns a pair consisting of a base (dictionary) form and a piece of morphosyntactic information: for example, *sat* might return (*sit*, PAST). Full-blown morphological analysis typically goes further and decomposes the word into smaller chunks, termed *morphs* or *morphemes*. A word such as *derivations* might be analyzed as *derive+ation+s*, with possible further annotations indicating the grammatical information about the pieces.

Most methods for full-blown decomposition have involved *finite-state transducers*, the most famous example of such a system being *Koskenniemi*'s [22.8] system for Finnish. Traditionally such systems have been handbuilt, a task that can obviously be very laborious for a language with a complex morphology. Recently, however, there has been a small amount of progress towards automating such acquisition in the form of a number of systems that can induce morphological alternations from unannotated corpora [22.9–11, inter alia]. However, such systems are still a long way from replacing linguistic experts in the construction of high-coverage morphological analyzers.

Lemmatizers have tended to use a variety of different methods, though all of these methods could, not surprisingly, be implemented using finite-state transducers. Some of the most interesting approaches to lemmatization include self-organizing methods such as those of *Yarowsky* and *Wicentowski* [22.12]. Part of this system is a decision tree that learns string suffixes of a word that are strongly associated with a particular suffix and stem change.

### 22.3.3 Syntactic Analysis, Accenting, and Phrasing

We have seen in various places in the preceding sections that written language is an imperfect representation of spoken language at the word level. Once one gets beyond the word level, this becomes even more true, especially when one considers the problems of *accenting* and *phrasing*. We discuss each of these topics in turn.

#### Accenting

In any language, different parts of a sentence are associated with different levels of *prominence*. So, for example, for the classic sentence *She had your dark suit in greasy wash water all year*, a fairly neutral reading of the sentence would have relatively high prominence on *had*, *dark*, *suit*, *greasy*, *wash* and *year*, and less prominence on the others. Such prominences are termed *accents*, and determining correct accent placement is a major issue in producing plausible-sounding synthetic speech. Accents themselves are typically implemented with some sort of *excursion* in fundamental frequency ($F_0$), possibly combined with an increase in amplitude and other phonetic effects.

The linguistic literature on accent often divides languages into three different basic classes; see [22.13–15, inter alia]. *Stress languages*, of which English is an example, have prominences associated with words, but how these are expressed in terms of pitch excursions varies with the context. In *pitch accent languages*, of which Japanese is an example, words are lexically marked for both accent placement and the $F_0$ excursion shape. Finally, in *tone languages* most or all syllables are lexically marked for $F_0$ information. Nevertheless, *all* languages will assign different prominence to words under different conditions, though the details of how this works out vary a lot from language to language [22.14, 15].

In the work on accent assignment in English, people have often made a three-way distinction between words that are fully accented, words that are deaccented, and words that are *cliticized*. Typically, deaccented words are not associated with an $F_0$ excursion. Cliticized words, in addition, tend to be prosodically very weak, with the consequence that they are quite short in duration, and behave as if they were really a part of an adjacent word.

A simple model of accentuation would make the decision on accenting on the basis of part-of-speech information. Typically, nouns, verbs adjectives, and other *open-class* words tend to be accented. Func-



Amazon EX1010 Page 486

tion words – determiners, pronouns, articles, etc. – tend to be deaccented or, if they are short, cliticized. A minimal requirement for accent assignment, then, is a part-of-speech tagger, or at least a word classifier that can distinguish between open- and closed-class words. Part-of-speech (POS) tagging is a well-studied problem [22.16–19, inter alia], and modern POS taggers perform quite well, with error rates in the range of perhaps 3% for newswire-type text, and any number of methods can be used in the context of TTS.

Of course, POS-based prediction can only get you so far: the purpose of accenting, at least to a large extent, is not to communicate broad lexical categories, but rather to communicate the importance of information. Indeed, it is a consequence of that fact that major lexical categories such as nouns tend to be accented, and minor categories such as prepositions tend to be deaccented: nouns typically carry more information than and are therefore more important than prepositions. So, much of the work on accenting has focused on ways to predict the information content. Early work such as [22.20] considered various categories of terms, including cue phrases (*And then* ...), and items that are *new* in the discourse, which tend to be given special prominence – versus those that are *given* and tend to be given less prominence. More recently, corpus-derived information-theoretic measures have been used to predict accent [22.21].

In English, a particularly interesting and complex set of cases involve complex noun phrases, i. e., a noun preceded by one or more adjectival or nominal modifiers. In a discourse-neutral context, some instances are accented on the final word (*Madison* ***Avenue***), some on the penultimate (***Wall*** *Street*, *kitchen* ***towel*** *rack*), and some on an even earlier word (***sump*** *pump factory*). For a binary nominal the primary determinants of accenting are largely semantic, although there is a fair amount of purely lexical specification [22.22–24]. So, right-hand accent is often found in cases where the left-hand element denotes a substance out of which the second element is made (cf. *apple* ***pie***, *steel* ***bar***), but there are numerous often quite systematic (***banana*** *bread*) exceptions. Computational models, e.g., [22.25, 26], have been somewhat successful at modeling the semantic and lexical generalizations; for example [22.26] uses a combination of hand-built lexical and semantic rules, as well as a statistical model based on a corpus of nominals hand-tagged with accenting information. Accenting on nominals longer than two words is generally predictable given that one can compute the nominal's structure (itself a nontrivial problem), and given that one knows the accentuation pattern of the binary nominals embedded in the larger construction [22.23, 26, 27].

### Phrasing

Speakers normally break up long sentences into a sequence of several prosodic phrases. There are two basic varieties of prosodic phrase commonly recognized in the literature: an utterance is typically assumed to consist of one or more *intonational phrases*, with each of these in turn consisting of one or more *intermediate phrases*. A linguistic analyzer for a TTS system must compute appropriate places to insert phrase boundaries.

If punctuation is used liberally so that there are relatively few words between the commas, semicolons or periods, then a reasonable guess at an appropriate phrasing would be simply to break the sentence at the punctuation marks, although this is not always appropriate. The real problem comes when long stretches occur without punctuation; in such cases, human readers would normally break the string of words into phrases, and the problem then arises of where to place these breaks. The simplest approach is to have a list of words, typically function words, that are likely indicators of good places to break [22.28, 29]. One has to use some caution however, since while a particular function word like *and* may coincide with a plausible phrase break in some cases, in other cases it might coincide with a particularly poor place to break, e.g., *I was forced to sit through a dog* ***and*** *pony show that lasted most of Wednesday afternoon.*

Most prosodic phrasing prediction systems attempt to do some amount of syntactic analysis of the input, ranging from full-blown parsing, to partial parsing to simply computing phrasing on the basis of part-of-speech tag sequences. An early system that used the output of a syntactic parser in phrasing prediction was the system reported in [22.30]. *Bachenko* and *Fitzpatrick* employed a wide-coverage deterministic syntactic parser (FIDDITCH, [22.31]) to construct a syntactic analysis for a sentence; the syntactic phrases were then transduced into prosodic phrases using a set of heuristics, including heuristics based on observations, originally due to *Gee* and *Grosjean* [22.32], that phrases tend to be somewhat balanced in length.

However, most systems have tended to eschew full-blown parsing, partly because of the computational overhead of parsing, and partly because the benefits of having a full parse have not been adequately



Amazon EX1010 Page 487

demonstrated. [22.33] reported on a corpus-based statistical approach that used classification and regression trees [22.34] to train a decision tree on transcribed speech data. In training, the dependent variable was the human prosodic phrase-boundary decision, and the independent variables were generally properties that were computable automatically from the text, including part of speech sequence around the boundary, shallow-parsing information such as the location of the edges of long noun phrases, the distance of the boundary from the edges of the sentence, and so forth.

More recently *Taylor* and *Black* [22.35] used generative Markov models to predict the most likely sequence of phrase breaks given a part of speech sequences. The Markov models also modeled phrase balancing heuristics of the kinds used in [22.30]. They experimented with various Markov model topologies, tag sets and smoothing methods, and concluded that the best setup allows for a 79% correct identification of breaks in a held out corpus, which is competitive with other approaches in the literature.



### 22.3.4 Sense Disambiguation: Dealing with Ambiguity in Written Language

Automatic sense disambiguation has become an important area of research in natural language processing, with a number of *bake-offs* (SENSEVAL and SEMEVAL), and the development of a number of sense-tagged corpora, and resources such as WordNets in many languages [22.36, 37].

Sense disambiguation has applications to several problems in TTS, but the most obvious application is to *homograph* disambiguation. For example, it is generally useful to know whether the word *mole* should be pronounced /mol/ (as in the mammal, chemistry or spy sense) or as /mole/ (as in the Mexican sauce sense); similarly, one wants to know if *IV* should be read as *four* (or *fourth*), or as *I. V.* (as in *IV drip*). In Japanese, one wants to know if the character *mountain* should be read as *san* or *yama*. In Russian one wants to know if *goroda* should be read with stress on the first syllable (*of a city*) or on the last (*cities*).

Sense disambiguation is essentially a language modeling problem. The writer of a text presumably knows which sense of each word is meant, so what they are trying to communicate can be thought of as coming with sense tags. What they actually produce, however, is written in an orthographic representation where distinct senses of words are conflated into a single orthographic form. It is the task of the reader, or the TTS algorithm, to reconstruct the underlying sense. As with language modeling more generally, a number of approaches have been taken, but particularly popular approaches involve *discriminative* methods that aim at finding contextual features that are particularly relevant to discriminating the two senses. The first such approach to be applied in TTS was due to *Yarowsky* [22.38, 39], who applied a more-general sense-disambiguation method based on *decision lists*. Features of the context, such as words within a particular window of the target word, words in a particular position relative to the target word, the part-of-speech sequence surrounding the target word, are examined individually. Features are ranked by the *log likelihood ratio* of the two senses, with the highest log likelihood ratios being assigned to those features that are the best discriminators. The list of features rank-ordered by log likelihood ratio is then used as a decision list, where the decision for a particular target word is decided on the basis of the first matching feature. A fragment of a decision list for *bass*, from [22.38] is given in Table 22.1. Since Yarowsky's work there has been a lot of research on sense disambiguation including a special issue of the journal *Computational Linguistics* [22.36], and several workshops, such as [22.37], and a number of different approaches have been explored. While most of this work has not specifically addressed the question of homograph disambiguation for TTS, any of these techniques could be so applied.

**Table 22.1** Decision list for *bass*, (after Yarowsky)

| Decision list for *bass* (*highly abbreviated*) | | |
|---|---|---|
| **log *L*** | **Evidence** | **Class** |
| 10.98 | *fish* in ±*k* words | ⇒ bæs |
| 10.92 | *striped* bass | ⇒ bæs |
| 9.70 | *guitar* in ±*k* words | ⇒ beIs |
| 9.20 | bass *player* | ⇒ beIs |
| 9.10 | *piano* in ±*k* words | ⇒ beIs |
| 9.01 | *tenor* in ±*k* words | ⇒ beIs |
| 8.87 | *sea* bass | ⇒ bæs |
| 8.49 | *play/V* + bass | ⇒ beIs |
| 8.31 | *river* in ±*k* words | ⇒ bæs |
| 8.28 | *violin* in ±*k* words | ⇒ beIs |
| 8.21 | *salmon* in ±*k* words | ⇒ bæs |
| 7.71 | *on* bass | ⇒ beIs |
| 5.32 | bass *are* | ⇒ bæs |

## 22.4 Architectures for Multilingual Linguistic Processing

There are naturally many ways in which one could implement any of the methods that we have been discussing, ranging from implementing special-purpose toolkits, to hard-coding many of the details of the algorithms in a programming language such as C++ or, in the open-source Festival system scheme [22.40].

There is some advantage to having a single computational mechanism for handling most or all aspects of text processing in all languages. One approach that has gained in popularity over the past decade are methods based on weighted finite-state transducers (WFSTs) [22.5, 41, 42]. Work on multilingual TTS was pioneered in the Bell Labs, but the approach was also adopted by the commercial system under development at Rhetorical, as well as in research systems such as the DeKo system at Stuttgart (http://www.ims.uni-stuttgart.de/projekte/DeKo/). The appeal of WFSTs is that they offer a common framework for handling a wide variety of phenomena that can be viewed as string-to-string transduction problems. Through the regular operations of union, concatenation, Kleene closure, and composition, complex systems can easily be constructed out of simple pieces. The algebraic and algorithmic properties of WFSTs are well understood [22.43–45]. Furthermore, many toolkits exist for compiling WFSTs out of grammars such as might be written by a linguistic expert – such as the Xerox/XRCE Toolkit [22.46, 47], van Noord's toolkit, or Lextools [22.5, 48]. WFSTs can also be constructed via a variety of statistical language-modeling methods. Since the regular operations listed above apply in the same way independently of how the WFSTs were constructed, one can have a single text-analysis module that covers all languages: one merely needs to plug in a new set of WFSTs constructed for the language of interest.

There are some *disadvantages* to using WFSTs as opposed to, for example, a system that is carefully hard-coded in a language such as C++. WFST models tend to be larger and may also be slower than such hard-coded systems, and this has led some system developers to eschew the use of general-purpose solutions. But given the move to ever-larger unit-selection databases for the production of the speech output, and the huge increases in computational resources that have become available, this view seems shortsighted. Indeed, WFSTs have been used in a number of commercial systems, including the systems once sold by Lucent Speech Solutions and Rhetorical Systems. The fact that these systems are no longer sold has nothing to do with technical reasons, and everything to do with the vagaries, and in the former case incompetence, of the business world.



## 22.5 Document-Level Processing

We have focused in this article on linguistic processing, but properties of the document are also important in conveying information effectively using speech. Different regions of a document often need to be rendered in different ways, and computing these regions is thus important for appropriate rendering.

For example, suppose one is reading an email message. In plain-text portions of the message one can ignore line feeds since they convey no information. But this is not the case in other portions of the message, for example in a signature block as in Fig. 22.1. Indeed, not only are line feeds significant – e.g., one would normally want some sort of break between the name *Charles Davies* and the following line – layout is also important. It would be wrong, for example, to read the line "Bell Laboratories, Lucent Technologies | tel (908) 582-1234" as one unit, since the portion to the right of the '|' is actually part of its own block, which includes the telephone, fax, and email address. As with text preprocessing, there has been relatively little sys-

```
Charles Davies
Dialogue Modeling Research Department
Multimedia Communications Research Laboratory
Bell Laboratories, Lucent Technologies  |tel (908) 582-1234
600 Mountain Avenue, Room 2d-500        |fax (908) 582-4321
Murray Hill, NJ 07974, USA              |ced@bell-labs.com
http://www.bell-labs.com/noname/mcs/
```

**Fig. 22.1** A sample signature block

tematic research on document-level processing, but one strand of research was work done at Bell Labs and reported in [22.49, 50]. This work combined statistical approaches to detecting different types of text regions (plain text, tables, signature blocks, etc.) with two-dimensional layout analysis for the complex structures in signature blocks.

A special note should be made here on the influential work of *T. V. Raman* [22.51] on the audio rendering of different classes of text, in particular mathematical expressions. Raman's work is particularly focused on auditory rendering for blind users, but of course has much broader applicability to any situation where the user cannot easily work with a visual display.

## 22.6 Future Prospects

Linguistic analysis is a critical component of a text-to-speech system. While much of the attention in TTS in recent years has been focused on synthesis techniques that improve voice quality, voice quality is not enough on its own for a system to be both natural sounding and pleasing to listen to. Supposing that one has a completely natural-sounding synthesizer in terms of voice quality, but that the system routinely expands abbreviations incorrectly, frequently chooses the wrong sense of a homograph, mispronounces words, and places inappropriate emphasis and phrase breaks; the best one could say about such a system is that it sounds like someone with a very nice voice, but whose brain is disconnected with reality.

Fortunately this hypothetical situation does not occur in general: over the past few decades, substantial improvements have been made in all aspects of linguistic analysis for TTS, and some of this progress has been reviewed here. However, even though TTS systems today can deal in a sensible way with a wide variety of both unstructured and structured text, there is still much that such systems cannot handle. Some examples of problems that as yet have no general solution are as follows:

- *Discourse-level reasoning*: For an example like *¿Cuantas tortas quiere Ud.? 200.* ('How many cakes do you want? 200.'), Spanish speakers would have no problem producing the correct form of *200 – doscientas*, the feminine form, since *tortas* (*cakes*) is feminine. In order to get this right, one must deduce the coreference relation between *200* and *tortas*. There has of course been a substantial amount of work on reference resolution (see [22.52] for a recent collection of papers), but the problem is hard, and we are far from a solution that covers all cases.
- *Emotion*: When humans read text, they frequently convey affect. This is especially true in genres, such as fiction, where it would generally be quite inappropriate to read the text in a dull-sounding voice that might be more fitting for a list of financial figures. There has been a substantial amount of work on the acoustic correlates of emotion (Chap. 25), but there has been a lot less on how one might predict appropriate emotion from text. One recent strand of work [22.52–54], has used machine-learning methods to attempt to predict the appropriate emotion given features computable from the input text. Still, it will likely be a long while before we will have systems that can perform at all like humans on this problem.
- *Accent and intonation*: Despite the massive amount of work on accenting and intonation, it is still very hard to predict appropriate accent placement, accent type, and intonational parameters from the text. Some of this relates to emotion, which we have just discussed, but not all of the issues are so readily characterized. Classic cases such as the following, which humans would typically have no problem with given appropriate context, are beyond the capabilities of the linguistic analysis components of current TTS systems.

1. *John called Bill a Republican, and then he insulted him.* (Interpretation: after calling Bill a Republican, John insulted Bill.)
2. *John called Bill a Republican, and then **he** insulted **him**.* (Interpretation: John called Bill a Republican, which is interpreted as an insult, so Bill insulted John back.)

By default TTS systems are likely to produce the first accentual variant. In a similar vein current TTS systems are incapable of deciding on the basis of textual analysis that the word *what* in the following example should reasonably be pronounced with an incredulous sounding intonation:

- *Marge ate the whole jar of habaneros.*
- *She did **what**?*



Amazon EX1010 Page 490

Ultimately the problem comes down to the point that we noted in the introduction: in order to read a text, one needs a very good model of the underlying language, and this is precisely what TTS systems lack.

To drive home this point with a final example, consider the following piece of dialog spoken by the character Jim in *Huckleberry Finn*:

> *"Doan' hurt me – don't! I hain't ever done no harm to a ghos'. I alwuz liked dead people, en done all I could for 'em. You go en git in de river agin, whah you b'longs, en doan' do nuffn to Ole Jim, 'at 'uz awluz yo' fren'." (Mark Twain, Huckleberry Finn, Chapter 8).*

Twain's spelling is intended to represent what he calls the Missouri Negro dialect. What is interesting about this example, and others like it, is that you do not actually have to know the dialect that Twain was trying to represent in order to be able to figure out what Jim is saying. Any competent speaker of English could read this text and understand it, and could at least read the words out in their own pronunciation, and if they have at least some familiarity with the dialect region, might even be able to use Twain's orthographic cues to make a guess as to how Jim might have said them. Of course, you are probably thinking that it would be straightforward to build a model that, given appropriate training data, could learn the relevant mappings between Jim's speech and standard English. But this is beside the point: competent speakers can generally adapt quite quickly to such nonstandard spellings without any training whatsoever in the particular spelling system being used. A robust knowledge of the underlying language is enough to overcome noisy input of this kind. Such ambient noise rarely throws off human speech recognition, though it can completely incapacitate ASR.

Just as with ASR, which has arguably reached the limits of what one can do with the current technology, so with linguistic processing for TTS my sense is that we will only see incremental improvements in the short term with tweaks to the current approaches. Underexplored problems will, of course, continue to show improvement as more researchers become interested in them. Emotion prediction from text, for example, is one area where we might expect to see substantial progress over the next few years. Richer datasets with annotations for discourse-level phenomena may, if developed, allow for some improvements in these areas too. Generally, however, we will fail to significantly close the gap between machine and human ability. Entirely new approaches to language understanding will be needed before that can occur. Needless to say, if and when such new approaches are developed they will have applicability well beyond TTS.



## References


22.1 T. Buckwalter: Issues in Arabic morphological analysis. In: *Arabic Computational Morphology: Knowledge-Based and Empirical Methods*, ed. by A. Soudi, G. Neumann, A. van den Bosch (ACM, New York 2006)

22.2 W. Hannas: *Asia's Orthographic Dilemma* (University Hawaii Press, Honolulu 1997)

22.3 K. W. Gan: Integrating word boundary identification with sentence understanding, Ph.D. Dissertation (National University of Singapore, Singapore 1995)

22.4 R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, C. Richards: Normalization of non-standard words, Comput. Speech Lang. **15**(3), 287–333 (2001)

22.5 R. Sproat (Ed.): *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach* (Kluwer, Boston 1997)

22.6 J. Allen, M.S. Hunnicutt, D. Klatt: *From Text to Speech: The MITalk System* (Cambridge Univ. Press, Cambridge 1987)

22.7 C. Coker, K. Church, M. Liberman: Morphology and rhyming: Two powerful alternatives to letter-to-sound rules for speech synthesis, Proc. ESCA Workshop on Speech Synthesis, ed. by G. Bailly, C. Benoit (ESCA, Autrans 1990) pp. 83–86

22.8 K. Koskenniemi: Two-level morphology: a general computational model for word-form recognition and production, Ph.D. dissertation (University of Helsinki, Helsinki 1983)

22.9 J. Goldsmith: Unsupervised acquisition of the morphology of a natural language, Comput. Linguist. **27**(2), 153–198 (2001)

22.10 P. Schone, D. Jurafsky: Knowledge-free induction of morphology using latent semantic analysis, Proc. Comput. Nat. Lang. Learning Conf., Lisbon (2000) pp. 67–72

22.11 D. Yarowsky, R. Wicentowski: Minimally supervised morphological analysis by multimodal alignment, Proc. ACL-2000, Hong Kong (2001) pp. 207–216

22.12 K. Vijay-Shanker, C.-N. Huang: Minimally supervised morphological analysis by multimodal alignment, Proc. 38th Meeting of the Association for Computational Linguistics, Hong Kong (2000) pp. 207–216


Amazon EX1010 Page 491

22.13 J. Pierrehumbert, M. Beckman: *Japanese Tone Structure*, Linguistic Inquiry Monograph Series (MIT Press, Cambridge 1988)
22.14 R. Ladd: *Intonational Phonology* (Cambridge Univ. Press, Cambridge 1996)
22.15 C. Shih: *Prosody Learning and Generation* (Springer, Berlin, Heidleberg 2007)
22.16 K. Church: A stochastic parts program and noun phrase parser for unrestricted text, Proc. Second Conf. Applied Natural Language Processing (ACL, Austin 1988) pp. 136–143
22.17 D. Cutting, J. Kupiec, J. Pedersen, P. Sibun: A practical part-of-speech tagger, Proc. Third Conf. Applied Natural Language Processing (1992)
22.18 E. Brill: A simple rule-based part of speech tagger, Proc. Third Conf. Applied Natural Language Processing (ACL, Trento 1992)
22.19 A. Ratnaparkhi: A maximum entropy part-of-speech tagger, Proc. First Empirical Methods in Natural Language Processing Conference, Philadelphia (1996)
22.20 J. Hirschberg: Pitch accent in context: Predicting intonational prominence from text, Artificial Intelligence **63**, 305–340 (1993)
22.21 S. Pan, K. McKeown: Word informativeness and automatic pitch accent modeling, EMNLP/VLC 99 (Association for Computational Linguistics, College Park 1999)
22.22 E. Fudge: *English Word-Stress* (Allen Unwin, London 1984)
22.23 M. Liberman, R. Sproat: The stress and structure of modified noun phrases in English. In: *Lexical Matters*, ed. by A. Szabolcsi, I. Sag (CSLI University of Chicago Press, Chicago 1992)
22.24 G. Cinque: A null theory of phrase and compound stress, Linguistic Inquiry **24**(2), 239–297 (1993)
22.25 A. Monaghan: Rhythm and stress-shift in speech synthesis, Comput. Speech Lang. **4**, 71–78 (1990)
22.26 R. Sproat: English noun-phrase accent prediction for text-to-speech, Comput. Speech Lang. **8**, 79–94 (1994)
22.27 M. Liberman, A. Prince: On stress and linguistic rhythm, Linguistic Inquiry **8**, 249–336 (1977)
22.28 D. Klatt: Review of text-to-speech conversion for English, J. Acoust. Soc. Am. **82**, 737–793 (1987)
22.29 D. O'Shaughnessy: Parsing with a small dictionary for applications such as text to speech, Comput. Linguist. **15**, 97–108 (1989)
22.30 J. Bachenko, E. Fitzpatrick: A computational grammar of discourse-neutral prosodic phrasing in English, Comput. Linguist. **16**, 155–170 (1990)
22.31 D. Hindle: A parser for text corpora. In: *Computational Approaches to the Lexicon*, ed. by B.T.S. Atkins, A. Zampolli (Oxford Univ. Press, New York 1994)
22.32 J.P. Gee, F. Grosjean: Performance structures: A psycholinguistic and linguistic appraisal, Cognitive Psychology **15**, 411–458 (1983)
22.33 M. Wang, J. Hirschberg: Automatic classification of intonational phrase boundaries, Comput. Speech Lang. **6**, 175–196 (1992)
22.34 L. Breiman, J. Friedman, R. Olshen, C. Stone: *Classification and Regression Trees* (Wadsworth Brooks, Pacific Grove 1984)
22.35 P. Taylor, A. Black: Assigning phrase breaks from part-of-speech sequences, Comput. Speech Lang. **12**, 99–117 (1998)
22.36 N. Ide, J. Véronis: Word sense disambiguation: The state of the art (1998) pp. 1–4
22.37 P. Edmonds, R. Mihalcea, P. Saint-Dizier (Eds.): *ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions* (Association for Computational Linguistics, Philadelphia 2002)
22.38 D. Yarowsky: Three machine learning algorithms for lexical ambiguity resolution, Ph.D. dissertation (University of Pennsylvania 1996)
22.39 D. Yarowsky: Homograph disambiguation in text-to-speech synthesis. In: *Progress in Speech Synthesis*, ed. by J. van Santen, R. Sproat, J. Olive, J. Hirschberg (Springer, New York 1997) pp. 157–172
22.40 P. Taylor, A. Black, R. Caley: The architecture of the Festival speech synthesis system, Proc. Third ESCA Workshop on Speech Synthesis, Jenolan Caves (1998) pp. 147–151
22.41 R. Sproat: Computational morphology. In: *Handbook of Natural Language Processing*, ed. by R. Dale, H. Moisl, H. Somers (Dekker, New York 1997), forthcoming
22.42 W. Skut: Finite-state text processing in a speech synthesis system. In: *Language Technology for Business Applications*, ed. by J. Piskorski, A. Przepiorkowski (Poznan 2004)
22.43 M. Mohri: Finite-state transducers in language and speech processing, Comput. Linguist. **23**, 1 (1997)
22.44 F. Pereira, M. Riley: Speech recognition by composition of weighted finite automata. In: *Finite-State Language Processing*, ed. by E. Roche, Y. Schabes (MIT Press, Cambridge 1997)
22.45 M. Mohri, F. Pereira, M. Riley: A rational design for a weighted finite-state transducer library, Lecture Notes in Computer Science, Vol. 1436 (1998)
22.46 R. Kaplan, M. Kay: Regular models of phonological rule systems, Comput. Linguist. **20**, 331–378 (1994)
22.47 K. Beesley, L. Karttunen: *Finite State Morphology*, CSLI Publications (University of Chicago Press, Chicago 2003)
22.48 M. Mohri, R. Sproat: An efficient compiler for weighted rewrite rules, 34th Annual Meeting of the Association for Computational Linguistics (ACL, Santa Cruz 1996) pp. 231–238
22.49 R. Sproat, J. Hu, H. Chen: EMU: An e-mail preprocessor for text-to-speech, IEEE Signal Processing Society 1998 Workshop on Multimedia Signal Processing, Los Angeles (1998)




Amazon EX1010 Page 492

22.50 H. Chen, J. Hu, R. Sproat: E-mail signature block analysis, ICPR'98, Brisbane (1998)
22.51 T.V. Raman: Audio system for technical readings, Ph.D. dissertation (Cornell University 1994)
22.52 S. Harabagiu, D. Farwell (Eds.): *Workshop on Reference Resolution and its Applications, ACL 2004, Barcelona* (2004)
22.53 C. O. Alm, D. Roth, R. Sproat: Emotions from text: machine learning for text-based emotion prediction, HLT/EMNLP 2005, Vancouver (2005)
22.54 C. O. Alm, R. Sproat: Emotional sequencing and development in fairy tales, First Int. Conf. Affective Computing and Intelligent Interaction, Beijing (2005)



Amazon EX1010 Page 493

# 23. Prosodic Processing

J. van Santen, T. Mishra, E. Klabbers

Speech synthesis systems have to generate natural-sounding speech output from text. One of the key aspects of speech is prosody, which must be both natural (i. e., sounding like a human) and meaningful (i. e., sounding like a human who understands the contents of the text). The computation of prosody from text can be divided into the computation of prosodic tags from text and the computation of acoustic speech features from these tags. This chapter focuses on the latter. It provides an overview of prosody in human–human communication, including the communicative functions of prosody and the acoustic correlates. Discussed next is a historical overview of the various methods that have been used for prosody generation in speech synthesis, as well as of current methods. Special attention is paid to prosody generation in unit selection synthesis methods, in which large corpora are searched for fragments of speech that match the phonemes and prosodic tags computed from text and that optimize various cost functions, and in which prosody is not modeled and speech not modified. We conclude the chapter by advocating hybrid approaches in which search capabilities of unit selection methods are combined with the speech modification methods from more-traditional approaches.

23.1 **Overview** ..... 471
23.1.1 What Is Prosody? ..... 471
23.1.2 Prosody in Human–Human Communication ..... 472

23.2 **Historical Overview** ..... 475
23.2.1 Rule-Based Approaches in Formant Synthesis ..... 475
23.2.2 Statistical Approaches in Diphone Synthesis ..... 475
23.2.3 Using as-is Prosody in Unit Selection Synthesis ..... 475

23.3 **Fundamental Challenges** ..... 476
23.3.1 Challenge to Unit Selection: Combinatorics of Language ..... 476
23.3.2 Challenge to Target Prosody-Based Approaches: Multitude of Interrelated Acoustic Prosodic Features ..... 476

23.4 **A Survey of Current Approaches** ..... 477
23.4.1 Timing ..... 477
23.4.2 Intonation ..... 479

23.5 **Future Approaches** ..... 484
23.5.1 Hybrid Approaches ..... 484

23.6 **Conclusions** ..... 485

**References** ..... 485



## 23.1 Overview

In natural speech, speakers use a multitude of acoustic cues to communicate a message. Some of these cues signal the identities of speech sounds, or phonemes. Additionally, however, there are cues that signal stressed syllables, prominent words, phrase boundaries, sentence mode, metalinguistic features such as irony, and the emotional state of the speaker. For synthetic speech to be fully comprehensible and sound natural, all these functions of prosody must be mimicked. Attention must be paid to how to compute *prosodic tags* such as syllable stress from text, to which *acoustic prosodic features* are used to express these tags (e.g., pitch, timing), and to the precise manifestation of these features (e.g., the shape of the pitch contour). Recent systems differ fundamentally from older systems in how they perform this mimicking task, and there is no reason to assume that future systems will not differ fundamentally from current systems; we will therefore look both into the past and into the future. This chapter will focus on the computation of acoustic manifestations from prosodic tags.

### 23.1.1 What Is Prosody?

Prosody is studied in a range of areas, including linguistics (both at the phonetic and the phonological level), speech pathology, and speech technology; in addition,

we can speak about prosody at the speech production, acoustic, and perceptual levels. As a result, terms do not always have the same meaning in different contexts. In our specific context, however, we can define prosody rather precisely, by explicitly referring to text-to-speech (TTS) components and their internal representations.

We will assume that a TTS system has a *text analysis component*, whose task it is to create a potentially very rich linguistic data structure, perhaps aided by markup of the input text, that contains tags (*target tags*) that characterize input text in terms of phoneme labels, parse trees, part-of-speech labels, syllable stress, and a range of additional features, perhaps even including emotional state or pragmatic features. Generally speaking, all tags except those referring to, or exclusively used for the computation of, phoneme identities can be called prosodic. Systems diverge in how these target tags are used to generate output speech. In some systems, such as unit selection systems, a tagged speech corpus is searched for units that match the target tags. In other systems, such as traditional diphone systems, a *target prosody component* computes from these target tags a specification of the output prosody, or *target prosody*, such as the duration of phonetic segments in ms or the pitch in Hz, which can be overlayed onto selected units by various means.



### 23.1.2 Prosody in Human–Human Communication

The role of prosody in human–human communication is manifold. First, acoustic prosodic features are used to distinguish between lexical items. Examples include pitch in tone languages, duration in languages that distinguish between long and short vowels or that have geminates, and both features – as well as additional features – to create different stress patterns for words that contain the same phonemes. Second, acoustic prosodic features are used to structure utterances in terms of phrases and to indicate relationships between phrases in utterances. For example, utterances containing lists differ in prosodic structure from utterances containing parenthetic remarks. Third, prosody is used to focus attention on certain words for a variety of purposes, such as highlighting a contrast (contrastive stress), emphasizing their importance, or enhancing their intelligibility because they are unpredictable from the context. All these uses share the property that they help the listener understand the contents of the message.

These roles are fairly straightforward, and can be mimicked reasonably well by current text analysis systems. Roles that have traditionally been ignored in TTS systems have to do with *pragmatic* and *affective* uses that go beyond helping to convey the literal contents of the message. There is a vast array of such uses in human–human communication. In fact, one should take note that, at the start of life, in the pre-verbal child, *all* communication is purely prosodic. It is thus not surprising that prosody continues its important role in communication throughout life. Examples include the communication of social roles and authority, making statements about the statement made (e.g., irony, doubt, sarcasm, firmness), instilling an emotion in the listener (e.g., reading a bedtime story to a child, or threatening someone by tone of voice), and, of course, expressing emotions. The reasons for this neglect may include both the difficulty of computing the relevant tags from text and the typical usage of TTS systems – for example, no sarcasm is desired for systems that provide flight arrival times.

We will now briefly review the literature on acoustic features of prosody that is of direct relevance for TTS.

#### Timing

Research of timing in human speech has typically focused on the durations of phonetic segments, less often of syllables or words, or sub-phonemic units. The reason for this is practical rather than fundamental. For example, boundaries between most segments are easy to determine, with the exception of transitions between segments having the same (e.g., n–m) or similar (e.g., vowel to glide) manners of production. At the end of this subsection we will revisit the issue of what the proper unit of timing is. Another limitation of this research is its neglect of pragmatic and affective prosody.

Most studies have focused on vowel duration in read speech. Two methodologies are commonly used. One is to design a balanced text corpus that allows one to test the presence of effects of a specific tag, e.g., word length (contrasting, e.g., brave, braver, bravery). The other method is to take natural text and use sophisticated statistical methods to predict durations from the tagged recorded speech. Both methodologies have shown that vowel duration depends on at least half a dozen factors, including: *identity-related tags* such as the identity of the vowel and of the surrounding (in particular following) consonants; *stress-related tags* such as vowel stress and word prominence; and *positional tags* such as the position of the syllable in the word and in the utterance structure. What is extremely important is that, together, these tags can predict vowel duration rather well. In read speech, a large number of studies have

shown vowel duration can be predicted with mean errors in the 10–20 ms or 15–25% range; correlations between observed and predicted durations tend to be in the 0.80–0.90 range [23.1–4]. Consonant durations, which require slightly different tags (such as within-syllable location: coda versus onset), have provided similar results [23.5].

In other words, durations of phonetic segments can be predicted accurately from text, using as criteria overall statistics such as the median absolute difference between observed (in a speech corpus) and predicted durations. There are at least two important caveats here, however. One is that such criteria may hide audible flaws in specific subsets of the data. For example, when we modify predictions by reducing the predicted duration of utterance-final vowels by 50%, there will be a pronounced effect perceptually but it will have little impact on the statistics because utterance-final vowel tokens are only a small percentage of the collection of all vowel tokens in a corpus. This suggests using a minimax error criterion (which minimizes the maximum error), after splitting up the data in appropriate subsets. Second, as shown by *Kato* et al. [23.6], listeners may be less attuned to absolute durations than to relative durations of neighboring units. This suggests using an error criterion that depends on specific durational ratios.

Why study durations of phonetic segments? One idea that has been discussed is to study the durations of longer units such as words or syllables. The merit of this idea for the spurposes of TTS is limited, because, even if it were the case that the durations of these longer units are extremely well behaved and hence highly predictable (e.g., exact isochrony at the syllable or foot level), one still needs to specify durations at the subunit level. For example, *van Santen* and *Shih* [23.7] showed that in stressed versus unstressed syllables mostly the onset and nucleus are stretched whereas in phrase-final versus phrase-medial syllables, primarily the nucleus and the coda, are stretched. In addition, one may even have to pay attention to timing at the sub-phonemic level, because it is also known that in contrasts such as mend versus meant mostly the final part of the nucleus as well as the nasal coda consonant are stretched, whereas in bide versus bite the early part of the diphthong is stretched [23.8].

Finally, the concept of a timing *unit*, whatever its size, inaccurately presupposes that the underlying articulatory gestures are always in lock-step with each other. The reason for neglecting this point of view in TTS is, of course, that articulatory timing does not easily map onto concatenative TTS architectures, or at least those architectures that compute target prosody and then impose the target timing on stored units by compressing or expanding them but not by modifying their spectral trajectories in ways that are reflective of articulatory asynchronies.

In summary, while in terms of overall statistical criteria the prediction of durations looks like a solved problem, several issues remain, including the search for perceptually more valid error criteria and a better understanding of sub-phonemic timing, in particular at the articulatory level.

### Pitch

Research on pitch in human–human communication goes back far longer than research on timing because it can be studied – certainly qualitatively – without digital methods for speech segmentation. Because of this long history, but also because this area is fraught with controversy and different schools of thought, we will not attempt to review the vast literature on this topic here. Instead, we will review only some of the basic facts about pitch.

Pitch is the perceptual correlate of fundamental frequency $F_0$. Intonation refers to patterns of pitch (at the perceptual level) or $F_0$ (at the production level) in speech. $F_0$ is more difficult to measure than segment duration for a variety of reasons. First, phonatory state (e.g., creaking, breathiness, low harmonic-to-noise ratio) may cause pitch tracking errors. Second, $F_0$ is absent in unvoiced regions, including in parts of nominally voiced regions that are often devoiced (e.g., voiced fricatives). Third, there are important segmental perturbations even in regions where $F_0$ can be reliably extracted. For example, vowel–nasal or nasal–vowel transitions are often accompanied by short-lived pitch movements of up to 10 Hz. Also, during the first 50 ms of a post-obstruental vowel, $F_0$ may be increased by 25 Hz or more. Finally, high vowels are associated with higher pitch values than low vowels, by as much as 10 Hz.

These difficulties result in $F_0$ contours that are often inaccurate and, even if accurate, are hard to interpret in terms of which features reflect intended prosody rather than segmental articulatory effects. What makes matters particularly complex is that these segmental articulatory effects are not random noise that can be removed by smoothing but have locally systematic effects that – if one were to make the traditional assumption that pitch peaks are critical for prosodic interpretation – may produce spurious pitch peaks or obscure true pitch peaks.

In part to deal with these difficulties, a number of descriptive schemes have been proposed. Some of these describe $F_0$ contours in terms of a small num-



Amazon EX1010 Page 496

ber of *pitch targets* [23.9]. A labeler, who is acutely aware of these measurement and interpretation difficulties, can reliably label a contour in terms of such targets. A different approach is to stylize $F_0$ contours using a sequence of straight line segments. If done carefully, these line segments may indeed cut through the regions polluted by segmental articulatory effects and generate a more-interpretable processed contour. Finally, quantitative models have been proposed that fit the entire curve; by having a relatively small number of parameters, the fitted curve will not mimic the fine details of the observed curve, and thereby – typically, but not necessarily – eliminate the segmental effects.

Of course, these schemes are not merely methods for generating more-robust descriptors of $F_0$ contours, but also constitute intonational theories, ranging from purely nonprocess phonological linguistic theories [23.9] to theories about the physiology of phonation [23.10]. While discussing these theories in detail is beyond the scope of this chapter, it must be made clear that these theories have implications for how one processes speech in TTS. For example, systems based on tones and break indices (ToBI= [23.11] – an intonational labeling scheme based on Pierrehumbert's work – compute *abstract intonational tags* (such as $H^*$ for a high tone associated with a stressed syllable) from text, and then either compute target prosody from these labels (in traditional diphone synthesis) or use the labels to search a speech corpus (in unit selection synthesis). This intermediate layer of abstract intonational tags is absent in typical applications of the Fujisaki model, where the values of the Fujisaki model's parameters are predicted directly from tags in the linguistic data structure relating to standard features such as syllable position, syllable stress, and other tags that do not include abstract intonational tags such as in ToBI.

We end this subsection by briefly listing a few of the basic intonational phenomena in standard US English. First, the global shape of $F_0$ contours. $F_0$ contours typically show local maxima close to stressed syllables. There is also usually a globally downward trend of the $F_0$ contour over the duration of a phrase. This downward trend is reversed in the final syllable or syllables in yes/no questions or in nonterminal phrases, but further accelerates downward in terminal phrases. In yes/no questions, there is a sharp rise at the end of the phrase; in nonterminal phrases, there is often a slight rise, or rise–fall–rise pattern (*continuation rise*) at the end. A further downward trend phenomenon occurs when speakers produce a list of items; here, typically each list item is presented in a register lower than that of the preceding item. A TTS system that mimics these basic shapes correctly can be considered adequate for conveying nonpragmatic or affective prosody. However, even a cursory inspection of affectively or pragmatically laden speech reveals an array of more-complex $F_0$ phenomena. For example, stressed syllables may be associated with local $F_0$ minima; there may exist $F_0$ plateaus, or regions with a global upward trend, that persist for several words. Generally, current TTS systems other than special unit selection based systems that have the right recordings (e.g., of emotional speech) do not mimic these phenomena.

A second basic phenomenon of $F_0$ is perceptual sensitivity to small changes in how pitch movement is aligned with the syllable or segmental boundaries [23.12, 13]. In studies of this phenomenon, it was found that changes of less than 100 ms in alignment were not only audible but also changed perceived meaning.

In summary, mimicking natural intonation is seriously challenged by the limited knowledge we have about natural intonation. Many challenges remain regarding how to measure and characterize pitch contours, how they vary as a function of prosodic content, and how intonation is perceived.

### Other Acoustic Features

Most TTS systems are concerned with timing and intonation. However, there is strong evidence that other acoustic features play a role in prosody as well. One of these has to do with the overall spectral shape of speech, reflecting multiple aspects of speech production such as phonation and lower-jaw position. For example, at the end of an utterance subglottal pressure tends to decrease and the vocal chords – specifically changes in the open quotient – contribute to a more-breathy voice quality. Also, a lowered jaw position results in a large oral aperture, which in turn results in less attenuation of higher frequencies.

A second feature has to do with reduction phenomena. Reduced vowels tend to have formant trajectories that are more-heavily influenced – coarticulated – by neighboring phonemes, and may as a result deviate substantially from their target values. These deviations contribute to the perception of a reduced syllable as being unstressed [23.14]. Likewise, reduced consonants may have less-complete closure in combination with less subglottal pressure, which leads to frication of stops and to *muffled* fricatives.

It is not certain how important the contributions of these features are to overall perceived naturalness of syn-



Amazon EX1010 Page 497

thetic speech. However, it is important to keep in mind that, even if they play a background role in affectively and pragmatically neutral speech, their role in charged speech is likely to be considerable, in particular for features such as harshness of voice, tenseness, and other phonatory features.

## 23.2 Historical Overview

Before giving an overview of current approaches to prosodic processing, we briefly mention older approaches in order to set the stage for a discussion of the fundamental challenges that these current approaches are facing (Sect. 23.3).

### 23.2.1 Rule-Based Approaches in Formant Synthesis

The first computer-based speech synthesis systems used rules to generate speech. For duration modeling, the best-known type of rule-based approach is a sequential rule system as applied by *Klatt* [23.15, 16] in the MITalk system [23.17]. His model assumes that

1. each phonetic segment type has an inherent duration specified as one of its distinctive features,
2. each rule results in a percentage increase or decrease in the duration of the segment, but
3. the segment cannot be compressed shorter than a certain minimum duration.

The rules were derived from small speech production experiments and the literature. More-detailed information about the Klatt duration model can be found in Sect. 23.4.1.

For intonation modeling, a well-known rule-based approach is presented by *Pierrehumbert* [23.9, 18]. This approach describes the intonation contour as a series of target values. Because of the declination effect, the $F_0$ range narrows and drifts downwards over the course of the phrase. Targets are viewed as belonging to two categories, high and low. When two targets are both high and are sufficiently separated in time, the computer program computes a sagging contour between them. Otherwise, the $F_0$ contour between the two targets is a monotonic curve. More-detailed information about Pierrehumbert's approach can be found in Sect. 23.4.2.

### 23.2.2 Statistical Approaches in Diphone Synthesis

In the past two decades advances in computer technology have led to the use of statistical approaches to model prosody. These approaches use large databases of speech to train prosodic models. For duration modeling, a well-known model is the sums-of-products approach [23.1] as implemented in the Bell Labs multilingual speech synthesis system [23.19]. This approach takes advantage of the fact that most interactions are directionally invariant, which allows these interactions to be described with equations consisting of sums and products. Phonemes that are affected similarly by the various factors are grouped into subclasses. The decisions concerning this grouping are based on exploratory data analysis and phonetic/phonological literature. For each subclass of segments, a separate sums-of-products model is trained. Exploratory data analysis is an essential step in order to appropriately decide which factors are important and how many levels on a factor should be distinguished. For each subclass this can lead to different results. Thus, a great deal of phonetic and phonological knowledge can be incorporated in the model. More-detailed information about the sums-of-products model can be found in Sect. 23.4.1.

A well-known statistical approach for modeling intonation is the tilt model [23.20]. In the tilt model, intonation is characterized by a sequence of phonetic intonational events. Like in Pierrehumbert's model these events are pitch accents and boundary tones. Each event has a fall and rise component which can vary in size and can be described by five *tilt* parameters. Speech corpora are analyzed automatically to find events and corresponding tilt parameters. A classification and regression tree (CART) can then be used to retrieve tilt parameters for synthesis [23.21]. More-detailed information about the tilt model can be found in Sect. 23.4.2.

### 23.2.3 Using as-is Prosody in Unit Selection Synthesis

The next step in speech synthesis development was to use large corpora of natural speech not just for training prosody models, but also as a source of units for concatenative synthesis. Instead of having one token for each diphone, the corpus contains several tokens with different phonetic and prosodic context characteristics.



Amazon EX1010 Page 498

One of the first systems to use such a unit selection synthesis approach was the CHATR system [23.22, 23]. The phoneme durations and $F_0$ values at 10 ms intervals are used as target values when searching for the most-optimal units. The best units are then concatenated and no further prosodic modification takes place. More-detailed information about prosody in unit selection synthesis can be found in Sect. 23.4.2.

## 23.3 Fundamental Challenges

### 23.3.1 Challenge to Unit Selection: Combinatorics of Language

As has been pointed out by several authors [23.24, 25], TTS systems have to face the challenge of generating speech for text that has not been pre-recorded.

For unit selection based systems, the combinatorial challenge is far more severe, because, in the absence of prosodic modification of the stored speech, recordings are needed of each unit in a large number of prosodically distinct contexts. The analyses by *van Santen* [23.24] show that, for unrestricted domain synthesis, the chance that the units needed to render an arbitrary sentence are available in the desired prosodic contexts in even a very large corpus is vanishingly small.

This does not mean that there are no combinatorial challenges for target prosody-based systems that modify the prosody of stored speech. In fact, because the number of combinations of prosodic tags is extremely large, a premium is put on the *generalization capabilities* of the statistical methods used for, e.g., duration prediction. This also means that one can deceive oneself severely when a system is evaluated on a test text corpus that is not too different from the corpus used for statistical estimation, because the test text corpus may have substantial overlap of prosodic context *types*. It has been shown [23.2] that statistical approaches that impose constraints on durational behavior, such as quasi-linear regression based methods (e.g., sums-of-products models; [23.2]) fare far better than approaches that do not impose constraints, such as classification and regression trees [23.26]. These constraints can be of several types. For example, the constraint of *directional invariance* states that, e.g., for two occurrences of the same vowel in two contexts that only differ in stress, the stressed occurrence will be longer in duration. In other words (where $D()$ indicates duration, $v$ a vowel, $S$ stress, and $c_1, \cdots, c_n$ other prosodic tags),

$$D(v, c_1, \cdots, c_n, +S) \geq D(v, c_1, \cdots, c_n, -S) \,.$$

A different constraint states that the ordering of durations over all prosodic contexts is the same for each phoneme within a given class (e.g., voice fricatives). For example (where PF denotes phrase finality),

$$\begin{aligned} D(+S, +\mathrm{PF}) &\geq D(-S, +\mathrm{PF}) \\ &\geq D(+S, -\mathrm{PF}) \\ &\geq D(-S, -\mathrm{PF}) \,. \end{aligned}$$

These constraints have the effect of providing reasonable extrapolation or interpolation to new types.

### 23.3.2 Challenge to Target Prosody-Based Approaches: Multitude of Interrelated Acoustic Prosodic Features

While avoiding the combinatorial explosion inherent in the requirement for unit selection-based approaches to have recordings of all units in a great variety of prosodic contexts, target prosody-based approaches face three different and equally severe fundamental problems. First, the target prosody is synthetic, not natural. Thus, the generated pitch contours are not guaranteed to have natural shapes, and the rhythm may be off.

Second, this approach allows the TTS system to only mimic those features of acoustic prosody that are explicitly modeled. Thus, if only $F_0$ and duration are computed, then the output speech may have inappropriate voice quality or loudness.

Third, the signal processing needed to impose target prosody on recorded units is well known to often cause audible distortions.



Amazon EX1010 Page 499

## 23.4 A Survey of Current Approaches

With these inherent limitations in mind, we now turn to a survey of current approaches to prosodic processing.

### 23.4.1 Timing

In this section we will give an overview of the predominant approaches to duration modeling.

#### The Klatt Duration Model

The Klatt duration model was developed by Dennis Klatt at MIT in the 1970s and 1980s [23.27]. It is part of the MITalk formant synthesizer [23.17]. The Klatt model is a sequential rule-based model. The model assumes that

1. each phonetic segment type has an inherent duration that is specified as one of its distinctive properties,
2. each rule tries to effect a percentage increase or decrease in the duration of the segment, but
3. segments cannot be compressed shorter than a certain minimum duration [23.15].

The model is summarized by the following equation:

$$\mathrm{DUR} = \mathrm{MINDUR} + \frac{(\mathrm{INHDUR} - \mathrm{MINDUR})\mathrm{PERC}}{100}, \tag{23.1}$$

where MINDUR is the minimal duration of the segment in an *accented* condition, INHDUR is the inherent duration of a segment in milliseconds, and PERC is the percentage shortening or lengthening determined by applying a set of rules in sequence. Klatt used a set of 10 rules that relate to effects of the phonetic environment, emphasis, stress level, etc. on the current segment's duration. Each rule adjusts the PERC term multiplicatively and the final result is the product of the rules plus the effect of one final rule that is applied after the calculation of DUR [23.17].

The problem with this type of approach is that the parameter values are based on small-scale studies and findings reported in the literature in which the factors pertaining to a particular rule are varied while most others are kept constant. Since there may be interactions between contextual factors, the obtained parameter values are likely to be inaccurate for contexts where the factors held constant have different levels. Models of this type have been developed for several languages including American English [23.17,28], Swedish [23.29], German [23.30], and French [23.31].

#### Classification and Regression Trees (CARTs)

CARTs were first introduced by *Breiman* et al. [23.32], and are tree-based nonlinear regression algorithms that have proven successful for a variety of prediction problems where there are nonlinear relations between the the variable you are trying to predict and the predictors. *Riley* [23.26] proposed the use of CART to model phoneme durations for text-to-speech synthesis. The CARTs can be constructed automatically. A binary-branching tree is grown with an algorithm that accepts phoneme labels with correct durations expressed as $z$-scores from a set of training data. The tree construction method looks for binary splits determined by a single factor that best correlates with the context by minimizing the variance of the estimated error within each cluster after the split. The result is a tree with questions at the node pertaining to the place and manner of articulation of current and surrounding segments, the position of a segment in the syllable, word, or phrase, or the lexical stress or accent on a syllable, etc., and with predicted duration values at the leaves. At synthesis time, the tree is traversed for each phoneme until a predicted $z$-score duration value is found. The main concern with training a CART is that the tree can become so large that it is overtrained, in which case it matches the training data well, but does not perform so well on new test data. Overtraining can be prevented by pruning the tree. A cross-validation scheme is used in which the tree is grown on 9/10 of the available data and tested on 1/10 of the data which is repeated 10 times for different subsets and averaging their variance. Branches that do not fit the unseen data well are then pruned. Because the procedure is completely automatic, CARTs are a very popular method of duration prediction. There are many programs available to train CARTs, including the *Wagon* procedure that is part of Festival [23.22]. However, overtraining or lack of coverage of the relevant phonetic/prosodic contexts may lead to suboptimal prediction. The CART approach to duration modeling has been used in TTS systems of several languages including Korean [23.33], Czech [23.34], Hindi and Telugu [23.35], and Italian [23.36]. The correlation between observed and predicted durations for these systems lies between 0.6 and 0.8 and the root-mean-squared error (RMSE) lies between 20 and 27 ms.

#### Neural Networks

One well-known example of a neural-network-based approach to duration modeling is the syllable-based approach developed by *Nick Campbell* as part of his



Amazon EX1010 Page 500

PhD thesis at the University of Edinburgh in the early 1990s [23.37]. The assumption is that neural networks can learn the underlying interactions between the contextual effects. The model predicts syllable durations first and then fits phoneme durations to the syllables. For each syllable a feature vector is computed which consists of information about the number of phonemes in the syllable, the nature of the syllabic peak (or nucleus, e.g., reduced, lax, tense vowel), the position in the tone group, the type of foot, stress level, and word class (function versus content word).

The syllable duration is calculated without prior knowledge of phonetic constraints. The phoneme durations within each predicted syllable class are then determined using an elasticity principle. For each phoneme class, there is a distribution of log durations that resembles a normal distribution and that is characterized by a mean duration $\mu$ and a standard deviation $\sigma$. In the majority of cases, the amount of compression or lengthening within a syllable can be expressed with a single constant $k$. This relies on the assumption that all segments in a syllable are lengthened or shortened equally in terms of standard deviation. Thus, the duration of the $i$-th segment is given by $\exp(\mu_i + k\sigma_i)$, where $k$ is computed such that the following expression matches the precomputed overall syllable duration:

$$\sum_{i=1}^{n} \exp(\mu_i + k\sigma_i) \,. \tag{23.2}$$

This approach does not take the segmental make-up of the syllable into account, which can contribute considerably to the overall syllable duration and to the amount of lengthening applied to one particular phoneme.

### The Sums-of-Products Approach

The sums-of-products approach was developed by *van Santen* et al. [23.2] as part of the Bell Labs multilingual text-to-speech synthesis system [23.19]. A sums-of-products model (23.3) gives the duration for a phoneme/context combination, as described by the feature vector $\boldsymbol{f}$. $K$ is a set of indices, each corresponding to a product term. $I_i$ is the set of indices of factors occurring in the $i$-th product term. In other words, a sums-of-products model is an equation whose parameters correspond to factor levels, which are combined by taking products of subgroups of parameters and then adding the products. Two examples of sums-of-products models are the additive model (where $K = \{1, \ldots, N\}$ and $I_i = \{i\}$) and the multiplicative model (where $K = \{1\}$ and $I_1 = \{1, \ldots, N\}$):

$$\mathrm{DUR}(\boldsymbol{f}) = \sum_{i \epsilon K} \prod_{j \epsilon I_i} S_{i,j}(f_j) \,. \tag{23.3}$$

To build a sums-of-products model for a TTS system, a large text corpus is used and all information relevant for duration prediction is computed and encoded in feature vectors. A set of sentences covering a subset of feature vectors is selected using a greedy algorithm. These sentences are then recorded and segmented into phonemes. Statistical analyses are carried out on the phoneme durations to manually construct a *category tree* in which all phonemes are grouped into subclasses that are affected similarly by contextual factors (not like in CART where phonemes with similar durations are grouped together) and to construct a set of sums-of-products models for each of the leaves in the tree. The statistical analyses ensure that the category tree and sums-of-products models best represent the data with as few parameters as possible.

The training is an iterative process in which exploratory data analysis is used to determine which levels have to be distinguished on a given factor for each phoneme category and which factors interact. By discarding irrelevant factors and factor levels, the number of missing data in the database will decrease and the number of observations per factor level will increase. By collapsing the correct factor levels, the predictions will be more accurate. The decisions are based on a number of information sources:

1. phonological/phonetic knowledge
2. the corrected means that are computed to determine the effect size of each factor level
3. the two-way corrected means that are computed to determine whether factors interact
4. the outcome of the multiplicative model in terms of prediction error

The calculations take place in the logarithmic domain to reduce the skewness of the duration distribution and to allow a more-accurate prediction of small durations.

The sums-of-products approach has been used for many different languages including American English, German ([23.3], $r = 0.90$, RMSE $= 19$ ms), Mandarin Chinese ([23.4], $r = 0.87$, RMSE $= 26$ ms), and Dutch ([23.38], $r = 0.77$, RMSE $= 23$ ms). *Maghbouleh* [23.39] has shown that the sums-of-products approach requires less training data than CART to achieve comparable results and performs better on unseen test data.



Amazon EX1010 Page 501

### 23.4.2 Intonation

There are many intonation models and theories, however, not all of them have been used for TTS. In this section, we will only present models that have been used in TTS synthesis systems. We will outline the implicit assumptions underlying each model, the explicit representation of $F_0$ chosen by each model, and the way in which each model can be used to compute intonation in a text-to-speech system. The presented models cover a wide spectrum of intonation approaches. We will be describing two phonological tone sequence models (the Pierrehumbert theory, the ToBI-based approaches), a phonetic sequential model that is not tone inspired [the recurrent neural network (RNN) model], a phonetic sequential model that involves acoustic stylization (the tilt model), one perceptually motivated acoustic stylization model [the Institute of Perception Research (IPO) model], three phonetic superpositional models [the Fujisaki model, the linear alignment model, the superposition of functional contours (SFC) model], one prosodic model (the Kiel model), and one physiological model [the soft template markup language (STEM-ML) model].

#### Pierrehumbert's Theory of Intonation

The Pierrehumbert theory of intonation [23.9], developed by Janet Pierrehumbert in 1980 as part of her doctoral dissertation, is a phonological model of intonation. It is based on autosegmental-metrical (AM) phonology [23.40, 41]. In keeping with the AM theory, the Pierrehumbert model considers intonation to be a sequence of high (H) and low (L) tones. The H and L are in *phonological opposition*, i. e., the difference in sound between them serves to distinguish intonational meaning. The two types of tones never interact with each other, rather they follow each other sequentially in an utterance.

The H and L tones are the building blocks of three larger tone units: *pitch accents*, *phrase accents*, and *boundary tones*. *Pitch accents* mark prominence. They are either single tones (H∗, L∗), or pairs of tones (L+H∗, L∗+H, H+L∗, H∗+L); the ∗ denotes the alignment of the tone with a stressed syllable. One or more pitch accents comprise an *intermediate phrase*. One or more intermediate phrases comprise an *intonational phrase*, the largest prosodic unit posited by this theory. The edges of the intonational phrase are marked by *boundary tones*. The boundary tones are single tones (%H, %L, H%, L%); the % denotes the alignment of the boundary tone with the pitch onset or offset of the intonational phrase. Pitch movement between a pitch accent and a boundary tone is indicated by a *phrase accent* (H−, L−), denoted by the diacritic, −.

To ensure that the model renders well-formed intonational representations, Pierrehumbert defined a finite state grammar that specifies the combinations in which pitch accents, phrase accents, and boundary tones can occur. She also devised a set of *phonetic realization rules* [23.18] to produce $F_0$ contours from the phonological model of intonation described above.

Generating the $F_0$ contour of a target utterance in a TTS system using the Pierrehumbert theory of intonation involves three main steps: first, determine the tonal representation of the utterance using the finite state grammar. Second, specify the target $F_0$ values of the high and low tones depending on the metrical prominence of the associated syllables, and the $F_0$ values of the preceding tones using the phonetic realization rules. Third, using the rules again, connect the target $F_0$ values to generate a $F_0$ contour: if two neighboring targets are far apart, connect them with a sagging contour implemented via a quadratic function. Otherwise, connect them via monotonic curves.

#### ToBI-Based Approaches

ToBI stands for *tones and break indices* [23.11]. Based on Pierrehumbert's theory of intonation, it was developed in four research meetings between 1991 and 1994 as a standard for describing American English intonation. It has since been extended to transcribe other languages and dialects [23.42–44].

ToBI consists of three parallel labeling tiers. The first tier is the *tone tier*. The tones specified by Pierrehumbert's theory are labeled in the tone tier. The second tier is the *break index tier*. In the break index tier, *break indices*, ranging from 0 to 4, are marked. *Break indices* mark the boundary strength between adjacent words; 0 indicates no boundary, 3 indicates an intermediate phrase boundary (- in Pierrehumbert's model), and 4 indicates a intonational phrase boundary (% in Pierrehumbert's model). The third tier is a miscellaneous tier, where hesitations, disfluencies, laughter, non-speech sounds, etc., are labeled.

It is important to note that ToBI is a labeling system and it *does not* specify the means to produce quantitative intonation from the ToBI labels. However, there are both rule-based and statistically trained approaches that can be applied to the ToBI labels to generate $F_0$ contours. An example of the rule-based approach is Jilka's handcrafted rule system for specifying the $F_0$ contour of American English from ToBI labels [23.45].



Amazon EX1010 Page 502

Jilka's approach is similar to Pierrehumbert's *phonetic realization rules*: rules specify the target $F_0$ values associated with ToBI labels, depending on pitch range and the voiced part of the syllable. The target $F_0$ values are calculated from left to right, taking into account only preceding ToBI labels, not subsequent ones. The $F_0$ contour is produced by linear interpolation between target points.

An example of the statistically trained approach is *Black* and *Hunt*'s linear-regression-based approach for generating $F_0$ contours from ToBI labels [23.46]. This approach simply involves predicting three target $F_0$ values for every syllable, one at the start of the syllable, one at mid-vowel position, and one at the end of the syllable, by means of linear regression. The prediction formula is defined as,

$$F_0 = I + w_1 f_1 + w_2 f_2 + w_3 f_3 + \ldots + w_n f_n \; . \tag{23.4}$$

The $f_i$ variables indicate the features that contribute to the $F_0$ value of a syllable, such as ToBI label associated with the syllable, syllable position in the phrase, syllable stress etc., $I$ and $w_i$ are parameters that are estimated by linear regression.

### The RNN Intonation Model

The recurrent neural network (RNN) intonation model, developed by *Traber* in 1991 [23.47, 48], uses *neural networks* to predict the $F_0$ contour. A *neural network* can be considered to be a nonlinear statistical model with many parameters. These parameters are estimated in the training phase so that they can return an optimal set of outputs from the corresponding input. The neural-network-based approach was motivated by the goal of using minimal human effort to obtain high-quality intonation, i. e., while humans would specify *which* phonological units were relevant for the phonetic realization of intonation, clever machine learning techniques would figure out *how* the phonological units map to the $F_0$ contour.

In the implementation of this intonation model in the SVOX TTS system [23.49], the $F_0$ contour was generated per syllable. For each syllable, the accent value of the syllable and segmental properties relating to its position in the phrase and the sentence, as well as the accent values of the neighboring syllables are given as input to the neural network. The network outputs the $F_0$ region related to the syllable (represented by 8 samples of the $F_0$ contour). The resultant $F_0$ regions are concatenated together to produce the complete $F_0$ contour.

### The Tilt Intonation Model

The tilt intonation model, developed by *Taylor* and *Black* [23.20] at the Center for Speech Technology Research of the University of Edinburgh, considers intonation to be a sequence of intonational events, which are parameterized by *tilt parameters*. The model posits four basic types of intonational events: pitch accents, boundary tones, connections (regions in the $F_0$ contour between two pitch accents, two boundary tones, or a pitch accent and a boundary tone), and silence.

Pitch accents and boundary tones are each modeled by piecewise combinations of quadratic functions; these quadratic functions may be rising or falling. Connections are modeled by straight-line interpolations. The amplitude and duration of the rising and falling quadratic functions, the position of the associated intonational event in the time–$F_0$ plane, together with a *tilt* value associated with each event constitutes the set of *tilt parameters* associated with the intonational events. The *tilt* parameter represents the amount of rise and fall of each accent. The tilt value is a difference in the amplitudes of the rise and fall functions, divided by their sum [23.21], as shown below:

$$\text{tilt} = \frac{|\text{Amp}_{\text{rise}}| - |\text{Amp}_{\text{fall}}|}{|\text{Amp}_{\text{rise}}| + |\text{Amp}_{\text{fall}}|} \; . \tag{23.5}$$

The tilt value ranges from $-1$ to 1, where $-1$ indicates a pure fall, 1 indicates a pure rise and 0 indicates a rise followed by a fall of equal magnitude. Thus, the tilt model uses continuous parameters rather than imposing categorical classification on the intonational events.

*Dusterhoff* and *Black* [23.21] have shown that the tilt model can be successfully used to predict $F_0$ contours in a text-to-speech system. The tilt-based $F_0$ generation process has two stages: a training stage and a testing stage. The training stage requires a training database. The database is labeled with tilt events, either automatically or by hand. For each syllable in the database marked with a tilt event, a set of linguistic-prosodic features are extracted. The features are grouped into separate training sets depending on event type. A CART [23.32] training algorithm is applied to each of the training sets to develop a decision tree for every tilt parameter. The decision trees thus describe the tilt parameters in terms of an optimal subset of the extracted features.

The training stage described above is performed offline. The tilt parameter descriptions obtained in the training stage are used in the testing stage for $F_0$ contour generation of given text. The given text is labeled with tilt events, and the same set of linguistic-prosodic



Amazon EX1010 Page 503

features (as in the training set) are extracted. The related tilt parameters are calculated from the extracted features using the descriptions obtained from training. The tilt parameters are then plugged into predetermined quadratic or linear functions to model the pitch accents and boundary tones, or connections respectively.

### The IPO Approach

The IPO approach [23.50, 51] was developed at the Institute of Perception Research (IPO) in Eindhoven, the Netherlands in the 1960s. It was originally used to model Dutch intonation. The IPO model is often classified as a *perceptual* intonation model because of the assumptions underlying the model.

1. Not all changes in $F_0$ are perceived by the human ear.
2. Only $F_0$ changes that are perceived by the human ear need to be modeled.
3. The human ear perceives tone variations (rise versus fall) and not tone intensities (high versus low).

Given these assumptions, the IPO approach models the raw $F_0$ contour as a piecewise linear approximation of the original contour, known as a *close copy contour*. It is called a *close copy contour* because, upon resynthesis, it is perceptually indistinguishable from the original $F_0$ contour. Generating the close copy contour also includes specifying the *declination line* (a line that represents the overall downward trend of the $F_0$ contour).

Close copy contours are classified into discrete, phonetically defined types of $F_0$ rises and falls. The classification parameters describe the deviation of the close copy from the declination line, and include descriptive factors such as its height and slope relative to the declination line, its span relative to the span of the declination line, its timing in relation to associated syllable(s) duration, its rate of change, etc. The particular parameters used for classification differ from language to language.

Once an inventory of $F_0$ rises and falls covering the entire combinatorial space of the classification parameters has been collected, a *grammar* specifying the possible and permissible combinations of the $F_0$ rises and falls is written in terms of the parameters. When the IPO intonation model is used in speech synthesis systems, this grammar is used for $F_0$ contour generation. $F_0$ contours predicted by this grammar must be perceptually equivalent to, and as acceptable as natural $F_0$ contours [23.19]. The IPO model has been implemented in speech synthesis systems for Dutch [23.52], English [23.53], and German [23.54].

### The Fujisaki Intonation Model

The Fujisaki intonation model, presented by Hiroya *Fujisaki* [23.10, 55] is the best known of the class of intonation models known as *superpositional* models (or *overlay* models) of $F_0$. Superpositional models consider $F_0$ to be a complicated function that can be decomposed into simpler component functions. In the Fujisaki model, the $F_0$ contour is considered to be an addition (in the log domain) of two components: the *phrase command* and the *accent command*.

The *phrase command* characterizes the overall trend of the intonation of an utterance, represented by the global movement of the associated $F_0$ contour. The *accent command* on the other hand, highlights particularly extreme excursions of intonation (to stress certain syllables or words) in the utterance, represented by the local peaks and valleys in the $F_0$ contour. The phrase command is modeled by pulses, while the accent command is modeled by step functions. The discontinuities in the two commands are then smoothed using separate filters to output phrase and accent components that appear continuous. The phrase and accent components are then added in the log domain to produce an *additive $F_0$ contour* – the defining characteristic of the Fujisaki model.

To use the Fujisaki model for $F_0$ prediction in text-to-speech synthesis, the pulses are placed at intonational phrase boundaries, while the step functions are associated with other key phonological units such as *accent groups*. Linguistic and other properties of the text determine the amplitude of each of the commands, and the width of the accent commands. This model has been successfully used for intonation modeling of many languages [23.56, 57].

### The Linear Alignment Model

The linear alignment model is another example of a superpositional model. It was developed by *van Santen* and *Möbius* [23.58] at Bell Laboratories. The distinguishing characteristic of this model is that it pays particular attention to the *alignment* between the pitch contour and the segmental stream underlying it.

Its concern with alignment is most effectively expressed in its modeling of the accent curve component. The accent component represents the same aspects of the $F_0$ contour as the accent component in the Fujisaki model, though it is modeled differently in this model. An accent curve is modeled by parameterized time warps of an accent curve *template*. The *template* can be defined as a sequence of anchor values, $T_p = \langle P_1, P_2, \ldots, P_n \rangle$ that describe the archetypical shape of the associated accent



Amazon EX1010 Page 504

curve type, $P$. Also associated with $P$ is an *alignment parameter matrix*, an ensemble of regression weights that describe the alignment of $P$ to the segmental region underlying it. All accent curves of type $P$ have the same template and the same alignment parameter matrix; they differ from each other only in terms of their duration.

Besides the accent curve, two other components of the additive $F_0$ contour are specified by the linear alignment model: the phrase curve and the *segmental perturbation curve*. As in the Fujisaki model, the phrase curve illustrates the long-term shape of the $F_0$ contour. The phrase curve is modeled by a piecewise linear function. The *segmental perturbation curve* described the segmental influences on the pitch contour such as pitch increase in vowels following voiceless plosives, and pitch lowering in nasals and glides. The segmental perturbation curves are modeled by exponential decay functions.

The linear alignment Model has been used for generating intonation in the Bell Labs multilingual text-to-speech system [23.59]. To synthesize speech from text, each of the three components of $F_0$ specified by the linear alignment model have to be related to linguistic entities. The phrase curve is anchored at three points: the start of the utterance, the start of the syllable that carries the nuclear pitch accent, and the end of the utterance. The accent curve is tied to a *left-headed foot*. A *left-headed foot* is defined as a sequence consisting of an accented syllable followed by all unaccented syllables that precede the next accented syllable or a phrase boundary. The degree of emphasis at a particular foot is obtained by multiplying the accent curve by a height factor. The phrase curve is tied to the intonational phrase. Minor and major phrase breaks are distinguished where major phrase breaks occur at sentence ends. Segmental perturbation curves are anchored at vowel onset. The amplitude of this function is determined by the broad class of the onset consonant; it has a maximal value for voiceless consonants, a smaller value for voiced obstruents, and a zero value for sonorants.

### The SFC Model

The superposition of functional contours (SFC) model of intonation was developed at the Institute for Speech Communication. It was proposed by *Aubergé* [23.60] and implemented by *Bailly* and *Holm* [23.61]. Like other superpositional models, the principal assumption of the SFC model is that the pitch contour is obtained by a superposition of simpler contours. In case of the SFC model, the simpler contours are multiparametric contours called *functional contours* (FCs).

Functional contours form the core of the distinguishing assumption of this model. They are assumed to directly encode specific *metalinguistic functions* tied to various discourse units, without any intermediate representation [23.62]. *Metalinguistic functions* refer to intonation functions that delimit phonological units and convey propositional and interactional information about these units within the discourse. Examples of metalinguistic functions are hierarchy, segmentation, emphasis, and speaker attitude.

Every functional contour has the following three properties:

1. It is *function-specific*, i. e., tied to a particular metalinguistic function.
2. It spans the extent of the unit(s) tied to the function it encodes – this extent is called the *scope* or domain of the FC.
3. The FC shape is a function of the metalinguistic function it encodes and its scope; however, it is important to note that the FC shape is not specified a priori in this model, rather it emerges in the training phase of the model's implementation.

The SFC model has been implemented for pitch prediction in TTS systems for German, French, and Mandarin Chinese. The metalinguistic functions that will be encoded by the functional contours are defined. One *contour generator* per function is implemented as a neural network. Each contour generator generates a family of functional contours that encode the same metalinguistic function and hence have the same shape, differing only in terms of their time domains. The input to each contour generator is information relating to the scope of the associated function, and the position of each syllable within the scope. The output is four output parameters per syllable (three $F_0$ values and a lengthening factor).

Training the contour generators to generate a particular pattern of functional contour is not a straightforward process as recovering the unique contributions of the contour generators to their sum (i. e. the $F_0$ contour) is an ill-posed problem. To determine the individual contributions of each contour generator and the particular pattern of the FC, an *analysis-by-synthesis loop* [23.61] is used.

### The Kiel Intonation Model (KIM)

The Kiel intonation model was developed by *Kohler* and his colleagues [23.63, 64] to model intonation patterns in German. In KIM, the $F_0$ contour is modeled as a sequence of *global intonational units*, each linked to one



Amazon EX1010 Page 505

emphasized word. The *global intonational units* are considered to be produced and perceived holistically, and cannot be split. These global units are either peaks or valleys or peak-valley combinations, and differ from each other in term of their pragmatic, semantic, syntactic, and meaning functions. They were determined in KIM by means of function-oriented phonetic experiments. KIM postulates that there is a prototypical intonational unit associated with a particular pragmatic–semantic–syntactic function combination. However, KIM does not ignore the *microprosodic phenomena* (e.g., $F_0$ shifts at the obstruent-vowel boundaries, $F_0$ changes in nasals and glides; effects of intrinsic pitch are also included in this category) observed in the $F_0$ contour; it is also incorporated in the model.

Since KIM was developed with a focus on TTS synthesis, the $F_0$ prediction rules are well specified. KIM applies two sets of rules, namely *symbolic feature rules* and *parametric rules*, for pitch prediction in TTS systems. The symbolic feature rules are applied to *phonological units* which have been annotated with syntactic, pragmatic and semantic markers. The *phonological units* are either segmental (vowels and consonants) or nonsegmental (morphological and phrase boundaries). The symbolic feature rules output the global intonational units associated with the phonological units, encoded as binary features (such as +/−terminal, +/−valley, +/−quest, +/−early, +/−late). These feature values are then used by the parametric rules to generate the $F_0$ contour of the target utterance. The parametric rules include rules for aligning the global intonational units with the segmental structure of the target utterance, downstepping of accent peaks, speech rate, prosodic boundaries and, finally, articulation-induced microprosody [23.65].

### The STEM-ML Intonation Model

Soft template markup language (STEM-ML) is a physiological model of intonation. It was developed by *Shih* and *Kochanski* in 2000 [23.66] to investigate the deviation of Mandarin Chinese tones from their expected canonical shape when occurring in natural sentences. However, this model has been designed to be language independent, and thus can be applied to non-tone languages such as English.

STEM-ML is founded on three key assumptions.

1. Human speech is pre-planned several syllables in advance.
2. Humans produce speech that optimally balances physiological effort required to speak against unambiguity of the spoken message. The speaker expends maximal effort to produce correct prosody at prosodically crucial events because the cost of ambiguity is high at these points. However, he minimizes effort between such events because the cost of ambiguity is low.
3. Speech prosody is continuous and smooth over short time periods.

STEM-ML includes a tagging system (see [23.67] for a complete description of the tag set) for intonation markup and specification, and a quantitative model to generate the $F_0$ contour. Two important building blocks of the STEM-ML model are parameters and *soft templates*. The parameters are associated with the tags in the tagging system. The *soft templates* are a part of the quantitative intonation generation model. The parameters and the soft templates together generate the $F_0$ contour.

In this model, the $F_0$ contour is considered a concatenation of the local accents. The local accents are represented by the *soft templates*. The soft templates are *soft* in the sense that the accent templates allow substantial distortion caused by neighboring accents. The concept of soft templates arise from the previously stated pre-planning assumption. An accent template is affected by past as well as future templates. The degree of distortion is controlled by a parameter called *strength*. The strength parameter reflects the cost of ambiguity in the previously stated assumption regarding optimally balanced speech. So, if strength (hence cost of ambiguity) is large, the template shape remains unchanged to reflect maximal articulatory effort, whereas if it is low, the accent shape is compromised to reflect minimal articulatory effort.

Besides local tags that control local accent shapes, there are global tags that control speaker-specific information. Thus, a STEM-ML model is built on a particular speech corpus. The implementation of the STEM-ML model involves two phases: the learning phase and the generation phase. In the learning phase, the values of the parameters are determined iteratively by minimizing the difference between the actual $F_0$ of every STEM-ML tagged utterance in the corpus and the $F_0$ predicted by the model. In the generation phase, when faced with a target utterance, the model first tags the text underlying the utterance, then uses the predetermined values of the parameters associated with the tags to modify the soft templates, and finally, concatenates the modified accent templates to produce the $F_0$ contour. STEM-ML has been used to model Man-



Amazon EX1010 Page 506

darin [23.68], Cantonese [23.69], and English [23.70] speech.

#### Using as-is Prosody in Unit Selection Synthesis

There are currently many unit selection synthesis systems that do not modify the intonation of the concatenated units at all. Some systems such as CHATR use direct duration and $F_0$ values in their target cost when searching for units, whereas other systems such as the Actor TTS system [23.71] use an abstract feature vector containing information about the location of the unit in the sentence to ensure selection of units that fit the target contour. The main advantage of this approach is that the output speech has a very natural voice quality as no signal modifications are made that degrade the signal. However, data sparsity is unavoidable [23.24,25] and as such there will always be contexts for which no appropriate units can be found, leading to audible prosodic discontinuities at unit boundaries and selection of units with inappropriate prosody that can change the intended meaning of the sentence. In most unit selection systems, the corpus consists of neutral news-style recordings to minimize these prosodic discontinuities at unit boundaries. However, there is an increasing need for expressive speech synthesis, which increases the number of prosodic contexts substantially, making it impossible to record a speech corpus large enough to avoid discontinuities.

#### Advantages and Disadvantages

As shown in the preceding descriptions, the different intonational models are implemented in TTS systems using different computational mechanisms: linguistic–prosodic rules, templates, neural nets, decision trees, and linear regression. Each of these computational mechanisms have their advantages and disadvantages. The advantages of rule-based approaches are that they are easy to implement and that they produce consistent intonational contours. The disadvantage of the rule-based approaches is that intonational contours lack the richness and variability of natural contours. The machine learning approaches (i. e., those using decision trees, neural nets and linear regression) on the other hand, have the advantage of being able to produce natural-sounding intonational contours, because they learn the mapping between annotated text and corresponding natural pitch contours in the training phase. However, the machine learning approaches have the disadvantage of requiring large amounts of training data to cover the combinatorial space of phoneme sequences and prosodic contexts. Since it is not feasible to cover this entire combinatorial space, machine learning approaches are often beset with data sparsity problems. The reason for capturing the contours related to the same phoneme sequence in different prosodic contexts is that the shape of the contour changes depending on context. One of the main advantages of the template-based approach is its ability to handle the changing shape of an intonational contour depending on context. The template-based approach posits a reasonable number of templates (or prototypical shapes) related to different prosodic events that can be distorted by means of parameters. However, one of the drawbacks of this approach is that the template and parameters may be fit to data in a way that is phonologically absurd.



## 23.5 Future Approaches

### 23.5.1 Hybrid Approaches

In order for unit selection synthesis to generate speech with natural and acceptable prosody, a different approach is needed. Some research has been devoted to select natural prosodic contours from dedicated prosodically balanced speech corpora in conjunction with the traditional unit search in a phonetically balanced speech corpus [23.72–74]. This significantly cuts down on the amount of speech data needed, as there is more emphasis on the phonetic context for the unit selection and more emphasis on the prosodic context for the prosody selection. The natural prosody contours are imposed on the units, so a high-quality speech modification algorithm is key for natural sounding speech. The use of dedicated prosody corpora allows for including different speaking styles and affective modes for synthesis. The use of the superpositional model of intonation by *van Santen* et al. [23.74] to select natural phrase and accent curves that are then superimposed to create the final intonation contour, ensures that the resulting contour is smooth and thus there will be no $F_0$ mismatch at unit boundaries.

Amazon EX1010 Page 507

## 23.6 Conclusions

This chapter provided a conceptual framework for prosodic processing in text-to-speech synthesis, and surveys both recent and older approaches. Over the years, many quite diverse approaches have been invented, ranging from approaches that generate prosody by rule to approaches that take prosodic patterns in the database as is. Despite this broad-based attack on this hard problem, there is clearly still much work to be done. Certainly, unit selection systems in which the database provides a very good coverage of a – necessarily restricted – domain will provide the best quality. As we argued, there are practical problems that limit this approach to certain types of applications where custom voices are not needed and where the overall speaking style is fairly muted. We sketched an alternative approach which is data driven, and in fact uses both prosodic patterns and phonetic sequences as found in the data base with a minimum of speech modification. Whether such an approach will be both more practical than, and comparable in quality to, the unit selection-based approach remains to be seen.

### References


23.1 J. van Santen: Contextual effects on vowel duration, Speech Commun. **11**(6), 513–546 (1992)

23.2 J. van Santen: Exploring N-way tables with Sums-of-Product models, J. Mathemat. Psychol. **37**(3), 327–371 (1993)

23.3 B. Möbius, J. van Santen: Modeling segmental duration in German text-to-speech synthesis, Proc. 1996 Int. Conf. Spoken Lang. Process. Philadelphia (1996) pp. 2395–2398

23.4 C. Shih, B. Ao: Duration study for the Bell Laboratories Mandarin text-to-speech system. In: *Progress in Speech Synthesis*, ed. by J. van Santen, R. Sproat, J. Olive, J. Hirschberg (Springer, New York 1996) pp. 383–397

23.5 J. van Santen: Assignment of segmental duration in text-to-speech synthesis, Computer Speech Language **8**, 95–128 (1994)

23.6 H. Kato, M. Tsuzaki, Y. Sagisaka: Acceptability for temporal modification of single vowel segments in isolated words, J. Acoust. Soc. Am. **104**(1), 540–549 (1998)

23.7 J. van Santen, C. Shih: Suprasegmental and segmental timing models in Mandarin Chinese and American English, J. Acoust. Soc. Am. **107**(2), 1012–1026 (2000)

23.8 J. van Santen: Segmental duration and speech timing. In: *Computing Prosody*, ed. by Y. Sagisaka, W.N. Campbell, N. Higuchi (Springer, New York 1996)

23.9 J.B. Pierrehumbert: *The phonetics and phonology of English intonation*, Ph.D. Thesis (MIT, Cambridge 1980)

23.10 H. Fujisaki: Dynamic characteristics of voice fundamental frequency in speech and singing. In: *The Production of Speech*, ed. by P.F. MacNeilage (Springer, New York 1983) pp. 39–55

23.11 K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, J. Hirschberg: ToBI: A standard for labeling English prosody, Proc. 1992 Int. Conf. Spoken Language Processing Banff (1992) pp. 867–870

23.12 K.J. Kohler: Macro and micro F0 in the synthesis of intonation. In: *Papers in Laboratory Phonology I: Between the Grammar and Physics of Speech*, ed. by J. Kingston, M.E. Beckman (Cambridge Univ. Press, New York 1990) pp. 115–138

23.13 M. d'Imperio, D. House: Perception of questions and statements in Neapolitan Italian, Proc. Fifth European Conference on Speech Communication and Technology Rhodes (1997)

23.14 D.J. Broad, F. Clermont: Linear scaling of vowel-formant ensembles (VFEs) in consonantal contexts, Speech Commun. **37**, 175–195 (2002)

23.15 D.H. Klatt: Interaction between two factors that influence vowel duration, J. Acoust. Soc. Am. **54**, 1102–1104 (1973)

23.16 D.H. Klatt: Linguistic uses of segmental duration in English: Acoustic and perceptual evidence, J. Acoust. Soc. Am. **59**, 1209–1221 (1976)

23.17 J. Allen, S. Hunnicut, D. Klatt: *Text-to-Speech: The MITalk System* (Cambridge Univ. Press, Cambridge 1987)

23.18 J.B. Pierrehumbert: Synthesizing intonation, J. Acoust. Soc. Am. **70**, 985–995 (1981)

23.19 R. Sproat (Ed.): *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach* (Kluwer, Dordrecht 1997)

23.20 P. Taylor: Analysis and synthesis of intonation using the Tilt model, J. Acoust. Soc. Am. **107**(3), 1697–1714 (2000)

23.21 K. Dusterhoff, A. Black: Generating F0 contours for speech synthesis using the Tilt intonation theory, Intonation: Theory, Models and Applications, Proc. ESCA Workshop, ed. by A. Botinis, G. Kouroupetroglou, G. Carayiannis (1997) pp. 107–110

23.22 A. Black, P. Taylor: CHATR: A generic speech synthesis system, Proc. COLING'94 Kyoto (1994) pp. 983–986




Amazon EX1010 Page 508

23.23 A. Black, N. Campbell: Prosody and the selection of source units for concatenative synthesis. In: *Progress in Speech Synthesis*, ed. by J. van Santen, R. Sproat, J. Olive, J. Hirschberg (Springer, New York 1995) pp. 279–292

23.24 J. van Santen: Combinatorial issues in text-to-speech synthesis, Proc. Eurospeech **97**, 2511–2514 (1997)

23.25 B. Möbius: Rare events and closed domains: Two delicate concepts in speech synthesis, Proc. 4rd ESCA Workshop on Speech Synthesis Pitlochry (2001)

23.26 M.D. Riley: Tree-based modeling for speech synthesis. In: *Talking Machines: Theories, Models, and Designs*, ed. by G. Bailly, C. Benoit, T. Sawallis (Elsevier, Amsterdam 1992) pp. 265–273

23.27 D.H. Klatt: Synthesis by rule of segmental durations in English sentences. In: *Frontiers of Speech Communication Research*, ed. by B. Lindblom, S. Öhman (Academic, New York 1979) pp. 287–300

23.28 J.P. Olive, M.Y. Liberman: Text to speech – An overview, J. Acoust. Soc. Am. **78**(Suppl. 1), 6 (1985)

23.29 R. Carlson, B. Granström: A search for durational rules in a real-speech database, Phonetica **43**, 140–154 (1986)

23.30 K.J. Kohler: Zeitstrukturierung in der Sprachsynthese, ITG-Fachbericht **105**, 165–170 (1994), in German

23.31 K. Bartkova, C. Sorin: A model of segmental duration for speech synthesis in French, Speech Commun. **6**, 245–260 (1987)

23.32 L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone: *Classification and Regression Trees* (Wadsworths Brooks, Monterey 1984)

23.33 H. Chung: Duration models and the perceptual evaluation of spoken Korean, Proc. Speech Prosody 2002 Aix-en-Provence (2002)

23.34 R. Batůšek: A duration model for Czech text-to-speech synthesis, Proc. Speech Prosody 2002 Aix-en-Provence (2002)

23.35 N.S. Krishna, H.A. Murthy: Duration modeling of Indian languages Hindi and Telugu, 5th ISCA Workshop of Speech Synthesis Pittsburgh (2005)

23.36 F. Tesser, P. Cosi, C. Drioli, G. Tisato: Prosodic data driven modelling of a narrative style in Festival TTS, Proc. 5th ISCA Workshop on Speech Synthesis Pittsburgh (2005)

23.37 W.N. Campbell: Syllable-based segmental durations. In: *Talking Machines: Theories, Models, and Designs*, ed. by G. Bailly, C. Benoit, T. Sawallis (Elsevier, Amsterdam 1992) pp. 43–60

23.38 E. Klabbers: *Segmental and Prosodic Improvements to Speech Generation*, Ph.D. Thesis (Eindhoven University of Technology, Eindhoven 2000)

23.39 A. Maghbouleh: An empirical comparison of automatic decision tree and linear regression models for vowel durations, Proc. second meeting of the ACL Special Interest Group in Computational Phonology Santa Cruz (1996)

23.40 D.R. Ladd: *Intonational Phonology* (Cambridge Univ. Press, Cambridge 1996)

23.41 J.A. Goldsmith: *Autosegmental and Metrical Phonology* (Blackwell, Oxford 1990)

23.42 N. Campbell, J. Venditti: J-ToBI: An intonation labelling system for Japanese, Proc. Autumn Meeting Acoust. Soc. Jpn. **1**, 317–318 (1995)

23.43 C. Mayo, M. Aylett, D.R. Ladd: Prosodic transcription of Glasgow English: An evaluation study of Glatobi, Proc. ESCA Workshop: Intonation: Theory, Models and Applications, ed. by A. Botinis, G. Kouroupetroglou, G. Carayiannis (ESCA, 1997) pp. 231–234

23.44 M. Reyelt, M. Grice, R. Benzmuller, J. Mayer, A. Batliner: Prosodische Etikettierung des Deutschen mit ToBI. In: *Natural Language and Speech Technology*, ed. by D. Gibbon (Mouton de Gruyter, Berlin 1996) pp. 144–155, in German

23.45 M. Jilka, G. Mohler, G. Dogil: Rules for the generation of ToBI-based American English intonation, Speech Commun. **28**, 83–108 (1999)

23.46 A. Black, A. Hunt: Generating F0 contours from the ToBI labels using linear regression, Proc. 4th Int. Conf. Spoken Language Process. **3**, 1385–1388 (1996)

23.47 C. Traber: F0 generation with a database of natural F0 patterns and with a neural network. In: *Talking Machines: Theories, Models, and Designs*, ed. by G. Bailly, C. Benoit, T. Sawallis (Elsevier, Amsterdam 1992) pp. 287–304

23.48 C. Traber: Syntactic processing and prosody control in the SVOX TTS system for German, Proc. Eurospeech **93**, 2099–2102 (1993)

23.49 C. Traber: *SVOX: The Implementation of a Text-to-Speech System for German*, Ph.D. Thesis (ETH Zurich, Zurich 1995)

23.50 A. Cohen, J. 't Hart: On the anatomy of intonation, Lingua **19**, 177–192 (1967)

23.51 J. 't Hart, R. Collier, A. Cohen: *A Perceptual Study of Intonation: An Experimental-Phonetic Approach to Speech Melody* (Cambridge Univ. Press, Cambridge 1990)

23.52 J. Terken: Synthesizing natural-sounding intonation for Dutch: rules and perceptual evaluation, Computer Speech Language **7**, 27–48 (1993)

23.53 N. Willems, R. Collier, J. 't Hart: A synthesis scheme for British English intonation, J. Acoust. Soc. Am. **84**(4), 1250–1261 (1988)

23.54 J. van Hemert, U. Adriaens-Porzig, L. Adriaens: Speech synthesis in the SPICOS project. In: *Analyse und Synthese gesprochener Sprache*, ed. by H. Tillmann, G. Willee (Georg Olms, Hildesheim 1987) pp. 34–39

23.55 H. Fujisaki, K. Hirose: Modelling the dynamic characteristics of voice fundamental frequency with applications to analysis and synthesis of intonation, Preprints of the Working Group on Intonation, 13th Intl. Congress of Linguists Tokyo (1982) pp. 57–70




Amazon EX1010 Page 509

23.56 H. Fujisaki: Modelling in the study of tonal features of speech with application to multilingual speech synthesis, Joint Conference of SNLP and Oriental COCOSDA Hua Hin Prachuapkirikhan (2002)

23.57 H. Mixdorff: Quantitative tone and intonation modeling across languages, Proc. Int. Symp. Tonal Aspects of Languages: With Emphasis on Tone Languages Beijing (2004) pp. 137–142

23.58 J. van Santen, B. Möbius: A quantitative model of $F_0$ generation and alignment. In: *Intonation: Analysis, Modeling and Technology*, ed. by A. Botinis (Kluwer Academic, Dordrecht 1999) pp. 269–288

23.59 J. van Santen, B. Möbius, J. Venditti, C. Shih: Description of the Bell Labs Intonation System, Proc. 3rd ESCA Speech Synthesis Workshop Jenolan Caves (1998) pp. 293–298

23.60 V. Aubergé: Prosody modeling with a dynamic lexicon of intonative forms: Application for text-to-speech synthesis, Proc. ESCA Workshop on Prosody (1993) pp. 62–65

23.61 B. Holm, G. Bailly: Generating prosody by superposing multi-parametric overlapping contours, Proc. Int. Conf. Speech and Language Processing Beijing (2000) pp. 203–206

23.62 G. Bailly, B. Holm: SFC: A trainable prosodic model, Speech Commun. **46**, 364–384 (2005)

23.63 K.J. Kohler: Studies in German intonation, Arbeitsberichte des Instituts für Phonetik und digitale Sprachverarbeitung, Universität Kiel **25**, 295–360 (1991)

23.64 K.J. Kohler: Parametric control of prosodic variables by symbolic input in TTS synthesis. In: *Progress in Speech Synthesis*, ed. by J. van Santen, R. Sproat, J. Olive, J. Hirschberg (Springer, New York 1997) pp. 459–475

23.65 Kohler K.J.: The Kiel Intonation Model (KIM), its Implementation in TTS Synthesis and its Application to the Study of Spontaneous Speech (1995), retrieved on July 15th, 2006 from http://www.ipds.uni-kiel.de/kjk/forschung/kim.en.html

23.66 G.P. Kochanski, C. Shih: Stem-ML: Language independent prosody description, Proc. Int. Conf. Spoken Lang. Process. **3**, 239–242 (2000)

23.67 G.P. Kochanski, C. Shih: Prosody modeling with soft templates, Speech Commun. **39**(3-4), 311–352 (2003)

23.68 G.P. Kochanski, C. Shih: Automated modelling of Chinese intonation in continuous speech, Proc. Eurospeech **01**, 911–914 (2001)

23.69 T. Lee, G. Kochanski, C. Shih, Y. Li: Modeling tones in continuous Cantonese speech, Proc. 2002 International Conference on Spoken Language Processing Denver (2002) pp. 2401–2404

23.70 C. Shih, G. Kochanski: Modeling intonation: Asking for confirmation in English, Proc. 15th Int. Congress of Phonetic Sciences Barcelona (2003)

23.71 S. Quazza, L. Donetti, L. Moisa, P.L. Salza: ACTOR: A multilingual unit-selection speech synthesis system, Proc. 4th ESCA Workshop on Speech Synthesis Pitlochry (2001)

23.72 F. Campillo-Díaz, E.R. Banga: Combined prosody and candidate unit selections for corpus-based text-to-speech systems, Proc. 7th Int. Conference on Spoken Language Processing (2002) pp. 141–144

23.73 A. Raux, A. Black: A unit selection approach to F0 modeling and its application to emphasis, ASRU 2003 St. Thomas (2003)

23.74 J. van Santen, A. Kain, E. Klabbers, T. Mishra: Synthesis of prosody using multi-level sequence units, Speech Commun. **46**(3-4), 365–375 (2005)




Amazon EX1010 Page 510

# 24. Voice Transformation

Y. Stylianou

Voice transformation refers to the various modifications one may apply to the sound produced by a person, speaking or singing. In this chapter we give a description of various ways in which one can modify a voice and provide details on how to implement these modifications using a simple, but quite efficient, parametric model based on a harmonic representation of speech. By discussing the quality issues of current voice transformation algorithms in conjunction with the properties of speech production and perception systems we try to pave the way for more-natural voice transformation algorithms in the future.

24.1 **Background** ..... 489

24.2 **Source–Filter Theory and Harmonic Models** ..... 490
24.2.1 Harmonic Model ..... 490
24.2.2 Analysis Based on the Harmonic Model ..... 491
24.2.3 Synthesis Based on the Harmonic Model ..... 491

24.3 **Definitions** ..... 492
24.3.1 Source Modifications ..... 492
24.3.2 Filter Modifications ..... 493
24.3.3 Combining Source and Filter Modifications ..... 494

24.4 **Source Modifications** ..... 494
24.4.1 Time-Scale Modification ..... 495
24.4.2 Pitch Modification ..... 496
24.4.3 Joint Pitch and Time-Scale Modification ..... 496
24.4.4 Energy Modification ..... 497
24.4.5 Generating the Source Modified Speech Signal ..... 497

24.5 **Filter Modifications** ..... 498
24.5.1 The Gaussian Mixture Model ..... 499

24.6 **Conversion Functions** ..... 499

24.7 **Voice Conversion** ..... 500

24.8 **Quality Issues in Voice Transformations** ..... 501

24.9 **Summary** ..... 502

**References** ..... 502

## 24.1 Background

Voice transformation refers to the various modifications one may apply to the sound produced by a person, speaking or singing. Voice transformation involves signal processing and the physics (or at least the understanding) of the speech production process and natural language processing. Driven mainly by its applications, signal processing has evolved faster than the physics of speech processing, even giving the impression that signal processing alone may be required to achieve high-quality voice transformation. To an external observer, this is similar to the problem of how to make an omelette without eggs. It is not surprising therefore that, although for some categories of voice transformation good quality of speech is produced, this is not true in general. While it is relatively easy to explain to a non-speech expert the necessity of speech modeling by providing examples from the history of telecommunications, this is not obvious for voice transformation. About two decades ago, it was easy to explain the applications of voice transformation technology to a speech processing engineer by providing examples from a specific area of speech technology: concatenating speech synthesis. In the late 2000s, providing a reason for voice transformation to both a speech expert and nonexpert faced the same difficulty. One reason for this was that the main application of voice transformation, that of concatenating speech synthesis, had evolved in a direction where it seemed that signal processing was no longer needed for this application. Such a point of view, however, was also supported by the quality problems perceived in a modified speech signal.

Recently, interest in voice transformation has increased substantially, and it is again the application of speech synthesis that is setting the pace. Voice



Amazon EX1010 Page 511

transformation is a flexible, possibly simple, and efficient way to produce the variety needed in the current text-to-speech (TTS) systems based on the concatenation of units (both large and small) [24.1]. In this chapter, we give a description of various ways in which one can modify a voice and provide details of how to implement these modifications using a simple, but quite efficient, parametric model based on a harmonic representation of speech. Discussing quality issues of current voice transformation algorithms in conjunction with properties of the speech production and perception systems we try to pave the way for more-natural voice transformation algorithms in the future.

## 24.2 Source–Filter Theory and Harmonic Models

### 24.2.1 Harmonic Model

When designing voice transformation techniques it is often convenient to refer to the source–filter model of speech production. According to this model, speech is viewed as the result of passing a glottal excitation signal (source) through a time-varying linear filter that models the resonant characteristics of the vocal tract. The most well-known source–filter system is that based on linear prediction (LP) of speech [24.2]. In its simplest form, a time-varying filter modeled as an autoregressive (AR) filter is excited by either quasiperiodic pulses (during voiced speech), or noise (during unvoiced speech). Many attempts have been made to improve the source (excitation) signal in the LP context. This includes multipulse LP [24.3], and code-excited linear prediction (CELP) [24.4]. A more-compact and at the same time flexible representation of the excitation signal has been proposed from a family of speech representations referred to as sinusoidal models (SM) [24.5]. In SM, the excitation signal for both voiced and unvoiced speech frames is represented by a sum of sinusoids:

$$e(t) = \sum_{k=0}^{K(t)} a_k(t) \mathrm{e}^{\mathrm{i}\phi_k(t)} \,, \tag{24.1}$$

where $a_k(t)$ and $\phi_k(t)$ are the instantaneous excitation amplitude and phase of the $k$-th sinusoid, respectively, and $K(t)$ is the number of sinusoids, which may vary in time. Especially for speech signals, a model where the sinusoids are harmonically related is quite valid (in the mean-squared-error (MSE) sense) while it allows a simple and convenient way of applying various modifications to the speech signal. In this case:

$$\dot{\phi}_k(t) = 2\pi k f_0(t) \,, \tag{24.2}$$

where $f_0(t)$ is the instantaneous fundamental frequency, which will also be referred to as the *pitch* in this chapter. Such a representation is still valid for both voiced and unvoiced speech frames. In the case of unvoiced speech frames a constant fundamental frequency is considered (i. e., 100 Hz) resulting in a Karhunen–Loeve representation of this speech category [24.5]. A further simplification of the excitation signal is convenient assuming that the excitation amplitude, $a_k(t)$, is constant over time and equal to unity: $a_k(t) = 1$. Based on these simplifications, the time-varying linear filter that models the resonant characteristics of the vocal tract approximates the combined effect of:

1. the transmission characteristics of the supraglottal cavities (including radiation at the mouth opening)
2. the glottal pulse shape

Its time-varying transfer function can be written

$$H(f;t) = G(f;t)\mathrm{e}^{\mathrm{i}\Psi(f;t)} \,, \tag{24.3}$$

where $G(f;t)$ and $\Psi(f;t)$ are, respectively, referred to as the time-varying amplitude and phase of the system. Speech processing is often (if not always) performed in a frame-by-frame basis, where each frame (i. e., about 20 ms) is considered to be a stationary process. In this case, inside a frame, the filter $H(f;t)$ is considered as linear time invariant (LTI). Then, the output speech signal $s(t)$ can be viewed as the convolution of the impulse response of the LTI filter, $h(t)$, and the excitation signal, $e(t)$:

$$s(t) = \int_0^t h(t-\tau)e(\tau)\mathrm{d}\tau \,. \tag{24.4}$$

Recognizing then that the excitation signal is just the sum of $K(t)$ eigenfunctions of the filter, $H(f)$, the



Amazon EX1010 Page 512

following speech model is obtained:

$$s(t) = \sum_{k=0}^{K(t)} G[f_k(t)]\, \mathrm{e}^{\mathrm{i}\{\phi_k(t)+\Psi[f_k(t)]\}}$$
$$= \sum_{k=0}^{K(t)} A_k(t)\, \mathrm{e}^{\mathrm{i}\theta_k(t)}\,, \qquad (24.5)$$

where $f_k(t) = k f_0(t)$ (the eigenfrequencies). The harmonic amplitude $A_k(t)$ of the $k$-th harmonic is the system amplitude $G[f_k(t)]$ (the eigenvalue). The phase $\theta_k(t)$ of the $k$-th harmonic is the sum of the excitation phase $\phi_k(t)$ and the system phase $\Psi[f_k(t)]$:

$$\theta_k(t) = \phi_k(t) + \Psi[f_k(t)]\,; \qquad (24.6)$$

$\theta_k(t)$ is often referred to as the *instantaneous phase* of the $k$-th harmonic.

### 24.2.2 Analysis Based on the Harmonic Model

Parameters of the harmonic model of speech may be estimated by minimizing a least-squares criterion [24.6] or by minimizing a mean-squared error that leads to a simple *peak-picking* approach [24.5]. The peak-picking approach results in a sinusoidal rather than a harmonic model. A second step is then required to fit a harmonic model to this sinusoidal model by selecting the fundamental frequency that best represents the set of estimated sinusoids. At each analysis time instant, $t_\mathrm{a}^i$, a set of parameters are estimated: the fundamental frequency, $f_0^i$, the harmonic amplitudes, $A_k^i$, and the harmonic phases, $\theta_k^i$.

Use of the harmonic model for voice transformations is simplified if the distance between two successive analysis time instants is equal to the local pitch period, $P(t_\mathrm{a}^i) = 2\pi / f_0^i$:

$$t_\mathrm{a}^{i+1} = t_\mathrm{a}^i + P\left(t_\mathrm{a}^i\right)\,. \qquad (24.7)$$

Another important step before synthesis is required: the estimation of the *amplitude and phase envelopes*, $A(f)$ and $\theta(f)$ (i. e., a continuous function of frequency) from the discrete set of amplitude and phase values, respectively.

While a number of methods can be used to estimate the amplitude envelope, for example, the linear prediction and homomorphic estimation techniques [24.7], it is desirable to use a method that yields an envelope that passes through the measured harmonic amplitudes. Such a technique was developed for the spectral envelope estimation vocoder (SEEVOC) [24.8] and was used in the sinusoidal model in [24.5]. Another approach was proposed in [24.9]: it provides a continuous frequency envelope when values of this envelope specified only at discrete frequencies (i. e., exactly the situation in the previously described harmonic representation). This approach makes use of cepstral coefficients and is based on a frequency-domain least-squares criterion combined with regularization to increase estimation robustness.

For the phase envelope $\theta(f)$, the previous techniques cannot be used since the phase values have been estimated modulo $2\pi$ (principal values). Therefore, a phase unwrapping algorithm has to be used. Two main approaches exist:

1. phase continuity by adding appropriate multiples of $2\pi$ to the principal phase values [24.10]
2. continuity by integration of the phase derivative

These algorithms try to obtain a continuous phase envelope in the frequency domain. An extension of these techniques to preserve the continuity in the time domain as well has been proposed using the information of phase from previous voiced frames [24.11].

An alternative to the phase envelope approach is the use of a minimum phase model for the system phase, while for the excitation phase a representation of the excitation in terms of its impulse locations (*onset times*) is used [24.12]. This approach, however, lacks robustness because estimation of the onset times requires precision that is not always easy to obtain.

Next, we will consider the case when a spectral envelope, $A(f)$, and phase envelope, $\theta(f)$, are provided.

### 24.2.3 Synthesis Based on the Harmonic Model

Without speech modification, synthesis time instants, $t_\mathrm{s}^i$ coincide with the analysis time instants $t_\mathrm{a}^i$, i.e., $t_\mathrm{s}^i = t_\mathrm{a}^i$, $\forall i$.

Let $(A_k^i, \theta_k^i, f_0^i)$ and $(A_k^{i+1}, \theta_k^{i+1}, f_0^{i+1})$ denote the set of parameters at synthesis time instant $t_\mathrm{s}^i$ and $t_\mathrm{s}^{i+1}$ for the $k$-th harmonic, respectively. Amplitudes and phases are obtained by sampling the phase and amplitude (spectral) envelopes at the harmonics of the fundamental frequencies $f_0^i$ and $f_0^{i+1}$. The instantaneous amplitude $A_k(t)$ is then obtained by linear interpolation of the



Amazon EX1010 Page 513

estimated amplitudes at the frame boundaries:

$$A_k(t) = A_k^i + \frac{A_k^{i+1} - A_k^{i+1}}{t_s^{i+1} - t_s^i} t \quad \text{for } t_s^i \le t < t_s^{i+1} . \tag{24.8}$$

In contrast to the third-order polynomial used in [24.13, 14], the harmonic model allows the use of a simple first-degree polynomial for the phase. First, the phase at $t_s^{i+1}$ is predicted from the estimated phase at $t_s^i$ by

$$\hat{\theta}_k^{i+1} = \theta_k^i + k2\pi f_{0\text{av}}\left(t_s^{i+1} - t_s^i\right) , \tag{24.9}$$

where $f_{0\text{av}}$ is the average value of the fundamental frequencies at $t_s^i$ and $t_s^{i+1}$:

$$f_{0\text{av}} = \frac{f_0^i + f_0^{i+1}}{2} . \tag{24.10}$$

Next, the phase $\theta_k^{i+1}$ is augmented by the term $2\pi M_k$ ($M_k$ is an integer) in order to approach the predicted value. Therefore, the value of $M_k$ is given by

$$M_k = \left\langle \frac{1}{2\pi}\left(\hat{\theta}_k^{i+1} - \theta_k^{i+1}\right) \right\rangle . \tag{24.11}$$

Then, the instantaneous phase $\theta_k(t)$ is simply obtained by linear interpolation

$$\theta_k(t) = \theta_k^i + \frac{\theta_k^{i+1} + 2\pi M_k - \theta_k^i}{t_s^{i+1} - t_s^i} t , \quad t_s^i \le t < t_s^{i+1} . \tag{24.12}$$

Having determined the instantaneous values of the harmonic amplitudes and phases the estimated speech signal (a harmonic representation of the speech signal) is then obtained by:

$$\hat{s}(t) = \sum_{k=0}^{K} A_k(t) \cos\left[\theta_k(t)\right] , \tag{24.13}$$

where $A_k(t)$ is given by (24.8) and $\theta_k(t)$ by (24.12).

Based on the source–filter model various speech modification methods can now be defined. Some of these refer only to the source signal, others only to the filter, while others apply to both the source and filter. Moreover, by developing the source–filter model in the context of the harmonic representation of speech signals, a mathematical notation regarding these modifications can be introduced that will be used throughout the rest of the chapter.

## 24.3 Definitions

### 24.3.1 Source Modifications

Modifications in the source signal are usually referred to as *prosodic modifications* and include three main types: time-scale modification, pitch modification, and intensity modification.

#### Time-Scale Modification

The goal of time-scale modification is to change the apparent rate of articulation without affecting the perceptual quality of the original speech. This requires the pitch contour to be stretched or compressed in time, and the formant structure to be changed at a slower or faster rate than the rate of the input speech, but otherwise not be modified. Figure 24.1 shows an example of time stretching where the pitch period contour is slowed down but not modified.

#### Pitch Modification

The goal of pitch modification is to alter the fundamental frequency in order to compress or expand the spacing between the harmonic components in the spectrum while preserving the short-time spectral envelope (the locations and bandwidths of the formants) as well as the time evolution. In contrast to time-scale modifica-

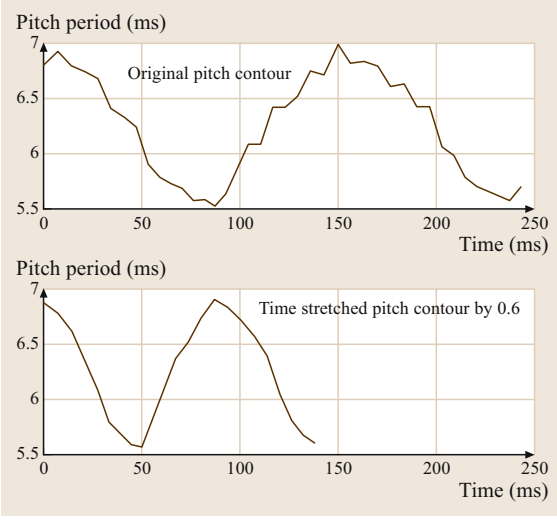



**Fig. 24.1** Pitch-period contour: original and time-stretched by a factor of 0.6



Amazon EX1010 Page 514

tions, in this case the pitch contour is modified *without* modifying, however, the time resolution of the pitch contour. Figure 24.2 shows an example of pitch modification by constant pitch-scale factor (0.6): the time evolution is preserved and the pitch-period contour is scaled by 0.6. In this case, the input fundamental frequency is increased by a factor of 1/0.6. This could give the impression that a male voice will sound more like a female voice, while a female voice will more sound like a child's voice.

#### Intensity Modification

It is widely considered that intensity modification is the simplest modification among the prosodic modifications. This is because it can be easily performed by associating an intensity scale factor at each analysis time instant of a signal. The signal is then just multiplied by this scale factor. In the case of a parametric model like the harmonic model developed previously, the scale factor is applied to the harmonic amplitudes $A_k(t)$ in (24.5). It may seem strange to modify a prosodic feature by changing a parameter corresponding to the vocal-tract filter. However, it should be remembered that the filter has been considered as an LTI filter; therefore, multiplying the amplitude of the excitation signal by a constant results in multiplying the harmonic amplitudes, $A_k(t)$, by the constant.

### 24.3.2 Filter Modifications

By filter modification we mean the modification of the magnitude spectrum of the frequency response of the vocal tract system, $|H(\omega)|$. It is widely accepted that $|H(\omega)|$ carries information of speaker individuality. Representations of the magnitude spectrum (i. e., mel frequency cepstral coefficients (MFCC), line spectrum frequencies (LSF), etc.) have been used a lot in the area of speaker identification and recognition as well as for speaker normalization for robust speech recognition. Therefore, by modifying the magnitude spectrum of the vocal tract, speaker identity may be controlled. We may distinguish two types of filter modification, which are described below.

#### Without a Specific Target

In this case, the filter characteristics of a speaker are modified in a general way without having a specific target (speaker) in mind. For example, we may wish to modify the overall quality of a speech signal produced by a female speaker so that it sounds as if it had been produced by an older female speaker. Based on the source–filter theory for the production of speech

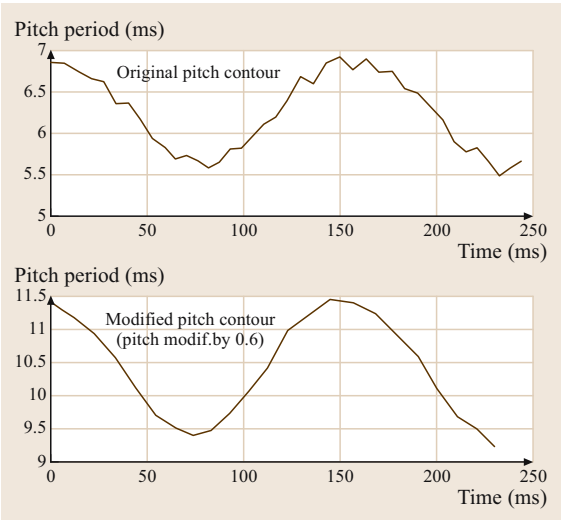



**Fig. 24.2** Pitch-period contour: original and modified by applying a pitch modification factor of 0.6

we know that the formants for a female voice are distributed in higher frequencies than the formants from an older female voice. Similar observations are valid for the harmonic frequencies. Therefore, one can modify in a general way the power spectrum of a speaker so that the resulting spectrum has the characteristics of a family of speakers (child's, old person's voices, etc.). Figure 24.3 shows an example of filter modification to modify the spectrum from a female voice to a spectrum *similar* to an old female person. For this one should compress the

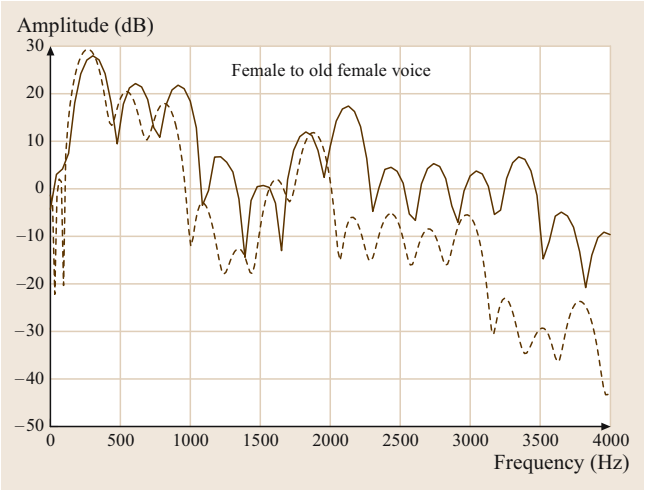



**Fig. 24.3** Female (*solid line*) to an old female (*dashed line*) filter modification

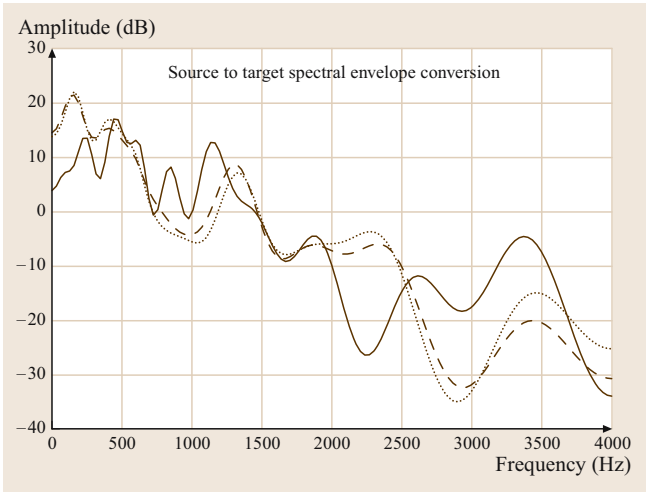



**Fig. 24.4** Source (*solid line*) to target (*dashed line*) filter transformation. The transformed filter is shown by a *dashed–dot line*

frequency information so that formants and harmonics are moved towards lower frequencies. The result shown in Fig. 24.3 was obtained by combining two operations: lowering the sampling frequency of the source signal (female voice, 44 100 Hz) to 18 000 Hz, and then applying time-scale modification with a factor of 8/9. The result signal can be played back at 16 000 Hz without any modification in the articulation rhythm. The quality of the modified signal is similar to that the initial signal.

#### With a Specific Target

In this case, the filter of a speaker (the source speaker) is modified in such a way that the modified filter approximates in the mean-squared sense the characteristics of the filter of another speaker (the target speaker). Usually we refer to this type of modification as a *transformation* or *conversion*. An example of a such transformation is depicted in Fig. 24.4. In this example the original spectrum is shown by a solid line, the target spectrum by a dashed line, and the transformed original to target spectrum by a dashed–dot line. To obtain the transformed spectrum, there is a learning process using many similar examples of the source and the target spectrum; therefore, the transformed spectrum is equal to the average spectrum of the target spectra used during the training process. Details about this type of spectral transformation will be provided in the next section.

### 24.3.3 Combining Source and Filter Modifications

To transform a female into a male voice, performing filter modification or only pitch modification alone may not provide convincing results. In most cases, source and filter modifications must be combined. For example, the prosody characteristics of a speaker may be a critical cue used for the identification of the speaker by others (speaking style) while at the same time, vocal-tract characteristics are also important for identification. Therefore, if we want to modify the voice of the speaker so that it sounds like the voice of another speaker, prosody and vocal tract modifications should be combined. If a target speaker is provided then this combined source and filter modifications is referred to as *voice conversion* or *transformation*. In contrast, when a specific target is not provided, this is usually referred to as *voice modification*.

*Voice morphing* is another type of combined source and/or filter modifications. In this case the same two sentences are uttered by two speakers and then a third speaker may be generated having characteristics from both speakers. This is mainly achieved by a dynamic time warping (DTW) algorithm between the two sentences, aligning the acoustic data and then applying a linear or other type of interpolation between the aligned data (source and/or filter characteristics). Sometimes this type of voice transformation is confused with voice conversion. Note that in voice conversion the sentence to be converted, uttered by the source speaker, *has never been uttered* by the target speaker. However, in voice morphing there are two source speakers that generate a new voice saying the same text as the two source voices. In voice conversion, there is only one source speaker and one target speaker, and the voice characteristics of the source speaker should be transformed into the voice characteristics of the target speaker (i. e., a new speaker is not generated in this case).

In the next section we will provide details about the main prosodic (time and pitch) modifications and the filter modifications. Then a system for voice conversion will be presented.

## 24.4 Source Modifications

*Pitch synchronous* analysis is the key to the simplicity of many source (prosodic) modification algorithms and is defined as follows. Given an analysis time instant, $t_a^i$, the next analysis time instant, $t_a^{i+1}$ is determined by the



Amazon EX1010 Page 516

local pitch period at $t_a^i$, $P(t_a^i)$, using (24.7). The length of the analysis window is proportional to the local pitch period (usually two local pitch periods are used). We may distinguish two types of pitch synchronous analysis. The first one may be referred to as *strict* pitch synchronous analysis, where the analysis time instants are supposed to coincide with the glottal closure instants (GCIs, sometimes called *pitch marks*). In the other, referred to as *relaxed* pitch synchronous analysis, the analysis time instants do not (necessarily) coincide with the GCIs. Since the estimation of pitch marks from the speech signal is not a robust process, resulting sometimes in an incoherent synthesis, relaxed pitch synchronous analysis seems to be easier to use than the fixed approach. However, this is not true. Pitch modification requires the re-estimation of phases. Coherent synthesis mainly means synthesis without linear phase mismatches. Strict pitch synchronous methods explicitly remove any linear phase mismatch between successive frames by using GCIs. Relaxed pitch synchronous methods, however, need to re-estimate the linear phase component for the new pitch values, which is not a trivial task. Phase models [24.12] and estimation of phase envelopes [24.11] try to overcome these problems.

For the system presented here, we will consider that the analysis time instants (Sect. 24.2.2) have been determined in a relaxed pitch synchronous way.

Next, we will see how the pitch synchronous scheme allows the use of simple and flexible techniques for time-scale and pitch-scale modifications. The first step consists of finding out the synthesis time instants $t_s^i$ (or synthesis pitch marks) according to the desired time-scale and pitch-scale modification factors. The modified signal is then obtained by using the new synthesis time instants.

### 24.4.1 Time-Scale Modification

We recall that the objective of time-scale modification is to alter the apparent rate of articulation without affecting the spectral content: the pitch contour and time evolution of the formant structure should be time scaled, but otherwise not modified [24.15].

From the stream of analysis time instants $t_a^i$ and the desired time-scale modification factor $\beta(t)$, $(\beta(t) > 0)$ the synthesis time instants $t_s^i$ will be determined. The mapping $t_a^i \rightarrow t_s^i = D(t)$ is referred to as the time-scale warping function, which is defined as the integral of $\beta(t)$:

$$D(t) = \int_0^t \beta(\tau)\,\mathrm{d}\tau \ . \tag{24.14}$$

Note that for a constant time modification rate $\beta(t) = \beta$, the time-scale warping function is linear: $D(t) = \beta t$. The case $\beta > 1$ corresponds to slowing down the rate of articulation by means of a time-scale expansion, while the case $\beta < 1$ corresponds to speeding up the rate of articulation by means of a time-scale compression. Thus, speech events that take place at a time $t_{or}$ in the original time scale will occur at a time $t_{mo} = \beta t_{or}$ in the new (modified) time scale.

As an example, let us assume that at each analysis tim -instant $t_a^i$ a time-scale modification factor $b_s$ has been specified. Thus, $\beta(t)$ is a piecewise constant function, i. e., $\beta(t) = \beta_s$, $t_a^i \le t < t_a^{i+1}$. It follows therefore that the time-scale warping function $D(t)$ can then be written

$$D(t) = D(t_a^i) + \beta_s\left(t - t_a^i\right) \ , \quad t_a^i \le t < t_a^{i+1} \tag{24.15}$$

with $D(t_a^1) = 0$.

Having specified the time-scale warping function $D(t)$, the next step consists of generating the stream of the synthesis time instants $t_s^i$, *while preserving the pitch contour*: the pitch in the time-scaled signal at time $t$ should be as close as possible to the pitch in the original signal at time $D^{-1}(t)$. In other words, $t \rightarrow P'(t) = P[D^{-1}(t)]$. We now have to find a stream of synthesis pitch marks (synthesis time instants) $t_s^i$, such that $t_s^{i+1} = t_s^i + P'(t_s^i)$. To solve this problem, the use of a stream of *virtual pitch marks*, $t_v^i$, in the original signal related to the synthesis pitch-marks by

$$\begin{aligned} t_s^i &= D\left(t_v^i\right) \ , \\ t_v^i &= D^{-1}\left(t_s^i\right) \ , \end{aligned} \tag{24.16}$$

is proposed in [24.15]. Assuming that $t_s^i$ and $t_v^i$ are known, we determine $t_s^{i+1}$ (and $t_v^{i+1}$), such that $t_s^{i+1} - t_s^i$ is approximately equal to the pitch in the original signal at time $t_v^i$. This can be expressed as

$$t_s^{i+1} - t_s^i = \frac{1}{t_v^{i+1} - t_v^i} \int_{t_v^i}^{t_v^{i+1}} P(t)\,\mathrm{d}t \tag{24.17}$$

with $t_s^{i+1} = D(t_v^{i+1})$. According to this equation, the synthesis pitch period $t_s^{i+1} - t_s^i$ at time $t_s^i$ is equal to the mean value of the pitch in the original signal calculated over the time interval $t_v^{i+1} - t_v^i$. Note that this time interval $t_v^{i+1} - t_v^i$ is mapped to $t_s^{i+1} - t_s^i$ by the mapping function $D(t)$.

The integral equation (24.17) is easily solved because $D(t)$ and $P(t)$ are piecewise linear functions. Figure 24.5 illustrates an example of the computation of synthesis pitch marks for time-scale modification by 1.5.

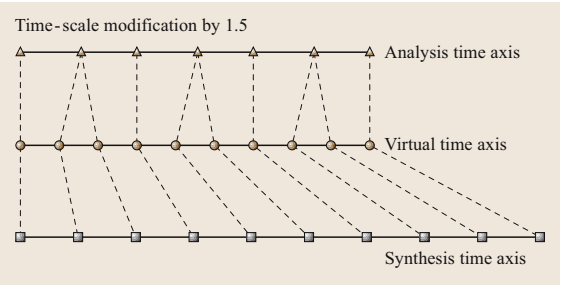



**Fig. 24.5** Computation of the synthesis pitch marks for time-scale modification by 1.5

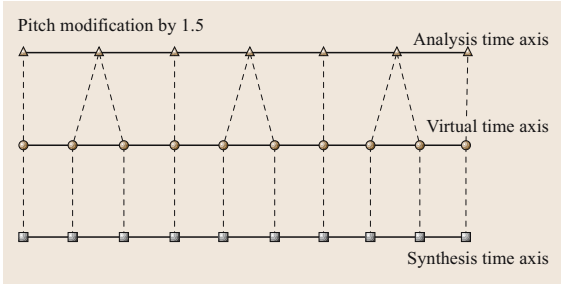



**Fig. 24.6** Computation of the synthesis pitch marks for pitch modification by 1.5

### 24.4.2 Pitch Modification

The goal of the pitch-scale modification is to alter the fundamental frequency of a speaker while the spectral envelope of the speaker's vocal-tract system function is unchanged. Obviously, pitch modification is only applied on the voiced speech frames. The first step consists of computing the synthesis time instants $t_s^i$ from the stream of the analysis time instants $t_a^i$ and the pitch-scale modification factors $a(t)$, with $a(t) > 0$. We recall that the analysis time instants are set in a pitch synchronous way. We require the same for the synthesis time instants: $t_s^{i+1} = t_s^i + P'(t_s^i)$, where $P'(t_s^i)$ is approximately equal to the pitch period in the original signal around time $t_a^i$ scaled by $1/\alpha(t_a^i)$:

$$P'\left(t_s^i\right) = \frac{P\left(t_a^i\right)}{\alpha\left(t_a^i\right)} \,. \tag{24.18}$$

Given the synthesis time instant $t_s^i$, the next synthesis time instant $t_s^{i+1}$ is obtained by setting the synthesis pitch period to be equal to the mean value of the scaled pitch period (by $1/\alpha(t_a^i)$) in the original signal calculated over the time frame $t_s^{i+1} - t_s^i$:

$$t_s^{i+1} - t_s^i = \frac{1}{t_s^{i+1} - t_s^i} \int_{t_s^i}^{t_s^{i+1}} \frac{P(t)}{\alpha(t)} \mathrm{d}t \,. \tag{24.19}$$

This integral equation is easily solved as $P(t)$ is a piecewise linear function and $\alpha(t)$ is a piecewise constant function:

$$\alpha(t) = a\left(t_a^i\right) \quad \text{for } t_a^i \le t < t_a^{i+1} \,. \tag{24.20}$$

Figure 24.6 shows an example of a mapping between the analysis and synthesis time instants for pitch modification by 1.5.

### 24.4.3 Joint Pitch and Time-Scale Modification

Based on the procedures presented above, joint pitch and time-scale modifications can easily be obtained. Given a pitch and time-scale modification factor at each analysis time instant and combining (24.17) and (24.19), the synthesis time instants can be obtained by solving the following integral equation

$$t_s^{i+1} - t_s^i = \frac{1}{t_v^{i+1} - t_v^i} \int_{t_v^i}^{t_v^{i+1}} \frac{P(t)}{\alpha(t)} \mathrm{d}t \,. \tag{24.21}$$

The synthesis time instants determined by the procedures above are not, in general, univocally associated with the analysis time instants (see for example Figs. 24.5 and 24.6). A solution consists of replacing the virtual pitch marks by the nearest analysis time instant. In this case, however, another problem arises: two or more successive frames could be the same and then the harmonic parameters (amplitudes and phases) will not vary within the frame, meaning that a high-quality modified synthetic signal is not produced. It follows therefore that the repeated analysis time instants should be elimi-

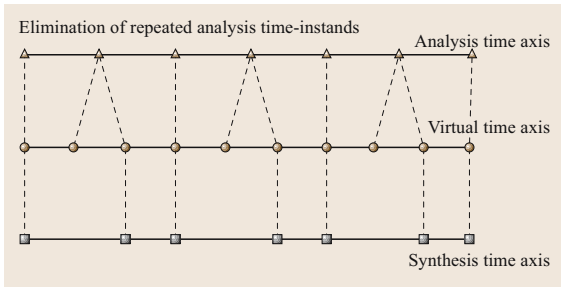



**Fig. 24.7** Elimination of repeated analysis time instants from the example in Fig. 24.6



Amazon EX1010 Page 518

nated. As an example, in Fig. 24.7 the repeated analysis time instants in Fig. 24.6 are eliminated.

### 24.4.4 Energy Modification

Energy modification is performed by associating an intensity scale factor, $c(t_s^i)$, at each synthesis time instant. Then, the harmonic amplitudes are multiplied by the square root of the current harmonic intensity scale factor

$$A'_k(t_s^i) = \sqrt{c(t_s^i)}\,A_k(t_s^i) \quad \text{for } k = 1, \ldots, K\,, \tag{24.22}$$

where $K$ is the number of harmonics. The modified amplitudes should then be used in estimating the amplitude envelope (Sect. 24.2.2).

### 24.4.5 Generating the Source Modified Speech Signal

In synthesis, the first step is the computation of the harmonic amplitudes and phases at the shifted harmonic frequencies. Note that in the case of time-scale modifications, the original amplitudes and phase may be preserved. In general, therefore, the amplitudes and phases are obtained by sampling the phase and amplitude envelopes at the corresponding harmonic frequencies. In the case of pitch modification, the phase and amplitude envelope should be sampled using the modified fundamental frequency: $f'_0(t_a^i) = \alpha(t_a^i) f_0(t_a^i)$. Given a spectrum, this results in a different number of harmonics from those initially included in the spectrum. When $\alpha(t_a^i) > 1$, fewer harmonics are included in the spectrum, and when $\alpha(t_a^i) < 1$, more harmonics are included. This means that the initial energy of the signal will be changed. Therefore, the amplitudes of the shifted harmonics are normalized in such a way that the final energy of the pitch-modified signal is equal to the energy of the unmodified one.

Using the new synthesis time instants and the modified set of parameters (amplitudes and phases) corresponding to each time instant, the source modified speech signal is obtained in exactly the same way as shown in Sect. 24.2.3. Examples of time-scale modification and pitch modification, both using a modification factor of 1.3, are depicted in Figs. 24.8 and 24.9, respectively.

An alternative to the parametric harmonic model for source modifications is the time-domain pitch synchronous overlap add (TD-PSOLA) [24.16]. TD-PSOLA relies heavily on the source–filter speech production model, although the parameters of this model are not estimated explicitly. TD-PSOLA is characterized by simplicity and low computational complexity, allowing good-quality prosodic modifications of speech. It is widely adopted for text-to-speech synthesis based on concatenation of acoustic units like diphones. Other methods similar to TD-PSOLA have also been proposed, including multiband resynthesis

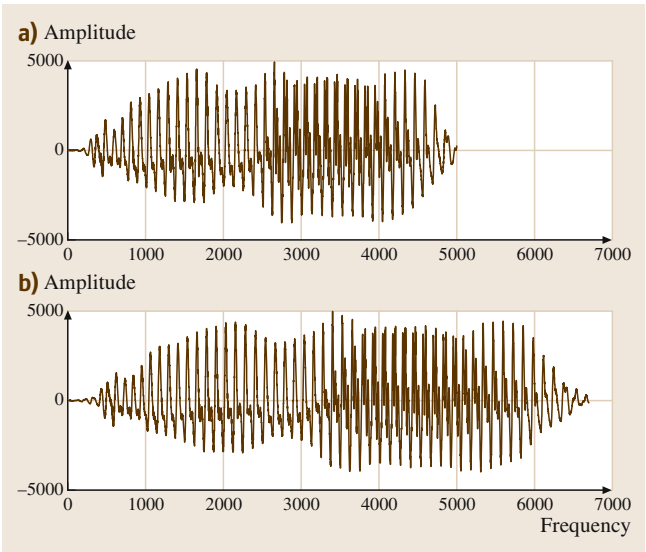



**Fig. 24.8** **(a)** Original speech signal. **(b)** Time-scaled by 1.3. Time is provided in samples (sampling frequency: 16 kHz)

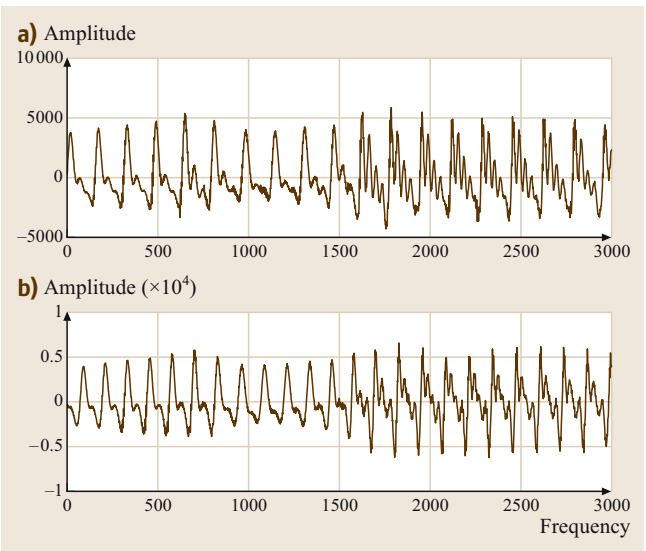



**Fig. 24.9** **(a)** Original speech signal. **(b)** Pitch modified by 1.3. Time is provided in samples (sampling frequency: 16 kHz)

overlap add (MBROLA) [24.1]. The synchronized overlap add (SOLA) [24.17] and the waveform similarity overlap add (WSOLA) [24.18] methods have mainly been proposed for time-scale modification. In SOLA and WSOLA, successive speech frames to be overlapped are cross-correlated, providing the time shift to ensure that the two overlapping frames are synchronized and thus add coherently. TD-PSOLA relies on GCIs (pitch marks for voice sounds) to synchronize the speech frames. The use of pitch marks allows TD-PSOLA to apply a simple mechanism for pitch modification; this is not possible for the SOLA and WSOLA techniques. Examples of how to apply TD-PSOLA for speech modifications using the mapping of analysis and synthesis time instants are provided in Chap. 19.

Nonparametric approaches such as TD-PSOLA, SOLA, and WSOLA, do not allow complex modifications of the signal, such as increasing the degree of friction, or changing the amplitude and phase relationships between the pitch harmonics. Another major drawback is the manipulation of *noise-like* sounds presented in speech. For example, TD-PSOLA eliminates/duplicates short-time waveforms extracted from the original speech signal by windowing. When this approach is applied to unvoiced fricatives a tonal noise is produced because the repetition of segments of a *noise-like* signal produces an artificial long-time autocorrelation in the output signal, perceived as some sort of periodicity [24.15]. A simple solution to this problem consists of reversing the time-axis whenever the TD-PSOLA algorithm needs to duplicate *unvoiced* short-time signals [24.15]. This solution *reduces* the undesirable correlation in the output signal but the tonal quality does not completely disappear. This solution cannot be applied when the time-scale factor is greater than 2. Moreover, this solution cannot be used when voiced fricative frames are processed.

On the other hand, sinusoidal models have been found to be an efficient representation of voiced speech [24.5]. For a flexible representation of the unvoiced sounds and for high-quality pitch and time-scale modification of speech, hybrid [i. e., harmonic plus noise (HNM) [24.6]] models are more suitable.

## 24.5 Filter Modifications

Next, we will consider the case of filter modification *with* a specific target. This provides a more-general framework for filter modification than without a specific target, since it has a higher time resolution; the modification filter should be changed faster than in the case where a nonspecific target is provided. In this context, a set of source and target spectral envelopes is assumed given that an appropriate representation of the vocal tract spectral envelope is provided (i. e., using cepstral coefficients, line spectrum frequencies, mel frequency cepstral coefficients). To convert the source spectral envelope to the target spectral envelope, a training (or learning step) is necessary. During this step, a conversion function is trained. For this purpose, the source and the target speaker utter the same sentences.

One of the earliest approaches to the filter conversion is the mapping codebook method [vector quantization (VQ)] of *Abe* et al. [24.19], which was originally introduced for speaker adaptation by *Shikano* et al. [24.20]. The basic idea of this technique is to make mapping codebooks that represent the correspondence between the two speakers. A conversion of acoustic features from one speaker to another is therefore reduced to the problem of mapping the codebooks of the two speakers [24.19]. The main shortcoming of this method is the fact that the acoustic space of the converted signal is limited to a discrete set of envelopes. To avoid the limitations of the discrete space represented by VQ, a fuzzy vector quantization (FVQ) has been proposed by *Kuwabara* et al. [24.21]. A quite different approach, also based on VQ, has been proposed by *Iwahashi* et al. [24.22] using speaker interpolation. The use of linear multivariate regression (LMR) for *mapping* one class from the VQ space of the source speaker to the *corresponding* class in the VQ space of the target speaker has been proposed by *Valbret* et al. [24.23]. In the same communication [24.23], *Valbret* et al. proposed a spectral transformation approach based on dynamic frequency warping (DFW). In LMR a simple linear transformation function for each class has been proposed, while in DFW a third-order polynomial is used. All these methods have been developed in the context of VQ. Most authors agree that the mapping codebook approach, although it provides an impressive perceptive voice conversion effect, is plagued by poor quality and lack of robustness [24.24]. Approaches based on LMR and DFW also introduce discontinuities into the spectral information, as the acoustic space of a speaker



Amazon EX1010 Page 520

is partitioned in discrete regions. DFW succeeds in moving the formant frequencies but it has little or no effect on their amplitudes and bandwidths. Mapping functions have also been proposed using more-robust modeling compared to VQ of the acoustic space of a speaker based on the Gaussian mixture model (GMM). Assuming that the source and target vectors obtained from the speakers acoustic space are jointly Gaussian, a continuous probabilistic mapping function based on GMM has been proposed [24.25, 26]. A similar mapping function has been proposed by *Kain* et al. [24.27], jointly modeling the source and target vectors with GMM.

All of these techniques are based on parallel training data, where both the source and target speaker utter the same sentence. Then, a DTW is used to align the two signals in time in order to extract the aligned source and target training vectors. Approaches without the requirement of parallel data have also been proposed in the literature [24.28, 29]. However, using the same mapping functions for parallel and nonparallel data, it has been shown that training with parallel data provides better conversion results [24.28].

In the following, we will present a state-of-the-art system for filter modification (spectral conversion) based on GMM making use of parallel data [24.26].

### 24.5.1 The Gaussian Mixture Model

The Gaussian mixture density is a weighted sum of $m$ component densities and given by the equation

$$p(\boldsymbol{x}|\Theta) = \sum_{i=1}^{m} \alpha_i \, p_i(\boldsymbol{x}|\theta_i) \,, \tag{24.23}$$

where $\boldsymbol{x} = [x_1 \; x_2 \; x_3 \; \cdots \; x_p]^{\mathrm{T}}$ is a $p$-by-1-dimensional random vector, $p_i(\boldsymbol{x}|\theta_i)$, for $i = 1, \cdots, m$, are the component densities and $\alpha_i$ are the mixture weights. Each component density, $p_i(\boldsymbol{x}|\theta_i)$, is a $p$-dimensional normal distribution

$$p_i(\boldsymbol{x}|\theta_i) = N(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{24.24}$$

with $\boldsymbol{\mu}_i$ the $p$-by-1 mean vector and $\boldsymbol{\Sigma}_i$ the $p$-by-$p$ covariance matrix. The mixture weights, $\alpha_i$, are normalized positive scalar weights ($\sum_{i=1}^{m} \alpha_i = 1$ and $\alpha_i \geq 0$). This ensures that the mixture is a true probability density function (PDF). The complete Gaussian mixture density is parameterized by the mixture weights, the mean vectors and the covariance matrices from all component densities, which is represented by the notation,

$$\Theta = (\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \,, \quad i = 1, \cdots, m \,. \tag{24.25}$$

The Gaussian mixture model (GMM) is a classic parametric model used in many pattern-recognition techniques [24.30] and speech applications such as speaker recognition [24.31]. In the GMM context, a speaker's voice is characterized by $m$ acoustic classes representing some broad phonetic events, such as vowels, nasal or fricatives. The probabilistic modeling of an acoustic class is important since there is variability in features coming from the same class due to variations in pronunciation and co-articulation. Thus, the mean vector $\boldsymbol{\mu}_i$ represents the average features for the acoustic class $\omega_i$, and the covariance matrix $\boldsymbol{\Sigma}_i$ models the variability of features within the acoustic class.

GMM parameters are usually estimated by a standard iterative parameter estimation procedure, which is a special case of the *expectation-maximization* (EM) algorithm [24.32, 33]. Initialization of the algorithm may be provided by VQ.

## 24.6 Conversion Functions

Let $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ be a set of $p$-dimensional vectors corresponding to the spectral envelopes of the source and the target speaker, respectively, where $t = 1, \ldots, n$. It is therefore assumed that these vectors have been obtained by speech samples from both speakers that have been aligned in time using a classic dynamic time-alignment algorithm [24.34]. We also assume that a Gaussian mixture model ($\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$, for $i = 1, \ldots, m$) has been fitted to the source vectors ($\boldsymbol{x}_t, t = 1, \ldots, n$).

It is worth noting that, if we take the limit case where the GMM is reduced to a single class and if the source vectors $\boldsymbol{x}_t$ follow a Gaussian distribution $N(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and that the source and target vectors are jointly Gaussian, the minimum mean-square error estimate of the target vector is given by [24.35]

$$E[\boldsymbol{y}|\boldsymbol{x} = \boldsymbol{x}_t] = \boldsymbol{\nu} + \boldsymbol{\Gamma}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}) \,, \tag{24.26}$$

where $E[]$ denotes expectation, and $\boldsymbol{\nu}$ and $\boldsymbol{\Gamma}$ are, respectively, the mean target vector

$$\boldsymbol{\nu} = E[\boldsymbol{y}] \,,$$

and the cross-covariance matrix of the source and target vectors

$$\boldsymbol{\Gamma} = E[(\boldsymbol{y} - \boldsymbol{v})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}] ,$$

where the superscript T denotes transposition [24.36].

In [24.27], a direct extension of (24.26) to the GMM case was proposed. However, such an extension is not supported by the theory of statistics. Moreover, such an extension makes the assumption of a one-to-one correspondence between the source and target space, which is not valid in practice. To overcome these difficulties, and motivated by (24.26) the following conversion function between the source and the target data has been proposed [24.25]:

$$\mathcal{F}(\boldsymbol{x}_t) = \sum_{i=1}^{m} P(\omega_i|\boldsymbol{x}_t)\left[\boldsymbol{v}_i + \boldsymbol{\Gamma}_i \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_i)\right] . \tag{24.27}$$

The conversion function $\mathcal{F}$ is entirely defined by the $p$-dimensional vectors $\boldsymbol{v}_i$ and the $p$-by-$p$ matrices $\boldsymbol{\Gamma}_i$, for $i = 1, \ldots, m$ (where $m$ is the number of mixture components). This means that $\boldsymbol{v}_i$ and $\boldsymbol{\Gamma}_i$ are the parameters to be estimated. The parameters of the conversion function are computed by least squares optimization on the learning data so as to minimize the total squared conversion error

$$\epsilon = \sum_{t=1}^{n} ||\boldsymbol{y}_t - \mathcal{F}(\boldsymbol{x}_t)||^2 . \tag{24.28}$$

Since the conversion function given by (24.27) is linear, the optimization of its parameters is equivalent to the resolution of a set of linear equations in the least-squares sense. Details of the minimization of (24.28) may be found in [24.37]. The mapping function (24.27) can be used with full or diagonal covariance matrices. Note that the conversion function is reduced to

$$\mathcal{F}(\boldsymbol{x}_t) = \sum_{i=1}^{m} P(\omega_i|\boldsymbol{x}_t)\boldsymbol{v}_i \tag{24.29}$$

if the correction term that depends on the difference between the source vector $\boldsymbol{x}_t$ and the mean of the GMM component $\boldsymbol{\mu}_i$ in (24.27) is omitted. This reduced conversion function is similar to the formula proposed by *Abe* et al. [24.19] in the mapping codebook approach. Comparing (24.29) and (24.27) it follows that in VQ-type mapping functions the variability of the transformed spectral envelope is strongly restricted.

An example of filter conversion based on the approach described in this section has already been presented in Fig. 24.4.

## 24.7 Voice Conversion

Combining source and filter modifications a system that controls speaker's quality and individuality can be obtained. Continuing with the harmonic model as an example of representing speech, a speech signal produced by the source speaker is analyzed in a pitch-synchronous way (Sect. 24.2.1) to extract a set of source spectral envelopes. Given the spectral conversion function in (24.27), the target spectral envelopes in each frame are estimated by applying (24.27) to the source spectral envelopes. The next step is to apply the appropriate prosodic or source modifications in order to capture the *prosodic patterns* of the target speaker (Sect. 24.4). For this, prosodic profile analysis (i. e., characteristic articulation rate, stress, and emotions, pitch fluctuations) is required for both speakers. Determining such a profile is not a trivial task and has not yet been achieved in a convincing way. Thus, most voice conversion systems today make use

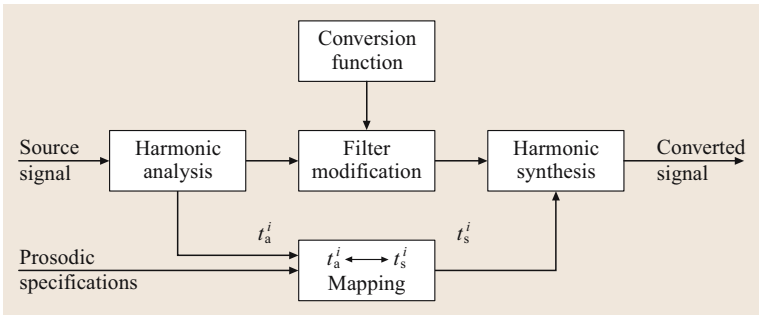



**Fig. 24.10** Block diagram of a voice conversion system; $t_a^i$ and $t_s^i$ represent the analysis and synthesis time instants, respectively



Amazon EX1010 Page 522

of average prosodic modification factors. A block diagram for the conversion system based on the harmonic model and the voice conversion function is presented in Fig. 24.10.

## 24.8 Quality Issues in Voice Transformations

Voice transformations are usually evaluated in subjective tests. The overall impression from the results obtained from these tests is that time-scale modification is quite successful for moderate scale factors, while pitch-modified signals by pitch scale factors over 1.5 and below 0.7, suffer from various artifacts, making listeners classify the modifications as not natural. Voice conversion reaches a high score for transforming the identity of the source speaker to that of the target speaker. However, there are serious quality problems, which are mostly referred to as *muffling effects*. To improve the quality of the speech produced by various proposed voice transformation algorithms a better understanding of speech production and perception mechanisms is necessary. For example, when we want to increase the loudness of our voice while sitting in a cafeteria, we add stress to a *part* of our speech signal, like consonants, and *not* to all the speech events we produce. One hypothesis for this is that consonants carry more of the information load, which is connected with the intelligibility of the message we would like to transmit. According to this hypothesis we only increase the stress to these sounds by an amount that is sufficient to mask the cafeteria noise. Increasing the stress does not mean that the amplitudes of all the frequencies for this sound are increased. Stress means an increase of the subglottal pressure, which will result in an abrupt glottal closure by accentuating the Bernoulli effect on airflow through the glottis [24.38]. This corresponds to more energy *mostly* at high frequencies. From this example, it is obvious that even a simple intensity modification is not as simple as we thought. Continuing the above example, the increase of the subglottal pressure will increase the tension in the vocal folds, resulting in an increase of the pitch. This shows that modifying one parameter may require the modification of another as well.

In most Western languages consonants (we recall that consonants carry important information load) are shorter in duration than vowels (which carry more-prosodic information). Our perceptual system requires some time to process the perceived sounds. When we want to speak faster, we somehow *protect* the consonants. *Pickett* [24.39] has done extensive studies on the degree of change in vowels and consonants in speaking at a faster or slower rate. In [24.39], it was reported that, when going from normal to the faster rate, the vowels were compressed by 50% while the consonants were compressed by 26%. However, going from the slowest to the fastest rate, both vowels and consonants were compressed by about 33% [24.38]. This shows that time-scale modifications should take into account phonetic information. Speaking at faster or slower rate again introduces modifications in pitch values since there are fluctuations in the subglottal pressure. This means that time-scale modifications should be performed jointly with pitch modifications.

In the source–filter theory presented at the beginning of the chapter, it was assumed that glottal airflow source is not influenced by the vocal tract. In reality, there is a nonlinear coupling between the source and the filter. Results from studies on the fine structure of the glottal airflow derivative waveform show that an increase in the first-formant bandwidth and modulation of the first-formant frequency occurs during the glottal open phase [24.40]. Obviously, when pitch modification is applied, these interactions should be respected.

Attempts have been made to incorporate some of these observations into the modification algorithms. In [24.41], a higher intelligibility score was achieved for time-scale-modified speech signals when nonstationarity measurements in the signal were taken into account. In [24.42], the interaction between pitch and spectral envelopes was modeled in a statistical way. This was used to postprocess pitch-modified signals. Perceptual tests have shown that this postprocessing improved the naturalness of the pitch-modified signal.

To improve further the quality of voice transformations more effort should be made taking into account nonlinear phenomena during the production process and results from the natural language processing area. In other words, voice transformation requires more than just modeling of the speech signal; it requires *understanding* of the speech process (production, perception, and language).

## 24.9 Summary

In this chapter, we have described voice transformations through a simple harmonic representation of speech. We began with the description of the basic source–filter theory for the production of speech and providing a mathematical description of speech production using this theory in the context of a harmonic model. We used this description to define voice transformation by specifying modifications for the source, for the filter, and their combination. We then provided formal definitions of these modifications and their application in the context of the harmonic model was also derived and a set of conditions for the pitch synchronous analysis of speech was described. Techniques for filter modifications were discussed and a state-of-the-art method based on a GMM description of the acoustic space of a speaker was developed. A mapping function that made use of the complete description of each component of the GMM was provided. Finally, we discussed speech quality issues related to voice transformations and noted that, for improving speech quality in the future, we need to give more realism to the source–filter model by taking into account the nonlinear coupling between the source and filter and to processes related to our perception system. In other words, just modeling the speech *signal* may be enough for transmission through the networks, but it is not sufficient for modifying the signal in a way that is perceived by humans to be natural. For this, we need to *understand* speech and then *develop* algorithms able to incorporate this understanding.

### References


24.1 T. Dutoit: *An Introduction to Text-to-Speech Synthesis* (Kluwer Academic, Dordrecht 1997)

24.2 J.D. Markel, A.M. Gray: *Linear Prediction of Speech* (Springer, Berlin, Heidelberg 1976)

24.3 B. Atal, J. Remde: A new model of LPC excitation for producing natural-sounding speech at low bit rates, Proc. IEEE ICASSP, Vol. 7 (1982) pp. 614–617

24.4 M.R. Schroeder, B.S. Atal: Code-excited linear prediction (CELP): High-quality speech at very low bit rates, Proc. IEEE ICASSP, Vol. 10 (1985) pp. 937–940

24.5 R.J. McAulay, T.F. Quatieri: Speech analysis/synthesis based on a sinusoidal representation, IEEE ICASSP **34**, 744–754 (1986)

24.6 Y. Stylianou: Modeling speech based on harmonic plus noise models. In: *Nonlinear Speech Modeling and Applications*, ed. by G. Chellot, A. Esposito, M. Faundez (Springer, Berlin, Heidelberg 2005) pp. 375–383

24.7 A.V. Oppenheim, R.W. Schafer: Homomorphic analysis of speech, IEEE Trans. Audio Electroacoust. **16**, 221–228 (1968)

24.8 D.B. Paul: The spectral envelope estimation vocoder, IEEE ICASSP **29**, 786–794 (1981)

24.9 O. Cappé, E. Moulines: Regularization techniques for discrete cepstrum estimation, IEEE Signal Process. Lett. **3**(4), 100–102 (1996)

24.10 A.V. Oppenheim, R.W. Schafer: *Discrete-Time Signal Processing* (Prentice Hall, Englewood Cliffs 1989)

24.11 Y. Stylianou, J. Laroche, E. Moulines: High-quality speech modification based on a harmonic + noise model, Proc. Eurospeech, Vol. 95 (1995) pp. 451–454

24.12 T.F. Quatieri, R.J. McAulay: Shape invariant time-scale and pitch modification of speech, IEEE ICASSP **40**, 497–510 (1992)

24.13 R.J. McAulay, T.F. Quatieri: Low-rate speech coding based on the sinusoidal model. In: *Advances in Speech Signal Processing*, ed. by S. Furui, M. Sondhi (Marcel Dekker, New York 1991) pp. 165–208, Chap. 6

24.14 L. Almeida, F. Silva: Variable-frequency synthesis: An improved harmonic coding scheme., Proc. IEEE ICASSP, Vol. 9 (1984) pp. 437–440

24.15 E. Moulines, J. Laroche: Techniques for pitch-scale and time-scale transformation of speech. Part I. Non parametric methods, Speech Commun. **16**, 175–205 (1995)

24.16 E. Moulines, F. Charpentier: Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones, Speech Commun. **9**, 453–467 (1990)

24.17 S. Roucos, A. Wilgus: High-quality time-scale modification of speech, Proc. IEEE ICASSP (1985) pp. 493–496

24.18 W. Verhelst, M. Roelands: An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech, Proc. IEEE ICASSP, Vol. 2 (1993) pp. 554–557

24.19 M. Abe, S. Nakamura, K. Shikano, H. Kuwabara: Voice conversion through vector quantization, Proc. IEEE ICASSP, Vol. 1 (1988) pp. 655–658

24.20 K. Shikano, K. Lee, R. Reddy: Speaker adaptation through vector quantization, Proc. IEEE ICASSP, Vol. 11 (1986) pp. 2643–2646




Amazon EX1010 Page 524

24.21 H. Kuwabara, Y. Sagisaka: Acoustic characteristics of speaker individuality: Control and conversion, Speech Commun. **16**(2), 165–173 (1995)

24.22 N. Iwahashi, Y. Sagisaka: Speech spectrum transformation based on speaker interpolation, Proc. IEEE ICASSP, Vol. 1 (1994) pp. 461–464

24.23 H. Valbret, E. Moulines, J. Tubach: Voice transformation using PSOLA techinques, Speech Commun. **11**(2-3), 175–187 (1992)

24.24 H. Mizuno, M. Abe: Voice conversion algorithm based on piecewise linear conversion rule of formant frequency and spectrum tilt, Speech Commun. **16**, 153–164 (1995)

24.25 Y. Stylianou, O. Cappé, E. Moulines: Statistical methods for voice quality transformation, Proc. Eurospeech, Vol. 95 (1995) pp. 447–450

24.26 Y. Stylianou, O. Cappé, E. Moulines: Continuous probabilistic transform for voice conversion, IEEE Trans. Speech Audio Process. **6**(2), 131–142 (1998)

24.27 A. Kain, M. Macon: Spectral voice conversion for text-to-speech synthesis, Proc. IEEE ICASSP, Vol. 5 (1998) pp. 285–288

24.28 A. Mouchtaris, J.V. derSpiegel, P. Mueller: Non parallel training for voice conversion based on a parameter adaptation, IEEE Trans. Audio Speech Language Process. **14**(3), 952–963 (2006)

24.29 D. Suendermann, H. Hoege, A. Bonafonte, H. Ney, A. Black, S. Narayanan: Text-independent voice conversion based on unit selection, Proc. IEEE ICASSP, Vol. 1 (2006) pp. 81–84

24.30 R.O. Duda, P.E. Hart: *Pattern Classification and Scene Analysis* (Wiley, New York 1973)

24.31 R.C. Rose, D.A. Reynolds: Text independent speaker identification using automatic acoustic segmentation, Proc. IEEE ICASSP, Vol. 1 (1990) pp. 293–296

24.32 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm (methodological), J. R. Stat. Soc. B **39**(1), 1–22 (1977)

24.33 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm (discussion), J. R. Stat. Soc. B **39**(1), 22–38 (1977)

24.34 L.R. Rabiner, B.-H. Juang: *Fundamentals of Speech Recognition* (Prentice Hall, Upper Saddle River 1993)

24.35 S.M. Kay: *Fundamentals of Statistical Signal Processing: Estimation Theory*, PH Signal Process. Ser. (Prentice Hall, Upper Saddle River 1993)

24.36 C. Chatfield, A.J. Collins: *Introduction to Multivariate Analysis* (Chapman Hall, Boca Raton 1980)

24.37 Y. Stylianou: *Harmonic plus Noise Models for Speech, Combined with Statistical Methods, for Speech and Speaker Modification*, Ph.D. Thesis (Ecole Nationale Supèrieure des Télécommunications, Paris 1996)

24.38 T.F. Quatieri: *Discrete-Time Speech Signal Processing* (Prentice Hall, Englewood Cliffs 2002)

24.39 J.M. Pickett: *The Sounds of Speech Communication* (Pro-Ed, Austin 1980)

24.40 C. Jankowski: *Fine Structure Features for Speaker Identification*, Ph.D. Thesis (Massachusetts Institute of Technology, Cambridge 1996)

24.41 D. Kapilow, Y. Stylianou, J. Schroeter: Detection of non-stationarity in speech signals and its application to time-scaling, Proc. Eurospeech, Vol. 99 (1999) pp. 2307–2310

24.42 A. Kain, Y. Stylianou: Stochastic modeling of spectral adjustment for high quality pitch modification, Proc. IEEE ICASSP, Vol. 2 (2000) pp. 949–952




Amazon EX1010 Page 525

# 25. Expressive/Affective Speech Synthesis

N. Campbell

The focus of speech synthesis research has recently shifted from *read speech* towards more *conversational* styles of speech, in order to reproduce those situations where a speech synthesis is used as part of a dialogue. When a speech synthesizer is used to represent the voice of a cognisant agent, whether human or simulated, there is need for more than just the intelligible portrayal of linguistic information; there is also a need for the expression of affect. This chapter reviews some recent advances in the synthesis of expressive speech and shows how the technology can be adapted to include the display of affect in conversational speech.

The chapter discusses how the presence of an interactive and active partner in a conversation can greatly affect the styles of human speech and presents a model of the cognitive processes that result in these differences, which concern not just the acoustic prosody and phonation quality of an utterance, but also its lexical selection and phrasing. It proposes a measure of the ratio of paralinguistic to linguistic content in an utterance as a means of quantifying the expressivity of a speaking style, and closes with a description of a phrase-level concatenative speech synthesis system that is currently in development.

25.1 **Overview** ..... 505
25.2 **Characteristics of Affective Speech** ..... 506
25.2.1 Intentions and Emotions ..... 506
25.2.2 Message and Filters ..... 507
25.2.3 Coding and Expression ..... 507
25.3 **The Communicative Functionality of Speech** ..... 508
25.3.1 Multiple Layers of Prosodic Information ..... 509
25.3.2 Text Data versus Speech Synthesis . 509
25.4 **Approaches to Synthesizing Expressive Speech** ..... 510
25.4.1 Emotion in Expressive Speech Synthesis ..... 510
25.5 **Modeling Human Speech** ..... 512
25.5.1 Discourse-Act Labeling ..... 513
25.5.2 Expressive Speech and Emotion .... 513
25.5.3 Concatenative Synthesis Using Expressive Speech Samples .. 515
25.6 **Conclusion** ..... 515
**References** ..... 515

## 25.1 Overview

Expressive speech synthesis, or the synthesis of affective speech is a relatively new area of research, prompted by a shift in the technology from *reading machines* towards *talking machines* [25.1]. This arose from the need for machines to be able to express more than just the phonetic content of an utterance and to be able to make use of more human-like subtlety in the generation of its prosodic characteristics. Accordingly, speech synthesis research has shifted from *read speech* towards more *conversational* styles of speech. For those situations where speech synthesis is used as part of a dialogue, i. e., where it represents the voice of a cognisant agent, whether human or simulated, then there is a need for more than just the intelligible portrayal of linguistic information; there is also a need for the expression of affect.

Not to be confused with the simpler term 'emotion', 'affect' as defined in this context refers to the human characteristics that govern the various subtleties of intonation and phrasing, which reveal extralinguistic and paralinguistic information about the speaker, about the speaker's relationships with the hearer, and about the progress of the discourse and the degrees of mutual understanding attained throughout its progress.

*Expressive* speech is that which reveals affect. The challenge in synthesizing expressive speech lies in the adequate specification of affective states in the input and in the manipulation or realization of prosodic characteristics to express them in the output speech. This chapter will start by describing the types of speech variation that will be necessary to synthesize, discuss



Amazon EX1010 Page 526

some techniques for eliciting them, and then close by describing some of the approaches that have been taken to tackle the problems of their modeling and realization.

## 25.2 Characteristics of Affective Speech

Speech per se is a meaningful type of noise which can take place with or without a hearer, but the nature of interactive speech specifically depends on the presence and purposes of a hearer. Broadcast speech represents one extreme of a hypothetical continuum of listener involvement, with romantic murmurings or infant-directed speech perhaps forming the other; currently only the former can be well modeled for synthesis, although there are calls for a more-personal and intimate form of speaking in many current applications of speech synthesis.

Broadcast speech assumes a general audience that is perhaps interested but not necessarily present, and certainly unable to provide any immediate relevant feedback. The focus here is on a unidirectional transmission of information. Conversational speech on the other hand typically requires an active and participating listener. The focus here is on the interaction. The listener may be remote, as in the case of telephone speech, but is required to give constant and timely feedback. The nature of the conversational speech changes according to these feedback reactions from the listener. Classroom speech or lectures can be considered intermediate on this continuum, since the audience is distant but present, able to show comprehension or otherwise, allowing the speaker to moderate and modify the content and speaking style accordingly.

To summarize, affective speech differs from traditional read speech in depending on a model of the listener being incorporated into the speech production process. The task of expressive speech synthesis is first to represent the higher-level interpersonal structures that motivate the speech [25.2], and then to model their lower-level consequences through prosodic control of the output speech signal.

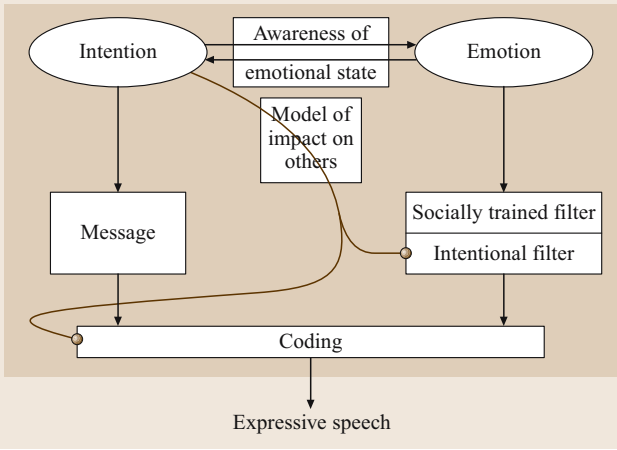



**Fig. 25.1** A proposed model to explain the interaction of affective and intentional information in the generation of a speech utterance. The ovals represent hidden processes or states that are not subject to conscious control but which serve as driving forces behind the production of the utterance. These are substantiated in the form of a message and filters, with constraints that are subject to a model of the potential impact on the listener, that determine the muscular coding for the production of the utterance with its resulting prosody and phonetic sequence

### 25.2.1 Intentions and Emotions

A representation of the cognitive processes involved in determining a specific speaking style for expressing affect-related information is given in Fig. 25.1, which schematizes the flow of activity that results in expressive speech. The interaction of emotions and intentions in the generation of an expressive speech utterance forms the higher-level part of this framework. In the figure, both *emotions* and *intentions* are represented by ovals, signifying their intangibility; i. e., that they act as a force but are not readily available for identification, control, or conscious appraisal.

The model posits these two underlying or hidden cognitive forces, intention and emotion, as drivers that provide the motivation for a basic communicative event that is to be realized in the form of an utterance in a discourse. They are distinguished from the more-tangible message and filters to be discussed in more detail below. These in turn govern a coding level of processing, which produces lower-level commands for the muscles that produce the speech and accompanying facial or bodily gestures.

A given combination of intentions and emotions represents an underlying sociopsychological state within the speaker, which is raw and unbound by linguistic or



Amazon EX1010 Page 527

social conventions. There may of course be some interaction between emotions and intentions, since intentions can be triggered by emotions, and emotions can in turn be subdued or amplified intentionally as part of a rational process, such as when a speaker forces herself to smile so that her voice will become happier. These are abstract causes that underly the more-tangible level of message and filters that particularly concerns us here, but they result in subtle differences in force of expression and need to be modeled carefully, since human listeners are particularly sensitive to their variations.

### 25.2.2 Message and Filters

The message gives form to the underlying intentions and constitutes a speech act, a discourse act, and a social event. It is not yet a text. It may be a greeting, a complaint, provision of information, request for information, etc., and may stand alone or function as a dependent element of a larger discourse. In many cases it will be as much phatic as informational in intent.

It is at the level of the message that the utterance begins to take shape, but its linguistic content and prosodic realization remain indeterminate at this level. For example, a greeting could take the form of 'Good morning', or 'Hi!', depending on who is being addressed, on the mood of the speaker, and on the contexts of the discourse (both social and environmental). These details are determined by the settings of the filters and realized at the level of the coding.

These filters are socially trainable. They depend to a large extent on language-specific, culture-specific and subculture-specific aspects, but can have an intentional element. They incorporate such modifiers as politeness constraints and serve to signal attitudinal relationships and interpersonal stances. The filters are shown as bilevel; depending both on social conditioning (above) and a degree of intentional control (below). It is at this lower level that the speaker takes into consideration the potential impact of an utterance on the listener [as illustrated (not coincidentally) in the center of the figure].

Whereas certain constraints may be ingrained, or determined by society and imprinted in the speaker at an early age, others are more open to conscious selection. For example, while young infants may readily and directly express whatever emotions they currently feel, older children and adults become more reserved, often concealing their true feelings or masking them for social reasons. A salesperson may wish to portray the proper company image, hiding particular strengths or weaknesses, or a call-center operator may be required to sound cheerful, even though the displayed emotion may be in conflict with that actually felt by the speaker at the time. This dichotomy provides part of the richness of spoken language and is surely parsed by the listener as part of (or alongside) the message.

Both filter levels function to control

1. what is displayed
2. what is concealed in the production of an utterance

They have an effect not just on the selection of lexical items and phrasing, but also on voice quality and prosodic aspects of phonation so that the utterance can be parsed appropriately as expressing the speaker's intentions subject to the prevailing social and psychological states and conditions. This requires a level of markup on the input far beyond that which is currently specified by the existing W3 speech synthesis markup language (SSML) conventions [25.3].

The sophistication of interactive human speech is a direct result of this multiplicity of cues that taken together contribute the intended interpretation of its linguistic and pragmatic content. The prosody and voice quality of expressive speech also encode subtle information related to the speaker's basic internal emotional states but masked by a more-superficial layer of information related to discourse goals and intentions, and social conventions governing politeness and sincerity of expression. The task of expressive speech processing, whether for synthesis or recognition, is to model the interrelationship of each of these sometimes conflicting sources of information.

### 25.2.3 Coding and Expression

In this model of expressive speech generation, the determination of lexical items, utterance complexity and length, phrasing, speaking rate and style, etc., is considered as taking place at the lowest level of utterance production, in a constraint-based way, subject to the higher-level constraints described above and illustrated in the main part of the figure.

Because there are usually several different ways of phrasing a proposition or eliciting backchannel information, the choice of a particular variant reveals much about the intentions and affective states of the speaker and about the contexts of the discourse. Both the text of the message and its prosodic encoding are constrained by the intentions of the speaker, subject to variations in emotional state and social as well as intentional constraints on utterance production.



Amazon EX1010 Page 528

Figure 25.1 shows the coding level, which ultimately produces sequences of muscular movements, to be fed by two streams of complementary information, as shown by the left and right vertical arrows; both are subject to a model of the impact of the utterance on the listener and others. This information can be similarly decoded to reveal not just the linguistic content, but also information about the speaker and the settings of the various filters.

## 25.3 The Communicative Functionality of Speech

The model implies that any given speech utterance contains information related not just to its propositional content, if any, but also to speaker-related and affective information, to speaker–hearer relationships, and to environmental factors etc.; i. e., in addition to the lexical content or word sequence, an utterance provides both linguistic prosody and social prosodic information [25.2, 4]. The former has a long history of being modeled in speech synthesis being predictable largely from the text (see, for example, [25.5–8]), but the latter remains a subject for continuing and intense research, being closely related to database design and corpus collection issues in the case of concatenative synthesis techniques [25.9–11].

These prosodic elements, along with related differences in voice quality, can be decoded by the listener to reveal the affective and interpersonal information that signals the speaker's position relative to the utterance and the discourse, and enables the listener to parse its intended pragmatic meaning from among the many possible linguistic candidate interpretations of the text string. The structure of conversational speech utterances (often called ill-formed) differs considerably from that of their written equivalents not because people are simple and unable to think efficiently in real time, but because speech as a medium has the benefits of multiple layers of prosodic information from which the underlying meanings and intentions of an utterance can be made clear.

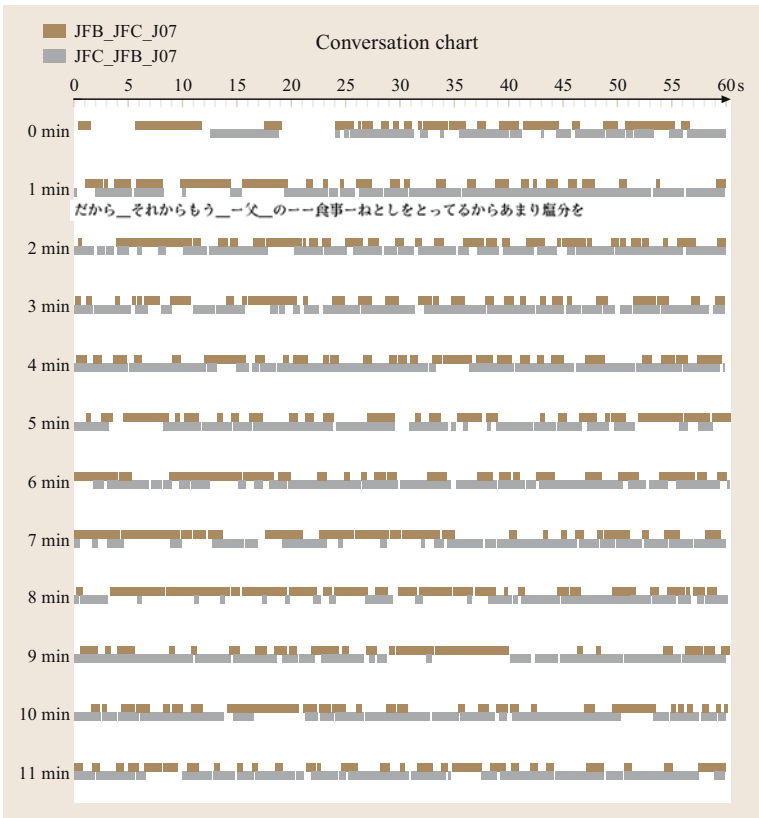



**Fig. 25.2** Time-aligned speech activity in a natural telephone conversation, showing how the dominant role is passed from one participant to the other in a not very clear-cut way. The lower speaker (grey) appears to be dominant, but there is considerable overlap as they negotiate their way through the conversation



Amazon EX1010 Page 529

Figure 25.2 shows the interplay of speaker and listener in a short telephone conversation taken from the expressive speech processing (ESP) corpus [25.12]. It is clear that the roles are often reversed, and that even in listener mode there is considerable speech activity. These backchannel feedback signals provide the speaker with cues to the listener's cognitive state, and comprehension of and agreement with the flow of the conversation as the two mutually develop a joint understanding, or meeting of the minds [25.13].

Conversational speech utterances are not just typically shorter than their written equivalents, they often contain idiomatic phrases, phatic elements, laughs, etc., that illustrate the discourse. Being so frequent and repetitive, these inessential speech noises allow the listener, even one who is unfamiliar with the speaker, to make judgements about the speaker's affective states and discourse intentions.

Whereas the true intentions and emotions of the speaker must remain hidden, much can be inferred about them from the combination of information in the message and in the choice of speaking style (i. e., from the visible effects of the inferred filters), the listener thereby has access not just to the text of the utterance, but also to:

1. intended meaning(s)
2. speaker state(s)
3. listener status and relationship(s) to the speaker.

This is what is now being covered in the developing studies of social prosody and what is yet to be modeled in expressive speech synthesis.

### 25.3.1 Multiple Layers of Prosodic Information

Since its original design as a reading machine, speech synthesis research has been focussed on the conversion of written words into speech sounds, using grapheme-to-phoneme conversion, prosody prediction, and waveform generation as its three main subprocesses.

Accordingly, there is a considerable body of prosody-related research in the field of speech synthesis, but almost all of it (with very few exceptions) is related to the forms of prosody that can be predicted from the text alone. The exceptions largely concern gender-related prosodic differences, or linguistic focus. Existing speech synthesis markup languages (e.g., [25.3]) provide for the modification of prosody, but only at the lowest level of mean pitch, phoneme duration, and amplitude. Little work has yet been done on the annotation of text for the expression of the types of affective information described above.

Traditional prosody for speech synthesis is directly related to the text structure of each utterance [25.14, 15], although nowadays it is predicted usually by use of statistical models that have been trained on speech data often unrelated to the voice or personality of the synthesis speaker (e.g., [25.16, 17]). The models provide duration, pitch, and sometimes power values for each segment of the utterance, often through an intermediate phonological representation such as ToBI [25.18, 19].

### 25.3.2 Text Data versus Speech Synthesis

Spoken language has been extensively studied through the use of corpora for several decades now, and the differences between the types of information that can be conveyed through written texts and those that are signalled through speech are beginning to be well understood. The different types of information that are signalled by different speaking styles are also well understood and are beginning to be modeled in speech technology applications, especially paralinguistic information, which is an essential component of speech communication that is largely carried through modulations of prosody, tone of voice, and speaking style. The more formal the speech, the more constrained the types of paralinguistic information that are conveyed.

At the formal extreme we might consider a public lecture where the speaker is (sometimes literally) talking from a script, to a large number of listeners (or even to a recording device with no listeners physically present) and has minimal feedback from, or two-way interaction with, the audience. This type of spontaneous speech is perhaps the most constrained, and most resembles text. At the most informal extreme, we might consider the mumblings of young lovers. Their conversation is largely phatic, and the words might carry little of linguistic content but are instead rich in feelings. For them, talk is almost a form of physical contact.

There are many steps along the continuum between these two hypothetical extremes of speaking-style variation, and they can be distinguished by the *ratio of paralinguistic to linguistic content*, i. e., the amount of personal information that is included in the speech. The lecture, having almost no personal information and a very high amount of propositional content will result in a very low value of this measure, while the phatic mutterings will score very high.



Amazon EX1010 Page 530

Speech research depends on the quality and types of the data on which it is modeled. If we are to collect data that contains sufficient examples of natural spoken interactions along the whole range of this continuum of values, then low-scoring material will prove very easy to collect, but most lovers might object strongly to the prospect of having a recording device intruding upon their privacy. Thus, by far the majority of speech corpora that have been used in previous research score very poorly on this scale and as a result the speech that they contain is not very far removed from pure text in its style and content.

## 25.4 Approaches to Synthesizing Expressive Speech

Expressive speech synthesis is covered under many patents, among them US patent no. 5 559 927 of Sept. 1996, for a *Computer system producing emotionally-expressive speech messages* which describes "a computer system in which the sounds of different speech messages are stored or synthesized, the system being adapted to reproduce a selected speech message and to impart emotional expressivity thereto whose character depends on the user's choice".

But what is emotional expressivity? Take as an example, the website of the Signal Processing Laboratory of the University of the Basque Country (UPV/EHU) in Bilbao [25.20], which announces expressive speech as its title, and which exemplifies what I believe to be a popular and very widespread misconception about expressive speech:

The page states that:

> *Expressive speech synthesis concerns the addition of features related to the emotional state of the (synthetic) speaker as well as the specific characteristics that identify the speaker to the neutral synthetic speech. As such, aspects as emotions (fear, anger, surprise, disgust, happiness, unhappiness . . . ) and voice quality features ( . . . ) should be considered when synthesizing expressive speech.*

It is often taken for granted that expressiveness in speech is the direct result of changes in emotional states, and furthermore that the emotions to be expressed are limited to happiness, sadness, fear, anger, surprise, and disgust (i. e., *Ekman*'s big six [25.21]). This may be an understandable assumption but it is one that does little justice to the sophistication of human interactive communication as described above, unless we are to limit expressive speech synthesis to the realms of storytelling, cartoons, and entertainment [25.22–24].

Ekman was attempting to explain facial expression when he published his seminal work, but the voice is different from the face, and in spoken conversation, what it reveals may not be the emotion so much as the interests of the speaker. In our analysis of a very large corpus of naturally occurring everyday conversations ([25.10, 25], see below), these strong emotions accounted for less than 1% of the expressive variability in the speech. However, they account for more than 90% of the related research in speech synthesis.

### 25.4.1 Emotion in Expressive Speech Synthesis

The modeling of expressive speech for speech synthesis perhaps started with the masters thesis of *Janet Cahn* at MIT in 1989 on generating expression in synthesized speech [25.26]. Her work resulted in an affect editor, which modified the prosodic parameters of DecTalk, changing the default settings of pitch, timing, voice quality and articulation according to the emotional states (angry, disgusted, fearful, glad, and sad) that were selected by the user.

Her first paper nearly 20 years ago [25.27] started with the words: "Synthesized speech is readily distinguished from human speech on the basis of inappropriate intonation and insufficient expressiveness", of which perhaps now only the latter part is still true, and she continued: "In synthesized speech intonation makes the message easier to understand; enhanced expressiveness contributes to dramatic effect, making the message easier to listen to," which is still completely true.

It is of particular interest to note that in the 1990 journal paper that described her thesis work [25.28], the simple list of the big six emotions is expanded (Fig. 25.3, which is Fig. 2 in her paper) to include more truly affective states: afraid, angry, annoyed, disgusted, distraught, glad, indignant, mild, plaintive, pleasant, pouting, sad, and surprised, although these were not mentioned in either the thesis or in the 1989 conference paper to the same society [25.29]. This necessary extension from emotion to affect, perhaps encouraged by a sensitive reviewer, has not yet been so readily taken up by the rest of the community of speech synthesis researchers.

A recent paper on expressive speech from AT&T [25.30] for example, describes an experiment



Amazon EX1010 Page 531

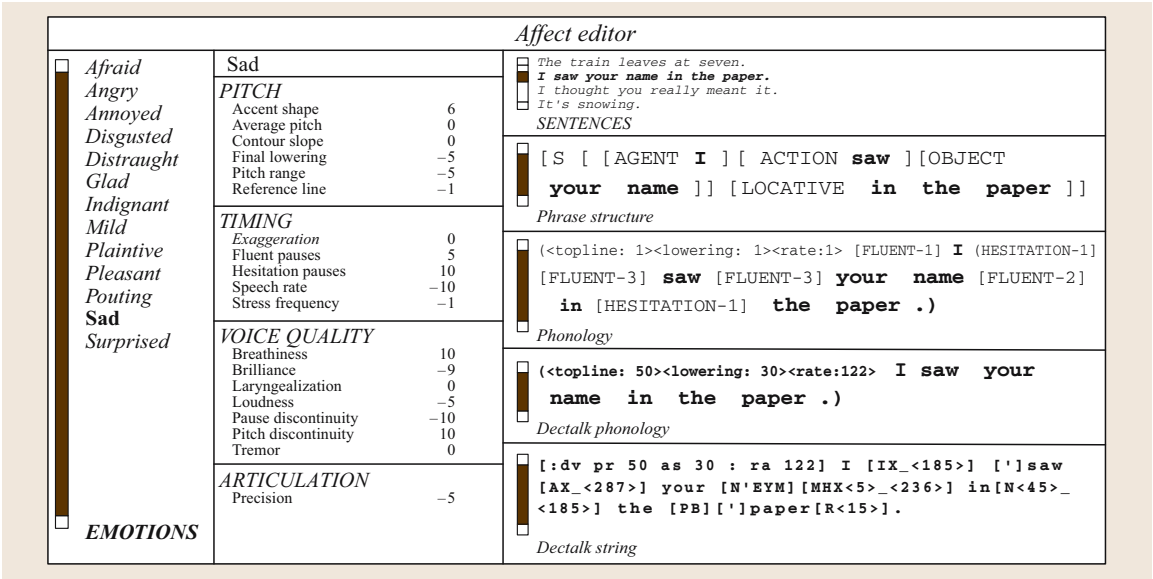



**Fig. 25.3** The Effect Editor user interface. In this example, the sentence, "I saw your name in the paper." will be spoken with a Sad affect. The parameters of the model and their quantities appear in the middle column, labeled with the curent emotion (Sad). (after [25.28, Fig. 2])

in synthesizing four emotional states – anger, happiness, sadness and 'neutral' [sic] – using a concatenative speech synthesizer. The results were evaluated by conducting listening tests with 33 naive listeners and the synthesized anger was recognized with 86.1% accuracy, sadness with 89.1%, happiness with 44.2%, and neutral emotion with 81.8% accuracy.

Recent developments at MIT for embodied communicative agents are extending Cahn's research, but still primarily from the standpoint of expressing raw emotions. The Kismet project [25.31] explains:

> *emotions have a global impact on speech since they modulate the respiratory system, larynx, vocal tract, muscular system, heart rate, and blood pressure. ... when a subject is in a state of fear, anger, or joy, the sympathetic nervous system is aroused. This induces an increased heart rate, higher blood pressure, changes in depth of respiratory movements, greater sub-glottal pressure, dryness of the mouth, and occasional muscle tremor. The resulting speech is faster, louder, and more precisely enunciated with strong high frequency energy, a higher average pitch, and wider pitch range. In contrast, when a subject is tired, bored, or sad, the parasympathetic nervous system is more active. This causes a decreased heart rate, lower blood pressure, and increased salivation. The resulting speech is typically slower, lower-pitched, more slurred, and with little high frequency energy.*

Since the Kismet goals are to give the appearance of sentience to an inanimate robot, these considerations may be appropriate, but in their recognition of human speech [25.32], they acknowledge concern with the affective states and communicative intentions of the speaker, categorizing incoming speech as one of: soothing, approving, prohibiting, or neutral (i. e., robot-directed). That is, they recognize that some speech is listener-directed rather than speaker-revealing.

Part of the reason for the dominance of raw emotions in expressivity research is the ease of collecting training data. *Campbell* and *Iida* collected an hour each of happy, sad, and angry speech in addition to the standard read-speech to produce an emotional voice for CHATR [25.33, 34]. By switching databases when producing synthesized speech, listeners correctly perceived the intended emotion. However, we soon found that our users, typically speaking-impaired persons such as ALS patients, had less need to express such simple emotions and wanted instead to show interest through tone of voice, or to show concern, warmth, indifference, to be able to communicate their feelings rather than their emotional states.



Amazon EX1010 Page 532

IBM Research, another team that has an excellent reputation for speech synthesis, has recognized this need for listener-based expression of affect, rather than speaker-based expression of emotion [25.35]. They acknowledge the intentional basis of human communication, and state their text-to-speech research goal to be "to make synthesized speech as intelligible, natural and pleasant to listen to as human speech and have it communicate just as meaningfully". They use concatenative techniques to tune the speech to fit the text content, currently defining expressiveness as distinguishing between: a good news statement, a bad news statement, a yes/no question, and Contrastive emphasis [25.36].

The need to distinguish between urgency and calm in the voice is also realized by the automotive industry. SVOX is continuing to research and develop new ways to extend unit selection algorithms to impart various emotions and expressive voice corpora [25.37]. They claim that:

> *With SVOX SpeechCreate, expressive text-to-speech prompts can be designed and integrated into our customers system. For automotive applications, as an example, an impression of urgency can be included with driving directions that should be followed immediately. In addition, paralinguistic sounds can be added to increase the human-like quality of the speech output.*

There are many similar examples of expressive speech on the web. The interested reader is recommended to visit Felix Burk Burkhardt's site (http://emosamples.syntheticspeech.de) first of all, then to explore the Humaine links [25.38] for a review research related to the expression of emotions in speech.

## 25.5 Modeling Human Speech

In our own expressive speech research, we first produced a corpus. The Japan Science and Technology Corporation core research for evolutional science and technology (JST/CREST) expressive speech corpus [25.39] was collected over a period of five years by fitting a small number of volunteers with head-mounted high-quality microphones and small minidisc walkman recorders to be worn while going about their ordinary daily social interactions. Further groups of paid volunteers transcribed and annotated the speech data for a variety of characteristics, including speech-act, speaker-state, emotion, relationship to the interlocutor, etc.

Altogether, we collected 1500 h of spontaneous natural conversational speech. All the data were manually transcribed, and about 10% was further annotated. Table 25.1 shows some of the categories that were used for annotation. Speech samples can be listened to at the project website, http://feast.atr.jp/non-verbal/. We found many words and short phrases repeated frequently throughout the corpus; utterances used by the speakers more for their discourse effect than for their linguistic or propositional content. These utterances proved most difficult for the labelers to adequately categorize. They function primarily as backchannel utterances, but also serve to display a wide range of attitudinal and affective states.

We have reported elsewhere [25.4] on studies that measure the extent to which their function can be similarly perceived by different groups of listeners belonging to different cultural backgrounds and languages. In terms of quantity, more than half of the utterances in the corpus were of this predominantly nonverbal type; short words or simple syllables that occurred alone or were repeated several times in succession, often not appearing at all in a dictionary of the formal language, but forming essential components of a two-way spoken interaction.

Our labelers tested several methods of describing these conversational utterances and eventually decided on the three-level system shown in Table 25.1, whereby the following could be distinguished

1. facts about the speaker
2. facts about the utterance, and
3. separate independent evaluations could be made about the information portrayed by differences in the voice quality.

Level 1 describes the state of the speaker, and requires long-term context in order to enable a judgement. It produces an estimation of the discourse purpose of the utterance (see details below), the speaker's emotional states and mood (these labels are free input, those in the table being examples), her interest in the discourse, and finally a label to denote labeler confidence in the current decisions. The numerical labels are forced choice on a scale of high to low (see the lower part of the table) with no default or zero setting.

Level 2 describes the style of the speech, its type and purpose, as can be estimated from a short time window (i. e., with no information regarding discourse context) so that it describes the information available from lis-



Amazon EX1010 Page 533

**Table 25.1** Three levels of labeling for describing each utterance, includingthe use of six-level forced-choice tendency scales

| **Level 1** | **State (about the speaker)** |
|---|---|
| Purpose | A discourse-act/DA label (see text) |
| Emotion | Happy/sad/angry/calm |
| Mood | Worried/tense/frustrated/troubled/ . . . |
| Interest | A 6-point scale from +3 to −3, omitting 0 |
| Confidence | A 6-point scale from +3 to −3, omitting 0 |
| **Level 2** | **Style (about the speech)** |
| Type | Speaking-style label (open class) |
| Purpose | A discourse-act label (closed class) |
| Sincerity | Insisting/telling/feeling/recalling/acting/ . . . |
| Manner | Polite/rude/casual/blunt/sloppy/childish/ sexy/ . . . |
| Mood | Happy/sad/confident/diffident/soft/ aggressive/ . . . |
| Bias | Friendly/warm/jealous/sarcastic/flattering/ aloof/ . . . |
| **Level 3** | **Voice (about the sound)** |
| Energy | A six-point scale from +3 to −3, omitting 0 |
| Tension | A six-point scale from +3 to −3, omitting 0 |
| Brightness | A six-point scale from +3 to −3, omitting 0 |
| **Level 0** | **Labeler** |
| Confidence | A six-point scale from +3 to −3, omitting 0 |

| **Six-point values** | **Negative** | **Positive** |
|---|---|---|
| Very noticeable | −3 | 3 |
| Noticeable | −2 | 2 |
| Only slightly noticeable | −1 | 1 |

tening to the isolated speech utterance alone, as distinct from the same utterance situated in a discourse (i. e., we are not interested in whether the speaker is actually, e.g., angry or not, but only in whether the particular segment in question sounds angry). The *sincerity* label describes an important functional aspect of the speech, such as can be distinguished between the verbs 'insisting', 'telling', 'quoting', 'saying', 'feeling', 'recalling', 'acting', 'pretending', etc.

Manner is a bucket category that includes politeness and sexiness (which are not at all mutually contradictory) as well as childishness, sloppiness, etc. to describe the perceived attitude(s) of the speaker towards the listener. This is complemented by mood and bias, of which the former indicates the affective states of the speaker, and the latter his or her attitudes.

Level 3 describes the physical characteristics of the speaker's voice quality and speaking style in perceptual terms.

### 25.5.1 Discourse-Act Labeling

In order to describe the purpose or function of each utterance, a decision is first made about its directionality, which may be either offering (to the listener) or seeking (from the listener). Utterances are then broadly categorized into seven classes of discourse intentions, including: questions, opinions, objections, advice, information, greetings, and grunts. These category labels are determined by necessity as examples of each appeared in the data. As noted above, the last category accounted for almost half of the utterances in the corpus.

Under the category of 'questions', we use the following labels: WH questions, Y/N questions, repetition requests, and information requests.

Under the category of 'opinions' we use the following labels: opinion, compliment, desire, will, thanks, and apology.

Under the category of 'objections' we use the following labels: objection, complaint.

Under the category of 'advice' we use the following labels: advice, command, suggestion, offer, and inducement

Under the category of 'information' we use the following labels: give information, reading, introduce self, introduce topic, and closing

Under the category of 'greetings' we use the following labels: greeting, talking to self, asking self, checking self.

Under the category of 'grunts' we use the following labels: notice, laugh, filler, disfluency, mimic, habit, response, and backchannel. Response and backchannel utterances are further subcategorized into the following types: agree, understand, convinced, accept, interested, not convinced, uncertain, negative, repeat, self-convinced, notice, thinking, unexpected, surprise, doubt, impressed, sympathy, compassion, exclamation, listening, and other.

### 25.5.2 Expressive Speech and Emotion

The experience gained from this labeling process has caused us to now rethink some of our original assumptions. We started out by attempting to overcome Labov's observer's paradox on the assumption that long-term exposure to a recording device would eventually cause the wearer to become so familiar with it that it no longer becomes a hindrance to normal spoken interaction, even of a highly personal kind. This has proved to be the case, and is confirmed by the variety of speech that we have collected.



Amazon EX1010 Page 534




**Fig. 25.4** The Chakai conversational speech synthesis interface. By clicking on a speech-act icon, a choice of emoticons is displayed in the *upper section* of the screen according to corpus availability, from which an utterance having the appropriate speech characteristics can be selected. Utterances are selected at random from among those in the same category within the corpus so that subsequent selection of the same combination will provide natural variety without unnecessary repetition

However, another paradox has arisen in its place. We originally believed that we would be able to capture truly natural and spontaneous emotional speech data by having a microphone active and in place before and while the emotional event took place. Instead, we find that by far the majority of our speech material is not marked for emotion as we then conceived it, but that it varies significantly in dimensions better related to affect and attitude, signaling the mood and interest of the speaker, his or her current relations with the listener, and controlling the variable flow of the discourse.

We started out by believing that emotion was the essential component lacking in our speech corpora for technology development, but we now consider that the human dimension that we were looking for is not best described by the term 'emotion' at all. Our data score very highly on the measure of paralinguistic to linguistic content described in the introduction, and are very far from the formal speech of less interactive situations, almost half being nonverbal and affect-related, but they lead us to conclude that the emotional state(s) of the speaker are not always directly expressed, and that social and interpersonal considerations override the supposed link between subjective emotion and displayed affective states. The social aspects of communication therefore take precedence over the blunt expression of feeling, and while the latter can perhaps be determined from an expressive utterance, the multiple levels of information in the former provide a richer source of data to be pro-



Amazon EX1010 Page 535

cessed if we are to better understand the person through her speech.

### 25.5.3 Concatenative Synthesis Using Expressive Speech Samples

With such a large corpus, including more than 600 h from one speaker alone, we were able to test some original approaches to concatenative synthesis of expressive speech. In this case, the units to be concatenated are no longer subsegmental, but can be as large as whole phrases. It might be debatable whether such a concatenative technique is still to be called speech synthesis, but since our unit-selection criteria make use of the labels described above, we would claim that this is still the case.

In parallel with the problem of determining the optimal unit size, is the equivalent problem of how to specify such units for input to the synthesizer. Plain text is no longer appropriate when the intention of the speaker is more important than the lexical sequence of the utterance. Instead, we needed to enable the user to quickly access a given corpus segment by means of higher-level intention-related constraints.

Figure 25.4 shows a recent proposal for Chakai, which is such a speech synthesis interface [25.40, 41] that allows for free input (by typing text into the white box shown at the bottom center) as well as the fast selection of various frequently used phrases and, in addition, an icon-based speech-act selection facility for the most common types of affective *grunt*. The name, not unrelated to CHATR, is composed of two Japanese syllables, meaning tea meeting, an event during which social and undirected chat is common. This format enables linking to a conventional CHATR-type synthesizer for creation of I-type utterances not found in the corpus, while providing a fast, three-click, interface for common A-type utterances which occur most frequently in ordinary conversational speech. Samples (and source code) are available from http://feast.atr.jp/chakai.

The selection of whole phrases from a large conversation speech corpus requires specification not just of the intention of the phrase (greeting, agreement, interest, question, etc.,) but also of the speaker's affective state (as desired to be represented) and the speaker's long- and short-term relationships with the listener at that particular time.

## 25.6 Conclusion

Humans are primarily social animals; they relate in groups and form close communities and subgroups. Much of human speech is concerned not with the transmission of propositional content or novel information, but with the transfer of affective information, establishing bonds, forming agreements, and reassuring each other of a positive and supporting (or other) environment.

In listening to a spoken utterance, human listeners parse not just its linguistic content, but also the way it has been spoken, voice qualities (including tone of voice) provide clues to its intent, in a way that is complementary to its content, to assist in the interpretation of the utterance.

In speech conversation, both the speaker and the listener are active at all times; the speaker predominantly so, conveying novel information, while the listener conveys feedback information about the flow of the discourse, and about the mutual states of comprehension.

In conversational speech, some of the time is devoted to imparting propositional content, but much of the time is devoted to managing the discourse; eliciting feedback, controlling turn-taking, and expressing affective states and stances.

In the synthesis of expressive speech, we must be able to generate sounds that reproduce all levels of information in the speech signal. At present there are no systems that can do this, but this is an active area of research and one for which there is a strong human need. If we can solve this problem, we will be able to produce a technology that people can relate to, can feel at ease with, and that can process information at levels far deeper than the mere linguistic.



## References

25.1 G. Bailly, C. Benoit, T.R. Sawallis (Eds.): *Talking Machines: Theories, Models, and Designs, Reports Papers from the first ISCA Speech Synthesis workshop in Autrans* (North-Holland, Amsterdam 1992)

Amazon EX1010 Page 536

25.2 N. Campbell: Getting to the heart of the matter; speech as expression of affect rather than just text or language, Language Res. Eval. **39**(1), 109–118 (2005)

25.3 SSML, The W3 Speech Synthesis Markup Language: www.w3.org/TR/speech-synthesis/ (See also the papers from the 2005 SSML Meeting at http://www.w3.org/2005/08/SSML/Papers/)

25.4 N. Campbell, D. Erickson: What do people hear? A study of the perception of non-verbal affective information in conversational speech, J. Phonetic Soc. Jpn. **7**(4), 9–28 (2004)

25.5 I.G. Mattingly: Experimental methods for speech synthesis by rules, IEEE Trans. AU **16**, 198–202 (1968)

25.6 J. Allen: Linguistic-based algorithms offer practical text-to-speech systems, Speech Technol. **1**(1), 12–16 (1981)

25.7 K. Church: Stress assignment in letter to sound rules for speech synthesis. In: *ACL Proc. 23rd Annual Meeting*, ed. by University of Chicago (Association for Computational Linguistics, Chicago 1985) pp. 246–253

25.8 G. Akers, M. Lennig: Intonation in text-to-speech synthesis: Evaluation of algorithms, J. Acoust. Soc. Am. **77**, 2157–2165 (1985)

25.9 N. Campbell: Recording techniques for capturing natural everyday speech. In: *Proc Language Resources and Evaluation Conference LREC-02)* (Las Palmas, Spain 2002) pp. 2029–2032

25.10 N. Campbell: Speech and expression; the value of a longitudinal corpus. In: *Proc. Language Resources and Evaluation Conference* (2004) pp. 183–186

25.11 R. Cowie, E. Douglas-Cowie, C. Cox: Beyond emotion archetypes; Databases for emotion modelling using neural networks, Neural Networks **18**, 371–388 (2005)

25.12 K. Ishimura, N. Campbell: Telephone dialogue data base of JST/CREST expressive speech processing project, Proc. Ann. Conf. JSAI **16**, 147–148 (2002)

25.13 D. McNeill, F. Quek, K.-E. McCullough, S. Duncan, N. Furuyama, R. Bryll, X.-F. Ma, R. Ansari: Catchments, prosody, and discourse, Gesture **1**, 9–33 (2001)

25.14 R. Carlson, B. Granstrom: A text-to-speech system based entirely on rules, Proc. IEEE-ICASSP **76**, 686–688 (1976)

25.15 J. Allen, M.S. Hunnicutt, D.H. Klatt: *From Text to Speech, The MITalk System* (Cambridge Univ. Press, Cambridge 1987)

25.16 K. Hirose, K. Sato, Y. Asano, N. Minematsu: Synthesis of F0 contours using generation process model parameters predicted from unlabeled corpora: Application to emotional speech synthesis, Speech Commun. **46**(3-4), 385–404 (2005-2007)

25.17 A. Sakurai, K. Hirose, N. Minematsu: Data-driven generation of F0 contours using a superpositional model, Speech Commun. **40**(4), 535–549 (2003)

25.18 K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, J. Hirschberg: ToBI: A standard for labelling English prosody, Proc. ICSLP 92 **2**, 867–870 (1992)

25.19 The official ToBI website: http://www.ling.ohio-state.edu/ tobi/

25.20 Webpage of the Signal Processing Laboratory of the University of the Basque Country (UPV/EHU) in Bilbao: http://bips.bi.ehu.es/aholab/TTS/Expressive-Speech-Synthesis.html

25.21 P. Ekman: Basic emotions. In: *Handbook of Cognition and Emotion*, ed. by T. Dalgleish, M. Power (Wiley, New York 1999) pp. 301–320

25.22 K. Sjolander, J. Gustafson: Voice creation for conversational fairy-tale characters. In: *Proc. SSW SYnthesis Workshop* (2005)

25.23 A. Silva, G. Raimundo, C. de Melo, A. Paiva: To tell or not to tell... Building an interactive virtual storyteller. In: *Proc. AISB Symp. Language Speech and Gesture for Expressive Characters* (2004)

25.24 M. Theune, K. Meijs, D. Heylen, R. Ordelman: Generating expressive speech for storytelling applications, IEEE Trans. Audio Speech Language Process. **14**(4), 1137–1144 (2006)

25.25 N. Campbell: Recording techniques for capturing natural everyday speech. In: *Proc. Language Resources and Evaluation Conference* (Las Palmas, Spain 2002) pp. 2029–2032

25.26 J.E. Cahn: *Generating expression in synthesized speech*, M.S. Thesis (Massachusetts Institute of Technology, Cambridge 1989), http://alumni.media.mit.edu/ cahn/emot-speech.html

25.27 J.E. Cahn: From sad to glad: Emotional computer voices. In: *Proc. Speech Tech '88 Voice Input/Output Applications Conference and Exhibition* (New York 1988) pp. 35–37

25.28 J.E. Cahn: The generation of affect in synthesized speech, J. Am. Voice I/O Soc. **8**, 1–19 (1990)

25.29 J.E. Cahn: Generation of affect in synthesized speech. In: *Proc. 1989 Conf. American Voice I/O Society* (Newport Beach, California 1989) pp. 251–256

25.30 M. Bulut, S.S. Narayanan, A.K. Syrdal: Expressive speech synthesis using a concatenative synthesizer, Proc. ICSLP **2002**, 1265–1268 (2002)

25.31 MIT's Kismet (the expressiveness and richness of the robot's vocal modality and how it supports social interaction): http://www.ai.mit.edu/projects/sociable/expressive-speech.html

25.32 MIT Kismet and Affective Intent in Speech: http://www.ai.mit.edu/projects/sociable/affective-intent.html

25.33 A. Iida, N. Campbell, M. Yasumura: Design and evaluation of synthesised speech with emotion, J. Inform. Process. Soc. Jpn. **40** (1998)

25.34 A. Iida, N. Higuchi, N. Campbell, M. Yasumura: Corpus-based speech synthesis system




Amazon EX1010 Page 537

with emotion, Speech Commun. **40**(1–2), 161–187 (2002)
25.35 E. Eide, A. Aaron, R. Bakis, W. Hamza, M.A. Picheny, J.F. Pitrelli: A corpus-based approach to AHEM expressive speech synthesis. In: *Proc. 5th ISCA Speech Synthesis Workshop* (Pittsburgh, USA 2004)
25.36 J.F. Pitrelli, R. Bakis, E.M. Eide, R. Fernandes, W. Hamza, M.A. Picheny: The IBM expressive text-to-speech synthesis system for American English, IEEE Trans. Audio Speech Language Process. **14**(4), 1099–1108 (2006)
25.37 SVOX: http://www.svox.com/Innovation.aspx
25.38 The HUMAINE Portal – Research on Emotions and Human-Machine Interaction: http://emotion-research.net/
25.39 The Expressive Speech Processing project web pages: http://feast.atr.jp/
25.40 N. Campbell: Specifying affect and emotion for expressive speech synthesis. In: *Computational Linguistics and Intelligent Text Processing*, ed. by A. Gelbukh (Springer, Berlin, Heidelberg 2004)
25.41 N. Campbell: Conversational speech synthesis and the need for some laughter, IEEE Trans. Audio Speech Language Process. **14**(4), 1171–1179 (2006)



Amazon EX1010 Page 538

# Part E Speech Recognition

## Part E Speech Recognition

Ed. by L. Rabiner, B.-H. Juang


**26 Historical Perspective of the Field of ASR/NLU**
L. Rabiner, Piscataway, USA
B.-H. Juang, Atlanta, USA

**27 HMMs and Related Speech Recognition Technologies**
S. Young, Cambridge, UK

**28 Speech Recognition with Weighted Finite-State Transducers**
M. Mohri, New York, USA
F. Pereira, Philadelphia, USA
M. Riley, New York, USA

**29 A Machine Learning Framework for Spoken-Dialog Classification**
C. Cortes, New York, USA
P. Haffner, Middletown, USA
M. Mohri, New York, USA

**30 Towards Superhuman Speech Recognition**
M. Picheny, Yorktown Heights, USA
D. Nahamoo, Yorktown Heights, USA

**31 Natural Language Understanding**
S. Roukos, Yorktown Heights, USA

**32 Transcription and Distillation of Spontaneous Speech**
S. Furui, Tokyo, Japan
T. Kawahara, Kyoto, Japan

**33 Environmental Robustness**
J. Droppo, Redmond, USA
A. Acero, Redmond, USA

**34 The Business of Speech Technologies**
J. Wilpon, Florham Park, USA
M. E. Gilbert, Florham Park, USA
J. Cohen, Menlo Park, USA

**35 Spoken Dialogue Systems**
V. Zue, Cambridge, USA
S. Seneff, Cambridge, USA


Amazon EX1010 Page 539

# 26. Historical Perspective of the Field of ASR/NLU



L. Rabiner, B.-H. Juang

The quest for a machine that can recognize and understand speech, from any speaker, and in any environment has been the holy grail of speech recognition research for more than 70 years. Although we have made great progress in understanding how speech is produced and analyzed, and although we have made enough advances to build and deploy in the field a number of viable speech recognition systems, we still remain far from the ultimate goal of a machine that communicates naturally with any human being. It is the goal of this section to document the history of research in speech recognition and natural language understanding, and to point out areas where great progress has been made, along with the challenges that remain to be solved in the future.

26.1 **ASR Methodologies** ..... 521
26.1.1 Issues in Speech Recognition ..... 522
26.2 **Important Milestones in Speech Recognition History** ..... 523
26.3 **Generation 1 – The Early History of Speech Recognition** ..... 524
26.4 **Generation 2 – The First Working Systems for Speech Recognition** ..... 524
26.5 **Generation 3 – The Pattern Recognition Approach to Speech Recognition** ..... 525
26.5.1 The ARPA SUR Project ..... 527
26.5.2 Research Outside of the ARPA Community ..... 529
26.6 **Generation 4 – The Era of the Statistical Model** ..... 530
26.6.1 DARPA Programs in Generation 4 ... 532
26.7 **Generation 5 – The Future** ..... 534
26.8 **Summary** ..... 534
**References** ..... 535

## 26.1 ASR Methodologies

Speech recognition research has been on-going for more than 70 years. Over that period there have been at least four generations of approaches, and we forecast a fifth generation that is being formulated based on current research themes. The five generations, and the technology themes associated with each of them, are as follows.

1. Generation 1 (1930s to 1950s): use of ad hoc methods to recognize sounds, or small vocabularies of isolated words.
2. Generation 2 (1950s to 1960s): use of acoustic–phonetic approaches to recognize phonemes, phones, or digit vocabularies.
3. Generation 3 (1960s to 1980s): use of pattern recognition approaches to speech recognition of small to medium-sized vocabularies of isolated and connected word sequences, including use of linear predictive coding (LPC) as the basic method of spectral analysis; use of LPC distance measures for pattern similarity scores; use of dynamic programming methods for time aligning patterns; use of pattern recognition methods for clustering multiple patterns into consistent reference patterns; use of vector quantization (VQ) codebook methods for data reduction and reduced computation.
4. Generation 4 (1980s to 2000s): use of hidden Markov model (HMM) statistical methods for modeling speech dynamics and statistics in a continuous speech recognition system; use of forward-backward and segmental $K$-means training methods; use of Viterbi alignment methods; use of maximum likelihood (ML) and various other performance criteria and methods for optimizing statistical models; introduction of neural network (NN) methods for estimating conditional probability densities; use of adaptation methods that modify the parameters associated with either the speech signal or the statistical model so as to enhance the compatibility between model and data for increased recognition accuracy.

Amazon EX1010 Page 540

5. Generation 5 (2000s to 2020s): use of parallel processing methods to increase recognition decision reliability; combinations of HMMs and acoustic–phonetic approaches to detect and correct linguistic irregularities; increased robustness for recognition of speech in noise; machine learning of optimal combinations of models.

The generations are not distinct, as most of the key ideas that define each generation were formulated in earlier generations. However, the indicated time periods for each generation represent the eras in which most of the research took place and the times when the resulting technology became the standard for most recognition systems of that era.

We discuss a range of technology issues and the way they have evolved in each of the speech recognition generations in the remainder of this paper.

In the area of natural language understanding (NLU) we have seen three highly overlapping generations of technology evolve including the following:

1. Generation 1: realize task constraints using simple nodal grammars independent of the acoustic realization of the spoken sentence and without regard to disfluencies in speaking.
2. Generation 2: incorporate statistical grammar and utilize finite state networks (FSN) to model acoustics, syntax, and semantics jointly in a single integrated model which can be configured to account for nongrammatical acoustic events, and which can be optimally searched to find the best path corresponding to the most likely spoken sentence consistent with the task grammar and semantics (and even possibly with task pragmatics).
3. Generation 3: utilize more-complex syntax and semantic representations that maximize language coverage and minimize language over-coverage.

Unfortunately, due to space limitations, we will not be able to talk about the technology evolution in the three generations of NLU research; however, Chap. 31 by *Roukos* will cover this area in great detail.

### 26.1.1 Issues in Speech Recognition

As we examine the progress made in implementing speech recognition and natural language understanding systems over the years, we will see that there are a number of issues that need to be addressed in order to define the operating range of each speech recognition system that is built. These issues include the following:

- Speech unit for recognition: ranging from words down to syllables and finally to phonemes or even phones. Early systems investigated all these types of units with the goal of understanding their robustness to context, speakers and speaking environments
- Vocabulary size: ranging from small (order of 2–100 words), medium (order of 100–1000) words, and large (anything above 1000 words up to unlimited vocabularies). Early systems tackled primarily small-vocabulary recognition problems; modern speech recognizers are all large-vocabulary systems
- Task syntax: ranging from simple tasks with almost no syntax (every word in the vocabulary can follow every other word) to highly complex tasks where the words follow a statistical $n$-gram language model
- Task perplexity (the average word branching factor): ranging from low values (for simple tasks) to values on the order of 100 for complex tasks whose perplexity approaches that of natural language task
- Speaking mode: ranging from isolated words (or short phrases), to connected word systems (e.g., sequences of digits that form identification codes or telephone numbers), to continuous speech (including both read passages and spontaneous conversational speech)
- Speaker mode: ranging from speaker-trained systems (works only on individual speakers who trained the system) to speaker-adaptive systems (works only after a period of adaptation to the individual speaker's voice) to speaker-independent systems, which can be used by anyone without any additional training. Most modern ASR (automatic speech recognition) systems are speaker independent and are utilized in a range of telecommunication applications. However, for dictation purposes, most systems are still largely speaker dependent and adapt over time to each individual speaker
- Speaking situation: ranging from human-to-machine dialogues to human-to-human dialogues (e.g., as might be needed for language translation systems)
- Speaking environment: ranging from a quiet room, to noisy places (e.g., offices, airline terminals), and even outdoors (e.g., via the use of cellphones)
- Transducer: ranging from high-quality microphones to telephones (wireline) to cellphones (mobile) to array microphones (which track the speaker location electronically)

Amazon EX1010 Page 541

- Transmission channel: ranging from simple telephone channels, with μ-law speech coders in the transmission path for wireline channels, to wireless channels with fading and with a sophisticated voice coder in the path

We will see that, whenever we discuss individual recognition systems, we will characterize them in terms of this list of issues (although not every issue will be specified for each system).

## 26.2 Important Milestones in Speech Recognition History

Although people have been interested in machines that listen and understand for a very long time, there have been a few events that have heightened public awareness of speech recognition systems in very clear and focused ways. Perhaps the most well-publicized event was a movie that introduced the general public to machines that could understand speech and respond accordingly, namely the Stanley Kubrick movie *2001: A Space Odyssey* which was first introduced in 1968. In this movie an intelligent machine, called HAL 9000, spoke in a natural sounding voice and was able to recognize and understand fluently spoken speech, and respond intelligently. This movie raised the public awareness of the potential of machines to take over a range of tasks that interacted with humans and provided a range of services and information, on demand. A second movie that raised public awareness of the potential of machines interacting with humans was the *Star Wars* saga of George Lucas. Here the machines (in the form of the mobile robots R2D2 and C3P0) were intelligent, able to speak naturally, able to recognize and understand fluent human speech, and able to move around and interact with their environment and with other machines.

A third important milestone that occurred totally within the speech recognition community was the creation of a vision for the future, by Apple Computer, in the form of a video of a scenario of life in the year 2011 entitled *Knowledge Navigator*. This video portrayed a man working from his home and interacting with his laptop computer via an intelligent agent interface. The intelligent agent was portrayed via an animated, natural-looking face that appeared on the screen and was able to speak naturally and understand all spoken inputs. What made the video so important was the user interfaces to information that were basically defined by this video, including a speech user interface (SUI) that was as natural and easy to use as speaking to another human being, as well as a multimodal user interface (MUI) where the user had a choice of speaking, pointing and clicking, and dragging and dropping objects. The video gave clear indications of how valuable it was to integrate all these interactive modes to maximize the efficiency of interacting with the machine to achieve desired goals. This video energized the speech research community and gave it a well-defined challenge that focused technology efforts throughout the world for more than a decade.

A fourth important milestone was the introduction of a broad-based speech recognition service within the telecommunications community, called Voice Recognition Call Processing (VRCP) in 1993 by AT&T [26.1,2]. This simple voice-enabled system enabled the 100 million customers of AT&T to make a range of operator-assisted toll calls (person-to-person, third-party billing, collect, reversed charges, or operator assisted) without having to interact with a human operator. Using a simple vocabulary of five phrases (one for each of the services), users could speak any of the phrases either as an isolated command or embedded in a carrier sentence (using so-called word or phrase spotting technology), and be reliably and accurately recognized more than 99% of the time. This system ultimately properly handled about 1.2 billion calls each year and made the general public aware that more-powerful speech recognition systems were just around the corner.

A fifth important milestone was the development and introduction to the public of several speech understanding systems (both demonstrations and real-world systems) that could hold a dialogue with a human user to complete a transaction or provide some desired information. Such systems, when properly designed and implemented, appeared to be intelligent and able to handle complex tasks with seemingly minimal effort. Among such systems were the Jupiter system for weather information from MIT [26.3], the Pegasus system for flight information (also from MIT [26.4]), and the *"How May I Help You"* system for call routing of customer help line requests from AT&T [26.5,6].

Today, in 2007, there are thousands of active speech recognition systems in use in telecom environments, in automobiles, on cellphones, in home environments and

office environments, showing just how far we have come over the past 70 years. Most of these systems primarily work in limited task domains (account entry, airlines information, stock-price quotations, directions, etc.) and in limited environments (e.g., quiet office or home), but their ubiquity and performance is testimony to the degree of success that has already been achieved with the technology.

## 26.3 Generation 1 – The Early History of Speech Recognition

Speech research in the period between 1930 and 1950 was dominated by research which attempted to determine the relationship between the linguistic identity of a sound (the phoneme or syllable being pronounced) and the spectral characterization of that sound, the so-called acoustic–phonetic representation of speech. There was also a great deal of research in trying to understand the factors that influenced the intelligibility of speech for different sounds and various noise environments. The two most prominent researchers during this period were Harvey *Fletcher* and Homer *Dudley*, both from AT&T Bell Laboratories [26.7–9]. The pioneering work of these two speech pioneers firmly established the relationships between sound classes and signal spectrum, and showed that reliable identification of the phonetic nature of a speech sound was inherently tied in with reliable estimation of the spectral properties of that sound. The work of Fletcher, Dudley, and numerous other speech processing pioneers was quantified in the landmark book *Visible Speech* [26.10] and this book served as the bible of speech processing for a number of years.

The first generation of speech recognition research basically defined the front end analysis for modern speech recognition systems as a time-varying spectral analysis of the speech signal. Ultimately a number of variants of short-time spectral analysis were devised in subsequent years including filter bank analyses, model-based estimation methods [including the technique of linear predictive coding (LPC)], and cepstral analysis.

## 26.4 Generation 2 – The First Working Systems for Speech Recognition

In the second generation of research on speech recognition systems we see a number of studies trying to create realizations of the analysis of Fletcher and Dudley, namely trying to establish algorithmic approaches to the quantification of the phonetic identity of a sound (or a sequence of sounds) based on measured spectral properties of the sound, as a function of time.

Among the systems built during this period were the following:

- In 1952 *Davis*, *Biddulph*, and *Balashak* of Bell Laboratories [26.11] designed and implemented a single speaker (i. e., a speaker-trained system), isolated digit recognizer based on learned formant trajectories during the vowel regions of the spoken digit. (The formant regions are the resonances of the vocal tract in producing the sounds and generally, but not always, correspond to prominent peaks in the speech spectral density of the sound.) Even with these crude reference patterns that were based on the spectral density of the speech signal, for some speakers, virtually perfect digit recognition scores could be obtained, whereas for other speakers the digit recognition accuracy fell to the 50% range, showing the difficulty of even a simple isolated digit recognition task.
- In 1956 *Olson* and *Belar* of RCA Laboratories [26.12] built a single speaker recognition system that attempted to recognize 10 monosyllables based on a crude spectral analysis using a filter bank with eight frequency bands over the frequency range of interest. Measurements of the spectral energy of the eight bandpass filters were made at five uniformly distributed times across the spoken input (a crude form of linear time normalization) and the resulting reference pattern of eight spectral measurements and five time points was quantized to 40 bits (i. e., 1 bit for each grid point), and compared to a stored pattern for each of the 10 monosyllables. Again the performance was highly variable with talkers, but at least one talker was able to achieve 98% accuracy using this system.
- One of the first attempts to build a speaker-independent speech recognition system was de-

scribed by *Forgie* and *Forgie* of MIT Lincoln Laboratory [26.13]. The vocabulary of interest was a set of 10 vowels (spoken in the carrier syllable /b/-vowel-/t/). The front-end spectral analysis was a 35 channel filter bank from which the prominent spectral regions were estimated and the first three formants were selected to represent the vowel sound in the utterance. In limited testing the system achieved 93% accuracy, a major accomplishment at its time in history.

- In 1959 *Denes* and *Fry* at University College in London built a system to recognize a sequence of one of four vowels followed by one of nine consonants [26.14]. The system consisted of a spectral analyzer and a spectral pattern matching algorithm. Although the performance of the resulting system was not very good, the most interesting part of their research was showing that when a linguistic model based on the actual probabilities of the sequence of a given vowel, $p_1$, and a given consonant, $p_2$ were utilized in the scoring method, the performance increased from 24% accuracy to 44% accuracy – a very significant rise due to the use of a grammar.

This one-to-one association of a section of the speech signal to a distinct sound (phoneme, syllable, or word for the most part), often denoted as a segmentation and labeling approach to speech recognition, proved to be a lot more difficult than anticipated, due to the high degree of variability of the spectral information for a given sound across speakers, accents, and linguistic content of the speech being analyzed.

## 26.5 Generation 3 – The Pattern Recognition Approach to Speech Recognition

A number of significant technological advances in speech processing systems (and their application to speech recognition systems) occurred over the period from 1960 to 1980. In the area of spectral analysis, the method of linear predictive coding was developed by *Atal* and *Itakura* (working independently) [26.15, 16] and rapidly became the method of choice for front-end spectral analysis of speech. The concept of comparing (determining the spectral distance between) two spectral representations was proposed and studied by *Itakura* [26.17] and was widely used in pattern-based systems. The concept of optimally aligning speech spectral patterns for utterance-long sentences, based on dynamic programming concepts, has come to be known as the set of dynamic time warping (DTW) methods and was originally proposed by *Vintsyuk* in the Soviet Union [26.18] and later by *Sakoe* and *Chiba* in Japan [26.19]. The concept of quantizing a spectral representation (in the form of a vector of spectral components) using an optimally designed codebook led to the methods of vector quantization (VQ), as originally proposed by Linde et al. [26.20] and studied by *Gersho* and *Gray* [26.21]. One last important technological contribution during generation 3 was the introduction of statistical pattern clustering techniques as the method of choice for creating speaker-independent reference patterns from multiple speaker-training tokens, as proposed by *Rabiner* et al. [26.22].

The 1960s saw the introduction of several Japanese laboratories into the speech recognition arena, generally building special purpose hardware (as opposed to computer simulations) to test out their systems. Among the many Japanese systems of the 1960s were the following:

- In 1961, *Suzuki* and *Nakata*, at the Radio Research Lab in Japan, built a 26 channel spectrum analyzer as the front end for a vowel recognition system [26.23]. The channels were grouped together and connected to a set of *vowel decision circuits* (using the voiced/unvoiced and pitch period information as controls) which periodically sampled the the vowel decision outputs and choosing the phoneme most frequently recognized.
- In 1962, *Sakai* and *Doshita* of Kyoto University built a vowel/consonant recognition system based on a segmentation of speech into vowels and consonants, based on zero crossing (ZC) and frequency band energy, and obtained recognition accuracies on the order of 90% for vowels and 70% for consonants (with some phonemes eliminated from the test set) [26.24].
- In 1963, *Nagata* et al. at NEC Labs in Japan built a hardware filter bank with 8 spectral bands and created one of the first systems that was shown capable of recognizing 10 Japanese isolated digits in a highly reliable manner, using early forms of many

of the techniques that were being developed over the course of generation 3 [26.25].

There were three other key developments in speech recognition research in the 1960s that had long-lasting influence on the directions of research over a decade or more. The first was the research of Tom *Martin* and his colleagues at the RCA Laboratories [26.26]. The novel aspect of this work was the understanding that reliable detection of speech beginning and ending points enabled a rudimentary form of nonuniform time alignment between reference and test patterns, which enabled significantly more-reliable speech recognizer performance. Martin ultimately developed a set of highly reliable algorithms for word recognition, built hardware solutions that implemented these new methods, and founded one of the first companies (Threshold Technology) that built, marketed and sold speech recognition products. One of the first products from Threshold Technology, called the VIP-100 recognizer, was used for a variety of real-world applications including baggage sorting at airports, package sorting at FedEx distribution centers, and for television face plate manufacturing quality control.

The second key development of the 1960s was the pioneering research of Raj *Reddy*, initially at Stanford University and ultimately at Carnegie Mellon University, in the field of continuous speech recognition based on dynamic tracking of phonemes [26.27]. Reddy laid the foundation for more than three decades of research in speech recognition at Carnegie Mellon University and was directly involved in many of the key recognition systems of the last three generations.

The third key development at the end of the 1960s was the work of *Velichko* and *Zagoruyko* in the Soviet Union [26.28]. These researchers used the earlier work of *Vintsyuk* [26.18] to build a speech recognizer based on a pattern recognition framework and using dynamic programming time alignment methods. Velichko and Zagoruyko ultimately built a 200-word recognizer and were able to achieve reliable and robust performance for a range of talkers.

In a classic review of the state of the art in speech recognition, S. R. *Hyde* looked at the progress that had been made in speech recognition systems at the end of the 1960s, and among his many observations and conclusions were the following [26.29] (with updated perspectives):

- The debate as to the basic unit of speech recognition, namely either the phoneme or the syllable, was not yet resolved. This debate would take another 20 years to resolve, but ultimately the decision was to use context-dependent phones as the basic speech recognition unit.
- Speech cannot be directly segmented into phonemes because of the importance and the influence of context dependency. Hence all segmentation and labeling systems were essentially doomed without applying linguistic constraints over multiple phonemes.
- Time normalization based on linear shrinking or stretching the waveform to match patterns does not work, particularly when carried out over the span of a word. Speech is a dynamic signal and certain sounds of speech can be expanded or compressed almost at will, while other sounds are highly constrained and cannot easily be expanded or compressed by arbitrary amounts. Hence a suitable form of time normalization, via nonlinear time alignment methods, was crucial to the ultimate success of any recognition system.
- In order for speech recognition to work reliably in a range of speaking environments, robust spectral measurements are crucial. This observation is at the heart of current research that is seeking methods of making standard spectral measurements be robust to noise, distortion, and reverberation.
- In order to recognize speech using just acoustic measurements, an effective method of talker normalization is essential. This observation is a reflection of the spectral variability of speech as a function of the vocal tract dimensions. Hence any effective system has to either normalize out this variability or account for it as part of the recognition model. Most modern systems build statistical models which account for the variability, since it is difficult, if not impossible, to properly normalize for every possible talker.
- Devising and perfecting speech recognition systems that are effective for single talkers is of little value in solving the problem in a speaker-independent manner. Hence we need to tackle the problem of interspeaker variability directly and not try to build in fixes to the single talker systems.
- We need to utilize linguistic information in order to properly and accurately recognize fluent speech. This is basically the only way of handling the word perplexity problem and making the recognition task manageable, even for large word vocabularies.

Many of the observations made by Hyde helped guide speech recognition research for a decade or more.

### 26.5.1 The ARPA SUR Project

By the end of the 1960s, the Advanced Research Projects Agency (ARPA) decided that speech recognition research had advanced sufficiently that it was time to pose a challenge, namely to see whether any research lab could build a demo of a working system that could reliably and accurately recognize continuous speech. The ARPA Speech Understanding Research (SUR) program began in 1971 and had a 5 year limit for achieving a defined set of goals, with the general guidelines of:

- transforming phonemes (or syllables) into words with the aid of linguistic information
- using stress to identify semantic content words and phrase boundaries
- using syntax to constrain word sequences
- using semantics to further constrain word sequences

The specific set of goals for the ARPA SUR project were [26.30]:

> *Accept connected speech, from many cooperative speakers, in a quiet room, using a good microphone, with slight tuning/speaker, accepting* 1000 *words, using an artificial syntax, in a constraining task, yielding* $<$ 10% *semantic error, in a few times real time, on a* 100 *MIPS machine.*

One interesting aspect of the ARPA SUR project was that each participating research group was free to choose the task that was to be solved using the final recognition system.

A large number of groups participated in the SUR project, and among the technological contributions that came out of the project were the following:

- use of LPC as the front-end spectral processor became the standard for speech recognition systems [26.15, 16, 31],
- methods for reliably estimating vocal tract areas and vocal tract lengths from LPC representations [26.32],
- use of the LPC residual as a sound similarity measure [26.17],
- use of dynamic programming for time alignment [26.18, 19],
- the cepstral and cosh distance measures [26.33, 34],
- the concept of statistical modeling which later evolved into hidden Markov models [26.35],
- the importance of detecting utterance endpoints [26.36], and
- the importance of applying phonological rules of language to limit the recognition search [26.37, 38].

The major participants in the ARPA SUR project were Carnegie Mellon University (CMU), Bolt Beranek and Newman (BBN) and the System Development Corporation (SDC), working jointly with the Stanford Research Institute (SRI). A wide variety of approaches were studied during the 5 year cycle including standard segmentation and labeling methods (used in the SDC and BBN systems), along with a novel blackboard approach based on parallel asynchronous processes for proposing and verifying word hypotheses, and for proposing and verifying sentences consistent with the syntax and semantics of the chosen task (used in the CMU Hearsay II system).

Interestingly, the most successful speech recognition system, and the only one that essentially met the challenge specifications of the ARPA SUR project was the HARPY system of Bruce *Lowerre* of CMU [26.39]. This system used a conventional segmentation and labeling approach (much like systems of the earlier generation). The key novelty of the system was a compilation of the acoustic models with the lexical representation of words, the syntactic production rules for sentences, and a set of word boundary rules, into one large homogeneous network that could be readily represented as a finite state network (FSN) and searched (from left to right) using a *beam search* that only retained search paths that were close to (within a beam width) the locally optimum search path at each frame of the spoken utterance. The high degree of match between the goals of the ARPA SUR project and the performance of the HARPY system is shown in Table 26.1.

**Table 26.1** Match between the goals of the ARPA SUR project and the performance of the HARPY system

| Goal | HARPY |
|---|---|
| Continuous speech | Yes |
| Many speakers | 5 (3 male, 2 female) |
| Cooperative | Yes |
| Quiet room | Computer terminal room |
| Good microphone | Close talking |
| Slight tuning/talker | 20 training sentences/talker |
| $>$ 1000 words | 1011 words |
| Artificial syntax | Average branching factor = 33 |
| Constrained task | Document retrieval |
| $<$ 10% semantic error | 5% semantic error |
| Few times real time | 80 times real time |
| 100 MIPS machine | 0.4 MIPS PDP-KA10 |

**Table 26.2** Performance of the various ARPA SUR systems at different tasks

| System | Sentence accuracy (%) | Word branching factor | Task | Typical sentence |
|---|---|---|---|---|
| CMU HARPY | 95 | 33 | Document retrieval | How many articles on psychology are there? |
| CMU Hearsay-II | 91, 74 | 33, 46 | Document retrieval | How many articles on psychology are there? |
| BBN HWIM | 44 | 195 | Travel budgets | What is the plane fare to Ottawa? |
| SDC | 24 | 105 | Ship facts | How fast is the Theodore Roosevelt? |

Table 26.2 shows a summary of the performance of two CMU systems (HARPY and HEARSAY II), the BBN *Hear What I Mean* (HWIM) system, and the SDC system. It can be seen that the performance of these speech understanding systems is a strong function of the vocabulary average word branching factor, going from a high of 95% semantic accuracy for HARPY with a word branching factor of 33 on the document retrieval task, to 24% semantic accuracy for the SDC system with a word branching factor of 105 on a ship database task. Interestingly the BBN HWIM system for travel budgets had a word branching factor of 195 but was able to achieve 44% semantic accuracy. The performance of the Hearsay II, HWIM, and SDC recognizers all failed to meet the ARPA SUR performance goal. However, the use of a set of parallel asynchronous processes (to simulate a set of component knowledge sources) with the use of a blackboard (to integrate the current state of knowledge about the words in the spoken sentence) was an innovative idea and one that defined the Hearsay II system as unique in its field. The BBN HWIM system also introduced new ideas into speech recognition systems including a lexical decoding network incorporating advanced phonological rules (whose goal was to improve the phoneme recognition rate by exploiting knowledge about contextual effects), a method of handling segmentation ambiguity using a lattice of alternate hypotheses, and the concept of performing word verification tests at the parametric level. Finally we note that at about the time that the ARPA project was ending, some new work was presented at CMU by Jim Baker, called the DRAGON system, based on statistical modeling of speech via hidden Markov models [26.35].

The ARPA SUR project had a huge impact on the field of speech recognition and natural language understanding. Among the long term achievements of this program were the following (taken from [26.40]):

1. established two viable architectures for speech recognition and natural language understanding including:
   - a uniform, precompiled, network representation of several levels of knowledge (acoustic, syntactic, and semantic)
   - a viable communication strategy between a set of parallel asynchronous processes via a common blackboard arrangement
2. established a network structure for representing task syntax (grammar) along with an understanding of the role of language perplexity on the performance of the recognizer for different tasks
3. established a viable control strategy for determining (via a network search) the best matching valid sentence in the language, using a left-to-right (or bottom up) search strategy mitigated by a beam search whose width was a function of the reliability of the recognition decision at each frame in time
4. established the importance of using semantics and context to reduce the size of the network search and to improve recognition system performance
5. established the value of precompiling the task syntax (at least for small to medium size tasks) and integrating the syntax into the overall network to be searched to find the best matching sentence in the language
6. established the value of a word verification strategy that enabled the system to backtrack and consider alternate paths when the word verification score was not above a threshold of reliability; this strategy also showed the clear benefits of applying phonological rules at word boundaries to account for pronunciation changes due to context
7. established the understanding that phonetic segmentation and labeling, as distinct processes in the recognition system, were not necessary for reliable word recognition in connected speech recognition tasks, e.g., connected digit recognition applications
8. established the value of statistical models of sounds (or words) via models like the hidden Markov model (HMM)
9. established LPC methods as the standard front-end processing for virtually all viable speech recognition systems

Amazon EX1010 Page 547

10. showed that talker normalization could be achieved with as few as 20 training sentences per new talker
11. showed that a full dialogue system, with appropriate response generation to each recognized sentence, was a viable model for human–machine communication systems.

By virtually any account, the ARPA SUR project was one of the most important events in the history of speech recognition and its achievements spurred the research agendas of most labs for more than a decade.

### 26.5.2 Research Outside of the ARPA Community

Outside of the ARPA community, two major efforts in speech recognition research were underway at IBM Research Lab and at AT&T Bell Labs. The two research efforts were aimed in completely different directions, with the effort at IBM striving to build a *voice typewriter* and the effort at Bell Labs striving to build speaker-independent speech recognizers that could be utilized within the telecommunications market for a range of applications such as voice dialing and command-and-control routing of phone calls.

The IBM research effort was led by Fred Jelinek and its goal was to build a system that could convert speech (initially in isolated word format, i. e., sentences spoken a word at a time with a slight pause between words, ultimately as continuous speech sentences) into a sequence of words (and punctuation marks) that could be entered into a text processing program and ultimately displayed on some output medium (computer monitor or paper) [26.41]. The resulting recognition system was essentially a speaker-trained system (requiring an enrollment period for each new talker) and was called the Tangora system [26.42]. The major technical achievements of the IBM research effort were the size of the vocabulary (virtually unlimited in size), with initial intended application being the creation of office correspondence using voice commands, and the creation of a statistically defined grammar. Further, IBM defined the structure of the recognition architecture as a statistical decoding algorithm that sought to maximize the likelihood of the recognized string over the set of all possible sentences that were valid within the language model of the task at hand. Using a statistical model to represent basic speech units, with a lexicon of word pronunciations and the statistical word grammar (called an $n$-gram language model because it proscribed the probability of a given word on the basis of the preceding $n-1$ words), the entire recognition problem was formulated as a maximum-likelihood decoding of the probability of a sentence (word sequence) in the language given the acoustic description of the speech signal (the spectral analysis vector representation of speech over time).

A block diagram of this fundamental approach to speech recognition is given in Fig. 26.1, which shows a sentence $W$ being converted to a speech signal $s[n]$ via the speech production process. The speech signal is then spectrally analyzed (by the acoustic processor) giving the spectral representation, $X = (X_1, X_2, \ldots, X_L)$ for the $L$ frames of the speech signal. The linguistic decoder solves for the maximum likelihood sentence, $\hat{W}$, that best matches $X$ (i. e., maximizes the probability of $W$ given $X$) via the Bayesian formulation:

$$\hat{W} = \arg\max_W P(W|X) \tag{26.1}$$

$$= \arg\max_W \frac{P(X|W)P(W)}{P(X)} \tag{26.2}$$

$$= \underbrace{\arg\max_W}_{\text{Step 3}} \underbrace{P_A(X|W)}_{\text{Step 1}} \underbrace{P_L(W)}_{\text{Step 2}} . \tag{26.3}$$

The maximization of (26.1) is converted to (26.2) using the Bayes rule, and since the denominator term $P(X)$ is independent of $W$, it can be removed leading to the three-step solution of (26.3). Here we explicitly denote the acoustic model by labeling $P(X|W)$ as $P_A(X|W)$, where $A$ denotes the set of acoustic models of the speech units used in the recognizer, and we denote $P(W)$ as $P_L(W)$ for the language model describing the probabilities of various word combinations. The process of determining the maximum-likelihood solution is to first train (offline) the set of acoustic models so that step 1 in (26.3) can be evaluated for each speech utterance. The next step is to train (again offline) the language model for step 2, so that the probability of every word sequence that forms a valid sentence in the language model can be evaluated. Finally step 3 is the heart of the computation, namely a search through all possible combinations of words in the language model to determine the word

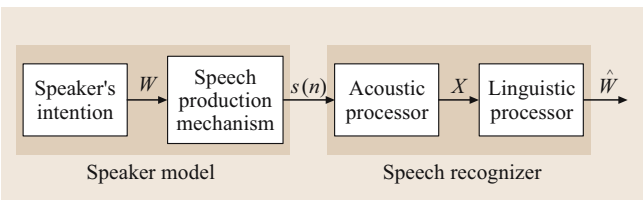



**Fig. 26.1** Block diagram of the statistical model for speech recognition

sequence, $W$, that maximizes the probability, and finally to set the solution, $\hat{W}$ to the maximum likelihood solution obtained in step 3. Very efficient procedures for solving (26.3) have been devised [26.41, 43].

A second large effort aimed at building a speech recognition system that could readily be used by a vast population of talkers (literally tens of millions) was the focus of research at AT&T Bell Labs in the late 1970s. Inherently the system had to be *speaker independent* and had to be able to deal with a range of talkers and regional accents. This requirement led to research into methods for clustering speech patterns (from a large number of talkers) in order to create word and sound reference patterns that could be used broadly [26.22]. Since most of the telecommunications applications of interest were highly constrained tasks, with small vocabularies and only very limited language models, most of the effort was concentrated on creating highly robust acoustic models for words and subword units. Further, since the desired speech recognition systems had to work for speakers who were likely to speak carrier phrases along with the set of valid vocabulary words/phrases that were to be recognized for a given service, it was essential to develop a set of keyword spotting methods [26.44]. For example, a customer might request a credit-card service by either speaking the two-word phrase "credit card" or, alternatively, he/she might say "I'd like to make a credit card call". For highly constrained tasks, like operator services or voice dialing, it is reasonably simple to spot a small set of keywords in continuous speech and provide a highly robust service capability, rather than making the customer repeat the command without the extraneous speech.

The real significance of the speech recognition efforts at IBM and at AT&T Bell labs was that a strong mathematical formalism and rigor guided both efforts, and helped to usher in the fourth generation of speech recognition technology.

## 26.6 Generation 4 – The Era of the Statistical Model

The key technology development of the fourth generation of speech recognition research was a paradigm shift away from the basic pattern recognition methods that used templates and spectral distance-based time alignment of patterns, to a rigorous, mathematically based, statistical modeling framework. Two statistical methodologies evolved over the fourth generation of research, namely the HMM approach (as a complete modeling system for ASR) and the artificial neural network (ANN) approach (as a method for estimating class conditional probability densities). We discuss each of these methodologies separately.

The fundamentals of statistical modeling based on HMMs was known and understood during the third research generation by people at the Institute for Defense Analyses (IDA) in Princeton [26.45], and, by extension, the methodology and theory of the HMM was introduced at Carnegie Mellon University via the work of Jim *Baker* on the DRAGON system [26.35], and extended via the work of the IBM speech recognition research team [26.41]. To commemorate the publication and release of information about the HMM methodology [26.45], a multiday seminar was held at the IDA facilities in Princeton, NJ with attendance by people from a number of speech research laboratories. As a result of this seminar series, work was started at AT&T Bell Labs on investigating the use of HMMs for speech recognition applications in telecom, resulting in the publication of two papers by the team of *Levinson*, *Sondhi*, and *Rabiner* [26.46, 47] that explained the basic theory and gave examples of how to apply the theory to speech recognition problems. The use of HMMs in the speech recognition community grew rapidly, with strong support from the DARPA program, after the publication of these two seminal papers, and by the end of the 1980s the HMM became the preferred model of choice for speech recognition systems. Due to the steady stream of improvements and refinements of the HMM technology, use of HMMs for both automatic speech recognition and natural language understanding systems has grown in popularity and remains a vital component of all speech recognition systems today.

The strength of the HMM methodology lies in the structure of the HMM, namely the doubly stochastic process that models the intrinsic variability of the speech signal (as seen from the analyzed spectral features of each frame) as well as the structure of spoken language, in an integrated and consistent statistical modeling framework [26.48]. The HMM formalism enables a consistent modeling of the linguistic structure of speech via a Markov chain of states, and a consistent statistical description of the variability of the acoustic (spectral) features of speech via a set of probability distributions. Given a training set of a sufficient number of text-labeled

utterances, there exists within the HMM methodology an efficient estimation method, known as the Baum–Welch (or forward–backward) algorithm [26.49], for obtaining the best set of parameters that define the corresponding set of models for the speech recognition task at hand. Thus, the training phase of the HMM methodology is the estimation of the set of HMM model parameters that maximize the likelihood of the labeled utterance, given the measured spectral characterization of the speech utterance. Once the set of models is determined, via the parameter estimation procedure, the resulting models can then be used to score the likelihood of unseen speech utterances (a test set), providing a likelihood (or equivalently a probability) score that an unknown utterance is a realization of the word (or sequence of words) represented by the appropriate set of models. It rapidly became clear to the speech recognition community that the HMM approach to training and scoring of speech recognition systems represented a major leap forward from the simple pattern recognition and acoustic–phonetic methods used in earlier generations of speech recognition technology. As such, the HMM method has become the paradigm of choice for virtually all speech recognition systems since the early 1990s.

One of the strengths of the HMM methodology was its ability to incorporate a larger speech decoding framework, i. e., a language model (including both syntactic and semantic descriptions of the task grammar and semantics), represented as a finite state network (FSN). Figure 26.2 shows an example of the resulting FSN for the utterance "Show all alerts", where the individual speech sounds (context-dependent phonemes in this example) are each represented by HMMs with three state structures, silence is represented as a single state model (which can occur at the beginning and end of the utterance, as well as between words when the speaker pauses) and the grammar network for the sentence is implemented via the concatenation of the phoneme HMM and silence HMMs into a large FSN as illustrated in Fig. 26.2. An efficient method for searching any FSN in an efficient manner was created by *Mohri* in 1997 [26.50]. The resulting search method is based on a unified transducer framework for the incorporation of a weighted search and is called a finite state machine (FSM) search method.

A second key technology that was introduced into speech recognition in generation 4 was the use of neural networks. The key technological advance that brought neural network technology into the forefront (having been initially introduced in the 1940s but with little success [26.51]) was the introduction of a novel

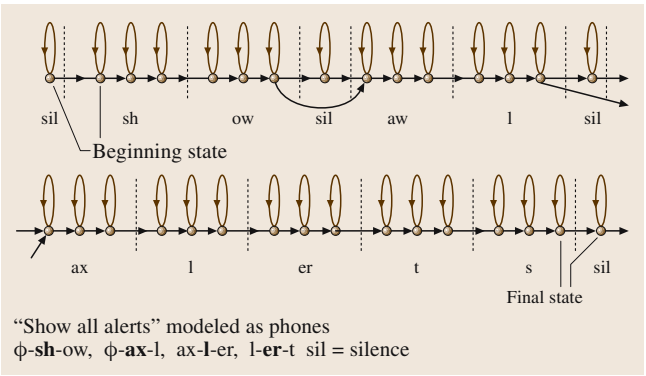



**Fig. 26.2** Composite FSN for the utterance "Show all alerts"

computing structure, known as the parallel distributed processing (PDP) model which was a dense interconnection of simple neural computational elements, and a corresponding training method known as the error backpropagation method. The most popular PDP implementation for speech recognition was the multilayer perceptron (MLP), as shown in Fig. 26.3, because of its proven capability of approximating any function of the input to an arbitrary precision, provided no limitation on the complexity of the processing configuration was imposed. MLP structures were utilized in some simple speech recognition systems (e.g., recognizing isolated and connected digit strings) with good success [26.52], but tended to work poorly in trying to recognize sounds with a lot of temporal variability (e.g., stop consonants) due to the lack of any mechanism for normalizing the time sequence of the speech input accordingly. One proposed solution to this problem of handling temporal variability with neural networks was the time-delay neural network structure proposed by *Waibel* et al. [26.53], which showed that with a proper temporal structure superimposed on an MLP implementation, excellent recognition of time-varying sounds could be achieved.

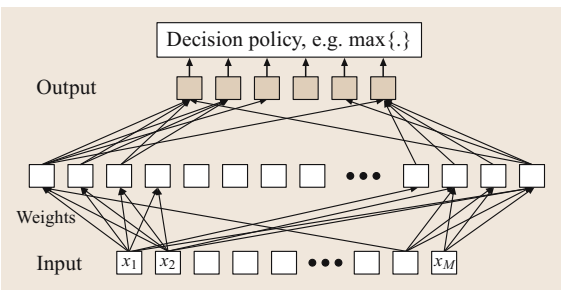



**Fig. 26.3** Structure of a multilayer perceptron (MLP) with one input layer and two processing layers

Also emerging, during this period, was the method of minimum error discriminative training, which improved the performance of maximum likelihood or maximum a posteriori methods of statistical pattern recognition [26.54]. This method shifted the paradigm of estimating the data distribution for speech recognition to one that aimed to minimize the recognition error directly. While HMMs still play a crucial role in representing the individual speech units in discriminative methods, their parameters are optimized to minimize the recognition error rate rather than simply to fit the hidden Markov model (distribution) to the training data. Improvements in speech recognition accuracy have been reported in a number of tasks and applications using this set of methods [26.55, 56].

### 26.6.1 DARPA Programs in Generation 4

Toward the end of the 1980s, it became clear to DARPA that speech recognition technology had reached a number of key milestones and was ready to be challenged to go to the next level, namely towards a machine that could both recognize the spoken words and understand the meaning behind the recognized sentence. Further, it was felt that success with machine understanding of speech could lead to a new generation of speech technology services that could communicate with a human and perform some desired task. As such DARPA created a new, multi-year program on large vocabulary, continuous speech recognition and understanding that ultimately led to several new speech recognition and understanding systems including:

- The CMU Sphinx system [26.57], which marked the first truly speaker-independent large-vocabulary continuous speech recognition system which worked well on a range of tasks with different vocabularies, language models, language perplexities, etc. The Sphinx system successfully integrated HMMs with network search. This system was able to train and embed context-dependent phone models in a sophisticated lexical decoding network.
- The BBN BYBLOS system [26.58] which was the first HMM-based system in the DARPA program [26.59]. This system demonstrated the effectiveness of using phonetic context to improve recognition accuracy [26.60].
- The SRI DECIPHER system [26.61].

DARPA selected and defined a broad range of speech recognition and natural language understanding tasks, throughout the 1990s and into the 2000s, in order to stimulate new thinking about how to make such systems highly reliable, accurate, and robust. Regular and periodic formal evaluations of the 3 DARPA-supported systems mentioned above, along with outside systems from AT&T Bell Labs (subsequently AT&T Labs Research), Cambridge University in England, IBM, MIT, and other institutions were conducted. These formal evaluations generally measured word and sentence error rates (for both the recognition and the natural language understanding components of each system) as the performance figure of merit.

A series of tasks of increasing complexity and difficulty were devised by DARPA and were serially introduced into the speech recognition community over a period of 10–15 years. The characteristics of these tasks were the following:

*Resource Management (RM).* This task could be used in a military environment in order to query a ships database about the locations and properties of naval ships throughout the world. The vocabulary was about 1000 words, and the spoken queries were read from a computer-generated list of possible commands to the system. The word error rate at the end of the trials for this task was on the order of 2%, a most remarkable achievement for its time, and was achieved by almost all the systems working on this task.

*Airline Travel Information System (ATIS).* This task enabled the user to formulate and make travel plans via an interactive dialogue with the task (which operated using an approximation to a real airline's schedule of flights). The speech mode for this task was essentially spontaneous (allowing the user to formulate the way in which each query was made), thereby causing the recognition system to have to handle out-of-vocabulary (OOV) words without derailing the meaning of the in-vocabulary speech. The task vocabulary was about 2500 words and the word error rate, at the end of the trials for this task, was about 2.5% word error rate, again a remarkable achievement on a considerably harder task than RM.

*North American Business (NAB).* This task enabled the user to speak a business story from one of several sources (e.g., the Wall Street Journal). The sentences were read directly from the NAB source journal, with a resulting vocabulary of about 64 000 words. Again there was an issue with OOV words, as well as new words which appeared in the news rather frequently as events in the world changed on a day-to-day basis. However, in spite

of these difficulties, the ultimate performance on this task was a respectable 6.6% word error rate.

*Broadcast News (BN).* This task literally picked up over-the-air news broadcasts from the Consumer News and Business Channel (CNBC) and other news stations, and provided a running conversion from speech to text at the bottom of the TV screen. The vocabulary size was about 210 000 words and the speech was conversational (or, more likely, news read from a teleprompter). On this task the final word error rate was about 13–17%, a most respectable performance on this very difficult task.

*Switchboard (SB).* This task utilized speech from a standard telecom switchboard, namely conversations between two individuals over a standard telephone channel. The speech was conversational speech, of telephone quality and bandwidth, and the vocabulary size was about 45 000 words. The final word error rate on this task was in the 25–29% range, reflecting the degree of difficulty of this task.

*Call Home (CH).* This task utilized conversational speech from individuals calling their home telephone number, and therefore speaking to someone within their family. Such conversational speech is filled with ungrammatical snippets, short sentences, etc., due to the inherent informality in the conversation between two family members. The vocabulary for this system was about 28 000 words and the resulting word error rate was about 40%, reflecting the extreme difficulty of recognizing the colloquial level of conversational speech associated with this task.

The general conclusion that can be made from the results of this series of DARPA tasks is that conversational speech, which clearly does not strictly adhere to normal linguistic constraints, is significantly more difficult to recognize (and understand) than read speech, or speech that follows a scenario or well-defined task, i. e., speech that obeys a set of well-defined syntactic and semantic rules. Also, the DARPA program consistently showed that increasing the amount of speech data used for training each of the tasks always led to reductions in the word error rate of the system. This seems to indicate that the capacity of mixture density hidden Markov models, in representing speech for the purpose of recognition, has not been reached. Also, progress was made in the DARPA program through the use of minimum-error training methods for large-vocabulary continuous speech recognition tasks [26.55].

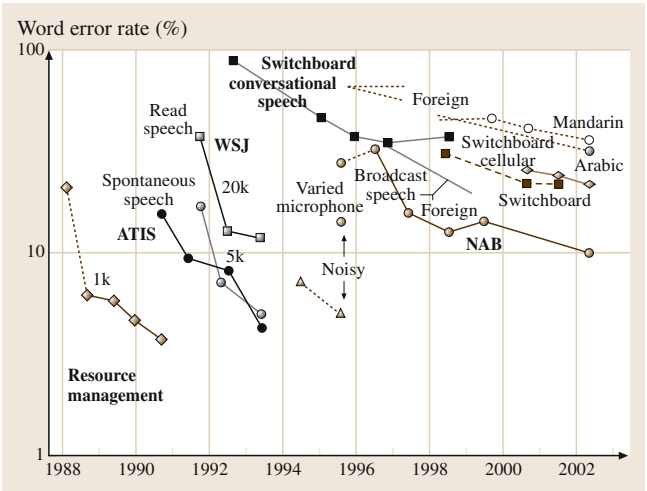



**Fig. 26.4** Word error rate performance over time for a range of DARPA large-vocabulary speech recognition tasks (courtesy NIST 1999 DARPA HUB-4 Rep., *Pallett* et al. and updates from DARPA)

Figure 26.4 shows a chart that summarizes the benchmark performance of several of these large-vocabulary continuous speech recognition tasks over time [26.62]. The performance scores represent the best performance among all the submissions to standard DARPA/National Institute of Standards and Technology (NIST) evaluation tests. The most startling feature of this chart is the clear evidence that, over time, the performance of each individual task improved at almost a common rate, independent of the task complexity or task vocabulary or mode of speech input. This continuous performance improvement over time has led IBM to try to see if they can track a single task over a very long period of time and achieve performance that rivals or exceeds that of a human listener.

One fallout from the DARPA program in the 1990s and beyond was the development of sophisticated software tools that rapidly became indispensable for advanced research and development of new concepts and algorithms. Fortunately several creators of such software tools have made their source code available for use in research environments. These software toolkits for speech recognition and natural language understanding include:

1. HTK (hidden Markov model toolkit), the HMM toolkit of Cambridge University [26.63]
2. Sphinx from Carnegie Mellon University [26.64]
3. the HMM toolkit from Mississippi State University [26.65]

## 26.7 Generation 5 – The Future

Several important things were learned about the speech communication process between humans and machines during the first four generations of speech recognition research and the three generations of natural language understanding research. We learned that, when speaking naturally to a machine, humans tended to speak sentences that often did not obey the grammatical constraints of the recognizer (e.g., OOV words, nongrammatical constructs, ill-formed sentences, etc.). We also learned that spoken utterances were often corrupted by linguistically irrelevant *noise* (including extraneous acoustic background sounds, interfering speech, background noise, etc.). Finally we learned that, in order to successfully complete any interesting or important task, we needed to provide the tools to maintain a dialog between the user and the machine in order to reach some desired state of understanding. Such a dialog system generally required operations such as query and confirmation, thus providing some leeway for speech recognition and understanding errors. Such dialog systems have been widely used in modern speech recognition systems and some examples include the *How May I Help You* system of AT&T and the experimental demonstrations systems (Pegasus and Jupiter) developed at MIT.

We have no simple solutions to the problems that remain. Robustness to noise and other external artifacts of speaking remains a challenge that is being attacked at the signal processing level (using algorithms like spectral subtraction) [26.66], at the feature level (using algorithms like cepstral mean subtraction), and at the model level (using model adaptation methods, [26.67]), but to date none of these fixes has really solved the problem [26.68]. We look to improved models of spectral analysis, especially those related to auditory modeling [26.69], to provide some relief in this dimension, but for the foreseeable future, this remains a key challenge to ubiquitous use of speech recognition and understanding systems.

The solution to the problem of nongrammatical speech with use of OOV words remains another difficult challenge for the future. Interestingly, one possible way of handling this type of problem is the use of multiple models for recognition. For this direction, a standard generation 4 approach would first try to recognize the spoken input, providing a list of the $N$-best distinct sentence matches. A secondary speech model, perhaps based on a more-conventional acoustic–phonetic model of speech, might provide a set of secondary recognition scores that could be utilized whenever the word verification score of the first model falls below a reliability threshold. Systems of this type are being examined and their ability to detect and correct linguistic errors is being studied [26.70].

## 26.8 Summary

Figure 26.5 shows a chart that outlines the progress that has been made in speech recognition and natural language understanding technology over the past four decades. The degree of progress in technology and system complexity is clear from this chart. We see that in the 1960s we were able to recognize small vocabularies (order of 10–100 words) of isolated words, based on simple acoustic–phonetic properties of speech sounds. The key technologies that were developed during this time frame were filter bank analyses, simple time normalization methods, and the beginnings of sophisticated dynamic programming methodologies. In the 1970s we were able to recognize medium-sized vocabularies (100–1000 words) using simple template-based, pattern recognition methods. The key technologies that were developed during this period were the pattern recognition models, the introduction of LPC methods for spectral representation, the pattern clustering methods for speaker-independent systems, and the introduction of dynamic programming methods for solving connected word recognition problems. In the 1980s we started to tackle large vocabulary (1000–unlimited number of words) speech recognition problems based on statistical methods, with a wide range of networks for handling language structures. The key technologies introduced during this period were the hidden Markov model and the stochastic language model, which together enabled powerful new methods for handling virtually any continuous speech recognition problem efficiently and with high performance. In the 1990s, we were able to build large-vocabulary systems with unconstrained language models, and constrained task syntax models for continuous speech recognition and natural language understanding. The key technologies devel-

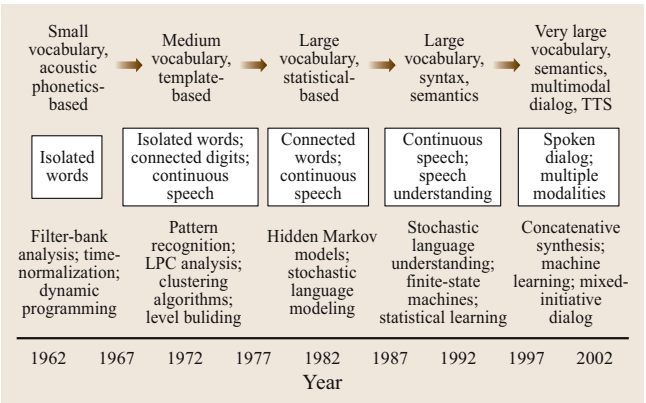



**Fig. 26.5** Milestones in speech recognition and understanding technology over the past 40 years

oped during this period were the methods for stochastic language understanding, statistical learning of acoustic and language models, and the introduction of the finite state transducer framework (and the resulting FSM library) and the methods for their determination and minimization for efficient implementation of large vocabulary speech understanding systems. Finally, in the last few years, we have seen the introduction of very large vocabulary systems with full semantic models, integrated with text-to-speech (TTS) synthesis systems, and multimodal inputs (including pointing, keyboards, mice, etc.). These systems enable spoken dialog systems with a range of input and output modalities for ease-of-use and flexibility in handling adverse environments where speech might not be as suitable as other input-output modalities. During this period, we have seen the emergence of highly natural concatenative speech synthesis systems, the use of machine learning to improve both speech understanding and speech dialogs, and the introduction of mixed initiative dialog systems to enable user control of the dialog, when necessary.

The challenge of designing a machine that truly functions like an intelligent human is still a major one going forward. Our accomplishments to date are only the beginning and it will take many years before a machine can pass the Turing test, namely achieving performance that rivals that of a human.

## References


26.1 J.G. Wilpon: Applications of voice-processing technology in telecommunications. In: *Voice Communications Between Humans and Machines*, ed. by D.B. Roe, J.G. Wilpon (National Academy, Washington 1994) pp. 280–310

26.2 J.G. Wilpon, D.B. Roe: AT&T telephone network applications of speech recognition, Proc. Nat. Acad. Sci. United States Am., Vol. 92 (1995)

26.3 V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, L. Hetherington: Jupiter: A telephone-based conversational interface for weather information, IEEE Trans. Speech Audio Process. **10**, 100–112 (2000)

26.4 V. Zue, S. Seneff, J. Polifroni, M. Phillips, C. Pao, D. Goddeau, J. Glass, E. Brill: Pegasus: A spoken language interface for on-line air travel planning, Proc. Workshop on Human Language Technology, ACM Digital Library (1994) pp. 201–206

26.5 A.L. Gorin, B. Parker, R. Sachs, J.G. Wilpon: How may I help you, Proc. Interactive Voice Technology for Telecommunications Applications (IVTTA) (1996) pp. 57–60

26.6 A.L. Gorin, G. Riccardi, J.H. Wright: How may I help you, Speech Commun. **23**(1), 113–127 (1997)

26.7 H. Fletcher: The nature of speech and its interpretations, Bell Syst. Tech. J. **1**, 129–144 (1922)

26.8 H. Dudley: The vocoder, Bell Labs Rec. **17**, 122–126 (1939)

26.9 H. Dudley, R.R. Riesz, S.A. Watkins: A synthetic speaker, J. Franklin Inst. **227**, 739–764 (1939)

26.10 R.K. Potter, G.A. Kopp, H.C. Green: *Visible Speech* (van Nostrand, Amsterdam 1947)

26.11 K.H. David, R. Biddulph, S. Balashek: Automatic recognition of spoken digits, J. Acoust. Soc. Am. **24**(6), 627–642 (1952)

26.12 H.F. Olson, H. Belar: Phonetic typewriter, J. Acoust. Soc. Am. **28**(6), 1071–1081 (1956)

26.13 J.W. Forgie, C.D. Forgie: Results obtained from a vowel recognition computer program, J. Acoust. Soc. Am. **31**(11), 1480–1489 (1959)

26.14 P.B. Denes, D.B. Fry: The design and operation of the mechanical speech recognizer at university college, J. Br. Inst. Radio Eng. **19**(4), 211–229 (1959)

26.15 B.S. Atal, S.L. Hanauer: Speech analysis and synthesis by linear prediction of the speech wave, J. Acoust. Soc. Am. **50**(2), 637–655 (1971)

26.16 F. Itakura, S. Saito: A statistical method for estimation of speech spectra density and formant frequencies, Electron. Commun. Jpn. **53A**, 36–43 (1970)

26.17 F. Itakura: Minimum prediction residual principle applied to speech recognition, IEEE Trans. Acoust. Speech Signal Process. **ASSP-23**, 57–72 (1975)

26.18 T.K. Vintsyuk: Speech Discrimination by Dynamic Programming, Kibernetika **4**(2), 81–88 (1968)

26.19 H. Sakoe, S. Chiba: Dynamic programming algorithm quantization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. **ASSP-26**(1), 43–49 (1978)

26.20 Y. Linde, A. Buzo, R.M. Gray: An algorithm for vector quantizer design, IEEE Trans. Commun. **COM-28**, 84–95 (1980)

26.21 A. Gersho, R.M. Gray: *Vector Quantization and Signal Compression* (Kluwer Academic, Boston 1991)

26.22 L.R. Rabiner, S.E. Levinson, A.E. Rosenberg, J.G. Wilpon: Speaker independent recognition of isolated words using clustering techniques, IEEE Trans. Acoust. Speech Signal Process. **ASSP-27**, 336–349 (1979)

26.23 J. Suzuki, K. Nakata: Recognition of Japanese vowels – Preliminary to the recognition of speech, J. Radio Res. Lab. **37**(8), 193–212 (1961)

26.24 T. Sakai, S. Doshita: The phonetic typewriter, Information Processing 1962, Proc. IFIP Congress (1962) pp. 445–450

26.25 K. Nagata, Y. Kato, S. Chiba: Spoken digit recognizer for Japanese language, J. Audio Eng. Soc. **12**(4), 336–342 (1964)

26.26 T.B. Martin, A.L. Nelson, H.J. Zadell: *Speech Recognition by Feature Abstraction Techniques*, Tech. Report AL-TDR-64-176 (Air Force Avionics Lab, 1964)

26.27 D.R. Reddy: *An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave*, Tech. Report No. C549 (Stanford University, 1966)

26.28 V.M. Velichko, N.G. Zagoruyko: Automatic recognition of 200 words, Int. J. Man-Machine Studies **2**, 222–234 (1970)

26.29 S.R. Hyde: Automatic speech recognition: A critical survey and discussion of the literature. In: *Human Communication: A Unified View*, ed. by E.E. David Jr., P.B. Denes (McGraw Hill, New York 1972) pp. 399–438

26.30 D.H. Klatt: Review of the ARPA speech understanding project, J. Acoust. Soc. Am. **62**, 1345–1366 (1977)

26.31 J. Makhoul: Spectral analysis of speech by linear prediction, IEEE Trans. Audio Electroacoust. **AU-21**(3), 140–148 (1973)

26.32 H. Wakita: Direct estimation of the vocal tract shape by inverse filtering of acoustic speech waveforms, IEEE Trans. Audio Electroacoust. **AU-21**(5), 417–427 (1973)

26.33 J.D. Markel, A.H. Gray Jr.: *Linear Prediction of Speech* (Springer, Berlin, Heidelberg 1976)

26.34 A.H. Gray Jr., J.D. Markel: Distance measures for speech processing, IEEE Trans. Acoust. Speech Signal Process. **ASSP-24**(5), 380–391 (1976)

26.35 J.K. Baker: The DRAGON system – An overview, IEEE Trans. Acoust. Speech Signal Process. **ASSP-23**, 24–29 (1975)

26.36 L.R. Rabiner, M.R. Sambur: An algorithm for determining the endpoints of isolated utterances, Bell Syst. Tech. J. **54**(2), 297–315 (1975)

26.37 B.T. Oshika, V. Zue, R. Weeks, H. Neu, J. Aurbach: The role of phonological rules in speech understanding research, IEEE Trans. Acoust. Speech Signal Process. **ASSP-23**(1), 104–112 (1975)

26.38 P.S. Cohen, R.L. Mercer: The phonological component of an automatic speech recognition system. In: *Speech Recognition*, ed. by D.R. Reddy (Academic, New York 1975)

26.39 B. Lowerre: The HARPY speech understanding system. In: *Readings in Speech Recognition*, ed. by A. Weibel, K.F. Lee (Morgan Kaufmann, Palo Alto 1990) pp. 576–586

26.40 G.M. White: Speech recognition: A tutorial overview, Computer **9**, 40–53 (1976)

26.41 F. Jelinek, L.R. Bahl, R.L. Mercer: Design of a linguistic statistical decoder for the recognition of continuous speech, IEEE Trans. Inform. Theory **IT-21**, 250–256 (1975)

26.42 S.K. Das, M.A. Picheny: Issues in practical large vocabulary isolated word recognition: The IBM tangora system. In: *Automatic Speech and Speaker Recognition Advanced Topics*, ed. by C.H. Lee, F.K. Soong, K.K. Paliwal (Kluwer, Boston 1996) pp. 457–479

26.43 L.R. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE **77**(2), 257–286 (1989)

26.44 J.G. Wilpon, L.R. Rabiner, C.H. Lee, E.R. Goldman: Automatic recognition of keywords in unconstrained speech using hidden Markov models, IEEE Trans. Acoust. Speech Signal Process. **38**(11), 1870–1878 (1990)

26.45 J.D. Ferguson: Hidden Markov analysis: An introduction. In: *Hidden Markov Models for Speech*, ed. by J.D. Ferguson (Institute for Defense Analyses, Princeton 1980)

26.46 S.E. Levinson, L.R. Rabiner, M.M. Sondhi: An introduction to the application of the theory of probabilistic functions of a Markov process, Bell Syst. Tech. J. **62**(4), 1035–1074 (1983)

26.47 L.R. Rabiner, S.E. Levinson, M.M. Sondhi: On the application of vector quantization and hidden Markov models to speaker independent, isolated word recognition, Bell Syst. Tech. J. **62**(4), 1075–1105 (1983)

26.48 L.R. Rabiner, B.H. Juang: Statistical methods for the recognition and understanding of speech. In: *Encyclopedia of Language and Linguistics*, ed. by K. Brown, J. Lai (Elsevier, Amsterdam 2005)

26.49 L.E. Baum: An inequality and associated maximization technique in statistical estimation for


Amazon EX1010 Page 555

probabilistic functions of Markov processes, Inequalities **3**, 1–8 (1972)
26.50 M. Mohri: Finite-state transducers in language and speech processing, Comput. Linguist. **23**(2), 269–312 (1997)
26.51 W.S. McCullough, W.H. Pitts: A logical calculus of ideas imminent in nervous activity, Bull. Math. Biophys. **5**, 115–133 (1943)
26.52 R.P. Lippmann: Review of neural networks for speech recognition. In: *Readings in Speech Recognition*, ed. by A. Weibel, K.F. Lee (Morgan Kaufmann, Palo Alto 1990) pp. 374–392
26.53 A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang: Phoneme recognition using time-delay neural networks, IEEE Trans. Acoust. Speech Signal Proc. **37**(3), 328–339 (1989)
26.54 B.H. Juang, S. Katagiri: Discriminative learning for minimum error classification, IEEE Trans. Signal Process. **40**(12), 3043–3054 (1992)
26.55 B.H. Juang, W. Chou, C.H. Lee: Minimum classification error rate methods for speech recognition, IEEE Trans. Speech Audio Process. **5**(3), 257–265(1997)
26.56 H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, G. Zweig: The IBM 2004 conversational telephony system for rich transcription, Proc. ICASSP, Vol. 2005 (2005) pp. I–205–I–208
26.57 K.F. Lee: *Large Vocabulary Speaker Independent Continuous Speech Recognition: The Sphinx System*, Ph.D. Thesis (Carnegie Mellon University, Pittsburgh 1988)
26.58 R. Schwartz, C. Barry, Y.-L. Chow, A. Derr, M.-W. Feng, O. Kimball, F. Kubala, J. Makhoul, J. Vandegrift: The BBN BYBLOS continuous speech recognition system, Proc. Speech and Natural Language Workshop (1989) pp. 94–99
26.59 J. Makhoul: Speech processing at BBN, IEEE Ann. Hist. Comput. **28**(1), 32–45 (2006)
26.60 R.M. Schwartz, Y.-L. Chow, O. Kimball, S. Roucos, M. Krasner, J. Makhoul: Context-dependent modeling for acoustic-phonetic recognition of continuous speech, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1985) pp. 1205–1208
26.61 H. Murveit, M. Cohen, P. Price, G. Baldwin, M. Weintraub, J. Bernstein: SRI's DECIPHER system, Proc. Speech and Natural Language Workshop (1989) pp. 238–242
26.62 D.S. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, A. Martin, A. Przybocki: The 1994 benchmark tests for the ARPA spoken language program, Proc. 1995 ARPA Human Language Technology Workshop (1995) pp. 5–38
26.63 S. Young: *The HTK Book* , http://htk.eng.cam.ac.uk/
26.64 Carnegie Mellon University: http://cmusphinx.soundforge.net/sphinx4/
26.65 J. Piccone: *http://www.isip.msstate.edu/projects/speech/* (Mississippi State University )
26.66 S.F. Boll: Suppression of acoustic noise in speech using spectral subtraction, IEEE Trans. Acoust. Speech Signal Process. **ASSP-27**, 113–120 (1979)
26.67 B.S. Atal: Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification, J. Acoust. Soc. Am. **55**(6), 1304–1312 (1974)
26.68 A. Acero, J. Droppo: Robustness in speech recognition. In: *Springer Handbook on Speech Processing and Speech Communication*, ed. by J. Benesty (Springer, Berlin, Heidelberg 2007)
26.69 S. Seneff: A joint synchrony/mean-rate model of auditory speech processing, J. Phonetics **16**, 55–76 (1988)
26.70 J. Hou, L.R. Rabiner, S. Dusan: Automatic speech attribute transcription (ASAT) – The front end processor. In: *Proc. Int. Conf. Acoust. Speech Signal Process.* (IEEE, 2006)

Amazon EX1010 Page 556

# 27. HMMs and Related Speech Recognition Technologies

S. Young

Almost all present-day continuous speech recognition (CSR) systems are based on hidden Markov models (HMMs). Although the fundamentals of HMM-based CSR have been understood for several decades, there has been steady progress in refining the technology both in terms of reducing the impact of the inherent assumptions, and in adapting the models for specific applications and environments. The aim of this chapter is to review the core architecture of an HMM-based CSR system and then outline the major areas of refinement incorporated into modern systems.

27.1 **Basic Framework** ..... 539
27.2 **Architecture of an HMM-Based Recognizer** ..... 540
27.2.1 Feature Extraction ..... 540
27.2.2 HMM Acoustic Models ..... 541
27.2.3 *N*-Gram Language Models ..... 543
27.2.4 Decoding and Lattice Generation .. 544
27.3 **HMM-Based Acoustic Modeling** ..... 547
27.3.1 Discriminative Training ..... 547
27.3.2 Covariance Modeling ..... 549
27.4 **Normalization** ..... 550
27.4.1 Mean and Variance Normalization . 550
27.4.2 Gaussianization ..... 550
27.4.3 Vocal-Tract-Length Normalization . 551
27.5 **Adaptation** ..... 551
27.5.1 Maximum A Posteriori (MAP) Adaptation ..... 552
27.5.2 ML-Based Linear Transforms ..... 552
27.5.3 Adaptive Training ..... 553
27.6 **Multipass Recognition Architectures** ..... 554
27.7 **Conclusions** ..... 554
**References** ..... 555



## 27.1 Basic Framework

Automatic continuous speech recognition (CSR) is sufficiently mature that a variety of real-world applications are now possible including command and control, dictation, transcription of recorded speech, and interactive spoken dialogues. This chapter describes the statistical models that underlie current-day systems: specifically, the hidden Markov model (HMM) and its related technologies.

The foundations of modern HMM-based continuous speech recognition technology were laid down in the 1970s by groups at Carnegie-Mellon, IBM, and Bell Labs [27.1–3]. Reflecting the computational power of the time, initial development in the 1980s focussed on whole-word small-vocabulary applications [27.4, 5]. In the early 1990s, attention switched to continuous speaker-independent recognition. Starting with the artificial 1000 word *resource management* task [27.6], the technology developed rapidly and by the mid-1990s, reasonable accuracy was being achieved for unrestricted dictation. Much of this development was driven by a series of Defense Advanced Research Projects Agency (DARPA) and National Security Agency (NSA) programmes [27.7], which set ever more challenging tasks, culminating most recently in systems for multilingual transcription of broadcast news programmes [27.8], and for spontaneous telephone conversations [27.9].

Although the basic framework for CSR has not changed significantly in the last 10 years, the detailed modeling techniques developed within this framework have evolved to a state of considerable sophistication (e.g., [27.10, 11]). The result has been steady and significant progress and it is the aim of this chapter to describe the main techniques by which this has been achieved. Many research groups have contributed to this progress, and each will typically have their own architectural perspective. For the sake of logical coherence, the presentation given here is somewhat biassed towards the architecture developed at Cambridge and supported by the HTK Software Toolkit [27.12] (available for free download at htk.eng.cam.ac.uk).

The chapter is organized as follows. In Sect. 27.2, the core architecture of a typical HMM-based recog-

nizer is described [27.13]. Subsequent sections then describe the various improvements that have been made to this core over recent years. Section 27.3 discusses methods of HMM parameter estimation and issues relating to improved covariance modeling. In Sects. 27.4 and 27.5, methods of normalization and adaptation are described which allow HMM-based acoustic models to more accurately represent specific speakers and environments. Finally in Sect. 27.6, the multipass architecture refinements adopted by modern transcription system is described. The chapter concludes in Sect. 27.7 with some general observations and conclusions.

## 27.2 Architecture of an HMM-Based Recognizer

The principal components of a large-vocabulary continuous speech recognizer are illustrated in Fig. 27.1. The input audio waveform from a microphone is converted into a sequence of fixed-size acoustic vectors $\boldsymbol{Y} = \boldsymbol{y}_1, \cdots, \boldsymbol{y}_T$ in a process called feature extraction. The decoder then attempts to find the sequence of words $W = w_1, \cdots, w_K$ that is most likely to have generated $\boldsymbol{Y}$, i. e., the decoder tries to find

$$\hat{W} = \arg\max_W [p(W|\boldsymbol{Y})] \,. \tag{27.1}$$

However, since $p(W|\boldsymbol{Y})$ is difficult to model directly, Bayes' rule is used to transform (27.1) into the equivalent problem of finding:

$$\hat{W} = \arg\max_W [p(\boldsymbol{Y}|W)p(W)] \,. \tag{27.2}$$

The likelihood $p(\boldsymbol{Y}|W)$ is determined by an *acoustic model* and the prior $p(W)$ is determined by a *language model*. In practice, the acoustic model is not normalized and the language model is often scaled by an empirically determined constant and a word-insertion penalty is added, i. e., in the log domain the total likelihood is calculated as $\log p(\boldsymbol{Y}|W) + \alpha p(W) + \beta|W|$ where $\alpha$ is typically in the range 8–20 and $\beta$ is typically in the range 0 to −20. The basic unit of sound represented by the acoustic model is the *phone*. For example, the word *bat* is composed of three phones /b/ /ae/ /t/. About 40 such phones are required for English.

For any given $W$, the corresponding acoustic model is synthesized by concatenating phone models to make words as defined by a pronunciation dictionary. The parameters of these phone models are estimated from training data consisting of speech waveforms and their orthographic transcriptions. The language model is typically an $N$-gram model in which the probability of each word is conditioned only on its $N-1$ predecessors. The $N$-gram parameters are estimated by counting $N$-tuples in appropriate text corpora. The decoder operates by searching through all possible word sequences using pruning to remove unlikely hypotheses thereby keeping the search tractable. When the end of the utterance is reached, the most likely word sequence is output. Alternatively, modern decoders can generate lattices containing a compact representation of the most likely hypotheses.

The following sections describe these processes and components in more detail.

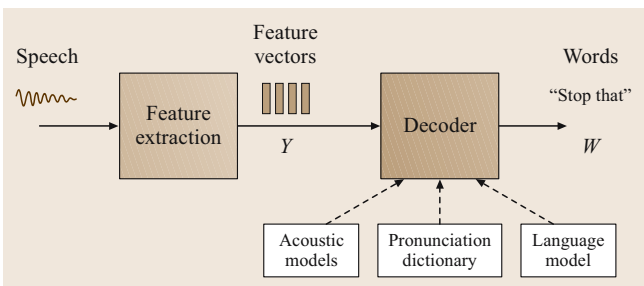



**Fig. 27.1** Architecture of an HMM-based recognizer

### 27.2.1 Feature Extraction

The feature extraction stage seeks to provide a compact encoding of the speech waveform. This encoding should minimize the information loss and provide a good match with the distributional assumptions made by the acoustic models. Feature vectors are typically computed every 10 ms using an overlapping analysis window of around 25 ms. One of the simplest and most widely used encoding schemes uses *mel-frequency cepstral coefficients (MFCCs)* [27.14]. These are generated by applying a truncated cosine transformation to a log spectral estimate computed by smoothing an FFT with around 20 frequency bins distributed nonlinearly across the speech spectrum. The nonlinear frequency scale used is called a *mel scale* and approximates the response of the human ear. The cosine transform is applied in order to smooth the spectral estimate and decorrelate the feature elements.

Further psychoacoustic constraints are incorporated into a related encoding called *perceptual linear prediction (PLP)* [27.15]. PLP computes linear prediction coefficients from a perceptually weighted nonlinearly

compressed power spectrum and then transforms the linear prediction coefficients to cepstral coefficients. In practice, PLP can give small improvements over MFCCs, especially in noisy environments and hence it is the preferred encoding for many systems.

In addition to the spectral coefficients, first-order (delta) and second-order (delta–delta) regression coefficients are often appended in a heuristic attempt to compensate for the conditional independence assumption made by the HMM-based acoustic models. The final result is a feature vector whose dimensionality is typically around 40 and which has been partially but not fully decorrelated.

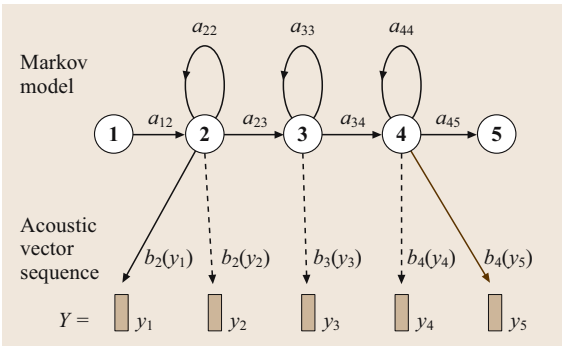



**Fig. 27.2** HMM-based phone model

## 27.2.2 HMM Acoustic Models

As noted above, the spoken words in $W$ are decomposed into a sequence of basic sounds called *base phones*. To allow for possible pronunciation variation, the likelihood $p(\boldsymbol{Y}|W)$ can be computed over multiple pronunciations

$$p(\boldsymbol{Y}|W) = \sum_{Q} p(\boldsymbol{Y}|Q)p(Q|W) \,. \tag{27.3}$$

Recognizers often approximate this by a max operation so that alternative pronunciations can be decoded as though they were alternative word hypotheses. Each $Q$ is a sequence of word pronunciations $Q_1, \cdots, Q_K$ where each pronunciation is a sequence of base phones $Q_k = q_1^{(k)} q_2^{(k)} \cdots$. Then

$$p(Q|W) = \prod_{k=1}^{K} p(Q_k|w_k) \,, \tag{27.4}$$

where $p(Q_k|w_k)$ is the probability that word $w_k$ is pronounced by base phone sequence $Q_k$. In practice, there will only be a very small number of possible $Q_k$ for each $w_k$ making the summation in (27.3) easily tractable.

Each base phone $q$ is represented by a continuous density hidden Markov model (HMM) of the form illustrated in Fig. 27.2 with transition parameters $\{a_{ij}\}$ and output observation distributions $\{b_j()\}$. The latter are typically mixtures of Gaussians

$$b_j(\boldsymbol{y}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \,, \tag{27.5}$$

where $\mathcal{N}$ denotes a normal distribution with mean $\boldsymbol{\mu}_{jm}$ and covariance $\boldsymbol{\Sigma}_{jm}$, and the number of components $M$ is typically in the range 10 to 20. Since the dimensionality of the acoustic vectors $\boldsymbol{y}$ is relatively high, the covariances are usually constrained to be diagonal. The entry and exit states are *nonemitting* and they are included to simplify the process of concatenating phone models to make words.

Given the composite HMM $Q$ formed by concatenating all of its constituent base phones, the acoustic likelihood is given by

$$p(\boldsymbol{Y}|Q) = \sum_{X} p(X, \boldsymbol{Y}|Q) \,, \tag{27.6}$$

where $X = x(0), \cdots, x(T)$ is a state sequence through the composite model and

$$p(X, \boldsymbol{Y}|Q) = a_{x(0),x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{y}_t) a_{x(t),x(t+1)} \,. \tag{27.7}$$

The acoustic model parameters $\{a_{ij}\}$ and $\{b_j()\}$ can be efficiently estimated from a corpus of training utterances using expectation maximization (EM) [27.16]. For each utterance, the sequence of base forms is found and the corresponding composite HMM constructed. A forward–backward alignment is used to compute state occupation probabilities (the E step) and the means and covariances are then estimated via simple weighted averages (the M step) [27.12, Chap. 7]. This iterative process can be initialized by assigning the global mean and covariance of the data to all Gaussian components and setting all transition probabilities to be equal. This gives a so-called *flat start* model. The number of component Gaussians in any mixture can easily be increased by cloning, perturbing the means and then re-estimating using EM.

This approach to acoustic modeling is often referred to as the *beads-on-a-string* model, so-called because all speech utterances are represented by concatenating a sequence of phone models together. The major problem with this is that decomposing each vocabulary word into

a sequence of context-independent base phones fails to capture the very large degree of context-dependent variation that exists in real speech. For example, the base form pronunciations for 'mood' and 'cool' would use the same vowel for 'oo', yet in practice the realizations of 'oo' in the two contexts are very different due to the influence of the preceding and following consonant. Context-independent phone models are referred to as *monophones*.

A simple way to mitigate this problem is to use a unique phone model for every possible pair of left and right neighbors. The resulting models are called *triphones* and, if there are $N$ base phones, there are logically $N^3$ potential triphones. To avoid the resulting data sparsity problems, the complete set of *logical* triphones $L$ can be mapped to a reduced set of physical models $P$ by clustering and tying together the parameters in each cluster. This mapping process is illustrated in Fig. 27.3 and the parameter tying is illustrated in Fig. 27.4 where the notation x-q+y denotes the triphone corresponding to phone q spoken in the context of a preceding phone x and a following phone y. The base phone pronunciations Q are derived by simple look-up from the pronunciation dictionary; these are then mapped to logical phones according to the context, and finally the logical phones are mapped to physical models. Notice that the context dependence spreads across word boundaries and this is essential to capture many important phonological processes. For example, the [p] in *stop that* has its burst suppressed by the following consonant.

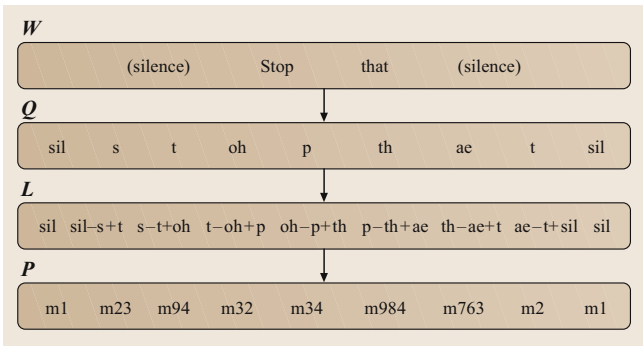



**Fig. 27.3** Context-dependent phone modeling

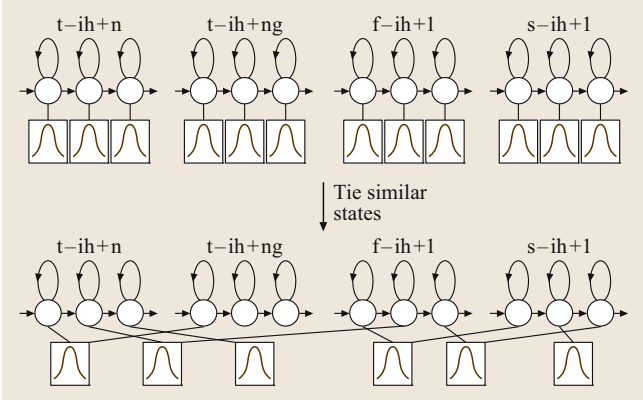



**Fig. 27.4** Formation of tied-state phone models

The clustering of logical to physical models typically operates at the state level rather than the model level since it is simpler and it allows a larger set of physical models to be robustly estimated. The choice of which states to tie is made using decision trees [27.17]. Each state position of each phone $q$ has a binary tree associated with it. Invariably each phone model has three states. Each node of the tree carries a question regarding the context. To cluster state $i$ of phone $q$, all states $i$ of all of the logical models derived from $q$ are collected into a single pool at the root node of the tree. Depending on the answer to the question at each node, the pool of states is successively split until all states have trickled down to leaf nodes. All states in each leaf node are then tied to form a physical model. The questions at each node are selected from a predetermined set to maximize the likelihood of the training data given the final set of state tyings. If the state output distributions are single-component Gaussians and the state occupation counts are known, then the increase in likelihood achieved by splitting the Gaussians at any node can be calculated simply from the counts and model parameters without reference to the training data. Thus, the decision trees can be grown very efficiently using a greedy iterative node-splitting algorithm. Figure 27.5 illustrates this tree-based clustering. In the figure, the logical phones s-aw+n and t-aw+n will both be assigned to leaf node 3 and hence they will share the same central state of the representative physical model. The total number of tied states in a large-vocabulary speaker-independent system typical ranges between 1000 and 5000 states.

The partitioning of states using phonetically driven decision trees has several advantages. In particular, logical models that are required but were not seen at all in the training data can be easily synthesized. One disadvantage is that the partitioning can be rather coarse. This problem can be reduced using so-called *soft tying* [27.18]. In this scheme, a postprocessing stage groups each state with its one or two nearest neighbors and pools all of their Gaussians. Thus, the single Gaussian models are converted to mixture Gaussian models

Amazon EX1010 Page 560

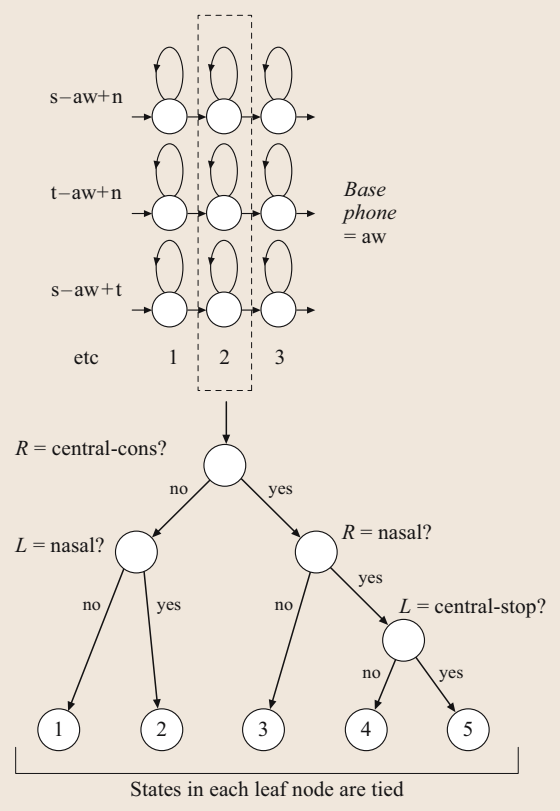



**Fig. 27.5** Decision tree clustering

whilst holding the total number of Gaussians in the system constant.

To summarize, the core acoustic models of a modern speech recognizer is typically comprised of a set of tied-state mixture Gaussian HMM-based acoustic models. This core is commonly built in the following steps [27.12, Chap. 3]:

1. A flat-start monophone set is created in which each base phone is a monophone single-Gaussian HMM with means and covariances equal to the mean and covariance of the training data.
2. The parameters of the single-Gaussian monophones are iteratively re-estimated using three or four iterations of EM.
3. Each single Gaussian monophone q is cloned once for each distinct triphone x-q+y that appears in the training data.
4. The set of training data single-Gaussian triphones is iteratively re-estimated using EM and the state occupation counts of the last iteration are saved.
5. A decision tree is created for each state in each base phone, the single-Gaussian triphones are mapped into a smaller set of tied-state triphones and iteratively re-estimated using EM.
6. Mixture components are iteratively split and re-estimated until performance peaks on a held-out development set.

The final result is the required tied-state context-dependent mixture Gaussian acoustic model set.

### 27.2.3 *N*-Gram Language Models

The prior probability of a word sequence $W = w_1, \cdots, w_K$ required in (27.2) is given by

$$p(W) = \prod_{k=1}^{K} p(w_k | w_{k-1}, \ldots, w_1) \,. \tag{27.8}$$

For large-vocabulary recognition, the conditioning word history in (27.8) is usually truncated to $N-1$ words to form an $N$-gram language model

$$p(W) = \prod_{k=1}^{K} p(w_k | w_{k-1}, w_{k-2}, \ldots, w_{k-N+1}) \,, \tag{27.9}$$

where $N$ is typically in the range 2–4. The $N$-gram probabilities are estimated from training texts by counting $N$-gram occurrences to form maximum likelihood (ML) parameter estimates. For example, let $C(w_{k-2}w_{k-1}w_k)$ represent the number of occurrences of the three words $w_{k-2}w_{k-1}w_k$ and similarly for $C(w_{k-2}w_{k-1})$, then

$$p(w_k | w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} \,. \tag{27.10}$$

The major problem with this simple ML estimation scheme is data sparsity. This can be mitigated by a combination of discounting and backing-off, known as *Katz smoothing* [27.19]

$$\begin{aligned} & p(w_k | w_{k-1}, w_{k-2}) \\ & = \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} \,, && \text{if } C > C' \,, \\ & = d\frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} \,, && \text{if } 0 < C \le C' \\ & = \alpha(w_{k-1}, w_{k-2}) p(w_k | w_{k-1}) && \text{otherwise} \,, \end{aligned} \tag{27.11}$$

where $C'$ is a count threshold, $d$ is a discount coefficient and $\alpha$ is a normalization constant. Thus, when the $N$-gram count exceeds some threshold, the ML estimate

is used. When the count is small the same ML estimate is used but discounted slightly. The discounted probability mass is then distributed to the unseen $N$-grams which are approximated by a weighted version of the corresponding bigram. This idea can be applied recursively to estimate any sparse $N$-gram in terms of a set of back-off weights and $(N-1)$-grams. The discounting coefficient is based on the Turing–Good estimate $d=(r+1)n_{r+1}/rn_r$ where $n_r$ is the number of $N$-grams that occur exactly $r$ times in the training data. Although Katz smoothing is effective, there are now known to be variations that work better. In particular, Kneser–Ney smoothing consistently outperforms Katz on most tasks [27.20, 21].

An alternative approach to robust language model estimation is to use class-based models in which, for every word $w_k$, there is a corresponding class $c_k$ [27.22, 23]. Then,

$$p(W)=\prod_{k=1}^{K} p(w_k|c_k)p(c_k|c_{k-1},\ldots,c_{k-N+1})\,. \tag{27.12}$$

As for word-based models, the class $N$-gram probabilities are estimated using ML but since there are far fewer classes (typically a few hundred) data sparsity is much less of an issue. The classes themselves are chosen to optimize the likelihood of the training set assuming a bigram class model. It can be shown that, when a word is moved from one class to another, the change in perplexity depends only on the counts of a relatively small number of bigrams. Hence, an iterative algorithm can be implemented which repeatedly scans through the vocabulary, testing each word to see if moving it to some other class would increase the likelihood [27.24].

In practice it is found that for reasonably sized training sets, an effective language model for large vocabulary applications consists of a word-based trigram or four-gram interpolated with a class-based trigram.

### 27.2.4 Decoding and Lattice Generation

As noted in the introduction to this section, the most likely word sequence $\hat{W}$ given a sequence of feature vectors $\boldsymbol{Y}=\boldsymbol{y}_1,\cdots,\boldsymbol{y}_T$ is found by searching all possible state sequences arising from all possible word sequences for the sequence that was most likely to have generated the observed data $\boldsymbol{Y}$. An efficient way to solve this problem is to use dynamic programming. Let $\phi_j(t)=\max_X\{p(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t,x(t)=j|\mathcal{M})\}$, i. e., the maximum probability of observing the partial sequence $\boldsymbol{y}_1,\cdots,\boldsymbol{y}_t$ and then being in state $j$ at time $t$ given the model $\mathcal{M}$. This probability can be efficiently computed using the Viterbi algorithm

$$\phi_j(t)=\max_i\left\{\phi_i(t-1)a_{ij}\right\}b_j(\boldsymbol{y}_t)\,. \tag{27.13}$$

It is initialized by setting $\phi_j(t)$ to 1 for the initial state and 0 for all other states. The probability of the most likely word sequence is then given by $\max_j\{\phi_j(T)\}$ and if every maximization decision is recorded, a traceback will yield the required best matching state/word sequence.

In practice, a direct implementation of the above algorithm becomes unmanageably complex for continuous speech where the topology of the models, the language model constraints, and the need to bound the computation must all be taken into account. Fortunately, much of this complexity can be abstracted away by a simple change of viewpoint.

First, the HMM topology can be made explicit by constructing a recognition network. For task-oriented applications, this network can represent the utterances that the user is allowed to say, i. e., it can represent a recognition grammar. For large-vocabulary applications, it will typically consist of all vocabulary words in parallel in a loop. In both cases, words are represented by a sequence of phone models as defined by the pronunciation dictionary (Fig. 27.6), and each phone model consists of a sequence of states as indicated by the inset. If a word has several pronunciations as in the general case described by (27.3), they are simply placed in parallel.

Given this network, at any time $t$ in the search, a single hypothesis consists of a path through the network representing an alignment of states with feature vectors $\boldsymbol{y}_1,\cdots,\boldsymbol{y}_t$, starting in the initial state and ending

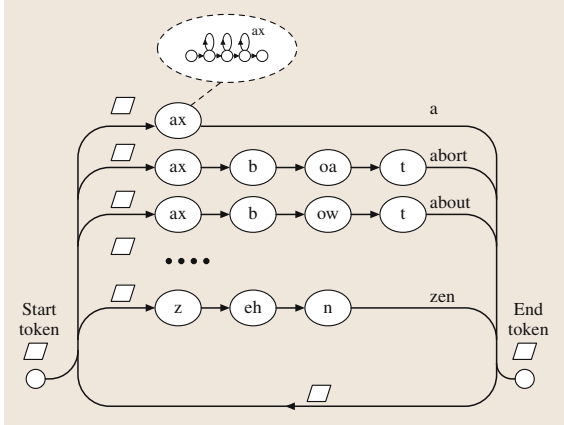



**Fig. 27.6** Basic recognition network

```
Put a start token <log(1), 0> in network entry node;
Put null tokens <log(0), 0> in all other nodes;
for each time t = 1 to T do
    – word internal token propagation
    for each non-entry node j do
        maxP = log(0);
        for each predecessor node i do
            Temp token Q = Q_i;
            Q.logP += log(a_ij) [+ log(b_j(yt)) if j emitting];
            If Q.logP > maxP then
                Q_j = Q; maxP = Q.logP;
        end;
    end;
    Copy tokens from word internal exits to following entries;
    – word external token from propagation
    for each word w with entry node j do
        maxP = log(0);
        for each predecessor word u with exit node i do
            Temp token Q = Q_i;
            Q.logP += α logp(w/u) + β;
            If Q.logP > maxP then
                Q_j = Q; maxP = Q.logP; u' = u
        end;
        – Record word boundary decision
        Create a record R;
        R.Q = Q_j; Rt = t; R word = u';
        Q.link = ↑R;
    end;
    Put null token in network entry node;
end;
Token in network exit state at time T represents the best path
```

**Fig. 27.7** Basic token-passing algorithm

at state $j$, and having a log likelihood of $\log \phi_j(t)$. This path can be made explicit via the notion of a *token* consisting of a pair of values $\langle \log P, \text{link} \rangle$, where $\log P$ is the log likelihood (often referred to as the *score*) and link is a pointer to a record of history information [27.25]. Each network node corresponding to a HMM state can store a single token and recognition proceeds by propagating these tokens around the network.

The basic Viterbi algorithm given above can now be recast for continuous speech recognition as the *token-passing* algorithm shown in outline in Fig. 27.7. The term *node* refers to a network node corresponding to a single HMM state. These nodes correspond to either entry, emitting or exit states. Essentially, tokens are passed from node to node and at each transition the token score is updated.

When a token transitions from the exit of a word to the start of the next word, its score is updated by the language model probability plus any word-insertion penalty. At the same time the transition is recorded in a record $R$ containing a copy of the token, the current time, and the identity of the preceding word. The *link* field of the token is then updated to point to the record $R$. As each token proceeds through the network it accumulates a chain of these records. The best token at time $T$ in a valid network exit node can then be examined and traced back to recover the most likely word sequence and the boundary times.

The above token-passing algorithm and associated recognition network is an exact implementation of the dynamic programming principle embodied in (27.13). To convert this to a practical decoder for speech recognition, the following steps are required:

1. For computational efficiency, only tokens that have some likelihood of being on the best path should be propagated. Thus, at every propagation cycle, the log probability of the most likely token is recorded. All tokens whose probabilities fall more than a constant below this are deleted. This results in a so-called *beam search* and the constant is called the *beam width*.
2. As a consequence of the beam search, 90% of the computation is actually spent on the first two phones of every word, after which most of the tokens fall outside of the beam and are pruned. To exploit this, the recognition network should be *tree-structured* so that word initial phones are shared (Fig. 27.8).
3. However, sharing initial phones makes it impossible to apply an exact language model probability during word-external token propagation since the identity of the following word is not known. Simply delaying the application of the language model until the end of the following word is not an option since that would make pruning ineffective. Instead, an incremental approach must be adopted in which the language model probability is taken to be the maximum possible probability given the set of possible following words. As tokens move through the tree-structured word graph, the set of possible next words reduces at each phone transition and the lan-

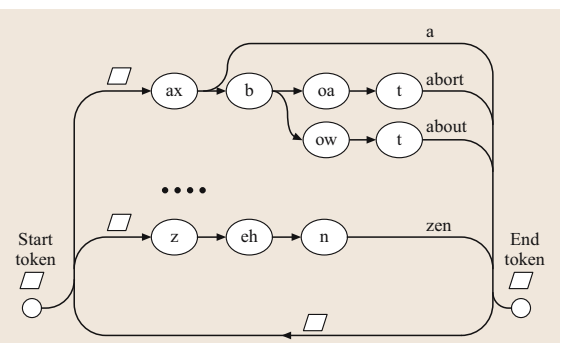



**Fig. 27.8** Tree-structured recognition network



Amazon EX1010 Page 563

guage model probability can be updated with a more accurate estimate.

4. The HMMs linked into the recognition network should be context dependent and for best performance, this dependency should span word boundaries. At first sight this requires a massive expansion of the network, but in fact a compact static network representation of cross-word triphones is possible [27.26].
5. The dynamic programming principle relies on the principle that the optimal path at any node can be extended knowing only the state information given at that node. The use of $N$-gram language models causes a problem here since unique network nodes would be needed to distinguish all possible $N-1$ word histories and for large-vocabulary decoders this is not tractable. Thus, the algorithm given in Fig. 27.7 will only work for bigram language models. A simple way to solve this problem is to store multiple tokens in each state, thereby allowing paths with differing histories to *stay alive* in parallel. Token propagation now requires a merge and sort operation that, although computationally expensive, can be made tractable.

The above description of a large-vocabulary decoder covers all of the essential elements needed to recognize continuous speech in real time using just a single pass over the data. For offline batch transcription of speech, significant improvements in accuracy can be achieved by performing multiple passes over the data. To make this possible, the decoder must be capable of generating and saving multiple recognition hypotheses. A compact and efficient structure for doing this is the *word lattice* [27.27–29].

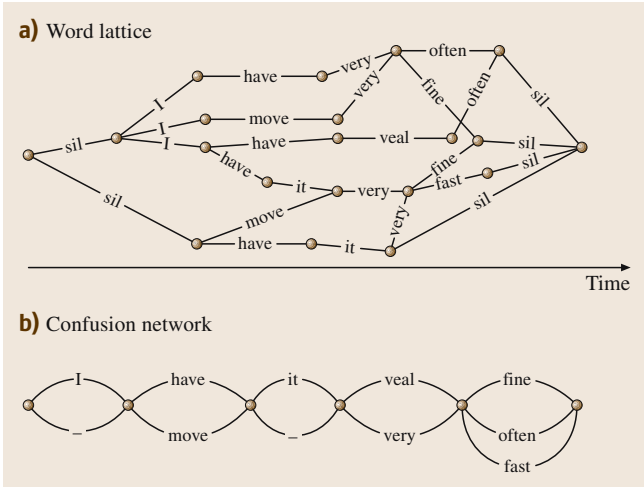



**Fig. 27.9** Example lattice and confusion network

A word lattice consists of a set of nodes representing points in time and a set of spanning arcs representing word hypotheses. An example is shown in Fig. 27.9a. In addition to the word identities (IDs) shown in the figure, each arc can also carry score information such as the acoustic and language model scores. Lattices are generated via the mechanism for recording word-boundary information outlined in Fig. 27.7, except that instead of recording just the best token that is actually propagated to following word entry nodes, all word-end tokens are recorded. For the simple single-token Viterbi scheme, the quality of lattices generated in this way will be poor because many of the close-matching second-best paths will have been pruned by exercising the dynamic programming principle. The multiple-token decoder does not suffer from this problem, especially if it is used with a relatively short-span bigram language model.

Lattices are extremely flexible. For example, they can be rescored by using them as an input recognition network and can be expanded to allow rescoring by a higher-order language model. They can also be compacted into a very efficient representation called a *confusion network* [27.30, 31]. This is illustrated in Fig. 27.9b where the – arc labels indicate null transitions. In a confusion network, the nodes no longer correspond to discrete points in time; instead they simply enforce word sequence constraints. Thus, parallel arcs in the confusion network do not necessarily correspond to the same acoustic segment. However, it is assumed that most of the time the overlap is sufficient to enable parallel arcs to be regarded as competing hypotheses. A confusion network has the property that, for every path through the original lattice, there exists a corresponding path through the confusion network. Each arc in the confusion network carries the posterior probability of the corresponding word $w$. This is computed by finding the *link probability* of $w$ in the lattice using a forward–backward procedure, summing over all occurrences of $w$ and then normalizing so that all competing word arcs in the confusion network sum to one. Confusion networks can be used for minimum-word-error decoding, to provide confidence scores, and for merging the outputs of different decoders [27.32–35] (Sect. 27.6).

Finally, it should be noted that all of the above relates to one specific approach to decoding. If simple Viterbi decoding was the only requirement, then there

Amazon EX1010 Page 564

would be little variation amongst decoder implementations. However, the requirement to support cross-word context-dependent acoustic models and long-span language models has led to a variety of design strategies. For example, rather than have multiple tokens, the network state can be dynamically expanded to represent explicitly the currently hypothesized cross-word acoustic and long-span language model contexts [27.36, 37]. These dynamic network decoders are more flexible than static network decoders, but they are harder to implement efficiently. Recent advances in weighted finite-state transducer technology offer the possibility of integrating all of the required information (acoustic models, pronunciation, language model probabilities, etc.) into a single, very large, but highly optimized network) [27.38]. This approach offers both flexibility and efficiency and is therefore extremely useful for both research and practical applications.

A completely different approach to decoding is to avoid the breadth-first strategy altogether and use a depth-first strategy. This gives rise to a class of recognizers called *stack decoders*. Stack decoders were popular in the very early developments of ASR since they can be very efficient. However, they require dynamic networks and their run-time search characteristics can be difficult to control [27.39, 40].

## 27.3 HMM-Based Acoustic Modeling

The key feature of the HMM-based speech recognition architecture described in the preceding section is the use of diagonal-covariance multiple-component mixture Gaussians for modeling the spectral distributions of the speech feature vectors. If speech really did have the statistics assumed by these HMMs and if there was sufficient training data, then the models estimated using maximum likelihood would be optimal in the sense of minimum variance and zero bias [27.41]. However, since this is not the case, there is scope for improving performance both by using alternative parameter-estimation schemes and by improving the models. In this section, both of these aspects of HMM design will be discussed. Firstly, discriminative training is described and then methods of improved covariance modeling will be explored.

### 27.3.1 Discriminative Training

Standard maximum-likelihood training attempts to find a parameter set $\lambda$ that maximizes the log likelihood of the training data, i. e., for training sentences $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_R$, the objective function is

$$\mathcal{F}_{\mathrm{ML}}(\lambda) = \sum_{r=1}^{R} \log p_\lambda(\boldsymbol{Y}_r | \mathcal{M}_{W_r}) \,, \tag{27.14}$$

where $W_r$ is the word sequence given by the transcription of the $r$-th training sentence and $\mathcal{M}_{W_r}$ is the corresponding composite HMM synthesized by concatenating phone models (denoted by $Q$ in Sect. 27.2.2). This objective function can be maximized straightforwardly using a version of EM known as the Baum–Welch algorithm [27.42]. This involves iteratively finding the probability of state-component occupation for each frame of training data using a forward–backward algorithm, and then computing weighted averages. For example, defining the following *counts*

$$\Theta^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) \boldsymbol{y}^r_t \Big|_{\mathcal{M}} \,, \tag{27.15}$$

and

$$\Gamma^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) \Big|_{\mathcal{M}} \,, \tag{27.16}$$

where $\gamma^r_{jm}(t)$ is the probability of the model occupying mixture component $m$ in state $j$ at time $t$ given the training sentence $\boldsymbol{Y}_r$ and model $\mathcal{M}$, then the updated mean estimate is given by ML as

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{r=1}^{R} \Theta^r_{jm}(\mathcal{M}_{W_r})}{\sum_{r=1}^{R} \Gamma^r_{jm}(\mathcal{M}_{W_r})} \,, \tag{27.17}$$

i. e., the average of the sum of the data weighted by the model component occupancy.

The key problem with the ML objective function is that it simply fits the model to the training data and takes no account of the model's ability to discriminate. An alternative objective function is to maximize the conditional likelihood using the maximum mutual

information (MMI) criterion [27.41, 43]

$$\mathcal{F}_{\mathrm{MMI}}(\lambda) = \sum_{r=1}^{R} \log \frac{p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_{W_r})p(W_r)}{\Sigma_W p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_W)p(W)} \,. \quad (27.18)$$

Here the numerator is the likelihood of the data given the correct word sequence $W_r$ whilst the denominator is the total likelihood of the data given all possible word sequences $W$. Thus, the objective function is maximized by making the correct model sequence likely and all other model sequences unlikely. It is therefore discriminative. Note also that it incorporates the effect of the language model and hence more closely represents recognition conditions.

There is no simple EM-based re-estimation scheme for (27.18), however, there is an approximate scheme known as the extended Baum–Welch algorithm in which stability is achieved for the parameter updates by adding a constant $D$ to both the numerator and denominator. For example, (27.17) in the extended scheme becomes

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{r=1}^{R}[\Theta^r_{jm}(\mathcal{M}_{\mathrm{num}}) - \Theta^r_{jm}(\mathcal{M}_{\mathrm{den}})] + D\boldsymbol{\mu}_{jm}}{\sum_{r=1}^{R}[\Gamma^r_{jm}(\mathcal{M}_{\mathrm{num}}) - \Gamma^r_{jm}(\mathcal{M}_{\mathrm{den}})] + D} \,, \quad (27.19)$$

where $\mathcal{M}_{\mathrm{num}}$ is the combined acoustic and language model used to compute the numerator of (27.18), and $\mathcal{M}_{\mathrm{den}}$ is the combined acoustic and language model used to compute the denominator. In the latter case, it is understood that the counts in $\Theta^r_{jm}(\mathcal{M}_{\mathrm{den}})$ are summed over all word sequences. For large-vocabulary continuous speech, this is approximated by computing lattices and summing over all lattice arcs. Although the numerator counts can be computed as in ML, in practice, the numerator counts are also computed using lattices since this provides a convenient way to take account of multiple pronunciations [27.44]. As can be seen, counts in the numerator are reduced if there are similar confusing counts in the denominator. The constant $D$ acts like an interpolation weight between the new estimate and the existing estimate.

In a little more detail, the MMI training process is as follows. Numerator and denominator lattices are generated for every training utterance using $\mathcal{M}_{\mathrm{num}}$ and $\mathcal{M}_{\mathrm{den}}$, respectively. $\mathcal{M}_{\mathrm{num}}$ consists of the current phone models integrated into a graph of alternative word pronunciations, and $\mathcal{M}_{\mathrm{den}}$ consists of the normal recognizer set-up with two exceptions. Firstly, a weaker language model is used. Typically the lattices are generated using a bigram language model and then rescored using a unigram [27.45]. Secondly, the likelihoods of the acoustic models are scaled by the inverse of the normal LM scaling factor [27.46]. Both of these modifications have been found to increase the number of confusions in the denominator lattice, thereby improving subsequent generalization. Once the word-level lattices have been generated, a Viterbi decode is performed on each lattice arc to obtain a phone-level segmentation. Forward–backward is then applied to obtain the component level counts and the model parameters are re-estimated using (27.19) (and similar formulae for the variances and mixture weights). This process is iterated and the state-component alignments are recomputed every three or four iterations. The word-level lattices are typically held fixed throughout the training process.

The constant $D$ must be chosen to be large enough to ensure convergence but small enough to ensure acceptably fast training. In practice, $D$ is chosen to ensure that variance updates are positive and is normally set specifically for each phone or Gaussian [27.46].

MMI training can provide consistent performance improvements compared to similar systems trained with ML [27.46]. However, it can be argued that it places too much emphasis on training utterances that have a low a posterior probability and not enough weight on training utterances near the decision boundary, as in for example minimum classification error (MCE) training [27.47]. MCE, however, focuses on overall sentence-level accuracy, and it is not appropriate for lattice-based training of large systems. Minimum-phone-error (MPE) training addresses this issue by attempting to maximize the a posterior utterance probability scaled by the *raw phone accuracy* (RPA) [27.48]

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\Sigma_W p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_W)p(W)\mathrm{RPA}(W, W_r)}{\Sigma_W p_\lambda(\boldsymbol{Y}_r|\mathcal{M}_W)p(W)} \,, \quad (27.20)$$

where, as in lattice-based MMI training, the sums over $W$ are taken over all word sequences appearing in the lattice generated by $\mathcal{M}_{\mathrm{den}}$. The RPA is a measure of the number of phones accurately transcribed in each word string hypothesis $W$. Given the times of the phone boundaries, each phone in $W$ is matched against the corresponding time segment in the transcription $W_r$. If the phones are the same, the RPA is incremented by the percentage overlap; otherwise it is decremented by the

Amazon EX1010 Page 566

percentage overlap. Parameter optimization is similar to the MMI process described above except that the counts are only computed on the denominator lattice. The numerator lattice provides the transcriptions needed for determining the RPA. Essentially, the counts are composed from the occupation probabilities scaled by the RPA. If they are positive, they contribute to the numerator terms in the update equations, and if they are negative, they contribute to the denominator terms (see [27.48] for details).

The generalization capabilities of discriminatively trained systems can be improved by interpolating with ML. For example, the H-criterion interpolates objective functions: $\alpha\mathcal{F}_{\text{MMI}} + (1-\alpha)\mathcal{F}_{\text{ML}}$ [27.49]. However, choosing an optimal value for $\alpha$ is difficult and the effectiveness of the technique decreases with increasing training data [27.50]. A more-effective technique is *I-smoothing* which increases the weight of the numerator counts depending on the amount of data available for each Gaussian component [27.48]. This is done by scaling the numerator counts $\Gamma^r_{jm}(\mathcal{M}_{\text{num}})$ and $\Theta^r_{jm}(\mathcal{M}_{\text{num}})$ by

$$1 + \frac{\tau}{\Sigma_r \Gamma^r_{jm}(\mathcal{M}_{\text{num}})} , \tag{27.21}$$

where $\tau$ is a constant (typically about 50). As $\tau$ increases from zero, more weight is given to ML.

### 27.3.2 Covariance Modeling

An alternative way of improving the acoustic models is to allow them to more closely match the true distribution of the data. The baseline acoustic models outlined in Sect. 27.2.2 use mixtures of diagonal-covariance Gaussians chosen as a compromise between complexity and modeling accuracy. Nevertheless, the data is clearly not diagonal and hence finding some way of improving the covariance modeling is desirable. In general the use of full-covariance Gaussians in large-vocabulary systems would be impractical due to the sheer size of the model set although given enough data it can be done [27.11]. However, the use of shared or *structured* covariance representations allow covariance modeling to be improved with very little overhead in terms of memory and computational load.

One the simplest and most effective structured covariance representations is the semi-tied covariance (STC) matrix [27.51]. STC models the covariance of the $m$-th Gaussian component as

$$\hat{\Sigma}_m = A^{-1}\Sigma_m(A^{-1})^T , \tag{27.22}$$

where $\Sigma_m$ is the component-specific diagonal covariance and $A$ is the STC matrix shared by all components. If the component mean is $\boldsymbol{\mu}_m = A\hat{\boldsymbol{\mu}}_m$ then component likelihoods can be computed by

$$\mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m) = |A|\mathcal{N}(A\boldsymbol{y}; \boldsymbol{\mu}_m, \Sigma_m) . \tag{27.23}$$

Hence, a single STC matrix can be viewed as a linear transform applied in feature space. The component parameters $\boldsymbol{\mu}_m$ and $\Sigma_m$ represent the means and variances in the transformed space and can be estimated in the normal way by simply transforming the training data. The STC matrix itself can be efficiently estimated using a row-by-row iterative scheme. Furthermore it is not necessary during training to store the full-covariance statistics at the component level. Instead, an interleaved scheme can be used in which the $A$ matrix statistics are updated on one pass through the training data, and then the component parameters are estimated on the next pass [27.51]. The means can in fact be updated on both passes. This can be integrated into the *mixing-up* operation described in Sect. 27.2.2. For example, a typical training scheme might start with a single Gaussian system and an identity $A$ matrix. The system is then iteratively refined by re-estimating the component parameters, updating the $A$ matrix, and mixing up until the required number of Gaussians per state is achieved. As well as having a single global $A$ matrix, the Gaussian components can be clustered and assigned one $A$ matrix per cluster. For example, there could be one $A$ matrix per phone or per state depending on the amount of training data available and the acceptable number of parameters. Overall, the use of STC can be expected to reduce word error rates by around 5–10% compared to the baseline system. In addition to STC, other types of structured covariance modeling include factor-analyzed HMMs [27.52], subspace-constrained precision and means (SPAM) [27.53], and extended maximum likelihood linear transform (EMLLT) [27.54].

It can be shown [27.51] that simultaneous optimization of the full set of STC parameters (i. e., $\{A, \boldsymbol{\mu}_m, \Sigma_m\}$) is equivalent to maximizing the auxiliary equation

$$\mathcal{Q}_{\text{STC}} = \sum_{t,m} \gamma_m(t) \log\left(\frac{|A|^2}{|\text{diag}(A\, W^{(m)} A^{\text{T}})|}\right) , \tag{27.24}$$

where

$$W^{(m)} = \frac{\Sigma_t \gamma_m(t) \bar{\boldsymbol{y}}_m(t) \bar{\boldsymbol{y}}_m(t)^{\text{T}}}{\Sigma_t \gamma_m(t)} \tag{27.25}$$



Amazon EX1010 Page 567

and where $\bar{\boldsymbol{y}}_m(t) = \boldsymbol{y}(t) - \hat{\boldsymbol{\mu}}_m$. If each Gaussian component is regarded as a class, then $W^{(m)}$ is the within-class covariance and it can be shown that (27.24) is the maximum-likelihood solution to a generalized form of linear discriminant analysis called heteroscedastic LDA (linear discriminant analysis) (HLDA) in which class covariances are not constrained to be equal [27.55]. The matrix $A$ can therefore be regarded as a feature space transform that discriminates Gaussian components and this suggests a simple extension by which $A$ can also perform a subspace projection, i. e.,

$$\hat{\boldsymbol{y}} = A\boldsymbol{y} = \begin{pmatrix} A_{[p]}\boldsymbol{y} \\ A_{[d-p]}\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{y}}_{[p]} \\ \hat{\boldsymbol{y}}_{[d-p]} \end{pmatrix} , \tag{27.26}$$

where the $d$-dimensional feature space is divided into $p$ *useful* dimensions and $d-p$ *nuisance* dimensions. The matrix $A_{[p]}$ projects $\boldsymbol{y}$ into a $p$-dimensional subspace that is modeled by the diagonal Gaussian mixture components of the acoustic models. The $d-p$ nuisance dimensions are modeled by a global nondiscriminating Gaussian. Equation (27.24) can therefore be factored as

$$\mathcal{Q}_{\text{HLDA}} = \sum_{t,m} \gamma_m(t) \times \log\left( \frac{|A|^2}{|\text{diag}(A_{[p]} W^{(m)} A_{[p]}^T)||\text{diag}(A_{[d-p]} T A_{[d-p]}^T)|} \right) , \tag{27.27}$$

where $T$ is the global covariance of the training data. The forms of (27.24) and (27.27) are similar, and the optimal value for $A_{[p]}$ can be estimated by the same row-by-row iteration used in the STC case.

For application to large vocabulary continuous speech recognition (LVCSR), $\boldsymbol{y}$ can be constructed either by concatenating successive feature vectors, or as is common in HTK-based systems, the standard 39-element feature vector comprised of static PLP coefficients plus their first and second derivatives is augmented by the third derivatives and then projected back into 39 dimensions using a $39 \times 52$ HLDA transform.

Finally note that, as with semi-tied covariances, multiple HLDA transforms can be used to allow the full acoustic space to be covered by a set of piecewise linear projections [27.56].

## 27.4 Normalization

Normalization attempts to condition the incoming speech signal to minimize the effects of variation caused by the environment and the physical characteristics of the speaker.

### 27.4.1 Mean and Variance Normalization

Mean normalization removes the average feature value and since most front-end feature sets are derived from log spectra, this has the effect of reducing sensitivity to channel variation. Cepstral variance normalization scales each individual feature coefficient to have a unit variance and empirically this has been found to reduce sensitivity to additive noise [27.57].

For transcription applications where multiple passes over the data are possible, the necessary mean and variance statistics should be computed over the longest possible segment of speech for which the speaker and environment conditions are constant. For example, in broadcast news transcription this will be a speaker segment and in telephone transcription it will be a whole side of a conversation. Note that for real-time systems that operate in a single, continuous pass over the data, the mean and variance statistics must be computed as running averages.

### 27.4.2 Gaussianization

Given that normalizing the first- and second-order statistics yields improved performance, an obvious extension is to normalize the higher-order statistics so that the features are Gaussian distributed. This so-called *Gaussianization* is performed by finding a transform $z = \phi(y)$, on a per-element basis, such that $p(z)$ is Gaussian. One simple way to achieve this is to estimate a cumulative distribution function (CDF) for each feature element $y$

$$F_0(y) = \frac{1}{N} \sum_{i=1}^{N} h(y - y_i) = \frac{\text{rank}(y_i)}{N} , \tag{27.28}$$

where $y_1, \cdots, y_N$ are the data to be normalized, $h(\cdot)$ is the step function and $\text{rank}(y_i)$ is the rank of $y_i$ when the data are sorted. The required transformation is then

$$z_i = \Phi^{-1}\left( \frac{\text{rank}(y_i)}{N} \right) , \tag{27.29}$$

where $\Phi(\cdot)$ is the CDF of a Gaussian [27.58].

One difficulty with this approach is that when the normalization data set is small the CDF estimate can be noisy. An alternative approach is to estimate an $M$-component Gaussian mixture model (GMM) on the data and then use this to approximate the CDF [27.59], that is

$$z_i = \Phi^{-1}\left(\int_{-\infty}^{y_i}\sum_{m=1}^{M} c_m \mathcal{N}(y;\mu_m,\sigma_m^2)\mathrm{d}y\right) , \quad (27.30)$$

where $\mu_m$ and $\sigma_m^2$ are the mean and variance of the $m$-th GMM component. This results in a smoother and more-compact representation of the Gaussianization transformation [27.60].

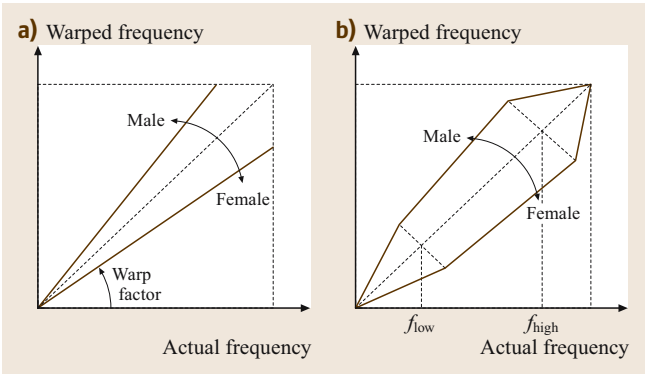



**Fig. 27.10a,b** Vocal-tract length normalization

### 27.4.3 Vocal-Tract-Length Normalization

Variations in vocal-tract length cause formant frequencies to shift in frequency in an approximately linear fashion. Thus, one simple form of normalization is to linearly scale the filter bank center frequencies within the front-end feature extractor to approximate a canonical formant frequency scaling [27.61]. This is called *vocal-tract-length normalization (VTLN)*.

To implement VTLN two issues need to be addressed: the definition of the scaling function and the estimation of the appropriate scaling function parameters for each speaker. Early attempts at VTLN used a simple linear mapping but, as shown in Fig. 27.10a, this results in a problem at high frequencies where female voices have no information in the upper frequency band and male voices have the upper frequency band truncated. This can be mitigated by using a piecewise linear function of the form shown in Fig. 27.10b [27.57]. Alternatively, a bilinear transform can be used [27.62]. Parameter estimation is performed using a grid search plotting log likelihoods against parameter values. Once the optimal values for all training speakers have been computed, the training data is normalized and the acoustic models re-estimated. This is repeated until the VTLN parameters have stabilized. Note here that, when comparing log likelihoods resulting from differing VTLN transformations, the Jacobean of the transform should strictly be included. This is however very complex to estimate and since the application of mean and variance normalization will reduce the affect of this approximation, it is usually ignored.

For very large systems, the overhead incurred from iteratively computing the optimal VTLN parameters can be considerable. An alternative is to approximate the effect of VTLN by a linear transform. The advantage of this approach is that the optimal transformation parameters can be determined from the auxiliary function in a single pass over the data [27.63].

VTLN is particularly effective for telephone speech where speakers can be clearly identified. It is less effective for other applications such as broadcast news transcription where speaker changes must be inferred from the data.

## 27.5 Adaptation

A fundamental idea in statistical pattern classification is that the training data should adequately represent the test data, otherwise a mismatch will occur and recognition accuracy will be degraded. In the case of speech recognition, there will always be new speakers who are poorly represented by the training data, and new hitherto unseen environments. The solution to these problems is *adaptation*. Adaptation allows a small amount of data from a target speaker to be used to transform an acoustic model set to make it more closely match that speaker. It can be used both in training to make more-specific and/or more-compact recognition sets, and it can be used in recognition to reduce mismatch and the consequent recognition errors.

There are various styles of adaptation that affect both the possible applications and the method of implementation. Firstly, adaptation can be *supervised*, in which case accurate transcriptions are available for all of the adaptation data, or it can be *unsupervised*, in which case the required transcriptions must be hypothesized. Secondly,



Amazon EX1010 Page 569

adaptation can be *incremental* or *batch mode*. In the former case, adaptation data becomes available in stages, for example, as is the case for a spoken dialogue system when a new caller comes on the line. In batch mode, all of the adaptation data is available from the start, as is the case in offline transcription.

This section describes the main approaches to adaptation and its application in both recognition and training.

### 27.5.1 Maximum A Posteriori (MAP) Adaptation

Given some adaptation data $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_R$ and a model $\mathcal{M}$ with parameters $\lambda$, MAP-based parameter estimation seeks to maximize the following objective function,

$$\mathcal{F}_{\text{MAP}}(\lambda) = \sum_{r=1}^{R} \log p(\boldsymbol{Y}_r|\mathcal{M}_{W_r}) p(\lambda) \,. \qquad (27.31)$$

Comparing this with the ML objection function given in (27.14), it can be seen that the likelihood is weighted by the prior. The choice of distribution for this prior is problematic since there is no conjugate prior density for a continuous Gaussian mixture HMM. However, if the mixture weights and Gaussian component parameters are assumed to be independent, then a finite mixture density of the form $p_{\text{D}}(\boldsymbol{c}_j) \prod_m p_{\text{W}}(\boldsymbol{\mu}_{jm}, \Sigma_{jm})$ can be used, where $p_{\text{D}}(\cdot)$ is a Dirichlet distribution over the vector of mixture weights $\boldsymbol{c}_j$ and $p_{\text{W}}(\cdot)$ is a normal–Wishart density. It can then be shown that this leads to parameter estimation formulae of the form

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\tau \boldsymbol{\mu}_{jm} + \sum_{r=1}^{R} \Theta^r_{jm}(\mathcal{M}_{W_r})}{\tau + \sum_{r=1}^{R} \Gamma^r_{jm}(\mathcal{M}_{W_r})} \,, \qquad (27.32)$$

where $\boldsymbol{\mu}_{jm}$ is the prior mean, and $\tau$ is a parameter of $p_{\text{W}}(\cdot)$, which is normally determined empirically. Similar, though rather more-complex, formulae can be derived for the variances and mixture weights [27.64].

Comparing (27.32) with (27.17), it can be seen that MAP adaptation effectively interpolates the original prior parameter values with those that would be obtained from the adaptation data alone. As the amount of adaptation data increases, the parameters tend asymptotically to the adaptation domain. This is a desirable property and it makes MAP especially useful for porting a well-trained model set to a new domain for which there is only a limited amount of data.

A major drawback of MAP adaptation is that every Gaussian component is updated individually. If the adaptation data is sparse, then many of the model parameters will not be updated. Various attempts have been made to overcome this [27.65,66] but MAP nevertheless remains ill-suited for rapid incremental adaptation.

### 27.5.2 ML-Based Linear Transforms

An alternative approach to adaptation is to build a set of linear transforms to map an existing model set into a new adapted model set such that the likelihood of the adaptation data is maximized. This is called *maximum-likelihood linear regression* (MLLR) and, unlike MAP, it is particularly well suited to unsupervised incremental adaptation.

There are two main variants of MLLR: unconstrained and constrained [27.67]. In unconstrained MLLR, separate transforms are trained for the means and variances,

$$\hat{\boldsymbol{\mu}}_{jm} = G\boldsymbol{\mu}_{jm} + \boldsymbol{b}; \; \hat{\Sigma}_{jm} = H\Sigma_{jm}H^{\text{T}} \,. \qquad (27.33)$$

The formulae for constrained MLLR (CMLLR) are identical except that $G = H$. The maximum-likelihood estimation formulae are given in [27.68]. Whereas there are closed-form solutions for unconstrained MLLR, the constrained case is similar to the semi-tied covariance transform discussed in Sect. 27.3.2 and requires an iterative solution. However, CMLLR has other advantages as discussed in Sect. 27.5.3 below. Linear transforms can also be estimated using discriminative criteria [27.69–71].

The key to the power of the MLLR adaptation approach is that a single transform matrix $G$ can be shared across a set of Gaussian mixture components. When the amount of adaptation data is limited, a global transform can be shared across all Gaussians in the system. As the amount of data increases, the HMM state components

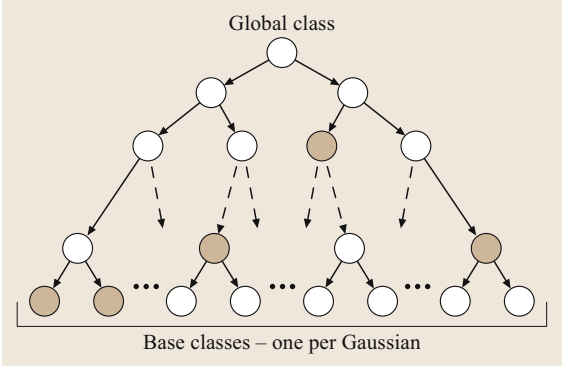



**Fig. 27.11** A regression class tree

can be grouped into classes, with each class having its own transform. As the amount of data increases further, the number of classes and therefore transforms increases correspondingly, leading to increasingly good adaptation.

The number of transforms to use for any specific adaptation set can be determined automatically using a *regression class tree* as illustrated in Fig. 27.11. Each node represents a regression class, i. e., a set of Gaussian components that will share a single transform. The total occupation count associated with any node in the tree can easily be computed since the counts are known at the leaf nodes. Then, for a given set of adaptation data, the tree is descended and the most specific set of nodes for which there is sufficient data is selected (for example, the shaded nodes in the figure).

When used in the unsupervised mode, MLLR is normally applied iteratively [27.72]. First, the unknown test speech is recognized, then the hypothesized transcription is used to estimate the MLLR transforms. The test speech is then re-recognized using the adapted models. This is repeated until convergence is achieved. A refinement of this is to use recognition lattices in place of the 1-best hypothesis to accumulate the adaptation statistics. This approach is more robust to recognition errors and avoids the need to re-recognize the data since the lattice can simply be rescored [27.73].

### 27.5.3 Adaptive Training

Ideally, an acoustic model set should encode just those dimensions that allow the different classes to be discriminated. However, in the case of speaker-independent (SI) speech recognition, the training data necessarily includes a large number of speakers and hence acoustic models trained directly on this set will have to *waste* a large number of parameters encoding the variability between speakers rather than the variability between spoken words, which is the true aim.

One way to overcome this is to replace the single SI model set with a cluster of more-specific models where each model can be trained on more-homogenous data. This is called *cluster adaptive training* (CAT). At recognition time, a linear combination of models is selected where the set of interpolation weights, in effect, forms a speaker-specific transform [27.74–76]. More recently discriminative techniques have been applied to CAT with some success [27.77].

An alternative approach to CAT is to use adaptation to transform each training set speaker to form a canonical model. This is called *speaker adaptive training*

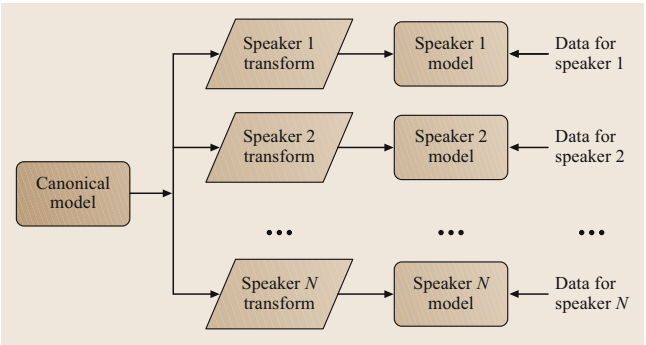



**Fig. 27.12** Adaptive training

(SAT) and the conceptual schema for this is shown in Fig. 27.12 [27.78]. When only mean transformations are used, SAT is straightforward. A transform is estimated for each speaker, and then the parameters of the canonical model set are estimated by modifying the statistics to account for the transform. For example, to estimate the canonical model means, the counts in (27.15) and (27.16) are modified as

$$\Theta^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) G^{(r)T} \Sigma^{-1}_{jm} (\boldsymbol{y}^r_t - \boldsymbol{b}^{(r)}) \Big|_{\mathcal{M}} , \tag{27.34}$$

and

$$\Gamma^r_{jm}(\mathcal{M}) = \sum_{t=1}^{T_r} \gamma^r_{jm}(t) G^{(r)T} \Sigma^{-1}_{jm} G^{(r)} \Big|_{\mathcal{M}} , \tag{27.35}$$

where $G^{(r)}, \boldsymbol{b}^{(r)}$ is the transform for the speaker uttering training sentence $r$. The mean is then estimated using (27.17) as normal.

Rather than modifying the statistics, the use of CMLLR allows adaptive training to be simplified further and allows combined mean and variance adaptation. Similar to the case for semi-tied covariances, the CMLLR transformed likelihood can be computed simply by regarding it as a feature space transformation, i. e., for any mixture component $m$ of state $j$

$$\mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{\mu}}_{jm}, \hat{\Sigma}_{jm}) = \frac{1}{|G|} \mathcal{N}(G^{-1}(\boldsymbol{y} - \boldsymbol{b}); \boldsymbol{\mu}_{jm}, \Sigma_{jm}) , \tag{27.36}$$

where $\hat{\boldsymbol{\mu}}_{jm}$ and $\hat{\Sigma}_{jm}$ are the transformed means and variances as in (27.33) (with $G = H$). Thus a SAT system can be built using CMMLR simply by iterating between the estimation of the canonical model using estimation

of the transformed training data and the transforms using the canonical model.

Finally, note that SAT trained systems incur the problem that they can only be used once transforms have been estimated for the test data. Thus, an SI model set is typically retained to generate the initial hypothesized transcription or lattice needed to compute the first set of transforms.



## 27.6 Multipass Recognition Architectures

The previous sections have reviewed some of the basic techniques available for both training and adapting an HMM-based recognition system. In general, any particular combination of model set and adaptation technique will have slightly different characteristics and make different errors. Furthermore, if the outputs of these systems are converted to confusion networks, as explained in Sect. 27.2.4, then it is straightforward to combine the confusion networks and then select the word sequence with the overall maximum a posterior likelihood. Thus, modern transcription systems typically utilize multiple model sets and make multiple passes over the data.

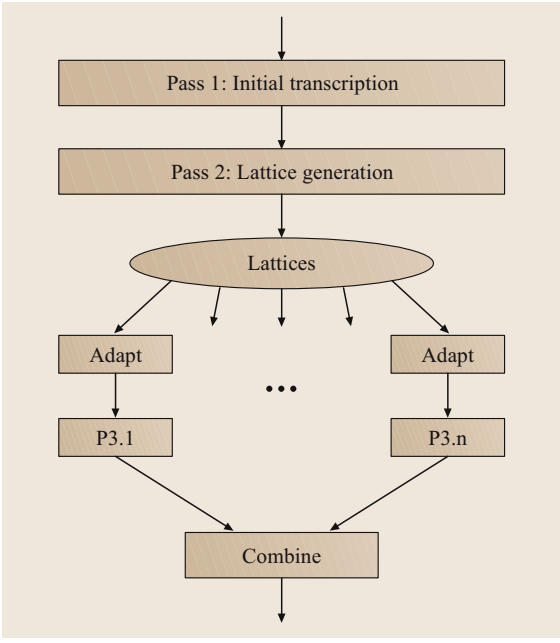



**Fig. 27.13** Multipass/system combination architecture

A typical architecture is shown in Fig. 27.13. A first pass is made over the data using a relatively simple SI model set. The 1-best output from this pass is used to perform a first round of adaptation. The adapted models are then used to generate lattices using a basic bigram or trigram word-based language model. Once the lattices have been generated, a range of more-complex models and adaptation techniques can be applied in parallel to provide $n$ candidate output confusion networks from which the best word sequence is extracted. These third pass models may include ML and MPE trained systems, SI and SAT trained models, triphone and quinphone models, lattice-based MLLR, CMLLR, four-gram language models interpolated with class-$n$-grams and many other variants. For examples of recent large-scale transcription systems see [27.11, 60, 79].

The gains obtained from this type of system combination can vary but overall performance is more robust across a range of task domains. Finally, note that adaptation can work more effectively if the required hypothesized transcriptions are generated by a different system. Thus, cross adaptation is also an increasingly popular architectural option.

## 27.7 Conclusions

This chapter has reviewed the core architecture of an HMM-based CSR system and outlined the major areas of refinement incorporated into modern systems. Diagonal-covariance continuous-density HMMs are based on the premise that each input frame is independent, and its components are decorrelated and have Gaussian distributions. Since none of these assumptions are true, the various refinements described above can all be viewed as attempts to reduce the impact of these false assumptions. Whilst many of the techniques are quite complex, they are nevertheless effective and overall substantial progress has been made. For example, error rates on the transcription of conversational telephone speech were around 45% in 1995. Today, with the benefit of more

data and the refinements described above, error rates are now well below 20%. Similarly, broadcast news transcription has improved from around 30% word error rate (WER) in 1997 to below 15% today.

Despite their dominance and the continued rate of improvement, many argue that the HMM architecture is fundamentally flawed and performance must asymptote. Of course, this is undeniably true since we have in our own heads an existence proof. However, no good alternative to the HMM has yet been found. In the meantime, the performance asymptote seems still to be some way away.



## References


27.1 J.K. Baker: The dragon system – an overview, IEEE Trans. Acoust. Speech Signal Process. **23**(1), 24–29 (1975)

27.2 F. Jelinek: Continuous speech recognition by statistical methods, Proc. IEEE **64**(4), 532–556 (1976)

27.3 B.T. Lowerre: *The Harpy Speech Recognition System, Ph.D. Dissertation* (Carnegie Mellon, Pittsburgh 1976)

27.4 L.R. Rabiner, B.-H. Juang, S.E. Levinson, M.M. Sondhi: Recognition of isolated digits using HMMs with continuous mixture densities, AT&T Tech. J. **64**(6), 1211–1233 (1985)

27.5 L.R. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE **77**(2), 257–286 (1989)

27.6 P.J. Price, W. Fisher, J. Bernstein, D.S. Pallet: The DARPA 1000-word resource management database for continuous speech recognition, Proc. IEEE ICASSP **1**, 651–654 (1988)

27.7 S.J. Young, L.L. Chase: Speech recognition evaluation: A review of the US CSR and LVCSR programmes, Comput. Speech Lang. **12**(4), 263–279 (1998)

27.8 D.S. Pallet, J.G. Fiscus, J. Garofolo, A. Martin, M. Przybocki: *1998 Broadcast News Benchmark Test Results: English and Non-English Word Error Rate Performance Measures, Tech. Rep.* (National Institute of Standards and Technology, Gaithersburg 1998)

27.9 J.J. Godfrey, E.C. Holliman, J. McDaniel: Switchboard, Proc. IEEE ICASSP **1**, 517–520 (1992)

27.10 G. Evermann, H.Y. Chan, M.J.F. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, P. Woodland: Development of the 2003 CU-HTK Conversational Telephone Speech Transcription System, Proc. IEEE ICASSP (2004)

27.11 H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, G. Zweig: The IBM 2004 conversational telephony system for rich transcription, Proc. IEEE ICASSP (2005)

27.12 S.J. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. Woodland: *The HTK Book Version 3.4* (Cambridge University, Cambridge 2006), http://htk.eng.cam.ac.uk

27.13 S.J. Young: Large vocabulary continuous speech recognition, IEEE Signal Process. Mag. **13**(5), 45–57 (1996)

27.14 S.B. Davis, P. Mermelstein: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Trans. Acoust. Speech Signal Process. **28**(4), 357–366 (1980)

27.15 H. Hermansky: Perceptual linear predictive (PLP) analysis of speech, J. Acoust. Soc. Am. **87**(4), 1738–1752 (1990)

27.16 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc. B **39**, 1–38 (1977)

27.17 S.J. Young, J.J. Odell, P.C. Woodland: Tree-based state tying for high accuracy acoustic modelling, Proc. Human Language Technology Workshop (Morgan Kaufman, San Francisco 1994) pp. 307–312

27.18 X. Luo, F. Jelinek: Probabilistic classification of HMM states for large vocabulary, Proc. Int. Conf. Acoust. Speech Signal Process. (1999) pp. 2044–2047

27.19 S.M. Katz: Estimation of probabilities from sparse data for the language model component of a speech recogniser, IEEE Trans. ASSP **35**(3), 400–401 (1987)

27.20 H. Ney, U. Essen, R. Kneser: On structuring probabilistic dependences in stochastic language modelling, Comput. Speech Lang. **8**(1), 1–38 (1994)

27.21 S.F. Chen, J. Goodman: An empirical study of smoothing techniques for language modelling, Comput. Speech Lang. **13**, 359–394 (1999)

27.22 P.F. Brown, V.J. Della Pietra, P.V. de Souza, J.C. Lai, R.L. Mercer: Class-based n-gram models of natural language, Comput. Linguist. **18**(4), 467–479 (1992)

27.23 R. Kneser, H. Ney: Improved clustering techniques for class-based statistical language modelling, Proc. Eurospeech **93**, 973–976 (1993)

27.24 S. Martin, J. Liermann, H. Ney: Algorithms for bigram and trigram word clustering, Proc. Eurospeech **2**, 1253–1256 (1995)

27.25 S.J. Young, N.H. Russell, J.H.S. Thornton: *Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems, Tech. Rep.*

*CUED/F-INFENG/TR38* (Cambridge University, Cambridge 1989)
27.26 K. Demuynck, J. Duchateau, D. van Compernolle: A static lexicon network representation for cross-word context dependent phones, Proc. Eurospeech **97**, 143–146 (1997)
27.27 S.J. Young: Generating multiple solutions from connected word DP recognition algorithms, Proc. IOA Autumn Conf. **6**, 351–354 (1984)
27.28 H. Thompson: Best-first enumeration of paths through a lattice – an active chart parsing solution, Comput. Speech Lang. **4**(3), 263–274 (1990)
27.29 F. Richardson, M. Ostendorf, J.R. Rohlicek: Lattice-based search strategies for large vocabulary recognition, Proc. IEEE ICASSP **1**, 576–579 (1995)
27.30 L. Mangu, E. Brill, A. Stolcke: Finding consensus among words: Lattice-based word error minimisation, Comput. Speech Lang. **14**(4), 373–400 (2000)
27.31 G. Evermann, P.C. Woodland: Posterior probability decoding confidence estimation and system combination, Proc. Speech Transcription Workshop (2000)
27.32 G. Evermann, P.C. Woodland: Large vocabulary decoding and confidence estimation using word posterior probabilities, Proc. IEEE ICASSP (2000) pp. 1655–1658
27.33 V. Goel, S. Kumar, B. Byrne: Segmental minimum bayes-risk ASR voting strategies, Proc. ICSLP (2000)
27.34 J. Fiscus: A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER), Proc. IEEE ASRU Workshop (1997) pp. 347–352
27.35 D. Hakkani-Tur, F. Bechet, G. Riccardi, G. Tur: Beyond ASR 1-best: Using word confusion networks in spoken language understanding, Comput. Speech Lang. **20**(4), 495–514 (2006)
27.36 J.J. Odell, V. Valtchev, P.C. Woodland, S.J. Young: A one-pass decoder design for large vocabulary recognition, Proc. Human Language Technology Workshop (Morgan Kaufman, San Francisco 1994) pp. 405–410
27.37 X. Aubert, H. Ney: Large vocabulary continuous speech recognition using word graphs, Proc. IEEE ICASSP **1**, 49–52 (1995)
27.38 M. Mohri, F. Pereira, M. Riley: Weighted finite state transducers in speech recognition, Comput. Speech Lang. **16**(1), 69–88 (2002)
27.39 F. Jelinek: A fast sequential decoding algorithm using a stack, IBM J. Res. Dev. **13**, 675–685 (1969)
27.40 D.B. Paul: Algorithms for an optimal A* search and linearizing the search in the stack decoder, Proc. IEEE ICASSP **91**, 693–996 (1991)
27.41 A. Nadas: A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood, IEEE Trans. Acoust. Speech Signal Process. **31**(4), 814–817 (1983)
27.42 B.-H. Juang, S.E. Levinson, M.M. Sondhi: Maximum likelihood estimation for multivariate mixture observations of Markov chains, IEEE Trans. Inform. Theory **32**(2), 307–309 (1986)
27.43 L.R. Bahl, P.F. Brown, P.V. de Souza, R.L. Mercer: Maximum mutual information estimation of hidden Markov model parameters for speech recognition, Proc. IEEE ICASSP **96**, 49–52 (1986)
27.44 V. Valtchev, J.J. Odell, P.C. Woodland, S.J. Young: MMIE training of large vocabulary recognition systems, Speech Commun. **22**, 303–314 (1997)
27.45 R. Schluter, B. Muller, F. Wessel, H. Ney: Interdependence of language models and discriminative training, Proc. IEEE ASRU Workshop (1999) pp. 119–122
27.46 P. Woodland, D. Povey: Large scale discriminative training of hidden Markov models for speech recognition, Comput. Speech Lang. **16**, 25–47 (2002)
27.47 W. Chou, C.H. Lee, B.-H. Juang: Minimum error rate training based on N-best string models, Proc. IEEE ICASSP **93**, 652–655 (1993)
27.48 D. Povey, P. Woodland: Minimum phone error and I-smoothing for improved discriminative training, Proc. IEEE ICASSP **2002**, I–105–I–108 (2002)
27.49 P.S. Gopalakrishnan, D. Kanevsky, A. Nadas, D. Nahamoo, M.A. Picheny: Decoder selection based on cross-entropies, Proc. IEEE ICASSP **1**, 20–23 (1988)
27.50 P.C. Woodland, D. Povey: Large scale discriminative training for speech recognition, ISCA ITRW Automatic Speech Recognition: Challenges for the Millenium (2000) pp. 7–16
27.51 M.J.F. Gales: Semi-tied covariance matrices for hidden Markov models, IEEE Trans. Speech Audio Process. **7**(3), 272–281 (1999)
27.52 A.V. Rosti, M. Gales: Factor analysed hidden Markov models for speech recognition, Comput. Speech Language **18**(2), 181–200 (2004)
27.53 S. Axelrod, R. Gopinath, P. Olsen: Modeling with a subspace constraint on inverse covariance matrices, Proc. ICSLP (2002)
27.54 P. Olsen, R. Gopinath: Modeling inverse covariance matrices by basis expansion, Proc. ICSLP (2002)
27.55 N. Kumar, A.G. Andreou: Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition, Speech Commun. **26**, 283–297 (1998)
27.56 M.J.F. Gales: Maximum likelihood multiple subspace projections for hidden Markov models, IEEE Trans. Speech Audio Process. **10**(2), 37–47 (2002)
27.57 T. Hain, P.C. Woodland, T.R. Niesler, E.W.D. Whittaker: The 1998 HTK system for transcription of conversational telephone speech, Proc. IEEE ICASSP **99**, 57–60 (1999)
27.58 G. Saon, A. Dharanipragada, D. Povey: Feature space Gaussianization, Proc. IEEE ICASSP (2004)
27.59 S.S. Chen, R. Gopinath: Gaussianization, Proc. Neural Information Processing Systems (MIT Press, 2000)

Amazon EX1010 Page 574

27.60 M.J.F. Gales, B. Jia, X. Liu, K.C. Sim, P. Woodland, K. Yu: Development of the CUHTK 2004 RT04 Mandarin conversational telephone speech transcription system, Proc. IEEE ICASSP (2005)
27.61 L. Lee, R.C. Rose: Speaker normalisation using efficient frequency warping procedures, Proc. IEEE ICASSP (1996)
27.62 J. McDonough, W. Byrne, X. Luo: Speaker normalisation with all pass transforms, Proc. ISCLP, Vol. 98 (1998)
27.63 D.Y. Kim, S. Umesh, M.J.F. Gales, T. Hain, P. Woodland: Using VTLN for broadcast news transcription, Proc. ICSLP (2004)
27.64 J.-L. Gauvain, C.-H. Lee: Maximum a posteriori estimation of multivariate Gaussian mixture observations of Markov chains, IEEE Trans. Speech Audio Process. **2**(2), 291–298 (1994)
27.65 S.M. Ahadi, P.C. Woodland: Combined Bayesian and predictive techniques for rapid speaker adaptation of continuous density hidden Markov models, Comput. Speech Lang. **11**(3), 187–206 (1997)
27.66 K. Shinoda, C.H. Lee: Structural MAP speaker adaptation using hierarchical priors, Proc. ASRU, Vol. 97 (1997)
27.67 C.J. Leggetter, P.C. Woodland: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models, Comput. Speech Lang. **9**(2), 171–185 (1995)
27.68 M.J.F. Gales: Maximum likelihood linear transformations for HMM-based speech recognition, Comput. Speech Lang. **12**, 75–98 (1998)
27.69 F. Wallhof, D. Willett, G. Rigoll: Frame-discriminative and confidence-driven adaptation for LVCSR, Proc. IEEE ICASSP (2000) pp. 1835–1838
27.70 L. Wang, P. Woodland: Discriminative adaptive training using the MPE criterion, Proc. ASRU (2003)
27.71 S. Tsakalidis, V. Doumpiotis, W.J. Byrne: Discriminative linear transforms for feature normalisation and speaker adaptation in HMM estimation, IEEE Trans. Speech Audio Process. **13**(3), 367–376 (2005)
27.72 P. Woodland, D. Pye, M.J.F. Gales: Iterative unsupervised adaptation using maximum likelihood linear regression, Proc. ICSLP (1996) pp. 1133–1136
27.73 M. Padmanabhan, G. Saon, G. Zweig: Lattice-based unsupervised MLLR for speaker adaptation, Proc. ITRW ASR2000: ASR Challenges for the New Millenium (2000) pp. 128–132
27.74 T.J. Hazen, J. Glass: A comparison of novel techniques for instantaneous speaker adaptation, Proc. Eurospeech **97**, 2047–2050 (1997)
27.75 R. Kuhn, L. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Finke, K. Field, M. Contolini: Eigenvoices for speaker adaptation, Proc. ICSLP (1998)
27.76 M.J.F. Gales: Cluster adaptive training of hidden Markov models, IEEE Trans. Speech Audio Process. **8**, 417–428 (2000)
27.77 K. Yu, M.J.F. Gales: Discriminative cluster adaptive training, IEEE Trans. Speech Audio Process. **14**, 1694–1703 (2006)
27.78 T. Anastasakos, J. McDonough, R. Schwartz, J. Makhoul: A compact model for speaker adaptive training, Proc. ICSLP (1996)
27.79 R. Sinha, M.J.F. Gales, D.Y. Kim, X. Liu, K.C. Sim, P.C. Woodland: The CU-HTK Mandarin broadcast news transcription system, Proc. IEEE ICASSP (2006)



Amazon EX1010 Page 575

# 28. Speech Recognition with Weighted Finite-State Transducers

M. Mohri, F. Pereira, M. Riley

This chapter describes a general representation and algorithmic framework for speech recognition based on weighted finite-state transducers. These transducers provide a common and natural representation for major components of speech recognition systems, including hidden Markov models (HMMs), context-dependency models, pronunciation dictionaries, statistical grammars, and word or phone lattices. General algorithms for building and optimizing transducer models are presented, including composition for combining models, weighted determinization and minimization for optimizing time and space requirements, and a weight pushing algorithm for redistributing transition weights optimally for speech recognition. The application of these methods to large-vocabulary recognition tasks is explained in detail, and experimental results are given, in particular for the North American Business News (NAB) task, in which these methods were used to combine HMMs, full cross-word triphones, a lexicon of 40 000 words, and a large trigram grammar into a single weighted transducer that is only somewhat larger than the trigram word grammar and that runs NAB in real time on a very simple decoder. Another example demonstrates that the same methods can be used to optimize lattices for second-pass recognition.

28.1 **Definitions** ..... 559
28.2 **Overview** ..... 560
28.2.1 Weighted Acceptors ..... 560
28.2.2 Weighted Transducers ..... 561
28.2.3 Composition ..... 562
28.2.4 Determinization ..... 563
28.2.5 Minimization ..... 565
28.2.6 Speech Recognition Transducers ..... 566
28.3 **Algorithms** ..... 567
28.3.1 Preliminaries ..... 567
28.3.2 Composition ..... 568
28.3.3 Determinization ..... 570
28.3.4 Weight Pushing ..... 572
28.3.5 Minimization ..... 573
28.4 **Applications to Speech Recognition** ..... 574
28.4.1 Speech Recognition Transducers ..... 574
28.4.2 Transducer Standardization ..... 577
28.5 **Conclusion** ..... 582
**References** ..... 582



## 28.1 Definitions

Much of current large-vocabulary speech recognition is based on models such as hidden Markov models (HMMs), lexicons, or $n$-gram statistical language models that can be represented by *weighted finite-state transducers*. Even when richer models are used, for instance context-free grammars for spoken-dialog applications, they are often restricted, for efficiency reasons, to regular subsets, either by design or by approximation [28.1–3].

A finite-state transducer is a finite automaton whose state transitions are labeled with both input and output symbols. Therefore, a path through the transducer encodes a mapping from an input symbol sequence, or *string*, to an output string. A *weighted* transducer puts weights on transitions in addition to the input and output symbols. Weights may encode probabilities, durations, penalties, or any other quantity that accumulates along paths to compute the overall weight of mapping an input string to an output string. Weighted transducers are thus a natural choice to represent the probabilistic finite-state models prevalent in speech processing.

We present a detailed view of the use of weighted finite-state transducers in speech recognition [28.4–10]. We show that common methods for combining and optimizing probabilistic models in speech processing can be generalized and efficiently implemented by translation to mathematically well-defined operations on weighted transducers. Furthermore, new optimization

opportunities arise from viewing all symbolic levels of speech recognition modeling as weighted transducers. Thus, weighted finite-state transducers define a common framework with shared algorithms for the representation and use of the models in speech recognition that has important algorithmic and software engineering benefits.

We begin with an overview in Sect. 28.2, which informally introduces weighted finite-state transducers and algorithms, and motivates the methods by showing how they are applied to speech recognition. This section may suffice for those only interested in a brief tour of these methods. In the subsequent two sections, we give a more-detailed and precise account. Section 28.3 gives formal definitions of weighted finite-state transducer concepts and corresponding algorithm descriptions. Section 28.4 gives a detailed description of how to apply these methods to large-vocabulary speech recognition and shows performance results. These sections are appropriate for those who wish to understand the algorithms more fully or wish to replicate the results.



## 28.2 Overview

We start with an informal overview of weighted automata and transducers, outlines of some of the key algorithms that support the ASR applications described in this chapter – *composition*, *determinization*, and *minimization* – and their application to speech recognition.

### 28.2.1 Weighted Acceptors

Weighted finite automata (or weighted acceptors) are used widely in automatic speech recognition (ASR). Figure 28.1 gives simple, familiar examples of weighted automata as used in ASR. The automaton in Fig. 28.1a is a toy finite-state *language model*. The legal word strings are specified by the words along each complete path, and their probabilities by the product of the corresponding transition probabilities. The automaton in Fig. 28.1b gives the possible pronunciations of one word, 'data', used in the language model. Each legal pronunciation is the phone string along a complete path, and its probability is given by the product of the corresponding transition probabilities. Finally, the automaton in Fig. 28.1c encodes a typical left-to-right, three-distribution HMM structure for one phone, with the labels along a complete path specifying legal strings of acoustic distributions for that phone.

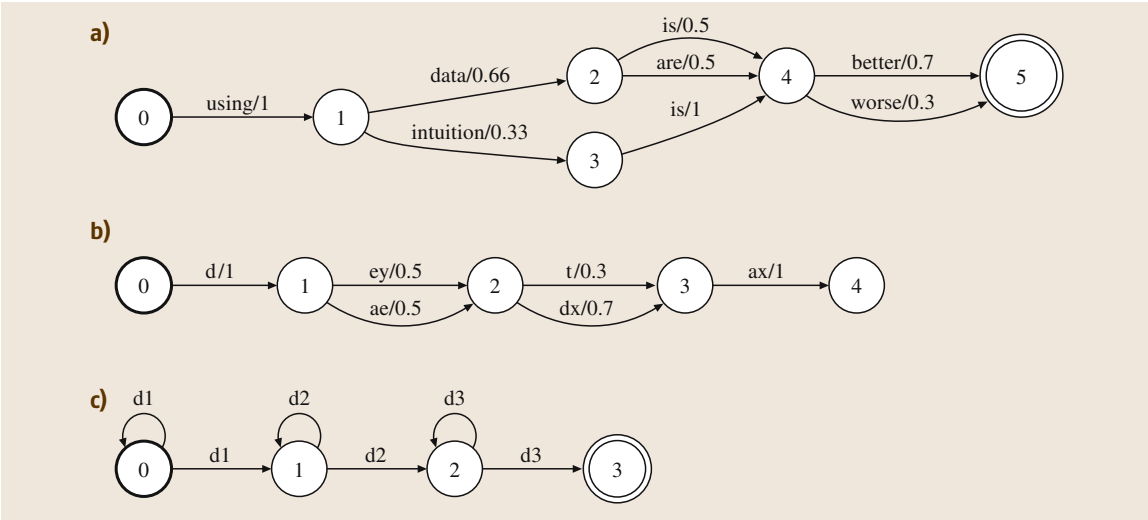



**Fig. 28.1a–c** Weighted finite-state acceptor examples. By convention, the states are represented by *circles* and marked with their unique number. The initial state is represented by a *bold circle*, final states by *double circles*. The label $l$ and weight $w$ of a transition are marked on the corresponding directed arc by $l/w$. When explicitly shown, the final weight $w$ of a final state $f$ is marked by $f/w$

These automata consist of a set of states, an initial state, a set of final states (with final weights), and a set of transitions between states. Each transition has a source state, a destination state, a label, and a weight. Such automata are called *weighted finite-state acceptors* (WFSAs), since they *accept* or *recognize* each string that can be read along a path from the start state to a final state. Each accepted string is assigned a weight, namely the accumulated weights along accepting paths for that string, including the final weights. An acceptor as a whole represents a set of strings, namely those that it accepts. As a weighted acceptor, it also associates to each accepted string the accumulated weights of their accepting paths.

Speech recognition architectures commonly give the run-time decoder the task of combining and optimizing automata such as those in Fig. 28.1. The decoder finds word pronunciations in its lexicon and substitutes them into the grammar. Phonetic tree representations may be used at this point to reduce path redundancy and thus improve search efficiency, especially for large-vocabulary recognition [28.11]. The decoder then identifies the correct context-dependent models to use for each phone in context, and finally substitutes them to create an HMM-level transducer. The software that performs these operations is usually tied to particular model topologies. For example, the context-dependent models might have to be triphonic, the grammar might be restricted to trigrams, and the alternative pronunciations might have to be enumerated in the lexicon. In addition, these automata combinations and optimizations are applied in a preprogrammed order to a prespecified number of levels.

## 28.2.2 Weighted Transducers

Our approach uses finite-state transducers, rather than acceptors, to represent the $n$-gram grammars, pronunciation dictionaries, context-dependency specifications, HMM topology, word, phone or HMM segmentations, lattices, and $n$-best output lists encountered in ASR. The transducer representation provides general methods for combining models and optimizing them, leading to both simple and flexible ASR decoder designs.

A weighted finite-state transducer (WFST) is quite similar to a weighted acceptor except that it has an input label, an output label, and a weight on each of its transitions. The examples in Fig. 28.2 encode (a superset of) the information in the WFSAs of Fig. 28.1a,b as WFSTs. Figure 28.2a represents the same language model as Fig. 28.1a by giving each transition identical input and output labels. This adds no new information, but is a convenient way we often use to treat acceptors and transducers uniformly.

Figure 28.2b represents a toy pronunciation lexicon as a mapping from phone strings to words in the lexicon, in this example 'data' and 'dew', with probabilities representing the likelihoods of alternative pronunciations. It *transduces* a phone string that can be read along a path from the start state to a final state to a word string with a particular weight. The word corresponding to a pronunciation is output by the transition that consumes the

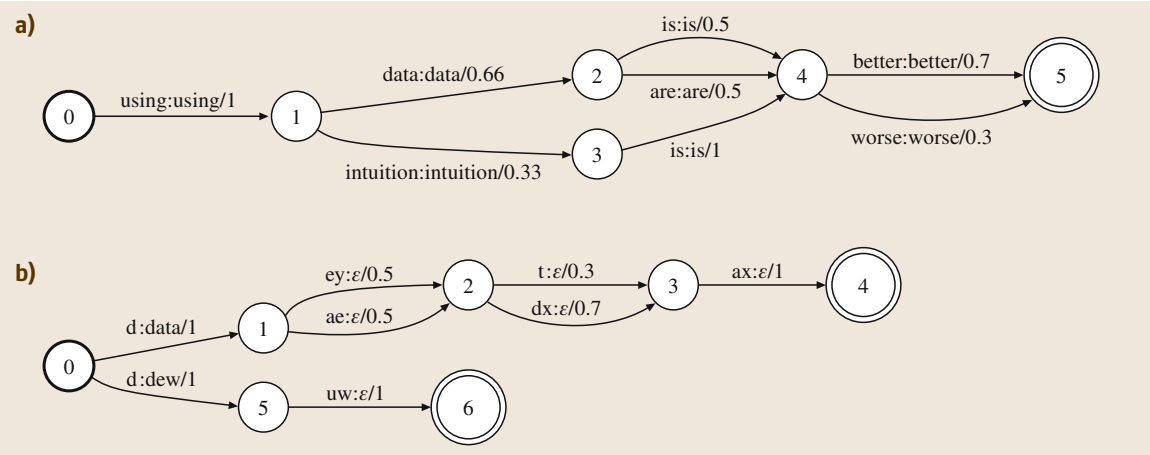



**Fig. 28.2a,b** Weighted finite-state transducer examples. These are similar to the weighted acceptors in Fig. 28.1 except output labels are introduced in each transition. The input label $i$, the output label $o$, and weight $w$ of a transition are marked on the corresponding directed arc by $i : o/w$

first phone for that pronunciation. The transitions that consume the remaining phones output no further symbols, indicated by the null symbol $\varepsilon$ as the transition's output label. In general, an $\varepsilon$ input label marks a transition that consumes no input, and an $\varepsilon$ output label marks a transition that produces no output.

This transducer has more information than the WFSA in Fig. 28.1b. Since words are encoded by the output label, it is possible to combine the pronunciation transducers for more than one word without losing word identity. Similarly, HMM structures of the form given in Fig. 28.1c can be combined into a single transducer that preserves phone model identity. This illustrates the key advantage of a transducer over an acceptor: the transducer can represent a relationship between two levels of representation, for instance between phones and words or between HMMs and context-independent phones. More precisely, a transducer specifies a binary relation between strings: two strings are in the relation when there is a path from an initial to a final state in the transducer that has the first string as the sequence of input labels along the path and the second string as the sequence of output labels along the path ($\varepsilon$ symbols are left out in both input and output). In general, this is a relation rather than a function since the same input string might be transduced to different strings along two distinct paths. For a weighted transducer, each string pair is also associated with a weight.

We rely on a common set of weighted transducer operations to combine, optimize, search, and prune them [28.4]. Each operation implements a single, well-defined function that has its foundations in the mathematical theory of rational power series [28.12–14]. Many of those operations are the weighted transducer generalizations of classical algorithms for unweighted acceptors. We have brought together those and a variety of auxiliary operations in a comprehensive weighted finite-state machine software library (FsmLib) available for non-commercial use from the AT&T Labs – research web site [28.4]. The OpenFST open-source library in C++ provides a new implementation of the main weighted transducer algorithms [28.15].

Basic *union*, *concatenation*, and *Kleene closure* operations combine transducers in parallel, in series, and with arbitrary repetition, respectively. Other operations convert transducers to acceptors by projecting onto the input or output label set, find the best or the $n$ best paths in a weighted transducer, remove unreachable states and transitions, and sort acyclic automata topologically.

Where possible, we provided *lazy* (also called *on-demand*) implementations of algorithms. Any finite-state object `fsm` can be accessed with the three main methods `fsm.start()`, `fsm.final(state)`, and `fsm.transitions(state)` that return the start state, the final weight of a state, and the transitions leaving a state, respectively. This interface can be implemented for concrete automata in an obvious way: the methods simply return the requested information from a stored representation of the automaton. However, the interface can also be given lazy implementations. For example, the lazy union of two automata returns a new lazy `fsm` object. When the object is first constructed, the lazy implementation just initializes internal bookkeeping data. It is only when the interface methods request the start state, the final weights, or the transitions (and their destination states) leaving a state, that this information is actually computed, and optionally cached inside the object for later reuse. This approach has the advantage that, if only a part of the result of an operation is needed (for example in a pruned search), then the unused part is never computed, saving time and space. We refer the interested reader to the library documentation and an overview of the library [28.4] for further details on lazy finite-state objects.

We now discuss the key transducer operations that are used in our speech applications for model combination, redundant-path removal, and size reduction.

### 28.2.3 Composition

Composition is the transducer operation for combining different levels of representation. For instance, a pronunciation lexicon can be composed with a word-level grammar to produce a phone-to-word transducer whose word strings are restricted to the grammar. A variety of ASR transducer combination techniques, both context independent and context dependent, can be conveniently and efficiently implemented with composition.

As previously noted, a transducer represents a binary relation between strings. The composition of two transducers represents their relational composition. In particular, the composition $T = T_1 \circ T_2$ of two transducers $T_1$ and $T_2$ has exactly one path mapping string $u$ to string $w$ for each pair of paths, the first in $T_1$ mapping $u$ to some string $v$ and the second in $T_2$ mapping $v$ to $w$. The weight of a path in $T$ is computed from the weights of the two corresponding paths in $T_1$ and $T_2$ with the same operation that computes the weight of a path from the weights of its transitions. If the transition weights represent probabilities, that operation is the product. If instead the weights represent log probabilities or negative log probabilities as is common in

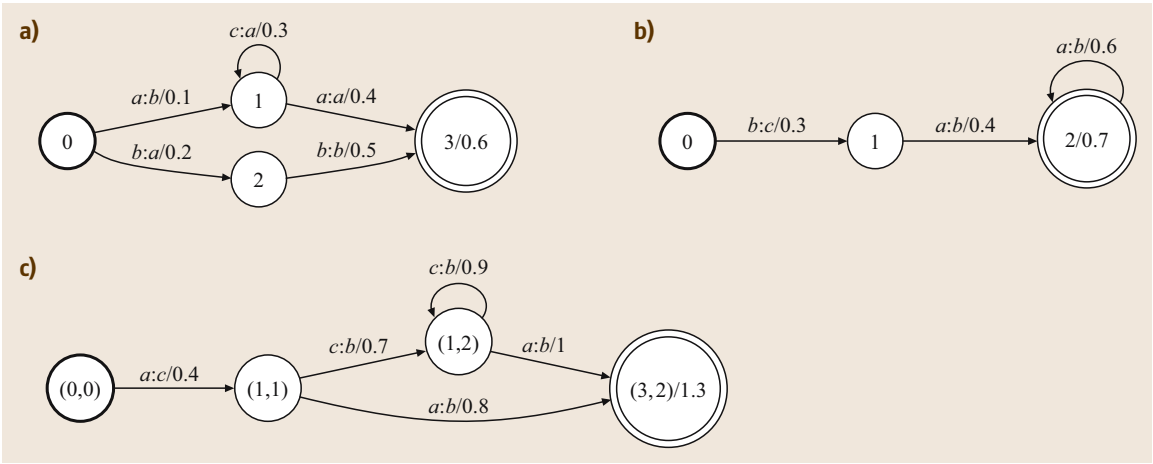



**Fig. 28.3a–c** Example of transducer composition

ASR for numerical stability, the operation is the sum. More generally, the weight operations for a weighted transducer can be specified by a *semiring* [28.12–14], as discussed in more detail in Sect. 28.3.

The weighted composition algorithm generalizes the classical state-pair construction for finite automata intersection [28.16] to weighted acceptors and transducers. The states of the composition $T$ are pairs of a $T_1$ state and a $T_2$ state. $T$ satisfies the following conditions:

1. its initial state is the pair of the initial state of $T_1$ and the initial state of $T_2$;
2. its final states are pairs of a final state of $T_1$ and a final state of $T_2$, and
3. there is a transition $t$ from $(q_1, q_2)$ to $(r_1, r_2)$ for each pair of transitions $t_1$ from $q_1$ to $r_1$ and $t_2$ from $q_2$ to $r_2$ such that the output label of $t_1$ matches the input label of $t_2$.

The transition $t$ takes its input label from $t_1$, its output label from $t_2$, and its weight is the combination of the weights of $t_1$ and $t_2$ done with the same operation that combines weights along a path. Since this computation is *local* – it involves only the transitions leaving two states being paired – it can be given a lazy implementation in which the composition is generated only as needed by other operations on the composed automaton. Transitions with $\varepsilon$-labels in $T_1$ or $T_2$ must be treated specially, as discussed in Sect. 28.3. Figure 28.3a and b show two simple transducers and the result of their composition, Fig. 28.3c. The weight of a path in the resulting transducer is the sum of the weights of the matching paths in $T_1$ and $T_2$ (as when the weights represent negative log probabilities).

Since we represent weighted acceptors by weighted transducers in which the input and output labels of each transition are identical, the intersection of two weighted acceptors is just the composition of the corresponding transducers.

### 28.2.4 Determinization

In a *deterministic automaton*, each state has at most one transition with any given input label and there are no input $\varepsilon$-labels. Figure 28.4a gives an example of a nondeterministic weighted acceptor: at state 0, for instance, there are two transitions with the same label $a$. The automaton in Fig. 28.4b, on the other hand, is deterministic.

The key advantage of a deterministic automaton over equivalent nondeterministic ones is its irredundancy: it contains at most one path matching any given input string, thereby reducing the time and space needed to process the string. This is particularly important in ASR due to pronunciation lexicon redundancy in large-vocabulary tasks. The familiar tree lexicon in ASR is a deterministic pronunciation representation [28.11].

To benefit from determinism, we use a determinization algorithm that transforms a nondeterministic weighted automaton into an equivalent deterministic automaton. Two weighted acceptors are *equivalent* if they associate the same weight to each input string; weights may be distributed differently along the paths of two equivalent acceptors. Two weighted transducers are equivalent if they associate the same output string and weights to each input string; the distribution of the

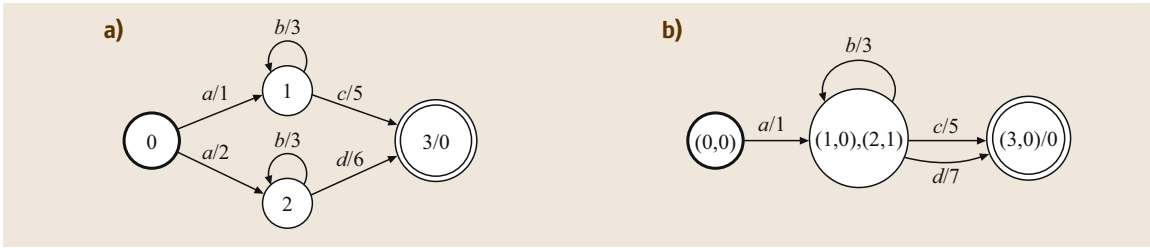



**Fig. 28.4a,b** Determinization of weighted automata. **(a)** Weighted automaton over the tropical semiring $A$. **(b)** Equivalent weighted automaton $B$ obtained by the determinization of $A$



weight or output labels along paths need not be the same in the two transducers.

If we apply the weighted determinization algorithm to the union of a set of chain automata, each representing a single word pronunciation, we obtain a tree-shaped automaton. However, the result of this algorithm on more-general automata may not be a tree, and in fact may be much more compact than a tree. The algorithm can produce results for a broad class of automata with cycles that have no tree representation.

Weighted determinization generalizes the classical subset method for determinizing unweighted finite automata [28.17]. Unlike in the unweighted case, not all weighted automata can be determinized. Conditions for determinizability are discussed in Sect. 28.3.3. Fortunately, most weighted automata used in speech processing can be either determinized directly or easily made determinizable with simple transformations, as we discuss in Sects. 28.3.3 and 28.4.1. In particular, any acyclic weighted automaton can be determinized.

To eliminate redundant paths, weighted determinization needs to calculate the combined weight of all paths with the same labeling. When each path represents a disjoint event with a probability given by its weight, the combined weight, representing the probability of the common labeling for that set of paths, would be the sum of the weights of the paths. Alternatively, we may just want to keep the most probable path, as is done in shortest path algorithms, leading to the so-called *Viterbi approximation*. When weights are negative log probabilities, these two alternatives correspond respectively to log summation and taking the minimum. In the general case, we use one operation, denoted $\otimes$, for combining weights along paths and for composition, and a second operation, denoted $\oplus$, to combine identically labeled paths. Some common choices of $(\oplus, \otimes)$ include $(\max, +)$, $(+, *)$, $(\min, +)$, and $(-\log(e^{-x} + e^{-y}), +)$. In speech applications, the first two are appropriate for probabilities, the last two for the corresponding negative log probabilities. More generally, as we will see in Sect. 28.3, many of the weighted automata algorithms apply when the two operations define an appropriate semiring. The choices $(\min, +)$, and $(-\log(e^{-x} + e^{-y}), +)$ are called the *tropical* and log semirings, respectively.

Our discussion and examples of determinization and, later, minimization will be illustrated with weighted acceptors. The *string* semiring, whose two operations are longest common prefix and concatenation, can be used to treat the output strings as weights. By this method, the transducer case can be handled as well; see *Mohri* [28.6] for details.

We will now work through an example of determinization with weights in the tropical semiring. Figure 28.4b shows the weighted determinization of automaton $A_1$ from Fig. 28.4a. In general, the determinization of a weighted automaton is equivalent to the original, that is, it associates the same weight to each input string. For example, there are two paths corresponding to the input string $ab$ in $A_1$, with weights $\{1+3=4, 2+3=5\}$. The minimum 4 is also the weight associated by $A_2$ to the string $ab$.

In the classical subset construction for determinizing unweighted automata, all the states reachable by a given input from the initial state are placed in the same subset. In the weighted case, transitions with the same input label can have different weights, but only the minimum of those weights is needed and the leftover weights must be tracked. Thus, the subsets in weighted determinization contain pairs $(q, w)$ of a state $q$ of the original automaton and a leftover weight $w$.

The initial subset is $[(i, 0)]$, where $i$ is the initial state of the original automaton. For example, in Fig. 28.4, the initial subset for automaton $B$ is $[(0, 0)]$. Each new subset $S$ is processed in turn. For each element $a$ of the input alphabet $\Sigma$ labeling at least one transition leaving a state of $S$, a new transition $t$ leaving $S$ is constructed

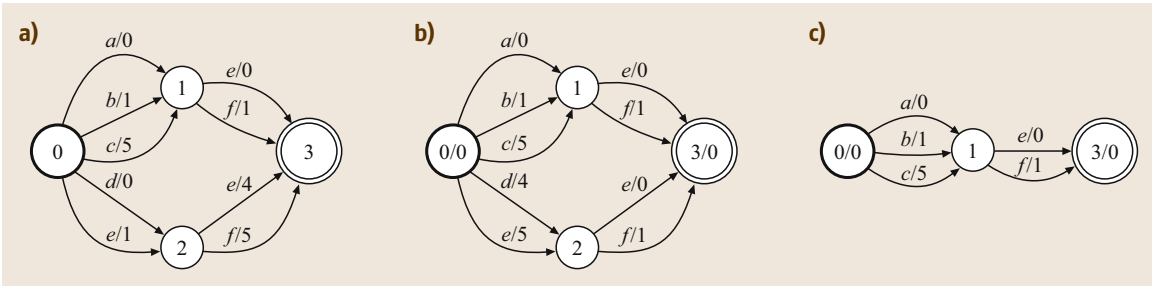



**Fig. 28.5a–c** Weight pushing and minimization. **(a)** Deterministic weighted automaton $A$. **(b)** Equivalent weighted automaton $B$ obtained by weight pushing in the tropical semiring. **(c)** Minimal weighted automaton equivalent to $A$

in the result automaton. The input label of $t$ is $a$ and its weight is the minimum of the sums $w+l$, where $w$ is $s$'s leftover weight, and $l$ is the weight of an $a$-transition leaving a state $s$ in $S$. The destination state of $t$ is the subset $S'$ containing those pairs $(q', w')$ in which $q'$ is a state reached by a transition labeled with $a$ from a state of $S$ and $w'$ is the appropriate leftover weight.

For example, in Fig. 28.4, the transition leaving $(0, 0)$ in $B$ labeled with $a$ is obtained from the two transitions labeled with $a$ leaving state 0 in $A$: its weight is the minimum of the weight of those two transitions, and its destination state is the subset $S' = \{(1, 1-1=0), (2, 2-1=1)\}$. The algorithm is described in more detail in Sect. 28.3.3.

It is clear that the transitions leaving a given state in the determinization of an automaton can be computed from the subset for the state and the transitions leaving the states in the subset, as is the case for the classical nondeterministic finite automata (NFA) determinization algorithm. In other words, the weighted determinization algorithm is locally like the composition algorithm, and can thus be given a lazy implementation that creates states and transitions only as needed.

### 28.2.5 Minimization

Given a deterministic automaton, we can reduce its size by minimization, which can save both space and time. Any deterministic unweighted automaton can be minimized using classical algorithms [28.18, 19]. In the same way, any deterministic weighted automaton $A$ can be minimized using our minimization algorithm, which extends the classical algorithm [28.6]. The resulting weighted automaton $B$ is equivalent to the automaton $A$, and has the least number of states and the least number of transitions among all deterministic weighted automata equivalent to $A$.

As we will see in Sect. 28.3.5, weighted minimization is quite efficient, indeed as efficient as classical deterministic finite automata (DFA) minimization.

We can view the deterministic weighted automaton $A$ in Fig. 28.5a as an unweighted automaton by interpreting each pair $(a, w)$ of a label $a$ and a weight $w$ as a single label. We can then apply the standard DFA minimization algorithm to this automaton. However, since the pairs for different transitions are all distinct, classical minimization would have no effect on $A$.

The size of $A$ can still be reduced by using true weighted minimization. This algorithm works in two steps: the first step *pushes* weight among transitions, and the second applies the classical minimization algorithm to the result with each distinct label–weight pair viewed as a distinct symbol, as described above. Weight pushing is useful not only as a first step of minimization but also to redistribute weight among transitions to improve search, especially pruned search. The algorithm is described in more detail in Sect. 28.3.4, and analyzed in [28.20]. Its applications to speech recognition are discussed in [28.21].

Pushing is a special case of *reweighting*. We describe reweighting in the case of the tropical semiring; similar definitions can be given for other semirings. A (nontrivial) weighted automaton can be reweighted in an infinite number of ways that produce equivalent automata. To see how, let $i$ be $A$'s initial state and assume for convenience that $A$ has a single final state $f$. Any automaton can be transformed into an equivalent automaton with a single final state by adding a super-final state, making all previously final states non-final, and adding from each previously final state $f$ with weight $\rho(f)$ an $\varepsilon$-transition with the weight $\rho(f)$ to the super-final state. Let $V : Q \to \mathbb{R}$ be an arbitrary *potential* function on states. Update each weight $w[t]$ of the transition $t$ from state $p[t]$ to $n[t]$ as

follows:

$$w[t] \leftarrow w[t] + [V(n[t]) - V(p[t])] \,, \tag{28.1}$$

and the final weight $\rho(f)$ as

$$\rho(f) \leftarrow \rho(f) + [V(i) - V(f)] \,. \tag{28.2}$$

It is easy to see that with this reweighting each potential internal to any successful path from the initial state to the final state is added and then subtracted, making the overall change in path weight:

$$[V(f) - V(i)] + [V(i) - V(f)] = 0 \,. \tag{28.3}$$

Thus, reweighting does not affect the total weight of a successful path and the resulting automaton is equivalent to the original.

To push the weight in $A$ towards the initial state as much as possible, a specific potential function is chosen, the one that assigns to each state the lowest path weight from that state to the final state. After pushing, the lowest-cost path (excluding the final weight) from every state to the final state will thus be 0.

Figure 28.5b shows the result of pushing for the input $A$. Thanks to pushing, the size of the automaton can then be reduced using classical minimization. Figure 28.5c illustrates the result of the final step of the algorithm. No approximation or heuristic is used: the resulting automaton $C$ is equivalent to $A$.

### 28.2.6 Speech Recognition Transducers

As an illustration of these methods applied to speech recognition, we describe how to construct a single, statically compiled and optimized recognition transducer that maps from context-dependent phones to words. This is an attractive choice for tasks that have fixed acoustic, lexical, and grammatical models since the static transducer can be searched simply and efficiently with no recognition-time overhead for model combination and optimization.

Consider the pronunciation lexicon in Fig. 28.2b. Suppose we form the union of this transducer with the pronunciation transducers for the remaining words in the grammar $G$ of Fig. 28.2a by creating a new super-initial state and connecting an $\varepsilon$-transition from that state to the former start states of each pronunciation transducer. We then take its Kleene closure by connecting an $\varepsilon$-transition from each final state to the initial state. The resulting pronunciation lexicon $L$ would pair any word string from that vocabulary to their corresponding pronunciations. Thus,

$$L \circ G \tag{28.4}$$

gives a transducer that maps from phones to word strings restricted to $G$.

We used composition here to implement a context-independent substitution. However, a major advantage of transducers in speech recognition is that they naturally generalize the notion of context-independent substitution of a label to the context-dependent case. In particular, the application of the familiar triphone models in ASR to the context-independent transducer, producing a context-dependent transducer, can be performed simply with composition.

To do so, we first construct a context-dependency transducer that maps from context-independent phones to context-dependent triphones. This transducer has a state for every pair of phones and a transition for every context-dependent model. In particular, if $ae/k\_t$ represents the triphonic model for $ae$ with left context $k$ and right context $t$, then there is a transition in the context-dependency transducer from state $(k, ae)$ to state $(ae, t)$ with output label $ae/k\_t$. This use of / to indicate *in the context of* in a triphone symbol offers a potential ambiguity with our use of / to separate a transition's weight from its input and output symbols. However, since context-dependency transducers are never weighted in this chapter, the confusion is not a problem in what follows, so we chose to stay with the notation of previous work rather than changing it to avoid the potential ambiguity. For the input label on this transition, we could choose the center phone $ae$ as depicted in Fig. 28.6a. This will correctly implement the transduction; but the transducer will be non-deterministic. Alternately, we can choose the right phone $t$ as depicted in Fig. 28.6b. This will also correctly implement the transduction, but the result will be deterministic. To see why these are correct, realize that when we enter a state, we have read (in the deterministic case) or must read (in the nondeterministic case) the two phones that label the state. Therefore, the source state and destination state of a transition determine the triphone context. In Sect. 28.4, we give the full details of the triphonic

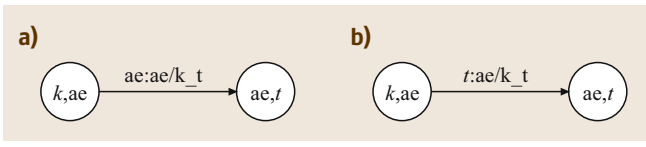



**Fig. 28.6a,b** Context-dependent triphone transducer transition: **(a)** nondeterministic, **(b)** deterministic



Amazon EX1010 Page 583

context-dependency transducer construction and further demonstrate its correctness.

The above context-dependency transducer maps from context-independent phones to context-dependent triphones. We can invert the relation by interchanging the transducer's input and output labels to create a new transducer that maps from context-dependent triphones to context-independent phones. We do this inversion so we can left compose it with our context-independent recognition transducer $L \circ G$. If we let $C$ represent a context-dependency transducer from context-dependent phones to context-independent phones, then:

$$C \circ (L \circ G) \tag{28.5}$$

gives a transducer that maps from context-dependent phones to word strings restricted to the grammar $G$. To complete our example, we optimize this transducer. Given our discussion of the benefits of determinization and minimization, we might try to apply those operations directly to the composed transducer:

$$N = \min(\det(C \circ (L \circ G))) \,. \tag{28.6}$$

This assumes that the recognition transducer can be determinized, which will be true if each of the component transducers can be determinized. If the context dependency $C$ is constructed as we have described and if the grammar $G$ is an $n$-gram language model, then they will be determinizable. However, $L$ may not be determinizable. In particular, if $L$ has ambiguities, namely homophones (two distinct words that have the same pronunciation), then it cannot be determinized as is. However, we can introduce auxiliary phone symbols at word ends to disambiguate homophones to create a transformed lexicon $\tilde{L}$. We also need to create a modified context-dependency transducer $\tilde{C}$ that additionally pairs the context-independent auxiliary symbols found in the lexicon with new context-dependent auxiliary symbols (which are later rewritten as epsilons after all optimizations). We leave the details to Sect. 28.4. The following expression specifies the optimized transducer:

$$N = \min(\det(\tilde{C} \circ (\tilde{L} \circ G))) \,. \tag{28.7}$$

In Sect. 28.4, we give illustrative experimental results with a fully composed, optimized (and *factored*) recognition transducer that maps from context-dependent units to words for the North American Business News (NAB) DARPA task. This transducer runs about 18 times faster than its unoptimized version and has only about 1.4× times as many transitions as its word-level grammar. We have found similar results with DARPA Broadcast News and Switchboard.



## 28.3 Algorithms

We now describe in detail the weighted automata and transducer algorithms introduced informally in Sect. 28.2 that are relevant to the design of speech recognition systems. We start with definitions and notation used in specifying and describing the algorithms.

### 28.3.1 Preliminaries

As noted earlier, all of our algorithms work with weights that are combined with operations satisfying the *semiring* conditions. A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is specified by a set of values $\mathbb{K}$, two binary operations $\oplus$ and $\otimes$, and two designated values $\overline{0}$ and $\overline{1}$. The operation $\oplus$ is associative, commutative, and has $\overline{0}$ as identity. The operation $\otimes$ is associative, has identity $\overline{1}$, distributes with respect to $\oplus$, and has $\overline{0}$ as annihilator: for all $a \in \mathbb{K}$, $a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$. If $\otimes$ is also commutative, we say that the semiring is *commutative*. All the semirings we discuss in the rest of this chapter are commutative.

Real numbers with addition and multiplication satisfy the semiring conditions, but of course they also satisfy several other important conditions (for exam-

**Table 28.1** Semiring examples. $\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(\mathrm{e}^{-x} + \mathrm{e}^{-y})$

| Semiring | Set | $\oplus$ | $\otimes$ | $\overline{0}$ | $\overline{1}$ |
|---|---|---|---|---|---|
| Boolean | $\{0, 1\}$ | $\vee$ | $\wedge$ | 0 | 1 |
| Probability | $\mathbb{R}_+$ | $+$ | $\times$ | 0 | 1 |
| Log | $\mathbb{R} \cup \{-\infty, +\infty\}$ | $\oplus_{\log}$ | $+$ | $+\infty$ | 0 |
| Tropical | $\mathbb{R} \cup \{-\infty, +\infty\}$ | min | $+$ | $+\infty$ | 0 |

ple, having additive inverses), which are not required for our transducer algorithms. Table 28.1 lists some familiar (commutative) semirings. In addition to the Boolean semiring, and the probability semiring used to combine probabilities, two semirings often used in text and speech processing applications are the *log semiring*, which is isomorphic to the probability semiring via the negative-log mapping, and the *tropical semiring*, which is derived from the log semiring using the *Viterbi approximation*.

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is *weakly left-divisible* if for any $x$ and $y$ in $\mathbb{K}$ such that $x \oplus y \neq \overline{0}$, there exists at least one $z$ such that $x = (x \oplus y) \otimes z$. The $\otimes$-operation is *cancellative* if $z$ is unique and we can write: $z = (x \oplus y)^{-1} \otimes x$. A semiring is *zero-sum-free* if, for any $x$ and $y$ in $\mathbb{K}$, $x \oplus y = \overline{0}$ implies $x = y = \overline{0}$.

For example, the tropical semiring is weakly left-divisible with $z = x - \min(x, y)$, which also shows that $\otimes$ for this semiring is cancellative. The probability semiring is also weakly left-divisible with $z = \frac{x}{x+y}$. Finally, the tropical semiring, the probability semiring, and the log semiring are zero sum free.

For any $x \in \mathbb{K}$, let $x^n$ denote

$$x^n = \underbrace{x \otimes \cdots \otimes x}_{n} \,. \tag{28.8}$$

When the infinite sum $\bigoplus_{n=0}^{+\infty} x^n$ is well defined and in $\mathbb{K}$, the closure of an element $x \in \mathbb{K}$ is defined as $x^* = \bigoplus_{n=0}^{+\infty} x^n$. A semiring is *closed* when infinite sums such as the one above, are well defined and if associativity, commutativity, and distributivity apply to countable sums (*Lehmann* [28.20, 22] and *Mohri* give precise definitions). The Boolean and tropical semirings are closed, while the probability and log semirings are not.

A *weighted finite-state transducer* $T = (\mathcal{A}, \mathcal{B}, Q, I, F, E, \lambda, \rho)$ over a semiring $\mathbb{K}$ is specified by a finite input alphabet $\mathcal{A}$, a finite output alphabet $\mathcal{B}$, a finite set of states $Q$, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, a finite set of transitions $E \subseteq Q \times (\mathcal{A} \cup \{\varepsilon\}) \times (\mathcal{B} \cup \{\varepsilon\}) \times \mathbb{K} \times Q$, an initial state weight assignment $\lambda : I \to \mathbb{K}$, and a final state weight assignment $\rho : F \to \mathbb{K}$. $E[q]$ denotes the set of transitions leaving state $q \in Q$. $|T|$ denotes the sum of the number of states and transitions of $T$.

*Weighted automata* (or weighted acceptors) are defined in a similar way by simply omitting the input or output labels. The *projection* operations $\Pi_1(T)$ and $\Pi_2(T)$ obtain a weighted automaton from a weighted transducer $T$ by omitting, respectively, the input or the output labels of $T$.

Given a transition $e \in E$, $p[e]$ denotes its origin or previous state, $n[e]$ its destination or next state, $i[e]$ its input label, $o[e]$ its output label, and $w[e]$ its weight. A *path* $\pi = e_1 \cdots e_k$ is a sequence of consecutive transitions: $n[e_{i-1}] = p[e_i], i = 2, \ldots, k$. The path $\pi$ is a *cycle* if $p[e_1] = n[e_k]$. An *$\varepsilon$-cycle* is a cycle in which the input and output labels of all transitions are $\varepsilon$.

The functions $n$, $p$, and $w$ on transitions can be extended to paths by setting $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$, and by defining the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. More generally, $w$ is extended to any finite set of paths $R$ by setting $w[R] = \bigoplus_{\pi \in R} w[\pi]$; if the semiring is closed, this is defined even for infinite $R$. We denote by $P(q, q')$ the set of paths from $q$ to $q'$ and by $P(q, x, y, q')$ the set of paths from $q$ to $q'$ with input label $x \in \mathcal{A}^*$ and output label $y \in \mathcal{B}^*$. For an acceptor, we denote by $P(q, x, q')$ the set of paths with input label $x$. These definitions can be extended to subsets $R, R' \subseteq Q$ by $P(R, R') = \cup_{q \in R,\, q' \in R'} P(q, q')$, $P(R, x, y, R') = \cup_{q \in R,\, q' \in R'} P(q, x, y, q')$, and, for an acceptor, $P(R, x, R') = \cup_{q \in R,\, q' \in R'} P(q, x, q')$. A transducer $T$ is *regulated* if the weight associated by $T$ to any pair of input–output strings $(x, y)$, given by

$$T(x, y) = \bigoplus_{\pi \in P(I, x, y, F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]] \,, \tag{28.9}$$

is well defined and in $\mathbb{K}$. If $P(I, x, y, F) = \emptyset$, then $T(x, y) = \overline{0}$. A weighted transducer without $\varepsilon$-cycles is regulated, as is any weighted transducer, over a closed semiring. Similarly, for a regulated acceptor, we define

$$T(x) = \bigoplus_{\pi \in P(I, x, F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]] \,. \tag{28.10}$$

The transducer $T$ is *trim* if every state occurs in some path $\pi \in P(I, F)$. In other words, a trim transducer has no useless states. The same definition applies to acceptors.

### 28.3.2 Composition

As we outlined in Sect. 28.2.3, composition is the core operation for relating multiple levels of representation in ASR. More generally, composition is the fundamental algorithm used to create complex weighted transducers from simpler ones [28.12, 14], and generalizes the composition algorithm for unweighted finite-state transducers [28.23, 24]. Let $\mathbb{K}$ be a commutative semiring and

let $T_1$ and $T_2$ be two weighted transducers defined over $\mathbb{K}$ such that the input alphabet $\mathcal{B}$ of $T_2$ coincides with the output alphabet of $T_1$. Assume that the infinite sum $\bigoplus_{z\in\mathcal{B}^*} T_1(x,z)\otimes T_2(z,y)$ is well defined and in $\mathbb{K}$ for all $(x,y)\in\mathcal{A}^*\times\mathcal{C}^*$, where $\mathcal{A}$ is the input alphabet of $T_1$ and $\mathcal{C}$ is the output alphabet of $T_2$. This will be the case if $\mathbb{K}$ is closed, or if $T_1$ has no $\varepsilon$-input cycles or $T_2$ has no $\varepsilon$-output cycles. Then, the result of the composition of $T_1$ and $T_2$ is a weighted transducer denoted by $T_1\circ T_2$ and specified for all $x$, $y$ by:

$$(T_1\circ T_2)(x,y)=\bigoplus_{z\in\mathcal{B}^*} T_1(x,z)\otimes T_2(z,y)\,. \qquad (28.11)$$

There is a general and efficient composition algorithm for weighted transducers [28.12, 14]. States in the composition $T_1\circ T_2$ of two weighted transducers $T_1$ and $T_2$ are identified with pairs of a state of $T_1$ and a state of $T_2$. Leaving aside transitions with $\varepsilon$ inputs or outputs, the following rule specifies how to compute a transition of $T_1\circ T_2$ from appropriate transitions of $T_1$ and $T_2$:

$$(q_1,a,b,w_1,r_1)\text{ and }(q_2,b,c,w_2,r_2)\Longrightarrow((q_1,q_2),a,c,w_1\otimes w_2,(r_1,r_2))\,. \qquad (28.12)$$

Figure 28.7 gives the pseudocode of the algorithm in the $\varepsilon$-free case.

The algorithm takes as input two weighted transducers

$$T_1=(\mathcal{A},\mathcal{B},Q_1,I_1,F_1,E_1,\lambda_1,\rho_1)\quad\text{and}$$
$$T_2=(\mathcal{B},\mathcal{C},Q_2,I_2,F_2,E_2,\lambda_2,\rho_2)\,, \qquad (28.13)$$

and outputs a weighted finite-state transducer $T=(\mathcal{A},\mathcal{C},Q,I,F,E,\lambda,\rho)$ implementing the composition of $T_1$ and $T_2$. $E$, $I$, and $F$ are all initialized to the empty set and grown as needed.

The algorithm uses a queue $S$ containing the set of pairs of states yet to be examined. The queue discipline of $S$ is arbitrary, and does not affect the termination of the algorithm. The state set $Q$ is initially the set of pairs of initial states of the original transducers, as is $S$ (lines 1–2). Each time through the loop in lines 3–16, a new pair of states $(q_1,q_2)$ is extracted from $S$ (lines 4–5). The initial weight of $(q_1,q_2)$ is computed by $\otimes$-multiplying the initial weights of $q_1$ and $q_2$ when they are both initial states (lines 6–8). Similar steps are followed for final states (lines 9–11). Then, for each pair of matching transitions $(e_1,e_2)$, a new transition is created according to the rule specified earlier (line 16). If the destination state $(n[e_1],n[e_2])$ has not been found previously, it is added to $Q$ and inserted in $S$ (lines 14–15).

```
Weighted-composition (T1, T2)
 1  Q ← I1 × I2
 2  S ← I1 × I2
 3  while S ≠ ∅ do
 4       (q1, q2) ← Head (S)
 5       Dequeue (S)
 6       if (q1, q2) ∈ I1 × I2 then
 7            I ← I ∪ {(q1, q2)}
 8            λ (q1, q2) ← λ1 (q1) ⊗ λ2 (q2)
 9       if (q1, q2) ∈ F1 × F2 then
10            F ← F ∪ {(q1, q2)}
11            ρ (q1, q2) ← ρ1 (q1) ⊗ ρ2 (q2)
12       for each (e1, e2) ∈ E [q1] × E [q2] such that o [e1] = i [e2] do
13            if (n [e1], n [e2]) ∉ Q then
14                 Q ← Q ∪ {(n[e1], n[e2])}
15                 Enqueue (S, (n[e1], n[e2]))
16            E ← E ∪ {((q1, q2),), i[e1], o[e2], w[e1] ⊗ w[e2], (n[e1], n[e2]))}
17  return T
```

**Fig. 28.7** Pseudocode for the composition algorithm

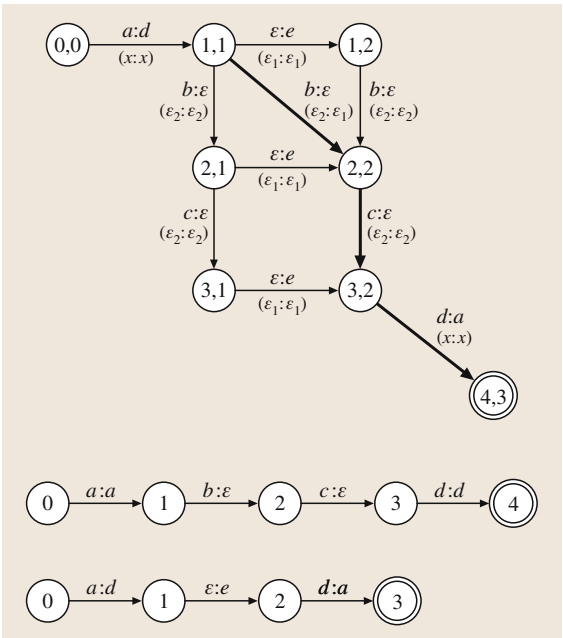



**Fig. 28.8** Redundant $\varepsilon$ paths. A straightforward generalization of the $\varepsilon$-free case could generate all the paths from (1, 1) to (3, 2) when composing the two simple transducers on the right-hand side

In the worst case, all transitions of $T_1$ leaving a state $q_1$ match all those of $T_2$ leaving state $q_1'$, thus the space and time complexity of composition is quadratic: $O(|T_1||T_2|)$. However, a lazy implementation of composition can be used to construct just the part of the composed transducer that is needed.



Amazon EX1010 Page 586

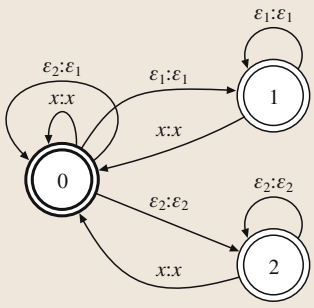



**Fig. 28.9** Filter for composition $F$



More care is needed when $T_1$ has output $\varepsilon$ labels and $T_2$ input $\varepsilon$ labels. Indeed, as illustrated by Fig. 28.8, a straightforward generalization of the $\varepsilon$-free case would generate redundant $\varepsilon$-paths and, in the case of non-idempotent semirings, would lead to an incorrect result. The weight of the matching paths of the original transducers would be counted $p$ times, where $p$ is the number of redundant paths in the composition.

To solve this problem, all but one $\varepsilon$-path must be filtered out of the composition. Figure 28.8 indicates in thick lines one possible choice for that path, which in this case is the shortest. Remarkably, that filtering mechanism can be encoded as a finite-state transducer.

Let $\tilde{T}_1$ and $\tilde{T}_2$ be the weighted transducers obtained from $T_1$ and $T_2$, respectively, by replacing the output $\varepsilon$ labels of $T_1$ with $\varepsilon_2$ and the input $\varepsilon$ labels of $T_2$ with $\varepsilon_1$. Consider the filter finite-state transducer $F$ represented in Fig. 28.9. Then $\tilde{T}_1 \circ F \circ \tilde{T}_2 = T_1 \circ T_2$. Since the two compositions in $\tilde{T}_1 \circ F \circ \tilde{T}_2$ do not involve $\varepsilon$ labels, the $\varepsilon$-free composition already described can be used to compute the resulting transducer.

Intersection (or the *Hadamard product*) of weighted automata and composition of finite-state transducers are both special cases of composition of weighted transducers. Intersection corresponds to the case where the input and output labels of transitions are identical and composition of unweighted transducers is obtained simply by omitting the weights.

```
Weighted-determinization (A)
 1  i' ← {(i, λ(i)) : i ∈ I}
 2  λ'(i') ← 1̄
 3  S ← {i'}
 4  while S ≠ ∅ do
 5      p' ← Head (S)
 6      Dequeue (S)
 7      for each x ∈ i[E[Q[p']]] do
 8          w' ← ⊕ {v ⊗ w : (p, v) ∈ p', (p, x, w, q) ∈ E}
 9          q' ← {(q, ⊕ {(w'^{-1} ⊗ (v ⊗ w) : (p, v) ∈ p', (p, x, w, q) ∈ E}) :
                q = n[e], i[e] = x, e ∈ E[Q[p']]}
10          E' ← E' ∪ {(p', x, w', q')}
11          if q' ∉ Q then
12              Q ← Q' ∪ {q'}
13              if Q[q'] ∩ F ≠ ∅ then
14                  F' ← F' ∪ {q'}
15                  ρ'(q') ← ⊕ {v ⊗ ρ(q) : (q, v) ∈ q', q ∈ F}
16              Enqueue (S, q')
17  return A'
```

**Fig. 28.10** Pseudocode for the weighted determinization algorithm [28.6]

## 28.3.3 Determinization

We now describe the generic determinization algorithm for weighted automata that we used informally when working through the example in Sect. 28.2.4. This algorithm is a generalization of the classical subset construction for unweighted nondeterministic finite automata (NFAs). The determinization of unweighted or weighted finite-state transducers can both be viewed as special instances of the generic algorithm presented here but, for simplicity, we will focus on the weighted acceptor case.

A weighted automaton is *deterministic* (also called *subsequential*) if it has a unique initial state and if no two transitions leaving any state share the same input label. The *determinization* algorithm we now present applies to weighted automata over a cancellative weakly left-divisible semiring that satisfies a mild technical condition. If $x \in P(I, Q)$, then $w[P(I, x, Q)] \neq \overline{0}$, which is satisfied by trim automata over the tropical semiring or any other zero-sum-free semiring. Figure 28.10 gives pseudocode for the algorithm.

A *weighted subset* $p$ of $Q$ is a set of pairs $(q, x) \in Q \times \mathbb{K}$. $Q[p]$ is the set of states $q$ in $p$, $E[Q[p]]$ is the set of transitions leaving those states, and $i[E[Q[p]]]$ is the set of input labels of those transitions.

The states of the result automaton are weighted subsets of the states of the original automaton. A state $r$ of the result automaton that can be reached from the start state by path $\pi$ is the weighted set of pairs $(q, x) \in Q \times \mathbb{K}$ such that $q$ can be reached from an initial state of the original automaton by a path $\sigma$ with $i[\sigma] = i[\pi]$ and $\lambda[p[\sigma]] \otimes w[\sigma] = \lambda[p[\pi]] \otimes w[\pi] \otimes x$. Thus, $x$ can be viewed as the *residual* weight at state $q$. The algorithm takes as input a weighted automaton $A = (\mathcal{A}, Q, I, F, E, \lambda, \rho)$ and, when it terminates, yields an equivalent deterministic weighted automaton $A' = (\mathcal{A}, Q', I', F', E', \lambda', \rho')$.

The algorithm uses a queue $S$ containing the set of states of the resulting automaton $A'$, yet to be examined.

Amazon EX1010 Page 587

The sets $Q'$, $I'$, $F'$, and $E'$ are initially empty. The queue discipline for $S$ can be chosen arbitrarily and does not affect the termination of the algorithm. The initial state set of $A'$ is $I' = \{i'\}$ where $i'$ is the weighted set of the initial states of $A$ with the respective initial weights. Its initial weight is $\overline{1}$ (lines 1–2). $S$ originally contains only the subset $I'$ (line 3). Each time through the loop in lines 4–16 a new weighted subset $p'$ is dequeued from $S$ (lines 5–6). For each $x$ labeling at least one of the transitions leaving a state $p$ in the weighted subset $p'$, a new transition with input label $x$ is constructed. The weight $w'$ associated to that transition is the sum of the weights of all transitions in $E[Q[p']]$ labeled with $x$ pre-$\otimes$-multiplied by the residual weight $v$ at each state $p$ (line 8). The destination state of the transition is the subset containing all the states $q$ reached by transitions in $E[Q[p']]$ labeled with $x$. The weight of each state $q$ of the subset is obtained by taking the $\oplus$-sum of the residual weights of the states $p$ $\otimes$-times the weight of the transition from $p$ leading to $q$ and by *dividing* that by $w'$. The new subset $q'$ is inserted into the queue $S$ when it is a new state (line 16). If any of the states in the subset $q'$ is final, $q'$ is made a final state and its final weight is obtained by summing the final weights of all the final states in $q'$, pre-$\otimes$-multiplied by their residual weight $v$ (line 14–15).

The worst-case complexity of determinization is exponential even in the unweighted case. However, in many practical cases such as for weighted automata used in large-vocabulary speech recognition, this blow-up does not occur. It is also important to notice that, just like composition, determinization has a natural lazy implementation in which only the transitions required by an application are expanded in the result automaton.

Unlike in the unweighted case, determinization does not halt on all input weighted automata. We say that a weighted automaton $A$ is *determinizable* if the determinization algorithm halts for the input $A$. With a determinizable input, the algorithm outputs an equivalent deterministic weighted automaton.

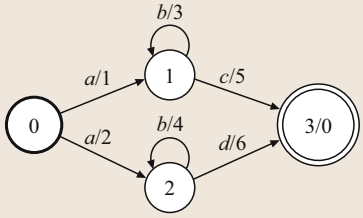



**Fig. 28.11** Nondeterminizable weighted automaton over the tropical semiring. States 1 and 2 are non-twin siblings

The *twins property* for weighted automata characterizes determinizable weighted automata under some general conditions [28.6]. Let $A$ be a weighted automaton over a weakly left-divisible semiring $\mathbb{K}$. Two states $q$ and $q'$ of $A$ are said to be *siblings* if there are strings $x$ and $y$ in $\mathcal{A}^*$ such that both $q$ and $q'$ can be reached from $I$ by paths labeled with $x$ and there is a cycle at $q$ and a cycle at $q'$ both labeled with $y$. When $\mathbb{K}$ is a commutative and cancellative semiring, two sibling states are said to be *twins* when, for every string $y$,

$$w[P(q, y, q)] = w[P(q', y, q')] \,. \tag{28.14}$$

$A$ has *the twins property* if any two sibling states of $A$ are twins. Figure 28.11 shows a weighted automaton over the tropical semiring that does not have the twins property: states 1 and 2 can be reached by paths labeled with $a$ from the initial state and have cycles with the same label $b$, but the weights of these cycles (3 and 4) are different.

The following theorems relate the twins property and weighted determinization [28.6].

*Theorem 28.1*
Let $A$ be a weighted automaton over the tropical semiring. If $A$ has the twins property, then $A$ is determinizable.

A weighted automaton is said to be *unambiguous* if, for any $x \in \Sigma^*$, it admits at most one accepting path labeled with $x$.

*Theorem 28.2*
Let $A$ be a trim unambiguous weighted automaton over the tropical semiring. Then $A$ is determinizable iff it has the twins property.

There is an efficient algorithm for testing the twins property for weighted automata [28.25]. Note that any acyclic weighted automaton over a zero-sum-free semiring has the twins property and is determinizable.

The *pre-determinization* algorithm can be used to make an arbitrary weighted transducer determinizable over the tropical semiring by inserting transitions labeled with special symbols [28.26]. The algorithm makes use of a general twins property [28.25] to insert new transitions when needed to guarantee that the resulting transducer has the twins property and thus is determinizable.



Amazon EX1010 Page 588

### 28.3.4 Weight Pushing

As discussed in Sect. 28.2.5, *weight pushing* is necessary in weighted minimization, and is also very useful to improve search. Weight pushing can also be used to test the equivalence of two weighted automata. Weight pushing is possible because the choice of the distribution of the total weight along each successful path of a weighted automaton does not affect the total weight of each successful path, and therefore preserves the definition of the automaton as a weighted set (weighted relation for a transducer).

Let $A$ be a weighted automaton over a zero-sum-free and weakly left-divisible semiring $\mathbb{K}$. For any state $q \in Q$, assume that the following sum is well defined and in $\mathbb{K}$:

$$d[q] = \bigoplus_{\pi \in P(q,F)} (w[\pi] \otimes \rho[n[\pi]]) \,. \tag{28.15}$$

The value $d[q]$ is the *shortest distance* from $q$ to $F$ [28.20]. This is well defined for all $q \in Q$ when $\mathbb{K}$ is a closed semiring. The weight-pushing algorithm consists of computing each shortest-distance $d[q]$ and of *reweighting* the transition weights, initial weights and final weights in the following way:

$$\begin{aligned} w[e] &\leftarrow d[p[e]]^{-1} \otimes w[e] \otimes d[n[e]] \text{ if } d[p[e]] \neq \overline{0} \\ \lambda[i] &\leftarrow \lambda[i] \otimes d[i] \,, \\ \rho[f] &\leftarrow d[f]^{-1} \otimes \rho[f] \text{ if } d[f] \neq \overline{0} \,, \end{aligned} \tag{28.16}$$

where $e$ is any transition, $i$ any initial state, and $f$ any final state. Each of these operations can be done in constant time. Therefore, reweighting can be done in linear time $O(T_{\otimes}|A|)$ where $T_{\otimes}$ denotes the worst cost of an $\otimes$-operation. The complexity of the shortest-distances computation depends on the semiring [28.20]. For the tropical semiring, $d[q]$ can be computed using a standard shortest-distance algorithm. The complexity is linear for acyclic automata, $O[|Q| + (T_{\oplus} + T_{\otimes})|E|]$, where $T_{\oplus}$ denotes the worst cost of an $\oplus$-operation. For general weighted automata over the tropical semiring, the complexity is $O(|E| + |Q| \log |Q|)$.

For semirings like the probability semiring, a generalization of the Floyd–Warshall algorithm for computing all-pairs shortest distances can be used [28.20]. Its complexity is $\Theta(|Q|^3(T_{\oplus} + T_{\otimes} + T_*))$ where $T_*$ denotes the worst cost of the closure operation. The space complexity of these algorithms is $\Theta(|Q|^2)$. Therefore, the Floyd–Warshall algorithm is impractical for automata with several hundred million states or transitions, which arise in large-vocabulary ASR. An approximate version of a generic single-source shortest-distance algorithm can be used instead to compute $d[q]$ efficiently [28.20].

Speaking informally, the algorithm pushes the weight on each path as much as possible towards the initial states. Figure 28.5a,b show weight pushing on the tropical semiring, while Fig. 28.12 shows weight pushing for the same automaton but on the probability semiring.

Note that if $d[q] = \overline{0}$, then, since $\mathbb{K}$ is zero-sum-free, the weight of all paths from $q$ to $F$ is $\overline{0}$. Let $A$ be a weighted automaton over the semiring $\mathbb{K}$. Assume that $\mathbb{K}$ is closed and that the shortest distances $d[q]$ are all well defined and in $\mathbb{K} - \{\overline{0}\}$. In both cases, we can use the distributivity over the infinite sums defining shortest distances. Let $e'$ ($\pi'$) denote the transition $e$ (path $\pi$) after application of the weight-pushing algorithm. $e'$ ($\pi'$) differs from $e$ (resp. $\pi$) only by its weight. Let $\lambda'$ denote the new initial weight function, and $\rho'$ the new final weight function. Then, the following proposition holds [28.6, 27].

*Proposition 28.1*

Let $B = (\mathcal{A}, Q, I, F, E', \lambda', \rho')$ be the result of the weight-pushing algorithm applied to the weighted automaton $A$, then

1. the weight of a successful path $\pi$ is unchanged after weight pushing:

$$\lambda'[p[\pi']] \otimes w[\pi'] \otimes \rho'[n[\pi']] = \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]] \,. \tag{28.17}$$

2. the weighted automaton $B$ is *stochastic*, that is,

$$\forall q \in Q, \bigoplus_{e' \in E'[q]} w[e'] = \overline{1} \,. \tag{28.18}$$

These two properties of weight pushing are illustrated by Figs. 28.5a,b and 28.12: the total weight of a successful path is unchanged after pushing; at each

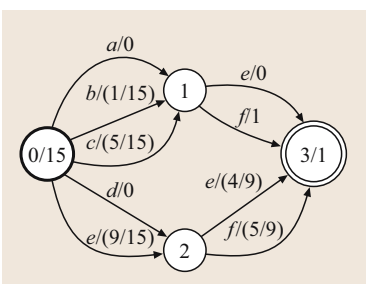



**Fig. 28.12** Weighted automaton $C$ obtained from $A$ of Fig. 28.5a by weight pushing in the probability semiring

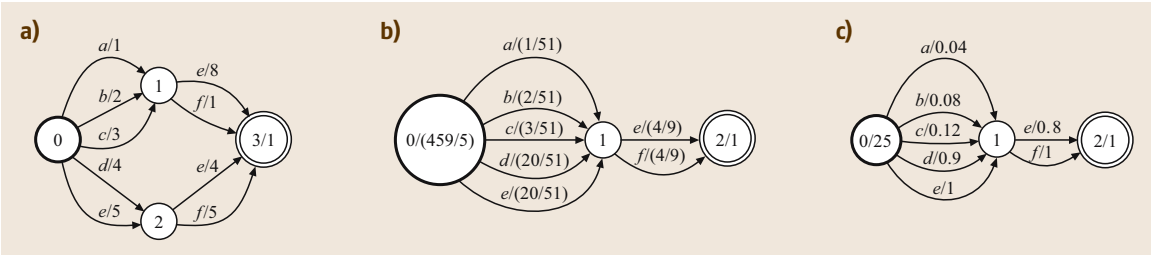



**Fig. 28.13a–c** Minimization of weighted automata. **(a)** Weighted automaton $A'$ over the probability semiring. **(b)** Minimal weighted automaton $B'$ equivalent to $A'$. **(c)** Minimal weighted automaton $C'$ equivalent to $A'$

state of the weighted automaton of Fig. 28.5b, the minimum weight of the outgoing transitions is 0, and at each state of the weighted automaton of Fig. 28.12, the weights of outgoing transitions sum to 1.

### 28.3.5 Minimization

Finally, we discuss in more detail the minimization algorithm introduced in Sect. 28.2.5. A deterministic weighted automaton is said to be *minimal* if there is no other deterministic weighted automaton with a smaller number of states that represents the same mapping from strings to weights. It can be shown that the minimal deterministic weighted automaton also has the minimal number of transitions among all equivalent deterministic weighted automata [28.6].

Two states of a deterministic weighted automaton are said to be *equivalent* if exactly the same set of strings label the paths from these states to a final state, and the total weight of the paths for each string, including the final weight of the final state, is the same. Thus, two equivalent states of a deterministic weighted automaton can be merged without affecting the function realized by that automaton. A weighted automaton is minimal when it is not possible to create two distinct equivalent states after any pushing of the weights along its paths.

As outlined in Sect. 28.2.5, the general minimization algorithm for weighted automata consists of first applying the weight-pushing algorithm to normalize the distribution of the weights along the paths of the input automaton, and then of treating each pair (label, weight) as a single label and applying classical (unweighted) automata minimization [28.6]. The minimization of both unweighted and weighted finite-state transducers can also be viewed as instances of the algorithm presented here, but, for simplicity, we will not discuss that further here. The following theorem holds [28.6].

*Theorem 28.3*

Let $A$ be a deterministic weighted automaton over a semiring $\mathbb{K}$. Assume that the conditions of application of the weight pushing algorithm hold. Then the execution of the following steps:

1. weight pushing,
2. (unweighted) automata minimization,

lead to a minimal weighted automaton equivalent to $A$.

The complexity of automata minimization is linear in the case of acyclic automata $O(|Q|+|E|)$ and is $O(|E|\log|Q|)$ in the general case. In view of the complexity results of the previous section, for the tropical semiring, the time complexity of the weighted minimization algorithm is linear $O(|Q|+|E|)$ in the acyclic case and $O(|E|\log|Q|)$ in the general case.

Figure 28.5 illustrates the algorithm in the tropical semiring. Automaton $A$ cannot be further minimized using the classical unweighted automata minimization since no two states are equivalent in that machine. After weight pushing, automaton $B$ has two states, 1 and 2, which can be merged by unweighted automaton minimization.

Figure 28.13 illustrates the minimization of an automaton defined over the probability semiring. Unlike the unweighted case, a minimal weighted automaton is not unique, but all minimal weighted automata have the same graph topology, and only differ in the weight distribution along each path. The weighted automata $B'$ and $C'$ are both minimal and equivalent to $A'$. $B'$ is obtained from $A'$ using the algorithm described above in the probability semiring and it is thus a stochastic weighted automaton in the probability semiring.

For a deterministic weighted automaton, the $\oplus$ operation can be arbitrarily chosen without affecting the



Amazon EX1010 Page 590

mapping from strings to weights defined by the automaton, because a deterministic weighted automaton has at most one path labeled by any given string. Thus, in the algorithm described in theorem 28.3, the weight pushing step can be executed in any semiring $\mathbb{K}'$ whose multiplicative operation matches that of $\mathbb{K}$. The minimal weighted automata obtained by pushing the weights in $\mathbb{K}'$ is also minimal in $\mathbb{K}$, since it can be interpreted as a (deterministic) weighted automaton over $\mathbb{K}$.

In particular, $A'$ can be interpreted as a weighted automaton over the semiring $(\mathbb{R}_+, \max, \times, 0, 1)$. The application of the weighted minimization algorithm to $A'$ in this semiring leads to the minimal weighted automaton $C'$ of Fig. 28.13c. $C'$ is also a *stochastic* weighted automaton in the sense that, at any state, the maximum weight of all outgoing transitions is one.

In the particular case of a weighted automaton over the probability semiring, it may be preferable to use weight pushing in the $(\max, \times)$-semiring since the complexity of the algorithm is then equivalent to that of classical single-source shortest-paths algorithms. The corresponding algorithm is a special instance of a generic shortest-distance algorithm [28.20]. Using the $(\max \times)$-semiring assumes the resulting distribution of weights along paths is not important. As discussed in the next section, the weight distribution that results from pushing in the $(+, \times)$ semiring has advantages when the resulting automaton is used in a pruned search.



## 28.4 Applications to Speech Recognition

We now describe the details of the application of weighted finite-state transducer representations and algorithms to speech recognition as introduced in Sect. 28.2.

### 28.4.1 Speech Recognition Transducers

As described in Sect. 28.2, we will represent various models in speech recognition as weighted finite-state transducers. Four principal models are the word-level grammar $G$, the pronunciation lexicon $L$, the context-dependency transducer $C$, and the HMM transducer $H$. We will now discuss the construction of each these transducers. Since these will be combined by composition and optimized by determinization, we ensure they are efficient to compose and allow weighted determinization.

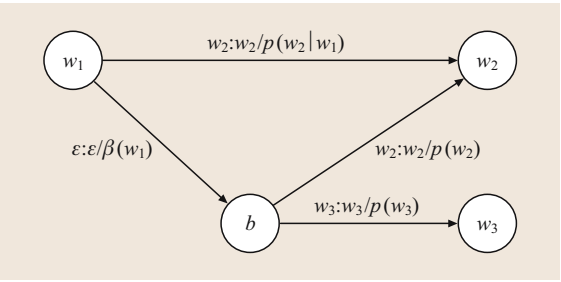



**Fig. 28.14** Word bigram transducer model: seen bigram $w_1w_2$ represented as a $w_2$-transition from the state $w_1$ to state $w_2$; unseen bigram $w_1w_3$ represented as an $\epsilon$-transition from state $w_1$ to backoff state $b$ and as a $w_3$ transition from state $b$ to state $w_3$

The word-level grammar $G$, whether handcrafted or learnt from data, is typically a finite-state model in speech recognition. Handcrafted finite-state models can be specified by regular expressions, rules or directly as automata. Stochastic $n$-gram models, common in large-vocabulary speech recognition, can be represented compactly by finite-state models. For example, a bigram grammar has a state for every word $w_i$ and a transition from state $w_1$ to state $w_2$ for every bigram $w_1w_2$ that is seen in the training corpus. The transition is labeled with $w_2$ and has weight $-\log(\hat{p}(w_2|w_1))$, the negative log of the estimated transition probability. The weight of a bigram $w_1w_3$ that is not seen in the training data can be estimated, for example, by backing-off to the unigram. That is, it has weight $-\log(\beta(w_1)\,\hat{p}(w_3))$, where $\hat{p}(w_3)$ is the estimated $w_3$ unigram probability and $\beta(w_1)$ is the $w_1$ back-off weight [28.28]. The unseen bigram could be represented as a transition from state $w_1$ to $w_3$ in the bigram automaton just as a seen bigram. However, this would result in $O(|V|^2)$ transitions in the automaton, where $|V|$ is the vocabulary size. A simple approximation, with the introduction of a *back-off* state $b$, avoids this. In this model, an unseen $w_1w_3$ bigram is represented as two transitions: an $\varepsilon$-transition from state $w_1$ to state $b$ with weight $-\log(\beta(w_1))$ and a transition from state $b$ to state $w_3$ with label $w_3$ and weight $-\log(\hat{p}(w_3))$. This configuration is depicted in Fig. 28.14. This is an approximation since seen bigrams may also be read as backed-off unigrams. However, since the seen bigram typically has higher probability than its backed-off unigram, it is usually a good approx-

Amazon EX1010 Page 591

imation. A similar construction is used for higher-order $n$-grams.

These grammars present no particular issues for composition. However, the back-off $\varepsilon$-transitions introduce nondeterminism in the $n$-gram model. If fully determinized without $\varepsilon$-transitions, $O(|V|^2)$ transitions would result. However, we can treat the back-off $\varepsilon$ labels as regular symbols during determinization, avoiding the explosion in the number of transitions.

As described in Sect. 28.2, we represent the pronunciation lexicon $L$ as the Kleene closure of the union of individual word pronunciations, as in Fig. 28.2b. In order for this transducer to efficiently compose with $G$, the output (word) labels must be placed on the initial transitions of the words; other locations would lead to delays in the composition matching, which could consume significant time and space.

In general, transducer $L$ is not determinizable. This is clear in the presence of homophones. But, even without homophones, it may not be determinizable because the first word of the output string might not be known before the entire phone string is scanned. Such unbounded output delays make $L$ nondeterminizable.

To make it possible to determinize $L$, we introduce an auxiliary phone symbol, denoted $\#_0$, marking the end of the phonetic transcription of each word. Other auxiliary symbols $\#_1 \ldots \#_{k-1}$ are used when necessary to distinguish homophones, as in the following example:

`r eh d` $\#_0$ *read*

`r eh d` $\#_1$ *red* .

At most $P$ auxiliary phones are needed, where $P$ is the maximum degree of homophony. The pronunciation dictionary transducer with these auxiliary symbols added



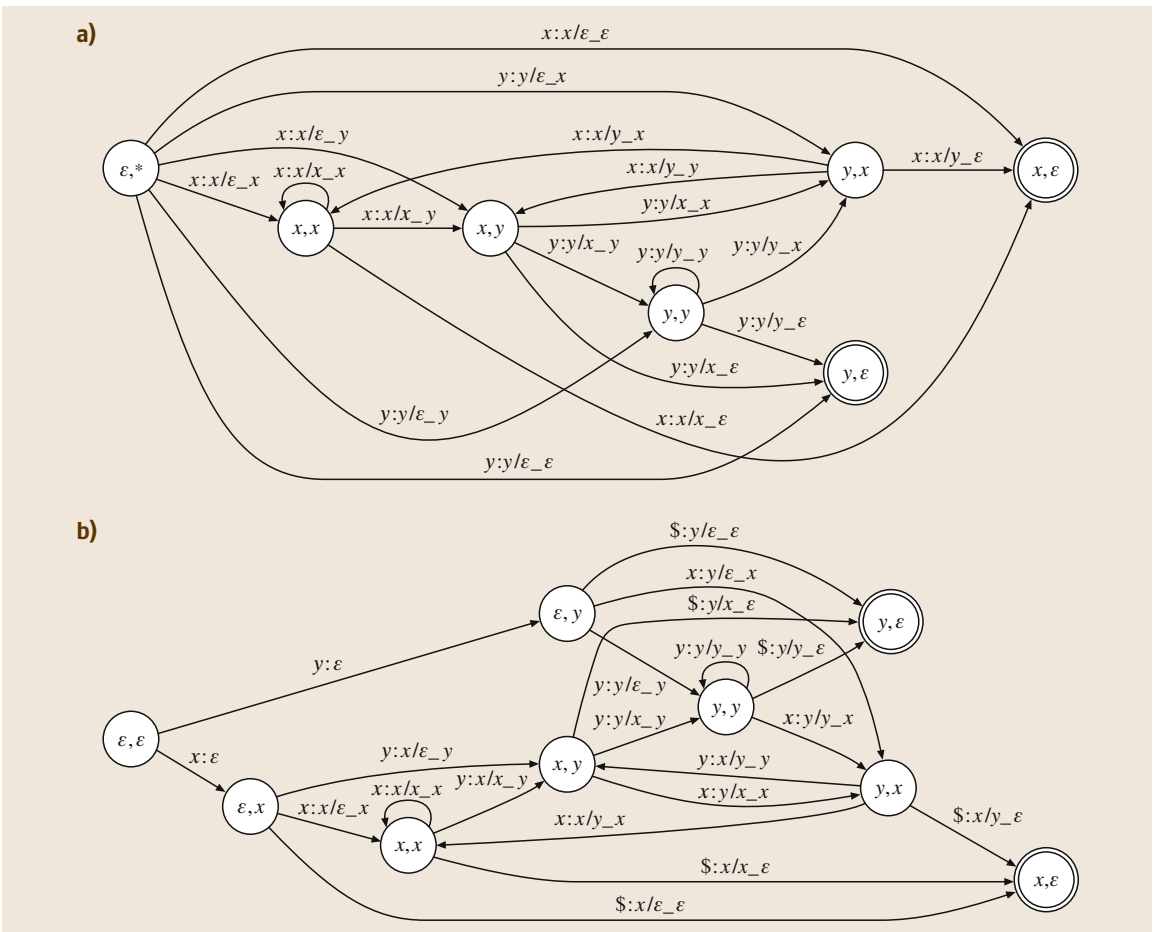



**Fig. 28.15a,b** Context-dependent triphone transducers: **(a)** nondeterministic, **(b)** deterministic

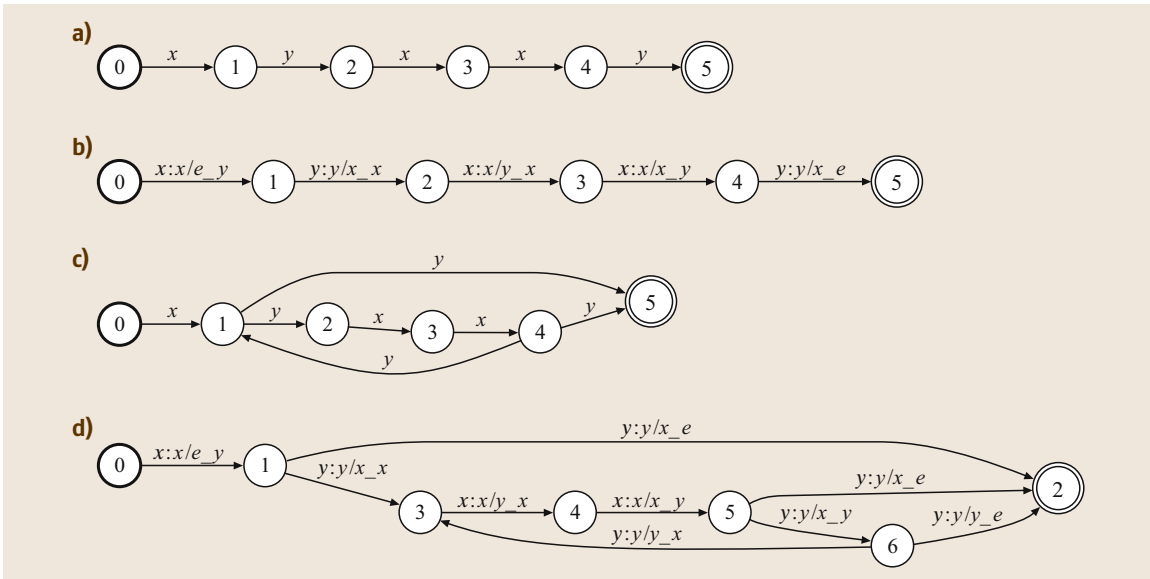



**Fig. 28.16a–d** Context-dependent composition examples: **(a)** context-independent 'string', **(b)** context dependency applied to 1, **(c)** context-independent automaton, **(d)** context dependency applied to **(c)**

is denoted by $\tilde{L}$. *Allauzen* et al. [28.29] describe more-general alternatives to the direct construction of $\tilde{L}$. In that work, as long as $L$ correctly defines the pronunciation transduction, it can be transformed algorithmically to something quite similar to $\tilde{L}$, regardless of the initial disposition of the output labels or the presence of homophony.

As introduced in Sect. 28.2, we can represent the mapping from context-independent phones to context-dependent units with a finite-state transducer, with Fig. 28.6 giving a transition of that transducer. Figure 28.15 gives complete context-dependency transducers where just two hypothetical phones $x$ and $y$ are shown for simplicity. The transducer in Fig. 28.15a is nondeterministic, while the one in Fig. 28.15b is deterministic. For illustration purposes, we will describe the nondeterministic version since it is somewhat simpler. As in Sect. 28.2, we denote the context-dependent units as *phone/left context_right context*. Each state in Fig. 28.15a encodes the knowledge of the previous and next phones. State labels in the figure are pairs $(a, b)$ of the past $a$ and the future $b$, with $\varepsilon$ representing the start or end of a phone string and $*$ an unspecified future. For instance, it is easy to see that the phone string $xyx$ is mapped by the transducer to $x/\varepsilon\_y\ y/x\_x\ x/y\_\varepsilon$ via the unique state sequence $(\varepsilon, *)(x, y)(y, x)(x, \varepsilon)$. More generally, when there are $n$ context-independent phones, this triphonic construction gives a transducer with $O(n^2)$ states and $O(n^3)$ transitions. A tetraphonic construction would give a transducer with $O(n^3)$ states and $O(n^4)$ transitions.

The following simple example shows the use of this context-dependency transducer. A context-independent string can be represented by the obvious single-path acceptor as in Fig. 28.16a. This can then be composed with the context-dependency transducer in Fig. 28.15. Before composition, we promote the acceptor in Fig. 28.16a to the corresponding transducer with identical input and output labels. The result is the transducer in Fig. 28.16b, which has a single path labeled with the context-independent labels on the input side and the corresponding context-dependent labels on the output side.

The context-dependency transducer can be composed with more-complex transducers than the trivial one in Fig. 28.16a. For example, composing the context-dependency transducer with the transducer in Fig. 28.16c results in the transducer in Fig. 28.16d. By definition of relational composition, this must correctly replace the context-independent units with the appropriate context-dependent units on all of its paths. Therefore, composition provides a convenient and general mechanism for applying context dependency to ASR transducers.

The nondeterminism of the transducer in Fig. 28.15a introduces a single symbol-matching delay in the com-



Amazon EX1010 Page 593

position with the lexicon. The deterministic transducer in Fig. 28.15b composes without a matching delay, which makes it the better choice in applications. However, it introduces a single-phone shift between a context-independent phone and its corresponding context-dependent unit in the result. This shift requires the introduction of a final *subsequential* symbol \$ to pad out the context. In practice, this might be mapped to a silence phone or an $\varepsilon$-transition.

If we let $C$ represent a context-dependency transducer from context-dependent phones to context-independent phones, then

$$C \circ L \circ G$$

gives a transducer that maps from context-dependent phones to word strings restricted to the grammar $G$. Note that $C$ is the inverse of a transducer such as in Fig. 28.15; that is, the input and output labels have been exchanged on all transitions. For notational convenience, we adopt this form of the context-dependency transducer when we use it in recognition cascades.

For correctness, the context-dependency transducer $C$ must also accept all paths containing the auxiliary symbols added to $\tilde{L}$ to make it determinizable. For determinizations at the context-dependent phone level and distribution level, each auxiliary phone must be mapped to a distinct context-dependent-level symbol. Thus, self-loops are added at each state of $C$ to map each auxiliary phone to a new auxiliary context-dependent phone. The augmented context-dependency transducer is denoted by $\tilde{C}$.

As we did for the pronunciation lexicon, we can represent the HMM set as $H$, the closure of the union of the individual HMMs (see Fig. 28.16c). Note that we do not explicitly represent the HMM-state self-loops in $H$. Instead, we simulate those in the run-time decoder. With $H$ in hand,

$$H \circ C \circ L \circ G$$

gives a transducer that maps from distributions to word strings restricted to $G$.

Each auxiliary context-dependent phone in $\tilde{C}$ must be mapped to a new distinct distribution name. Self-loops are added at the initial state of $H$ with auxiliary distribution name input labels and auxiliary context-dependent phone output labels to allow for this mapping. The modified HMM model is denoted by $\tilde{H}$.

We thus can use composition to combine all levels of our ASR transducers into an integrated transducer in a convenient, efficient, and general manner. When these automata are statically provided, we can apply the optimizations discussed in the next section to reduce decoding time and space requirements. If the transducer needs to be modified dynamically, for example by adding the results of a database lookup to the lexicon and grammar in an extended dialogue, we adopt a hybrid approach that optimizes the fixed parts of the transducer and uses lazy composition to combine them with the dynamic portions during recognition [28.30, 31].

### 28.4.2 Transducer Standardization

To optimize an integrated transducer, we use three additional steps: (a) determinization, (b) minimization, and (c) factoring.

#### Determinization

We use weighted transducer determinization at each step of the composition of each pair of transducers. The main purpose of determinization is to eliminate redundant paths in the composed transducer, thereby substantially reducing recognition time. In addition, its use in intermediate steps of the construction also helps to improve the efficiency of composition and to reduce transducer size.

First, $\tilde{L}$ is composed with $G$ and determinized, yielding $\det(\tilde{L} \circ G)$. The benefit of this determinization is the reduction of the number of alternative transitions at each state to at most the number of distinct phones at that state, while the original transducer may have as many as $V$ outgoing transitions at some states, where $V$ is the vocabulary size. For large tasks in which the vocabulary has $10^5$ to $10^6$ words, the advantages of this optimization are clear.

$\tilde{C}$ is then composed with the resulting transducer and determinized. Similarly, $\tilde{H}$ is composed with the context-dependent transducer and determinized. This last determinization increases sharing among HMM models that start with the same distributions. At each state of the resulting integrated transducer, there is at most one outgoing transition labeled with any given distribution name, reducing the recognition time even more.

In a final step, we use the erasing operation $\pi_\varepsilon$ that replace the auxiliary distribution symbols by $\varepsilon$'s. The complete sequence of operations is summarized by the following construction formula:

$$N = \pi_\varepsilon(\det(\tilde{H} \circ \det(\tilde{C} \circ \det(\tilde{L} \circ G)))) \,, \qquad (28.19)$$

where parentheses indicate the order in which the operations are performed. The result $N$ is an integrated recognition transducer that can be constructed even in



Amazon EX1010 Page 594

very large-vocabulary tasks and leads to a substantial reduction in recognition time, as the experimental results below show.

### Minimization

Once we have determinized the integrated transducer, we can reduce it further by minimization. The auxiliary symbols are left in place, the minimization algorithm is applied, and then the auxiliary symbols are removed:

$$N = \pi_\varepsilon(\min(\det(\tilde{H} \circ \det(\tilde{C} \circ \det(\tilde{L} \circ G))))) \,. \quad (28.20)$$

Weighted minimization can be used in different semirings. Both minimization in the tropical semiring and minimization in the log semiring can be used in this context. It is not hard to prove that the results of these two minimizations have exactly the same number of states and transitions and only differ in how weight is distributed along paths. The difference in weights arises from differences in the definition of the weight-pushing operation for different semirings.

Weight pushing in the log semiring has a very large, beneficial impact on the pruning efficacy of a standard



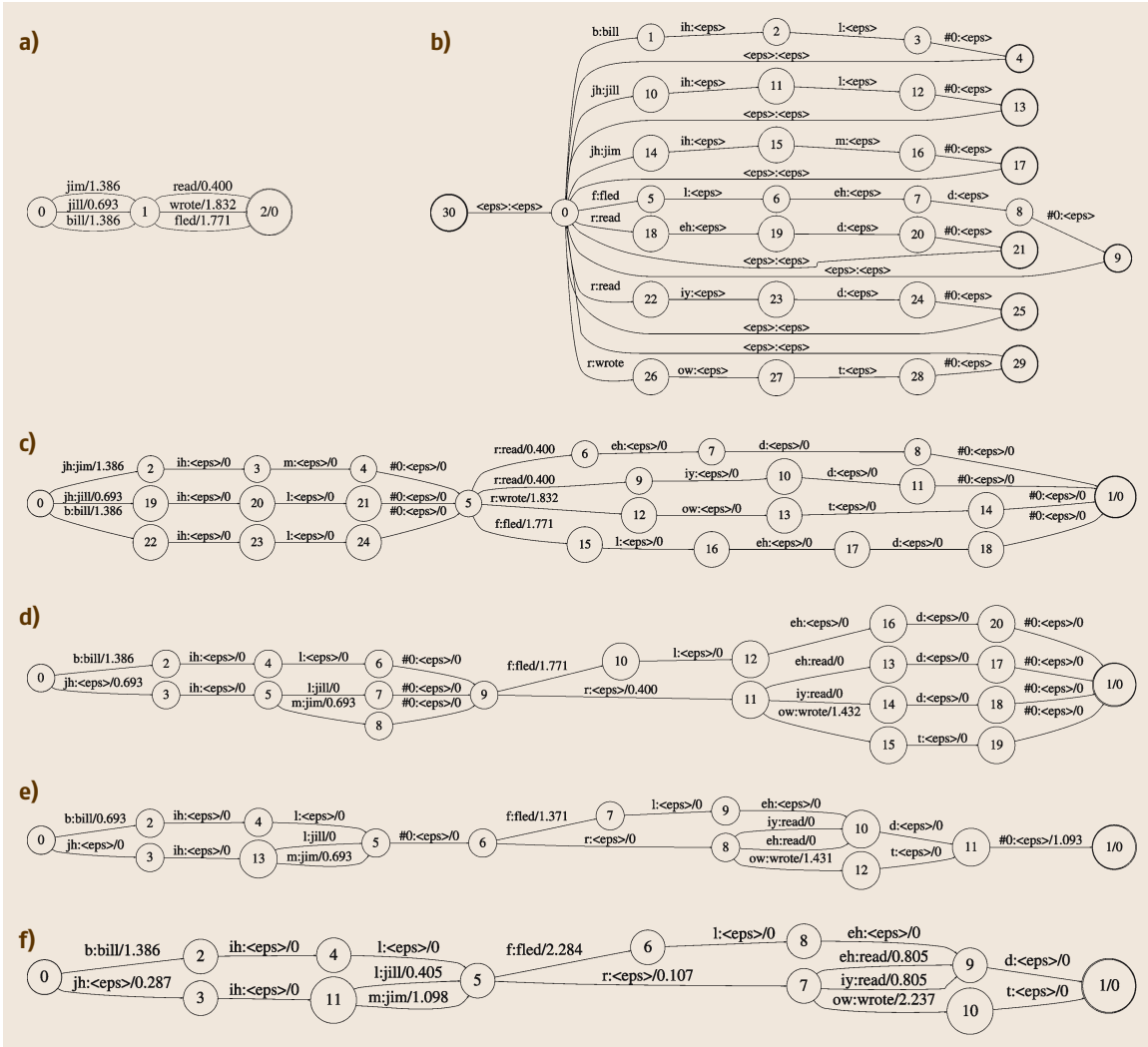



**Fig. 28.17a–f** Recognition transducer construction: **(a)** grammar $G$, **(b)** lexicon $\tilde{L}$, **(c)** $\tilde{L} \circ G$, **(d)** $\det(\tilde{L} \circ G)$, **(e)** $\min_{\text{tropical}}(\det(\tilde{L} \circ G))$, **(f)** $\min_{\log}(\det(\tilde{L} \circ G))$

Viterbi beam search. In contrast, weight pushing in the tropical semiring, which is based on lowest weights between paths described earlier, produces a transducer that may slow down beam-pruned Viterbi decoding many fold.

To push weights in the log semiring instead of the tropical semiring, the potential function is the $-\log$ of the total probability of paths from each state to the super-final state rather than the lowest weight from the state to the super-final state. In other words, the transducer is pushed in terms of probabilities along all future paths from a given state rather than the highest probability over the single best path. By using $-\log$ probability pushing, we preserve a desirable property of the language model, namely that the weights of the transitions leaving each state are normalized as in a probabilistic automaton [28.32]. We have observed that probability pushing makes pruning more effective [28.21], and conjecture that this is because the acoustic likelihoods and the transducer probabilities are now *synchronized* to obtain the optimal likelihood ratio test for deciding whether to prune. We further conjecture that this reweighting is the best possible for pruning. A proof of these conjectures will require a careful mathematical analysis of pruning.

We have thus *standardized* the integrated transducer in our construction – it is the *unique* deterministic, minimal transducer for which the weights for all transitions leaving any state sum to 1 in probability, up to state relabeling. If one accepts that these are desirable properties of an integrated decoding transducer, then our methods obtain the *optimal* solution among all integrated transducers.

Figure 28.17 illustrates the steps in this construction. For simplicity, we consider a small toy grammar and show the construction only down to the context-independent phone level. Figure 28.17a shows the toy grammar $G$ and Fig. 28.17b shows the lexicon $\tilde{L}$. Note the word labels on the lexicon are on the initial transitions and that disambiguating auxiliary symbols have been added at the word ends. Figure 28.17c shows their composition $\tilde{L} \circ G$. Figure 28.17d shows the resulting determinization, $\det(\tilde{L} \circ G)$; observe how phone redundancy is removed. Figure 28.17e,f shows the minimization step, $\min(\det(\tilde{L} \circ G))$; identical futures are combined. In Fig. 28.17e, the minimization uses weight pushing over the tropical semiring, while in Fig. 28.17f, the log semiring is used.

### Factoring

For efficiency reasons, our decoder has a separate representation for variable-length left-to-right HMMs, which we will call the *HMM specification*. The integrated transducer of the previous section does not take good advantage of this since, having combined the HMMs into the recognition transducer proper, the HMM specification consists of trivial one-state HMMs. However, by suitably *factoring* the integrated transducer, we can again take good advantage of this feature.

A path whose states other than the first and last have at most one outgoing and one incoming transition is called a *chain*. The integrated recognition transducer just described may contain many chains after the composition with $\tilde{H}$, and after determinization. As mentioned before, we do not explicitly represent the HMM-state self-loops but simulate them in the run-time decoder. The set of all chains in $N$ is denoted by $\text{chain}(N)$.

The input labels of $N$ name one-state HMMs. We can replace the input of each length-$n$ chain in $N$ by a single label naming an $n$-state HMM. The same label is used for all chains with the same input string. The result of that replacement is a more-compact transducer denoted by $F$. The factoring operation on $N$ leads to the following decomposition:

$$N = H' \circ F \,, \tag{28.21}$$

where $H'$ is a transducer mapping variable-length left-to-right HMM state distribution names to $n$-state HMMs. Since $H'$ can be separately represented in the decoder's HMM specification, the actual recognition transducer is just $F$.

Chain inputs are in fact replaced by a single label only when this helps to reduce the size of the transducer. This can be measured by defining the *gain* of the replacement of an input string $\sigma$ of a chain by:

$$G(\sigma) = \sum_{\pi \in \text{chain}(N), i[\pi]=\sigma} |\sigma| - |o[\pi]| - 1 \,, \tag{28.22}$$

where $|\sigma|$ denotes the length of the string $\sigma$, $i[\pi]$ the input label, and $o[\pi]$ the output label of a path $\pi$. The replacement of a string $\sigma$ helps to reduce the size of the transducer if $G(\sigma) > 0$.

Our implementation of the factoring algorithm allows one to specify the maximum number $r$ of replacements done (the $r$ chains with the highest gain are replaced), as well as the maximum length of the chains that are factored.

Factoring does not affect recognition time. It can however significantly reduce the size of the recognition transducer. We believe that even better factoring methods may be found in the future.



Amazon EX1010 Page 596

**Table 28.2** Size of the first-pass recognition transducers in the NAB 40 000-word vocabulary task

| Transducer | States | Transitions |
|---|---|---|
| $G$ | 1 339 664 | 3 926 010 |
| $L \circ G$ | 8 606 729 | 11 406 721 |
| $\det(L \circ G)$ | 7 082 404 | 9 836 629 |
| $C \circ \det(L \circ G)$ | 7 273 035 | 10 201 269 |
| $\det(H \circ C \circ L \circ G)$ | 18 317 359 | 21 237 992 |
| $F$ | 3 188 274 | 6 108 907 |
| $\min(F)$ | 2 616 948 | 5 497 952 |

**Table 28.3** **(a)** Recognition speed of the first-pass transducers in the NAB 40, 000-word vocabulary task at 83% word accuracy. **(b)** Recognition speed of the second-pass transducers in the NAB 160 000-word vocabulary task at 88% word accuracy

| **a)** Transducer | × real time |
|---|---|
| $C \circ L \circ G$ | 12.5 |
| $C \circ \det(L \circ G)$ | 1.2 |
| $\det(H \circ C \circ L \circ G)$ | 1.0 |
| $\min(F)$ | 0.7 |
| **b)** Transducer | × real time |
| $C \circ L \circ G$ | 0.18 |
| $C \circ \det(L \circ G)$ | 0.13 |
| $C \circ \min(\det(L \circ G))$ | 0.02 |

### Experimental Results – First-Pass Transducers

We have used the techniques discussed in the previous sections to build many recognizers. To illustrate the effectiveness of the techniques and explain some practical details, we discuss here an integrated, optimized recognition transducer for a 40 000-word vocabulary North American Business News (NAB) task. The following models are used:

- An acoustic model of 7208 distinct HMM states, each with an emission mixture distribution of up to 12 Gaussians.
- Triphonic context-dependency transducer $C$ with 1525 states and 80 225 transitions.
- A 40 000-word pronunciation dictionary $L$ with an average of 1.056 pronunciations per word and an out-of-vocabulary rate of 2.3% on the NAB Eval '95 test set.
- A trigram language model $G$ with 3 926 010 transitions built by Katz's back-off method with frequency cutoffs of 2 for bigrams and 4 for trigrams, shrunk with an epsilon of 40 using the method of [28.33], which retained all the unigrams, 22.3% of the bigrams, and 19.1% of the trigrams. The perplexity on the NAB Eval '95 test set is 164.4 (142.1 before shrinking).

We applied the transducer optimization steps as described in the previous section except that we applied the minimization and weight pushing after factoring the transducer. Table 28.2 gives the size of the intermediate and final transducers.

Observe that the factored transducer $\min(F)$ has only about 40% more transitions than $G$. The HMM specifi-

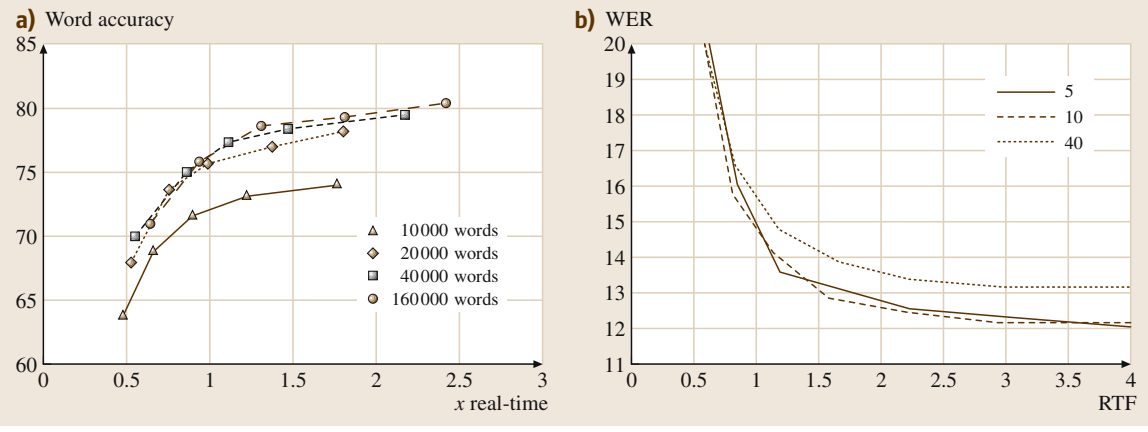



**Fig. 28.18** **(a)** Effect of vocabulary size: NAB bigram recognition results for different vocabulary sizes (LG optimized only). **(b)** NAB Eval 1995 recognition results for the Seymore and Rosenfeld shrink factors of 5, 10, and 40 (thanks to RWTH; uses RWTH acoustic models)



Amazon EX1010 Page 597

cation $H'$ consists of 430 676 HMMs with an average of 7.2 states per HMM. It occupies only about 10% of the memory of $\min(F)$ in the decoder (due to the compact representation possible from its specialized topology). Thus, the overall memory reduction from factoring is substantial.

We used these transducers in a simple, general-purpose, one-pass Viterbi decoder applied to the DARPA NAB Eval '95 test set. Table 28.3a shows the recognition speed on a Compaq Alpha 21264 processor for the various optimizations, where the word accuracy has been fixed at 83.0%. We see that the fully optimized recognition transducer, $\min(F)$, substantially speeds up recognition.

To obtain improved accuracy, we might widen the decoder beam, use a larger vocabulary, or use a less-shrunken language model. These models have an asymptotic wide-beam accuracy of 85.3%. Figure 28.18a shows the affect of vocabulary size (with a bigram LM and optimization only to the $L \circ G$ level). We see that beyond 40 000 words, there is little benefit to increasing the vocabulary either in real-time performance or asymptotically. Figure 28.18b shows the affect of the language model shrinking parameter. These curves were produced by Stephan Kanthak of RWTH using our transducer construction, but RWTH's acoustic models, as part of a comparison with lexical tree methods [28.34]. As we can see, decreasing the shrink parameter from 40 as used above to 10 has a significant effect, while further reducing it to 5 has very little effect. An alternative to using a larger LM is to use a two-pass system to obtain improved accuracy, as described in the next section. This has the advantage tha tit allows quite compact shrunken bigram LMs in the first pass, while the second pass performs as well as the larger-model single-pass systems.

While our examples here have been on NAB, we have also applied these methods to Broadcast News [28.35], Switchboard, and various AT&T-specific large-vocabulary tasks [28.29]. In our experience, fully optimized and factored recognition transducers provide very fast decoding while often having substantially less than twice the number of transitions as their word-level grammars.

### Experimental Results – Rescoring Transducers

The weighted transducer approach is also easily applied to multipass recognition. To illustrate this, we now show how to implement lattice rescoring for a 160 000-word vocabulary NAB task. The following models are used to build lattices in a first pass:

- An acoustic model of 5520 distinct HMM states, each with an emission mixture distribution of up to four Gaussians.
- A triphonic context-dependency transducer $C$ with 1525 states and 80 225 transitions.
- A 160 000-word pronunciation dictionary $L$ with an average of 1.056 pronunciations per word and an out-of-vocabulary rate of 0.8% on the NAB Eval '95 test set.
- A bigram language model $G$ with 1 238 010 transitions built by Katz's back-off method with frequency cutoffs of 2 for bigrams. It is shrunk with an epsilon of 160 using the method of *Seymore* and *Rosenfeld* [28.33], which retained all the unigrams and 13.9% of the bigrams. Perplexity on the NAB Eval '95 test set is 309.9.

We used an efficient approximate lattice-generation method [28.36] to generate word lattices. These word lattices are then used as the grammar in a second rescoring pass. The following models are used in the second pass:

- An acoustic model of 7208 distinct HMM states, each with an emission mixture distribution of up to 12 Gaussians. The model is adapted to each speaker using a single full-matrix MLLR (maximum likelihood linear regression) transform [28.37].
- A triphonic context-dependency transducer $C$ with 1525 states and 80 225 transitions.
- A 160 000-word stochastic, TIMIT-trained, multiple-pronunciation lexicon $L$ [28.38].
- A 6-gram language model $G$ with 40 383 635 transitions built by Katz's back-off method with frequency cutoffs of 1 for bigrams and trigrams, 2 for 4-grams, and 3 for 5-grams and 6-grams. It is shrunk with an epsilon of 5 using the method of Seymore and Rosenfeld, which retained all the unigrams, 34.6% of the bigrams, 13.6% of the trigrams, 19.5% of the 4-grams, 23.1% of the 5-grams, and 11.73% of the 6-grams. The perplexity on the NAB Eval '95 test set is 156.83.

We applied the transducer optimization steps described in the previous section but only to the level of $L \circ G$ (where $G$ is each lattice). Table 28.3b shows the speed of second-pass recognition on a Compaq Alpha 21264 processor for these optimizations when



Amazon EX1010 Page 598

the word accuracy is fixed at 88.0% on the DARPA Eval '95 test set. The recognition speed excludes the offline transducer construction time. We see that the optimized recognition transducers again substantially speed up recognition. The median number of lattice states and arcs is reduced by $\sim 50\%$ by the optimizations.

## 28.5 Conclusion

We presented an overview of weighted finite-state transducer methods and their application to speech recognition. The methods are quite general, and can also be applied in other areas of speech and language processing, including information extraction, speech synthesis [28.39, 40], phonological and morphological analysis [28.41, 42], optical character recognition, biological sequence analysis, and other pattern-matching and string-processing applications [28.43], and image processing [28.44], to mention just some of the most active application areas.



### References


28.1 F. Pereira, R. Wright: Finite-state approximation of phrase-structure grammars. In: *Finite-State Language Processing*, ed. by E. Roche, Y. Schabes (MIT Press, Cambridge, MA 1997) pp. 149–173

28.2 M-J Nederhof: Practical experiments with regular approximation of context-free languages, Computational Linguistics, 26(1) (2000)

28.3 M. Mohri, M.-J. Nederhof: Regular approximation of context-free grammars through transformation. In: *Robustness in Language and Speech Technology*, ed. by J.C. Junqua, G. van Noord (Kluwer Academic, The Netherlands 2001) pp. 153–163

28.4 M. Mohri, F. Pereira, M. Riley: The design principles of a weighted finite-state transducer library, Theoret. Comput. Sci. **231**, 17–32 (2000)

28.5 F. Pereira, M. Riley: *Finite State Language Processing, chapter Speech Recognition by Composition of Weighted Finite Automata* (MIT Press, Cambridge, MA 1997)

28.6 M. Mohri: Finite-state transducers in language and speech processing, Computational Linguistics 23(2) (1997)

28.7 M. Mohri, F. Pereira, Michael Riley: Weighted automata in text and speech processing, ECAI-96 Workshop, Budapest ECAI (1996)

28.8 M. Mohri, M. Riley: Network optimizations for large vocabulary speech recognition, Speech Commun. 25(3) (1998)

28.9 M. Mohri, M. Riley, D. Hindle, A. Ljolje, F. Pereira: full expansion of context-dependent networks in large vocabulary speech recognition, Procs. ICASSP (ICASSP '98), Seattle (1998)

28.10 M. Mohri, M. Riley: Integrated context-dependent networks in very large vocabulary speech recognition, Proc. 6th Europ. Conf. Speech Commun. Technol. (Eurospeech '99), Budapest (1999)

28.11 S. Ortmanns, H. Ney, A. Eiden: Language-model look-ahead for large vocabulary speech recognition, Proc. Int. Conf. Spoken Lang. Proces. (ICSLP'96), University of Delaware and Alfred I. duPont Institute (1996) pp. 2095–2098

28.12 A. Salomaa, M. Soittola: *Automata-Theoretic Aspects of Formal Power Series* (Springer, New York 1978)

28.13 J. Berstel, C. Reutenauer: *Rational Series and Their Languages* (Springer, Heidelberg 1988)

28.14 W. Kuich, A. Salomaa: *Semirings, Automata, Languages, EATCS Monographs on Theoretical Computer Science*, Vol. 5 (Springer, Berlin, Heidelberg 1986)

28.15 C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, M. Mohri: OpenFST – a General and Efficient Weighted Finite-State Transducer Library, Proceedings of the International Conference on Implementation and Application of Automata, Prague (2007)

28.16 J.E. Hopcroft, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation* (Addison Wesley, Reading 1979)

28.17 A.V. Aho, R. Sethi, J.D. Ullman: *Compilers, Principles, Techniques and Tools* (Addison Wesley, Reading 1986)

28.18 A.V. Aho, J.E. Hopcroft, J.D. Ullman: *The Design and Analysis of Computer Algorithms* (Addison Wesley, Reading 1974)

28.19 D. Revuz: Minimisation of acyclic deterministic automata in linear time, Theoret. Comput. Sci. **92**, 181–189 (1992)

28.20 M. Mohri: Semiring frameworks and algorithms for shortest-distance problems, J. Automata Lang. Combinat. **7**(3), 321–350 (2002)

28.21 M. Mohri, M. Riley: A weight pushing algorithm for large vocabulary speech recognition, Proc. 7th Europ. Conf. Speech Commun. Technol. (Eurospeech '01), Aalborg (2001)


Amazon EX1010 Page 599

28.22 D.J. Lehmann: Algebraic structures for transitive closures, Theoret. Comput. Sci. **4**, 59–76 (1977)
28.23 S. Eilenberg: Automata, *Languages and Machines* (Academic, San Diego 1974–1976) vol. A–B
28.24 J. Berstel: *Transductions and Context-Free Languages* (Teubner Studienbucher, Stuttgart 1979)
28.25 C. Allauzen, M. Mohri: Efficient algorithms for testing the twins property, J. Automata Languages Combinat. **8**(2), 117–144 (2003)
28.26 C. Allauzen, M. Mohri: An optimal pre-determinization algorithm for weighted transducers, Theor. Comput. Sci. **328**(1-2), 3–18 (2004)
28.27 M. Mohri: Statistical natural language processing. In *Applied Combinatorics on Words*, ed. by M. Lothaire (Cambridge Univ. Press Cambridge 2005)
28.28 S.M. Katz: Estimation of probabilities from sparse data for the language model component of a speech recogniser, IEEE Trans Acoust. Speech Signal Process. **35**(3), 400–401 (1987)
28.29 C. Allauzen, M. Mohri, B. Roark, M. Riley: A generalized construction of integrated speech recognition transducers, Proc. ICASSP (ICASSP 2004), Montréal (2004)
28.30 M. Riley, F. Pereira, M. Mohri: Transducer composition for context-dependent network expansion, Proc. Eurospeech'97, Rhodes (1997)
28.31 M. Mohri, F. C.N. Pereira: Dynamic compilation of weighted context-free grammars, 36th Annual Meeting ACL and 17th Int. Conf. Computat. Linguist., vol. 2 (1998) p. 891–897
28.32 J.W. Carlyle, A. Paz: Realizations by stochastic finite automaton, J. Comput. Syst. Sci. **5**, 26–40 (1971)
28.33 K. Seymore, R. Rosenfeld: Scalable backoff language models, Proc. ICSLP, Philadelphia (1996)
28.34 S. Kanthak, H. Ney, M. Riley, M. Mohri: A comparison of two LVR search optimization techniques, Proc. Int. Conf. Spoken Lang. Proces. 2002 (ICSLP '02), Denver (2002)
28.35 M. Saraclar, M. Riley, E. Bocchieri, V. Goffin: Towards automatic closed captioning : Low latency real time broadcast news transcription, In Proceedings of the International Conference on Spoken Language Processing (ICSLP'02) (2002)
28.36 A. Ljolje, F. Pereira, M. Riley: Efficient general lattice generation and rescoring, Proc. Europ. Conf. Speech Commun. Technol. (Eurospeech '99), Budapest (1999)
28.37 C. Leggetter, P. Woodland: Maximum likelihood linear regession for speaker adaptation of continuous density HMMs, Comput. Speech Lang. **9**(2), 171–186 (1995)
28.38 M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraclar, C. Wooters, G. Zavaliagkos: Stochastic pronunciation modelling form hand-labelled phonetic corpora, Speech Commun. **29**, 209–224 (1999)
28.39 R. Sproat: Multilingual text analysis for text-to-speech synthesis, J. Nat. Lang. Eng. **2**(4), 369–380 (1997)
28.40 C. Allauzen, M. Mohri, M. Riley: Statistical modeling for unit selection in speech synthesis, In 42nd Meeting of the Association for Computational Linguistics (ACL 2004), Proceedings of the Conference, Barcelona (2004)
28.41 R. M. Kaplan, M. Kay: Regular models of phonological rule systems, Computational Linguistics 20(3) (1994)
28.42 L. Karttunen: The replace operator, In 33rd Meeting of the Association for Computational Linguistics (ACL 95), Proceedings of the Conference, MIT, Cambridge ACL (1995)
28.43 M. Crochemore, W. Rytter: *Text Algorithms* (Oxford Univ. Press, Oxford 1994)
28.44 K.I.I. Culik, J. Kari: Digital images and formal languages. In: *Handbook of Formal Languages*, ed. by G. Rozenberg, A Salomaa (Springer, Berlin, Heidelberg 1997) pp. 599–616




Amazon EX1010 Page 600

# 29. A Machine Learning Framework for Spoken-Dialog Classification

C. Cortes, P. Haffner, M. Mohri

One of the key tasks in the design of large-scale dialog systems is classification. This consists of assigning, out of a finite set, a specific category to each spoken utterance, based on the output of a speech recognizer. Classification in general is a standard machine-learning problem, but the objects to classify in this particular case are word lattices, or weighted automata, and not the fixed-size vectors for which learning algorithms were originally designed. This chapter presents a general kernel-based learning framework for the design of classification algorithms for weighted automata. It introduces a family of kernels, *rational kernels*, that combined with support vector machines form powerful techniques for spoken-dialog classification and other classification tasks in text and speech processing. It describes efficient algorithms for their computation and reports the results of their use in several difficult spoken-dialog classification tasks based on deployed systems. Our results show that rational kernels are easy to design and implement, and lead to substantial improvements of the classification accuracy. The chapter also provides some theoretical results helpful for the design of rational kernels.

29.1 Motivation ..... 585
29.2 Introduction to Kernel Methods ..... 586
29.3 Rational Kernels ..... 587
29.4 Algorithms ..... 589
29.5 Experiments ..... 591
29.6 Theoretical Results for Rational Kernels .. 593
29.7 Conclusion ..... 594
References ..... 595



## 29.1 Motivation

A critical problem for the design of large-scale spoken-dialog systems is to assign a category, out of a finite set, to each spoken utterance. These categories help guide the dialog manager in formulating a response to the speaker. The choice of categories depends on the application, they could be for example *referral* or *pre-certification* for a health-care company dialog system, or *billing services* or *credit* for an operator-service system.

To determine the category of a spoken utterance, one needs to analyze the output of a speech recognizer. Figure 29.1 is taken from a customer-care application. It illustrates the output of a state-of-the-art speech recognizer in a very simple case where the spoken utterance is "Hi, this is my number." The output is an acyclic weighted automaton called a *word lattice*. It compactly represents the recognizer's best guesses. Each path is la-

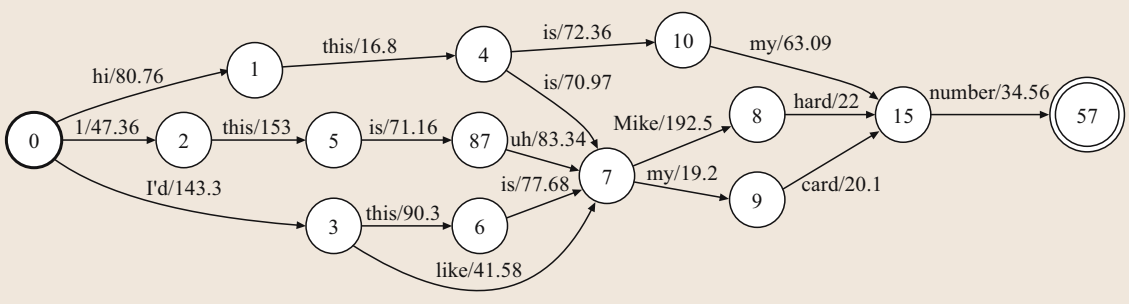



**Fig. 29.1** Word lattice output of a speech recognition system for the spoken utterance "Hi, this is my number"

beled with a sequence of words and has a score obtained by summing the weights of the constituent transitions. The path with the lowest score is the recognizer's best guess, in this case: *I'd like my card number*.

This example makes evident that the error rate of conversational speech recognition systems is still too high in many tasks to rely only on the one-best output of the recognizer. Instead, one can use the full word lattice, which contains the correct transcription in most cases. This is indeed the case in Fig. 29.1, since the top path is labeled with the correct sentence. Thus, in this chapter, spoken-dialog classification is formulated as the problem of assigning a category to each word lattice.

Classification in general is a standard machine-learning problem. A classification algorithm receives a finite number of labeled examples which it uses for training, and selects a hypothesis expected to make few errors on future examples. For the design of modern spoken-dialog systems, this training sample is often available. It is the result of careful human labeling of spoken utterances with a finite number of predetermined categories of the type already mentioned.

However, most classification algorithms were originally designed to classify fixed-size vectors. The objects to analyze for spoken-dialog classification are word lattices, each a collection of a large number of sentences with some weight or probability. How can standard classification algorithms such as support vector machines [29.1] be extended to handle such objects?

This chapter presents a general framework and solution for this problem, which is based on *kernels methods* [29.2, 3]. Thus, we shall start with a brief introduction to kernel methods (Sect. 29.2). Section 29.3 will then present a kernel framework, *rational kernels*, that is appropriate for word lattices and other weighted automata. Efficient algorithms for the computation of these kernels will be described in Sect. 29.4. We also report the results of our experiments using these methods in several difficult large-vocabulary spoken-dialog classification tasks based on deployed systems in Sect. 29.5. There are several theoretical results that can guide the design of kernels for spoken-dialog classification. These results are discussed in Sect. 29.6.



## 29.2 Introduction to Kernel Methods

Let us start with a very simple two-group classification problem illustrated by Fig. 29.2 where one wishes to distinguish two populations, the blue and red circles. In this very simple example, one can choose a hyperplane to separate the two populations correctly, but there are infinitely many choices for the selection of that hyperplane. There is good theory though supporting the choice of the hyperplane that maximizes the *margin*, that is the distance between each population and the separating hyperplane. Indeed, let $\mathcal{F}$ denote the class of real-valued functions on the ball of radius $R$ in $\mathbb{R}^N$:

$$\mathcal{F} = \{x \mapsto w \cdot x : \|w\| \leq 1, \|x\| \leq R\} . \tag{29.1}$$

Then, it can be shown [29.4] that there is a constant $c$ such that, for all distributions $D$ over $X$, with probability at least $1-\delta$, if a classifier $\mathrm{sgn}(f)$, with $f \in \mathcal{F}$, has margin at least $\rho$ over $m$ independently generated training examples, then the generalization error of $\mathrm{sgn}(f)$, or error on any future example, is no more than

$$\frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2 m + \log\frac{1}{\delta}\right) . \tag{29.2}$$

This bound justifies large-margin classification algorithms such as support vector machines (SVMs). Let $w \cdot x + b = 0$ be the equation of the hyperplane, where $w \in \mathbb{R}^N$ is a vector normal to the hyperplane and $b \in \mathbb{R}$ a scalar offset. The classifier $\mathrm{sgn}(h)$ corresponding to this hyperplane is unique and can be defined with respect to the training points $x_1, \ldots, x_m$:

$$h(x) = w \cdot x + b = \sum_{i=1}^{m} \alpha_i (x_i \cdot x) + b , \tag{29.3}$$

where the $\alpha_i$ are real-valued coefficients. The main point we are interested in here is that, both for the construction

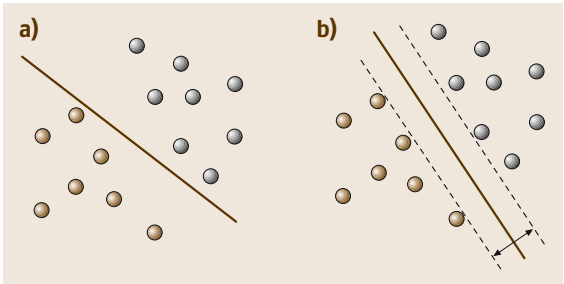



**Fig. 29.2a,b** Large-margin linear classification. **(a)** An arbitrary hyperplane can be chosen to separate the two groups. **(b)** The maximal-margin hyperplane provides better theoretical guarantees

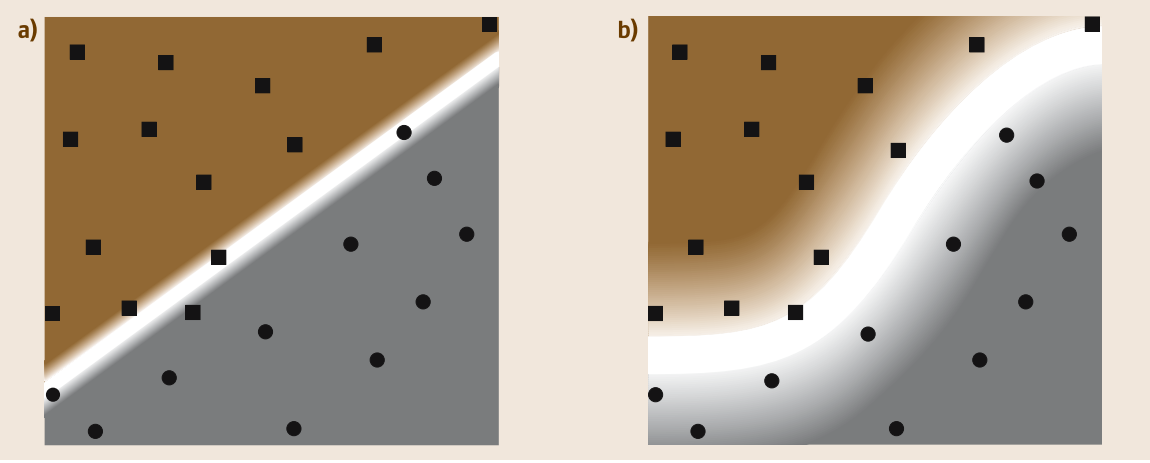



**Fig. 29.3a,b** Nonlinearly separable case. The classification task consists of discriminating between the solid squares and solid circles. **(a)** No hyperplane can separate the two populations. **(b)** A nonlinear mapping can be used instead

of the hypothesis and the later use of that hypothesis for classification of new examples, one needs only to compute a number of dot products between examples.

In practice, nonlinear separation of the training data is often not possible. Figure 29.3a shows an example where any hyperplane crosses both populations. However, one can use more-complex functions to separate the two sets as in Fig. 29.3b. One way to do that is to use a nonlinear mapping $\Phi : X \to F$ from the input space $X$ to a higher-dimensional space $F$ where linear separation is possible.

The dimension of $F$ can truly be very large in practice. For example, in the case of document classification, one may use as features, sequences of three consecutive words (trigrams). Thus, with a vocabulary of just 100 000 words, the dimension of the feature space $F$ is $10^{15}$. On the positive side, as indicated by the error bound of (29.2), the generalization ability of large-margin classifiers such as SVMs does not depend on the dimension of the feature space but only on the margin $\rho$ and the number of training examples $m$. However, taking a large number of dot products in a very high-dimensional space to define the hyperplane may be very costly.

A solution to this problem is to use the so-called *kernel trick* or *kernel methods*. The idea is to define a function $K : X \times X \to \mathbb{R}$ called a *kernel*, such that the kernel function on two examples $x$ and $y$ in input space, $K(x, y)$, is equal to the dot product of two examples $\Phi(x)$ and $\Phi(y)$ in feature space:

$$\forall x, y \in X, \quad K(x, y) = \Phi(x) \cdot \Phi(y) . \tag{29.4}$$

$K$ is often viewed as a similarity measure. A crucial advantage of $K$ is efficiency: there is no need anymore to define and explicitly compute $\Phi(x)$, $\Phi(y)$, and $\Phi(x) \cdot \Phi(y)$. Another benefit of $K$ is flexibility: $K$ can be arbitrarily chosen as long as the existence of $\Phi$ is guaranteed, which is called Mercer's condition. This condition is important to guarantee the convergence of training for algorithms such as SVMs. Some standard Mercer kernels over a vector space are the polynomial kernels of degree $d \in \mathbb{N}$, $K_d(x, y) = (x \cdot y + 1)^d$, and Gaussian kernels $K_\sigma(x, y) = \exp(-\|x - y\|^2/\sigma^2)$, $\sigma \in \mathbb{R}_+$.

A condition equivalent to Mercer's condition is that the kernel $K$ be *positive definite and symmetric*, that is, in the discrete case, the matrix $[K(x_i, x_j)]_{1 \le i,j \le n}$ must be symmetric and positive semidefinite for any choice of $n$ points $x_1, \ldots, x_n$ in $X$. Said differently, the matrix must be symmetric and its eigenvalues nonnegative. Thus, for the problem that we are interested in, the question is how to define positive-definite symmetric kernels for word lattices or weighted automata.

## 29.3 Rational Kernels

This section introduces a family of kernels for weighted automata, *rational kernels*. We will start with some preliminary definitions of automata and transducers. For the most part, we will adopt the notation

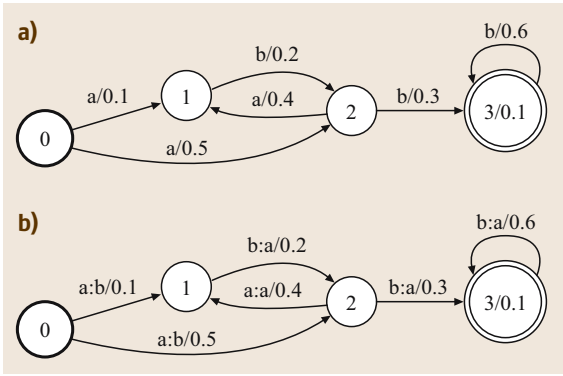



**Fig. 29.4a,b** Weighted automata and transducers. **(a)** Example of a weighted automaton $A$. $[\![A]\!](\text{abb}) = 0.1\times0.2\times0.3\times0.1+0.5\times0.3\times0.6\times0.1$. **(b)** Example of a weighted transducer $T$. $[\![T]\!](\text{abb}, \text{baa}) = [\![A]\!](abb)$

introduced in Chap. 28 for automata and transducers.

Figure 29.4a shows a simple weighted automaton. It is a weighted directed graph in which edges or transitions are augmented with a label and carry some weight indicated after the slash symbol. A bold circle indicates an initial state and a double circle a final state. A final state may also carry a weight indicated after the slash symbol representing the state number.

A path from an initial state to a final state is called a *successful path*. The weight of a path is obtained by multiplying the weights of constituent transitions and the final weight. The weight associated by $A$ to a string $x$, denoted by $[\![A]\!](x)$, is obtained by summing the weights of all paths labeled with $x$. In this case, the weight associated to the string 'abb' is the sum of the weight of two paths. The weight of each path is obtained by multiplying transition weights and the final weight 0.1.

Similarly, Fig. 29.4b shows an example of a weighted transducer. Weighted transducers are similar to weighted automata but each transition is augmented with an output label in addition to the familiar input label and the weight. The output label of each transition is indicated after the colon separator. The weight associated to a pair of strings $(x, y)$, denoted by $[\![T]\!](x, y)$, is obtained by summing the weights of all paths with input label $x$ and output label $y$. Thus, for example, the weight associated by the transducer $T$ to the pair (abb, baa) is obtained as in the automata case by summing the weights of the two paths labeled with (abb, baa).

To help us gain some intuition into the definition of a family of kernels for weighted automata, let us first consider kernels for sequences. As mentioned earlier in Sect. 29.2, a kernel can be viewed as a similarity measure. In the case of sequences, we may say that two strings are similar if they share many common substrings or subsequences. The kernel could then be for example the sum of the product of the counts of these common substrings. But how can we generalize that to weighted automata?

Similarity measures such as the one just discussed can be computed by using weighted finite-state transducers. Thus, a natural idea is to use weighted transducers to define similarity measures for sequences. We will say a that kernel $K$ is *rational* when there exists a weighted transducer $T$ such that

$$K(x, y) = [\![T]\!](x, y)\,, \tag{29.5}$$

for all sequences $x$ and $y$. This definition generalizes naturally to the case where the objects to handle are weighted automata. We say that $K$ is *rational* when there exists a weighted transducer $T$ such that $K(A, B)$, the similarity measure of two automata $A$ and $B$, is given by:

$$K(A, B) = \sum_{x,y} [\![A]\!](x)\cdot[\![T]\!](x, y)\cdot[\![B]\!](y)\,. \tag{29.6}$$

$T(x, y)$ is the similarity measure between two strings $x$ and $y$. But, in addition, we need to take into account the weight associated by $A$ to $x$ and by $B$ to $y$, and sum over all pairs $(x, y)$. This definition can be generalized to the case of an arbitrary semiring where general operations other than the usual sum and multiplication are applied, which has important theoretical and algorithmic consequences and also interesting software-engineering implications [29.5].

Our definition of kernels for weighted automata is a first step towards extending SVMs to handle weighted automata. However, we also need to provide efficient algorithms for computing the kernels. The weighted automata that we are dealing with in spoken-dialog systems are word lattices that may have hundreds of thousands of states and transitions, and millions of paths, thus the efficiency of the computation is critical. In particular, the computational cost of applying existing string kernel algorithms to each pair of paths of two automata $A$ and $B$ is clearly prohibitive.

We also have to provide effective kernels for spoken-dialog classification and prove that they are indeed positive-definite symmetric so they can be combined with SVM for high-accuracy classification systems.



Amazon EX1010 Page 604

## 29.4 Algorithms

This section describes general, efficient algorithms for the computation of rational kernels between weighted automata and provide examples of effective positive definite symmetric kernels for spoken-dialog classification.

The outline of the algorithm is as follows. The main observation is that the sum defining rational kernels can be viewed as the sum of the weights of all the paths of a single transducer, obtained by *composing* $A$ with $T$ with $B$ (see Chap. 28):

$$\sum_{x,y} [\![A]\!](x)[\![T]\!](x, y)[\![B]\!](y) = \sum_{x,y} [\![A \circ T \circ B]\!](x, y) \,. \tag{29.7}$$

The sum of the weights of all the paths of a transducer can be computed using a general single-source shortest-distance algorithm over the ordinary $(+, \times)$ semiring or the so-called forward–backward algorithm in the acyclic case [29.6]. Thus, this leads to the following general algorithm to compute $K(A, B)$ when $K$ is a rational kernel associated to the transducer $T$.

- Use the composition algorithm (see Chap. 28) to compute $U = A \circ T \circ B$ in time $O(|T||A||B|)$.
- Use a general single-source shortest-distance algorithm to compute the sum of the weights of all successful paths of $U$ [29.6]. This can be done in linear time ($O(|U|)$) when $U$ is acyclic, which is the case when $A$ and $B$ are acyclic automata (word lattices).

In combination, this provides a general, efficient algorithm for computing rational kernels whose total complexity for acyclic word lattices is quadratic: $O(|T||A||B|)$. Here $|T|$ is a constant independent of the automata $A$ and $B$ for which $K(A, B)$ needs to be computed.

The next question that arises is how to define and compute kernels based on counts of substrings or subsequences as previously discussed in Sect. 29.3. For weighted automata, we need to generalize the notion of counts to take into account the weight of the paths in each automaton and use instead the *expected counts* of a sequence.

Let $|u|_x$ denote the number of occurrences of a substring $x$ in $u$. Since a weighted automaton $A$ defines a distribution over the set of strings $u$, the *expected count* of a sequence $x$ in a weighted automaton $A$ can be defined naturally as

$$c_A(x) = \sum_{u \in \Sigma^*} |u|_x [\![A]\!](u) \,, \tag{29.8}$$

where the sum runs over $\Sigma^*$, the set of all strings over the alphabet $\Sigma$. We mentioned earlier that weighted transducers can often be used to count the occurrences of some substrings of interest in a string. Let us assume that one could find a transducer $T$ that could compute the expected counts of all substrings of interest appearing in a weighted automaton $A$. Thus, $A \circ T$ would provide the expected counts of these substrings in $A$. Similarly, $T^{-1} \circ B$ would provide the expected counts of these substrings in $B$. Recall that $T^{-1}$ is the transducer obtained by swapping the input and output labels of each transition Chap. 28. Composition of $A \circ T$ and $T^{-1} \circ B$ matches paths labeled with the same substring in $A \circ T$ and $T^{-1} \circ B$ and multiplies their weights. Thus, by definition of composition (see Chap. 28), we could compute the sum of the expected counts in $A$ and $B$ of the matching substrings by computing

$$A \circ T \circ T^{-1} \circ B \,, \tag{29.9}$$

and summing the weights (expected counts) of all paths using a general single-source shortest-distance algorithm.

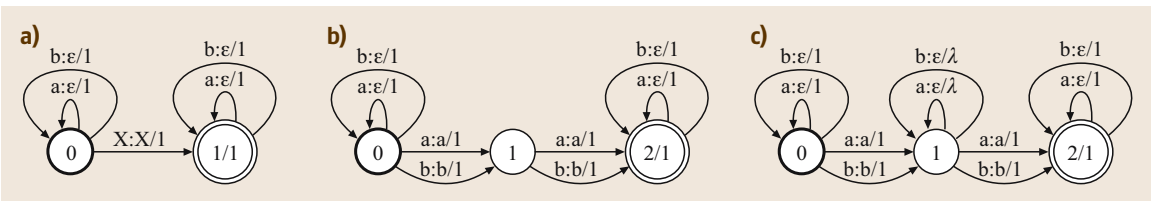



**Fig. 29.5a–c** Weighted transducers computing the expected counts of **(a)** all sequences accepted by the regular expression $X$; **(b)** all bigrams; **(c)** all gappy bigrams with penalty gap $\lambda$, over the alphabet $\{a, b\}$

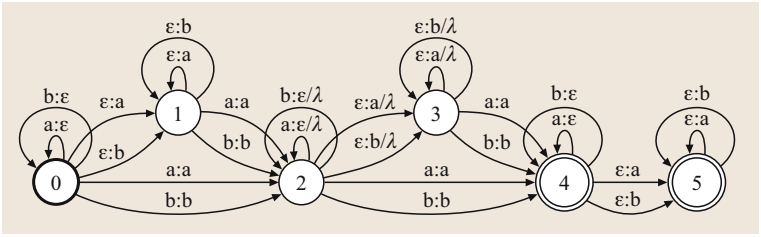



**Fig. 29.6** Simple weighted transducer corresponding to the gappy bigram kernel used by [29.7] obtained by composition of the transducer of Fig. 29.5c and its inverse

Comparing this formula with (29.7) shows that the count-based similarity measure we are interested in is the rational kernel whose corresponding weighted transducer $S$ is:

$$S = T \circ T^{-1} \,. \tag{29.10}$$

Thus, if we can determine a $T$ with this property, this would help us naturally define the similarity measure $S$. In Sect. 29.6, we will prove that the kernel corresponding to $S$ will actually be positive-definite symmetric, and can hence be used in combination with SVMs.

In the following, we will provide a weighted transducer $T$ for computing the expected counts of substrings. It turns out that there exists a very simple transducer $T$ that can be used for that purpose. Figure 29.5a shows a simple transducer that can be used to compute the expected counts of all sequences accepted by the regular expression $X$ over the alphabet $\{a, b\}$. In the figure, the transition labeled with $X : X/1$ symbolizes a finite automaton representing $X$ with identical input and output labels and all weights equal to one.

Here is how transducer $T$ counts the occurrences of a substring $x$ recognized by $X$ in a sequence $u$. State 0 reads a prefix $u_1$ of $u$ and outputs the empty string $\epsilon$. Then, an occurrence of $x$ is read and output identically. Then state 1 reads a suffix $u_2$ of $u$ and outputs $\epsilon$. In how many different ways can $u$ be decomposed into $u = u_1 x u_2$? In exactly as many ways as there are occurrences of $x$ in $u$. Thus, $T$ applied to $u$ generates exactly $|u|_x$ successful paths labeled with $x$. This holds for all strings $x$ accepted by $X$. If $T$ is applied to (or composed with) a weighted automaton $A$ instead, then for any $x$, it generates $|u|_x$ paths for each path of $A$ labeled with $u$. Furthermore, by definition of composition, each path generated is weighted with the weight of the path in $A$ labeled with $u$. Thus, the sum of the weights of the paths generated that are labeled with $x$ is exactly the expected count of $x$:

$$[\![A \circ T]\!](x) = c_A(x) \,. \tag{29.11}$$

Thus, there exists a simple weighted transducer $T$ that can count, as desired, the expected counts of all strings recognized by an arbitrary regular expression $X$ in a weighted automaton $A$. In particular, since the set of bigrams over an alphabet $\Sigma$ is a regular language, we can construct a weighted transducer $T$ computing the expected counts of all bigrams. Figure 29.5b shows that transducer, which has only three states regardless of the alphabet size.

In some applications, one may wish to allow for a gap between the occurrences of two symbols and view two weighted automata as similar if they share such substrings (*gappy bigrams*) with relatively large expected counts. The gap or distance between two symbols is penalized using a fixed penalty factor $\lambda$, $0 \leq \lambda < 1$. A sequence kernel based on these ideas was used successfully by [29.7] for text categorization. Interestingly, constructing a kernel based on gappy bigrams is straightforward in our framework. Figure 29.5c shows the transducer $T$ counting expected counts of gappy bigrams from which the kernel can be efficiently constructed. The same can be done similarly for higher-order gappy $n$-grams or other gappy substrings.

The methods presented in this section can be used to construct efficiently and, often in a simple manner, relatively complex weighted automata kernels $K$ based on expected counts or other ideas. A single, general algorithm can then be used as described to compute efficiently $K(A, B)$ for any two weighted automata A and B, without the need to design a new algorithm for each new kernel, as previously done in the literature. As an example, the gappy kernel used by [29.7] is the rational kernel corresponding to the six-state transducer $S = T \circ T^{-1}$ of Fig. 29.6.



Amazon EX1010 Page 606

## 29.5 Experiments

This section describes the applications of the kernel framework and techniques to spoken-dialog classification.

In most of our experiments, we used simple $n$-gram rational kernels. An $n$-gram kernel $k_n$ for two weighted automata or lattices A and B is defined by:

$$k_n(\mathrm{A}, \mathrm{B}) = \sum_{|x|=n} c_{\mathrm{A}}(x)\, c_{\mathrm{B}}(x)\,. \tag{29.12}$$

As described in Sect. 29.4, $k_n$ is a kernel of the form $T \circ T^{-1}$ and can be computed efficiently. An $n$-gram rational kernel $K_n$ is simply the sum of kernels $k_m$, with $1 \leq m \leq n$:

$$K_n = \sum_{m=1}^{n} k_m\,.$$

Thus, the feature space associated with $K_n$ is the set of all $m$-gram sequences with $m \leq n$. As discussed in the previous section, it is straightforward, using the same algorithms and representations, to extend these kernels to kernels with gaps and to many other more-complex rational kernels more closely adapted to the applications considered.

We did a series of experiments in several large-vocabulary spoken-dialog tasks using rational kernels with a twofold objective [29.8]: to improve classification accuracy in those tasks, and to evaluate the impact on classification accuracy of the use of a word lattice rather than the one-best output of the automatic speech recognition (ASR) system.

The first task we considered was that of a deployed customer-care application (*How may I help you?*, HMIHY 0300). In this task, users interact with a spoken-dialog system via the telephone, speaking naturally, to ask about their bills, their calling plans, or other similar topics. Their responses to the open-ended prompts of the system are not constrained by the system, they may be any natural language sequence. The objective of the spoken-dialog classification is to assign one or several categories or call-types, e.g., *billing credit*, or *calling plans*, to the users' spoken utterances. The set of categories is predetermined, and in this first application there are 64 categories. The calls are classified based on the user's response to the first greeting prompt: "Hello, this is AT&T. How may I help you?"

Table 29.1 indicates the size of the HMIHY 0300 datasets we used for training and testing. The training set is relatively large with more than 35 000 utterances;

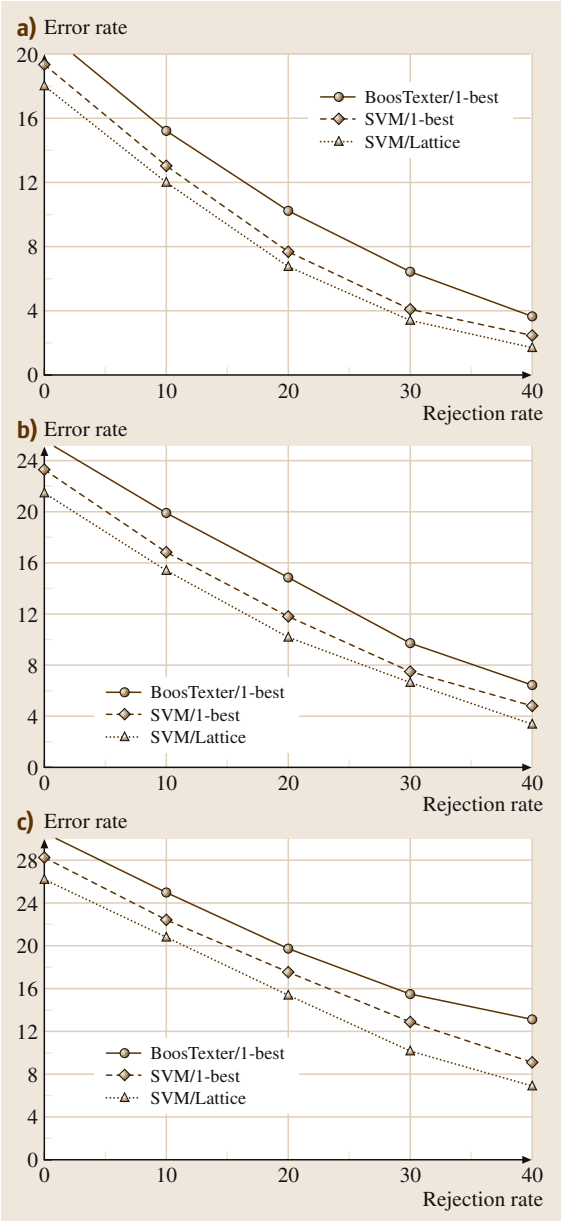



**Fig. 29.7a–c** Classification error rate as a function of rejection rate in **(a)** HMIHY 0300, **(b)** VoiceTone1, and **(c)** VoiceTone2

this is an extension of the one we used in our previous classification experiments with HMIHY 0300 [29.9]. In our experiments, we used the $n$-gram rational kernels

**Table 29.1** Key characteristics of the three datasets used in the experiments. The fifth column displays the total number of unigrams, bigrams, and trigrams found in the one-best output of the ASR for the utterances of the training set, that is the number of features used by BoosTexter or SVMs used with the one-best outputs. The training and testing sizes reported in columns 3 and 4 are described in the number of utterances, or equivalently the number of word lattices

| Dataset | Number of classes | Training size | Testing size | Number of $n$-grams | ASR word accuracy % |
|---|---|---|---|---|---|
| HMIHY 0300 | 64 | 35551 | 5000 | 24177 | 72.5 |
| VoiceTone1 | 97 | 29561 | 5537 | 22007 | 70.5 |
| VoiceTone2 | 82 | 9093 | 5172 | 8689 | 68.8 |

just described with $n = 3$. Thus, the feature set we used was that of all $n$-grams with $n \leq 3$. Table 29.1 indicates the total number of distinct features of this type found in the datasets. The word accuracy of the system based on the best hypothesis of the speech recognizer was 72.5%. This motivated our use of the word lattices, which contain the correct transcription in most cases. The average number of transitions of a word lattice in this task was about 260.

Table 29.1 reports similar information for two other datasets, VoiceTone1, and VoiceTone2. These are more recently deployed spoken-dialog systems in different areas, e.g., VoiceTone1 is a task where users interact with a system related to health-care with a larger set of categories (97). The size of the VoiceTone1 datasets we used and the word accuracy of the recognizer (70.5%) make this task otherwise similar to HMIHY 0300. The datasets provided for VoiceTone2 are significantly smaller with a higher word error rate. The word error rate is indicative of the difficulty of classification task since a higher error rate implies a more noisy input. The average number of transitions of a word lattice in VoiceTone1 was about 210 and in VoiceTone2 about 360.

Each utterance of the dataset may be labeled with several classes. The evaluation is based on the following criterion: it is considered an error if the highest scoring class given by the classifier is none of these labels.

We used the AT&T FSM library [29.10] and the GRM library [29.11] for the implementation of the $n$-gram rational kernels $K_n$ used. We used these kernels with SVMs, using a general learning library for large-margin classification(LLAMA), which offers an optimized multiclass recombination of binary SVMs [29.12]. Training time took a few hours on a single processor of a 2.4 GHz Intel Pentium processor Linux cluster with 2 GB of memory and 512 KB cache.

In our experiments, we used the trigram kernel $K_3$ with a second-degree polynomial. Preliminary experiments showed that the top performance was reached for trigram kernels and that 4-gram kernels, $K_4$, did not significantly improve performance. We also found that the combination of a second-degree polynomial kernel with the trigram kernel significantly improves performance over a linear classifier, but that no further improvement could be obtained with a third-degree polynomial.

We used the same kernels in the three datasets previously described and applied them to both the speech recognizer's single best hypothesis (one-best results), and to the full word lattices output by the speech recognizer. We also ran, for the sake of comparison, the BoosTexter algorithm [29.13] on the same datasets by applying it to the one-best hypothesis. This served as a baseline for our experiments.

Figure 29.7a shows the result of our experiments in the HMIHY 0300 task. It gives classification error rate as a function of rejection rate (utterances for which the top score is lower than a given threshold are rejected) in HMIHY 0300 for: BoosTexter, SVM combined with our kernels when applied to the one-best hypothesis, and SVM combined with kernels applied to the full lattices.

SVM with trigram kernels applied to the one-best hypothesis leads to better classification than BoosTexter everywhere in the range of 0–40% rejection rate. The accuracy is about 2–3% absolute value better than that of BoosTexter in the range of interest for this task, which is roughly between 20% and 40% rejection rate. The results also show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of SVMs applied to the one-best alone by about 1% absolute value.

Figures 29.7b,c show the results of our experiments in the VoiceTone1 and VoiceTone2 tasks using the same techniques and comparisons. As observed previously, in many regards, VoiceTone1 is similar to the HMIHY 0300 task, and our results for VoiceTone1 are comparable to those for HMIHY 0300. The results show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of BoosTexter, by more than 4% absolute value at a rejection rate of about 20%. They also demonstrate



Amazon EX1010 Page 608

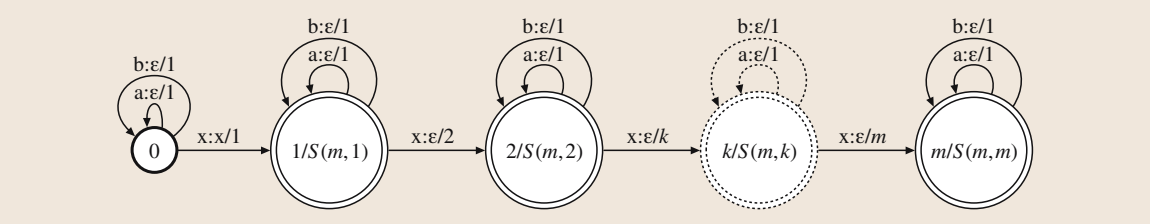



**Fig. 29.8** Weighted transducer $T^m$ for computing the $m$-th moment of the count of an aperiodic substring $x$. The final weight at state $k$, $k = 1, \ldots, m$, indicated after /, is $S(m, k)$, the Stirling number of the second kind, that is the number of ways of partitioning a set of $m$ elements into $k$ nonempty subsets, $S(m, k) = \frac{1}{k!} \sum_{i=0}^{k} \binom{k}{i} (-1)^i (k-i)^m$. The first-order transducer $T_1$ coincides with the transducer of Fig. 29.5a, since $S(1, 1) = 1$

more clearly the benefits of the use of the word lattices for classification in this task. This advantage is even more manifest for the VoiceTone2 task for which the speech recognition accuracy is lower. VoiceTone2 is also a harder classification task, as can be seen by comparing the plots of Fig. 29.7b. The classification accuracy of SVMs with kernels applied to lattices is more than 6% absolute value better than that of BoosTexter near the 40% rejection rate, and about 3% better than SVMs applied to the one-best hypothesis.

Thus, our experiments in spoken-dialog classification in three distinct large-vocabulary tasks demonstrated that using rational kernels with SVMs consistently leads to very competitive classifiers. They also show that their application to the full word lattices instead of the single best hypothesis output by the recognizer systematically improves classification accuracy.

We further explored the use of kernels based on other moments of the counts of substrings in sequences, generalizing $n$-gram kernels [29.14]. Let $m$ be a positive integer. Let $c_{\mathrm{A}}^m(x)$ denote the $m$-th moment of the count of the sequence $x$ in $A$ defined by:

$$c_{\mathrm{A}}^m(x) = \sum_{u \in \Sigma^*} |u|_x^m \, [\![\mathrm{A}]\!](u) \,. \tag{29.13}$$

We can define a general family of kernels, denoted by $K_n^m$, $n, m \geq 1$ and defined by:

$$K_n^m(\mathrm{A}, \mathrm{B}) = \sum_{|x|=n} c_{\mathrm{A}}^m(x) c_{\mathrm{B}}^m(x) \,, \tag{29.14}$$

which exploit the $m$-th moment of the counts of substrings $x$ in weighted automata A and B to define their similarity. *Cortes* and *Mohri* [29.14] showed that there exist weighted transducers $T_n^m$ that can be used to compute $c_{\mathrm{A}}^m(x)$ efficiently for all $n$-gram sequence $x$ and weighted automaton A. Thus, these kernels are rational kernels and their associated transducers are $T_n^m \circ T_n^{m-1}$:

$$K_n^m(\mathrm{A}, \mathrm{B}) = \sum_{x,y} [\![\mathrm{A} \circ [T_n^m \circ (T_n^m)^{-1}] \circ \mathrm{B}]\!](x, y) \,. \tag{29.15}$$

Figure 29.8 shows the weighted transducer $T_1^m$ for aperiodic strings $x$, which has only $m|x| + 1$ states. An aperiodic string is a string that does not admit a nonempty prefix as a suffix. The $m$-th moment of other strings (periodic strings) can also be computed using weighted transducers. By (29.15), the transducer $T_1^m$ can be used to compute $K_n^m(\mathrm{A}, \mathrm{B})$ by first computing the composed transducer $A \circ [T_n^m \circ (T_n^m)^{-1}] \circ \mathrm{B}$ and then summing the weights of all the paths of this transducer using a shortest-distance algorithm [29.6] .

The application of moment kernels to the HMIHY 0300 task resulted in a further improvement of the classification accuracy. In particular, at a 15% rejection rate, the error rate was reduced by 1% absolute, which is about 6.2% relative, which is significant in this task, by using *variance kernels*, that is moment kernels of second order ($m = 2$).

## 29.6 Theoretical Results for Rational Kernels

In the previous sections, we introduced a number of rational kernels, e.g., kernels based on expected counts or moments of the counts and applied them to spoken-dialog tasks. Several questions arise in relation to these kernels. As pointed out earlier, to guarantee the convergence of algorithms such as SVMs, the kernels used

must be positive-definite symmetric (PDS). But, how can we construct PDS rational kernels? Are $n$-gram kernels and similar kernels PDS? Can we combine simpler PDS rational kernels to create more-complex ones? Is there a characterization of PDS rational kernels?

All these questions have been investigated by Cortes et al. [29.5]. The following theorem provides a general method for constructing PDS rational kernels.

*Theorem 29.1*

Let $T$ be a weighted finite-state transducer over $(+, \times)$. Assume that the weighted transducer $T \circ T^{-1}$ is regulated, then $S = T \circ T^{-1}$ defines a PDS rational kernel. A weighted transducer $T$ is said to be *regulated* when $[\![T]\!](x, y)$, the sum of the weights of the paths with input $x$ and output $y$, is well defined.

*Proof.* We give a sketch of the proof. A full proof is given in [29.5]. Let $K$ be the kernel associated with $S = T \circ T^{-1}$. By definition of $T^{-1}$ and composition,

$$\forall x, y \in X, K(x, y) = \sum_z [\![T]\!](x, z)[\![T]\!](y, z) , \tag{29.16}$$

where the sum is over all strings $z$. Let $K_n$ be the function defined by restricting the sum to strings of length at most $k$:

$$\forall x, y \in X, K_k(x, y) = \sum_{|z| \le k} [\![T]\!](x, z)[\![T]\!](y, z) . \tag{29.17}$$

Consider any ordering of all strings of length at most $k$: $z_1, \ldots, z_l$. For any set of $n$ strings $x_1, \ldots, x_n$, let $A$ be the matrix defined by $A = [[\![T]\!](x_i, z_j)]_{i \in [1,n], j \in [1,l]}$. Then, the eigenvalues of the matrix $M_n$ defined by

$$M_n = [K_n(x_i, x_j)]_{i,j \in [1,n]} \tag{29.18}$$

are necessarily nonnegative since $M_n = AA^\top$. Thus, for any $n \le 0$, $K_n$ is a PDS kernel. Since $K$ is a pointwise limit of $K_n$, $K$ is also PDS [29.5, 15]. □

The theorem shows that the rational kernels we considered in previous sections, e.g., count-based similarity kernels, $n$-gram kernels, and gappy $n$-gram kernels, are all PDS rational kernels, which justifies a posteriori their use in combination with SVMs. Conversely, we have conjectured elsewhere that all PDS rational kernels are rational kernels associated to transducers of the type $S = T \circ T^{-1}$ [29.5], and proved several results in support of that conjecture. In particular, for acyclic transducer, this indeed provides a characterization of PDS rational kernels.

It can also be shown that a finite sum of PDS rational kernels is a PDS rational kernel, which we used for defining $n$-gram kernels. More generally, the following theorem holds [29.5].

*Theorem 29.2*

PDS rational kernels are closed under sum, product, and Kleene closure.

Thus, one can use rational operations to create complex PDS rational kernels from simpler ones.

## 29.7 Conclusion

Rational kernels form an effective tool and framework for spoken-dialog classification. They are based on a general theory that guarantees in particular the positive definiteness of rational kernels based on an arbitrary $(+, \times)$-weighted transducer and thus the convergence of training for algorithms such as SVMs. General, efficient algorithms can be readily used for their computation.

Experiments in several large-vocabulary spoken-dialog tasks show that rational kernels can be combined with SVMs to form powerful classifiers and that they perform well in several difficult tasks. They also demonstrate the benefits of the use of kernels applied to word lattices.

Rational kernels form a rich family of kernels. The kernels used in the experiments we described are only special instances of this general class of kernels. Rational kernels adapted to a specific spoken-dialog task can be designed. In fact, it is often straightforward to craft prior knowledge about a task in the transducer defining these kernels. One may, for example, exclude some word sequences or regular expressions from the similarity measure defined by these kernels or emphasize the importance of others by increasing their corresponding weight in the weighted transducer.



Amazon EX1010 Page 610

## References


29.1 C. Cortes, V.N. Vapnik: Support-vector networks, Machine Learning **20**(3), 273–297 (1995)

29.2 B.E. Boser, I. Guyon, V.N. Vapnik: A training algorithm for optimal margin classifiers. In: *Proc. 5th Annual Workshop Computational Learning Theory*, Vol. 5 (ACM, Pittsburg 1992) pp. 144–152

29.3 B. Schölkopf, A. Smola: *Learning with Kernels* (MIT Press, Cambridge 2002)

29.4 P. Bartlett, J. Shawe-Taylor: Generalization performance of support vector machines and other pattern classifiers. In: *Advances in Kernel Methods: Support Vector Learning* (MIT Press, Cambridge 1999) pp. 43–54

29.5 C. Cortes, P. Haffner, M. Mohri: Positive definite rational kernels, Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003) (2003), LNCS **2777**, 41–56

29.6 M. Mohri: Semiring frameworks and algorithms for shortest-distance problems, J. Automata Lang. Combinat. **7**(3), 321–350 (2002)

29.7 H. Lodhi, J. Shawe-Taylor, N. Cristianini, C. Watkins: *Text Classification Using String Kernels (NIPS 2000)* (MIT Press, Boston 2001) pp. 563–569

29.8 C. Cortes, P. Haffner, M. Mohri: Rational kernels: Theory and algorithms, J. Machine Learning Res. (JMLR) **5**, 1035–1062 (2004)

29.9 C. Cortes, P. Haffner, M. Mohri: Rational kernels. In: *Advances in Neural Information Processing Systems (NIPS 2002)*, Vol. 15 (MIT Press, Boston 2002)

29.10 M. Mohri, F.C.N. Pereira, M. Riley: The design principles of a weighted finite-state transducer library, Theoret. Comput. Sci. **231**, 17–32 (2000)

29.11 C. Allauzen, M. Mohri, B. Roark: A general weighted grammar library. In: *Proceedings of the Ninth International Conference on Automata* (CIAA, Kingston 2004)

29.12 P. Haffner, G. Tur, J. Wright: Optimizing SVMs for complex Call Classification, Proceedings ICASSP'03 (2003)

29.13 R.E. Schapire, Y. Singer: Boostexter: A boosting-based system for text categorization, Machine Learning **39**(2/3), 135–168 (2000)

29.14 C. Cortes, M. Mohri: Moment kernels for regular distributions, Machine Learning **60**(1-3), 117–134 (2005)

29.15 C. Berg, J.P.R. Christensen, P. Ressel: *Harmonic Analysis on Semigroups* (Springer, Berlin, Heidelberg 1984)




Amazon EX1010 Page 611

# 30. Towards Superhuman Speech Recognition

M. Picheny, D. Nahamoo

After over 40 years of research, human speech recognition performance still substantially outstrips machine performance. Although enormous progress has been made, the ultimate goal of achieving or exceeding human performance – *superhuman* speech recognition – eludes us. On a more-prosaic level, many industrial concerns have been trying to make a go of various speech recognition businesses for many years, yet there is no clear *killer app* for speech. If the technology were as reliable as human perception, would such *killer apps* emerge?

Either way, there would be enormous value in producing a recognizer with superhuman capabilities. This chapter describes an ongoing research program at IBM that attempts to address achieving superhuman speech recognition performance in the context of the metric of word error rate. First, a multidomain conversational test set to drive the research program is described. Then, a series of human listening experiments and speech recognition experiments based on the test set is presented. Large improvements in recognition performance can be achieved through a combination of adaptation, discriminative training, a combination of knowledge sources, and simple addition of more data. Unfortunately, devising a set of informative listening tests synchronized with the multidomain test set proved to be more difficult than expected because of the highly informal nature of the underlying speech. The problems encountered in performing the listening tests are presented along with suggestions for future listening tests. The chapter concludes with a set of speculations on the best way for speech recognition research to proceed in the future in this area.

30.1 **Current Status** ..... 597
30.2 **A Multidomain Conversational Test Set** ... 598
30.3 **Listening Experiments** ..... 599
30.3.1 Baseline Listening Tests ..... 599
30.3.2 Listening Tests to Determine Knowledge Source Contributions ... 600
30.4 **Recognition Experiments** ..... 601
30.4.1 Preliminary Recognition Results .... 602
30.4.2 Results on the Multidomain Test Set ..... 602
30.4.3 System Redesign ..... 606
30.4.4 Coda ..... 606
30.5 **Speculation** ..... 607
30.5.1 Proposed Human Listening Experiments ..... 607
30.5.2 Promising Incremental Approaches 608
30.5.3 Promising Disruptive Approaches .. 612
**References** ..... 614



## 30.1 Current Status

After over 40 years of research, human speech recognition performance still substantially outstrips machine performance. Although enormous progress has been made, the ultimate goal of achieving or exceeding human performance – *superhuman* speech recognition – eludes us. In addition, it is fair to say that there have been no recent *breakthroughs* in the speech recognition area – progress over the last several years, though continual, has been evolutionary rather than revolutionary. Can we achieve levels of superhuman speech recognition in our lifetimes through evolutionary approaches, or do we need to make radical changes to our methodologies?

On a more-prosaic level, many industrial concerns have been trying to make a go of various speech recognition businesses for many years, yet there is no clear *killer app* for speech that would guarantee the huge revenue needed to justify the expense of investing in the technology. Is this because there really is no value in the

technology, or that the technology just needs additional incremental improvements, or is this because the levels of accuracy must reach human performance or higher to be regarded as sufficiently reliable to serve as a user interface?

Either way, there would be enormous value in producing a recognizer with superhuman capabilities. This chapter describes an ongoing research program at IBM that attempts to address some of these questions, focusing on word error rate as a metric. It is recognized that this is only one of many dimensions in which speech recognition performance can be measured, and it is hoped that this chapter will trigger corresponding studies in related areas, such as concept and meaning extraction. First, we describe a multidomain conversational test set established to drive the research program (Sect. 30.2). Then, we describe a series of human listening experiments that attempt to determine the best set of research investments to achieve the goal of superhuman speech recognition (Sect. 30.3), a set of recognition experiments that begin to address methodologies for designing systems to achieve superhuman performance (Sect. 30.4), and a set of speculations on the best way for research to proceed in the future in this area (Sect. 30.5).

## 30.2 A Multidomain Conversational Test Set

We had a number of goals in mind when we designed the test set. First, the test set had to cover a reasonably broad range of conversational applications and contain data representing key challenges to reliable recognition including various forms of acoustic interference, speech from non-native speakers, and a large recognition vocabulary. Second, the test set had to include at least one component that is readily available to other researchers to facilitate comparisons between our recognizers and those developed externally. Third, the test set needed to be reasonably small, to facilitate rapid turnaround of experiments. For all experiments reported here, we used a test set composed of the following five parts.

**swb98**: The switchboard portion of the 1998 hub 5e evaluation set [30.1], consisting of 2 h of telephone-bandwidth (8 kHz) audio. The data were collected from two-person conversations between strangers on a preassigned topic. A variety of telephone channels and regional dialects are represented in the data.

**mtg**: An initial release of the Bmr007 meeting from the ICSI (International Computer Science Institute) meeting corpus [30.2], consisting of 95 minutes of audio. The data were collected from eight speakers wearing either lapel microphones or close-talking headsets. This meeting involved eight speakers: five native speakers of American English (two females and three males), and three non-native speakers (all males). Although the data is wide bandwidth (16 kHz), the primary challenge in this test set is the presence of background speech in many of the utterances. The crosstalk problem is especially severe for speakers recorded using lapel microphones.

**cc1**: 0.5 h of audio from a call center. The data are collected from customer-service representatives (CSRs) and customers calling with service requests or problems. The primary challenge in this test is acoustic interference: a combination of nonlinear distortion from speech compression, background noise from both the CSR and customer sides, and intermittent beeps on the channel, which are played to remind the customer that the call is being recorded. The data is telephony bandwidth but otherwise relatively quiet.

**cc2**: 0.5 h of audio from a second call center. The recordings are from a different center than the cc1 test set, but cover similar subject matter and have similar, poor acoustics. This data set has no information associating speakers with sets of utterances, which poses problems for speaker and channel adaptation. The data is telephony bandwidth but otherwise relatively quiet.

**vm**: Test data from the IBM voicemail corpus, consisting of 1 hour of audio. This material was previously reported on as the E-VM1 test set [30.3], and is a superset of the test data in the voicemail corpus part I and part II distributed by the LDC (Linguistic Data Consortium). Unlike the other tests, the voicemail data are conversational monologues. The acoustic quality of the data is generally quite high, although loud clicks caused by the speaker hanging up at the end of some messages can pose problems for feature normalization – especially

**Table 30.1** Characteristics of the multidomain conversational test set used for listening and recognition experiments

| Task | Number of hours | Number of segments |
|---|---|---|
| swb98 | 2.0 | 3500 |
| mtg | 1.5 | 2060 |
| cc1 | 0.5 | 978 |
| cc2 | 0.5 | 1033 |
| vm | 1.0 | 1033 |



Amazon EX1010 Page 613

normalization of c0 based on the maximum value of c0 within an utterance. This test set also has no information associating speakers with sets of utterances. The data is telephony bandwidth but otherwise relatively quiet.

The test sets were segmented into utterances suitable for recognition using a variety of means. The default LDC segmentation was used for the swb98 data. An automatic segmenter was used on the mtg data. The cc1 and cc2 data came in the form of calls and were automatically segmented into smaller units; the vm data came in the form of individual messages and was also automatically segmented into smaller units. A summary of the total number of hours and segments for each test set is given in Table 30.1.

## 30.3 Listening Experiments

Humans use whatever information is available to aid recognition performance. In a recent classic paper, *Lippmann* [30.4] compared machine and human recognition performance across a wide variety of stimuli. He demonstrated that human performance far exceeded machine performance with minimal linguistic information (digits, letters, nonsense sentences), minimal acoustic information (speech in noise), and various combinations of both (telephone conversations). Table 30.2 shows human versus machine performance for digits, letters, sentences from the *Wall Street Journal*, and telephony conversations. In all four cases, human performance was significantly better than machine performance, sometimes by more than an order of magnitude. Although these numbers were obtained some years ago, one would be hard pressed to argue that more than a factor of two improvement in recognition has occurred in the last 10 years, so the gap is still substantial.

Many studies have suggested that humans can accurately identify words with as little as two seconds of surrounding context [30.5, 6]. *Allen* [30.7] presents strong evidence that humans can highly accurately recognize phonemes without any linguistic context, and also suggests that the enormous robustness of human speech perception across degradations in channel conditions arises from the clever processing of many independent frequency bands in the auditory system. Internal experiments performed at IBM by Jelinek and his associates in the late 1980s suggest that human performance in word prediction from text (the *Shannon Game* [30.8]) is over three times better than the language models then (and unfortunately, still currently) used in speech recognition when humans are presented with full sentential context. No studies seem to exist that assess the relative importance of these information sources, so we initiated a series of tests to try to ascertain the performance of humans on the multidomain test set.

**Table 30.2** Human versus machine recognition word error rates across a variety of tasks (after *Lippmann* [30.4])

| Task | Machine performance % | Human performance % |
|---|---|---|
| Connected digits | 0.72 | 0.009 |
| Letters | 5 | 1.6 |
| Resource management | 3.6 | 0.1 |
| WSJ (Wall Street Journal) | 7.2 | 0.9 |
| Switchboard | 43 | 4 |

### 30.3.1 Baseline Listening Tests

The first set of experiments attempted to obtain baseline performance figures on how well humans could actually recognize the multidomain test set. Random segments were chosen from the test set and played to two listeners over headphones. In one experiment, the listeners were allowed to listen only one time to the utterances, and in the second experiment, the listeners were allowed to hear the utterances multiple times. The utterances were randomized across the test set both in terms of domain and in terms of order. In addition, long utterances were segmented into no longer than 2–3 second chunks, as the memory load for utterances that were longer essentially made the single-pass task intractable.

The summary results are shown in Table 30.3. The error rates are much higher than those presented in [30.4] on similar data. In addition, no appreciable reduction in error rate is seen when the listener is allowed to listen multiple times to the test utterance. The error rates

**Table 30.3** Summary of the multidomain test set base listening test results

| Condition | Word error rate |
|---|---|
| Listen once | 25.6 |
| Listen multiple times | 21.5 |



Amazon EX1010 Page 614

**Table 30.4** Breakdown of base listening test results across corpora (listen-once case)

| Corpus | Word error rate |
|---|---|
| swb98 | 25.1 |
| mtg | 28.1 |
| cc1 | 19.6 |
| cc2 | 32.2 |
| vm | 18.8 |

by corpus were broken down to see if there were any obvious dependencies on the domain materials. The breakdown is shown in Table 30.4. As can be seen, there is no clear dependency on any one particular corpus, though it is clear, for example, that the voicemail data is more comprehensible than the call center data, perhaps because the talkers know that they are leaving a message that is to be listened to after the fact.

Why is there such a discrepancy in the results here and the results reported in [30.4]? There are a number of possible causes. First, the earlier experiments were performed on longer segments of speech, listened to multiple times by each listener. Second, the earlier experiments may not have randomized the segments across speaker. Therefore, the listener may have been able to take advantage of both task adaptation and speaker/channel adaptation in the earlier results. In an attempt to try to tease these issues apart, we embarked upon a more-ambitious series of listening tests, described in the next subsection.

### 30.3.2 Listening Tests to Determine Knowledge Source Contributions

The processes that determine how humans recognize speech are still subjects of ongoing research and will remain so for some time to come. The goal of a speech recognition professional is not so much to understand how humans recognize speech but to try to utilize human strategies of speech recognition as a guide to improve

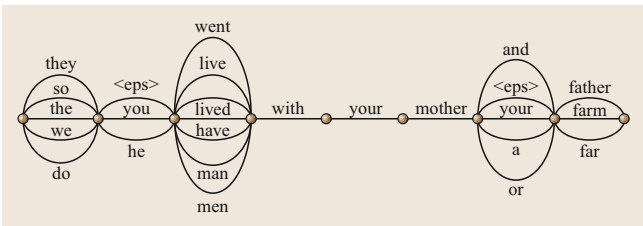



**Fig. 30.1** A typical word confusion network, also known as a *sausage*

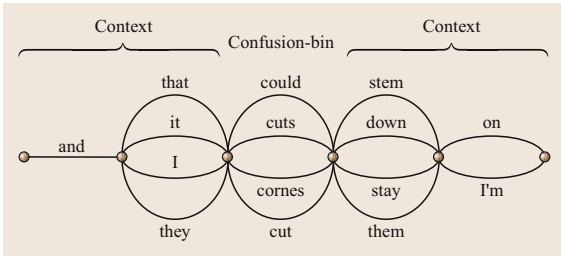



**Fig. 30.2** Word recognition test for the *listening* condition depicting only the audio information and sausage structure presented to the subjects

the performance of today's speech recognition systems. As such, the next set of listening experiments attempt to determine what sources of knowledge humans can use in the context of performance of today's speech recognition systems.

Currently, one of the most interesting aspects of speech recognition systems is that even when they produce relatively high word error rates (e.g., 30%), it is possible for them to produce extremely compact representations of the search space in which the best possible, or oracle, error rate is very low (e.g., 10%). These representations take the form of a confusion network (typically referred to as a *sausage*), as illustrated in Fig. 30.1. In this example, the correct word sequence is *so you lived with your mother and father,* although *do you live with your mother and father* and many other sequences are also possibilities. (The word that was ranked highest by the recognizer appears at the top of each segment.)

In this set of listening experiments, sausage structures produced by a state-of-the-art recognizer were presented to human subjects, and they were asked to select the best word sequence either with accompanying audio (*listening*) or with accompanying long language model context (*comprehension*). The hope was to assess the relative contributions of acoustic and language information as possible information sources on top of a pre-existing speech recognition system.

More specifically, in the listening condition (Fig. 30.2) subjects were presented with audio containing two words to the left, the target word and two words to the right of the word target, and asked to identify the target center word. In the comprehension condition (Fig. 30.3), the subject was given five segments of text history and asked to determine the best path through the sausage network without corresponding audio. In these experiments, we used audio data from the RT03 evaluation data set from LDC [30.9] and from the



Amazon EX1010 Page 615

**Table 30.5** Word and oracle recognition error rates for listening tests to determine relative knowledge source contributions

| Corpus | Word error rate | Oracle error rate |
|---|---|---|
| RT03 | 29.2 | 8.0 |
| MALACH | 28.3 | 9.5 |

**Table 30.6** Listener error rates when presented with audio context in the form of the surrounding words (listening) and word context in the form of sausages (comprehension)

| Corpus | Base recognition word error rate | Listening | Comprehension |
|---|---|---|---|
| RT03 | 29.2 | 24.2 | 30.6 |
| MALACH | 28.3 | 27.3 | 32.0 |

MALACH corpus [30.10]. A summary of the word error rates and oracle error rates for the sausages are shown in Table 30.5. As can be seen, the oracle error rates are much lower than the word error rates, implying that, if there is useful information that humans can utilize in the surrounding context, error rates can be made to drop significantly.

Sausages as described above were presented to a set of 20 listeners in random order with respect to speaker, channel, and topic. Each listener saw or heard 100 sausages either under the listening or comprehension conditions. The results are shown in Table 30.6. As can be seen from the results, neither source of information dramatically improved the listener's ability to choose the correct path through the sausage data. The presence of acoustic information helped somewhat more than textual context, but not in some definitive sense. Subjectively speaking, the task was surprisingly difficult to perform – the conversations had significant sections of quite unintelligible murmuring which may not be critical for comprehension but when segments are selected completely at random may significantly affect the overall error rate. In retrospect, it might have been useful to include a condition in which both long-span acoustic and language model information was included to ensure that humans could perform the task with *infinite* context, and also have a contrastive condition in which the target speaker is fixed across a variety of utterances. There were issues associated with occasional misalignments of the target word text and the underlying audio and some segmentation artifacts, but these did not occur frequently enough to affect the error rates in a significant fashion.

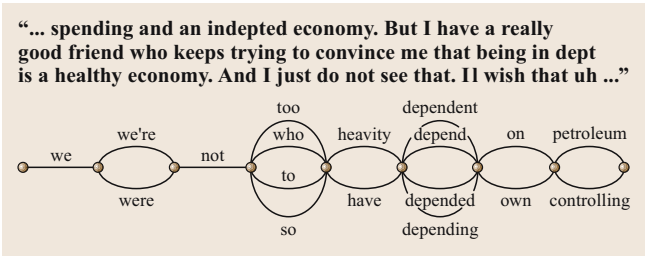



**Fig. 30.3** Word recognition test for the comprehension condition depicting only the text information and the sausage structure presented to the subjects

What these initial experiments illustrate are some of the difficulties in performing human listening experiments as a guide for a technical research agenda in speech recognition. Unfortunately, we were unable to adequately tease out where research efforts should be focused to achieve the maximum benefit, and were left to rely on our technical intuition as to how best to proceed.

## 30.4 Recognition Experiments

To achieve superhuman speech recognition, it is useful to decompose speech recognition into a set of semi-independent tasks. This is shown in Table 30.7. The error rates are rough guides and should not be taken

**Table 30.7** Recognition performance in terms of word error rates for different styles and types of speech

| Task | Speech style | Target | Channel | Word error rate |
|---|---|---|---|---|
| Dictation | Well formed | Computer | Full BW (bandwidth) | < 4 [30.11] |
| Broadcast news | Usually well formed | Spontaneous | Audience | 8.6 [30.12] |
| DARPA communicator | Spontaneous | Computer | Telephone BW | 15.1 [30.13] |
| SWB | Spontaneous | Person | Telephone BW | 15.2 [30.14] |
| Voicemail | Spontaneous | Person | Telephone BW | 27.9 [30.3] |
| Meetings | Spontaneous | People | Far-field | 50 [30.15] |

literally. As can be seen, the main determinants of recognition difficulty are the mode of speech (is one talking to a machine or to a person), and the channel/environment. Needless to say, since people obviously communicate in such modes and across such environments quite freely, a superhuman speech recognizer at a minimum must be robust across these dimensions.

### 30.4.1 Preliminary Recognition Results

In an attempt to get a feeling for the domain sensitivity of current speech recognition systems, a set of experiments was performed. Three corpora were available for training – transaction data [commands, digits, Defense Advanced Research Projects Agency (DARPA) communicator data], switchboard data, and voicemail data. Four corpora were available as test sets – continuous digits, names, switchboard test data (swb98, above) and voicemail test data (vm, above). Systems were trained individually on these corpora. In addition, a common technique – multistyle training – was employed to produce systems in which the training data from the switchboard and voicemail were combined, and one in which all three training corpora were combined. The training and decoding were done as described in Sect. 30.4.2. In all cases, for decoding, a language model suitable for the test data was utilized.

The results are shown in Table 30.8. As can be seen, significant degradation can result when data from one mode of speaking is decoded with models trained on a different mode of speaking. Note also that multistyle training across different corpora to some extent reduces the cross-corpus training degradation effects, but does not always allow one to completely recover the performance levels obtained when test and training data are matched to each other. This implies that blind combination of a huge amount of data from multiple sources is not likely to allow us to reach levels of superhuman performance in cross-domain experiments (and the best way in which to combine language model data has not yet been addressed).

### 30.4.2 Results on the Multidomain Test Set

A conscious decision was made to focus on telephony transcription data as an initial challenge. Therefore, we continued to utilize the multidomain test corpus described in Sect. 30.2. To deal with the broad range of material present in the multidomain test set, we employed a recognition strategy based on multiple passes of recognition interleaved with unsupervised acoustic model adaptation and on a combination of recognition hypotheses from systems using disparate feature sets and acoustic models. We first describe the basic techniques used in the benchmark system for signal processing, acoustic modeling, adaptation, and language modeling, then we describe the architecture of the recognition system and present the performance of the system on the multidomain test corpus at various stages of processing.

#### Signal Processing

The systems in this work use either mel-frequency cepstral coefficients (MFCC) or perceptual linear prediction (PLP) features as raw features. The MFCC features are based on a bank of 24 mel filters spanning 0–4.0 kHz. The PLP features are based on a bank of 18 mel filters spanning 0.125–3.8 kHz and use a 12th-order autoregressive analysis to model the auditory spectrum. Both feature sets are based on an initial spectral analysis that uses 25 ms frames smoothed with a Hamming window, a 10 ms frame step, and adds the equivalent of 1 bit of noise to the power spectra as a form of flooring. Both feature sets also are computed using periodogram averaging to reduce the variance of the spectral estimates. The final recognition feature set for all systems are generated by concatenating raw features from nine consecutive frames and projecting to a 60-dimensional (60-D) feature space. The projection is a composition of a discriminant projection (either linear discriminant analysis or heteroscedastic discriminant analysis [30.16]) and a diagonalizing transform [30.17, 18].

Prior to the projection to the final, 60-D recognition feature space, the raw features are normalized. Three

**Table 30.8** Preliminary test results. The training corpora are listed across the top and the test corpora are represented by the rows

| | Training corpus | | | | |
|---|---|---|---|---|---|
| | Transactions | SWB | Voicemail | Voicemail + SWB | All |
| Names | 4.4 | 6.4 | 8.6 | | 5.3 |
| Digits | 1.3 | 1.9 | 2.4 | | 1.4 |
| SWB | | 39 | 57 | 46 | |
| Voicemail | | 47 | 36 | 37 | |



Amazon EX1010 Page 617

different normalization schemes are used by different systems in this work:

1. utterance-based mean normalization of all features;
2. utterance-based mean normalization of all features except c0 and maximal normalization of c0; and
3. side-based mean and variance normalization of all features except c0 and maximal normalization of c0. In maximal normalization of c0, the maximum value of c0 within an utterance is subtracted from c0 for all frames in the utterance. The estimate of variance is based solely on frames for which c0 exceeds a threshold with respect to the maximum value of c0 in the utterance. This is intended to ensure that the variance is only computed from speech frames.

### Acoustic Modeling

We use an alphabet of 45 phones to represent words in the lexicon. Each phone is modeled as a three-state, left-to-right hidden Markov model (HMM). Acoustic variants of the HMM states are identified using decision trees that ask questions about the surrounding phones within an 11-phone context window (±5 phones around the current one). Systems may employ *word-internal* context, in which variants are conditioned only on phones within the current word, or *left* context, in which variants are conditioned on phones within the current and the preceding words.

The majority of the systems described in this work model the leaves of the phonetic decision trees using mixtures of diagonal-covariance Gaussian distributions that are trained using maximum-likelihood estimation (MLE). Subject to a constraint on the maximum number of Gaussians assigned to a leaf, the number of mixture components used to model a leaf is chosen to maximize the Bayesian information criterion (BIC),

$$F(\theta) = \log P(X_s|s,\theta) - \frac{\lambda}{2}|\theta|\log(N_s)\,, \qquad (30.1)$$

where $P(X_s|s,\theta)$ is the total likelihood of the data points $X_s$ that align to leaf $s$ under model $\theta$, $N_s$ is the number of such points, and $|\theta|$ is the total number of parameters in model $\theta$. The overall size of an acoustic model may be adjusted by changing the weight on the BIC penalty term, $\lambda$. The acoustic models for all recognizers are trained on 247 h of switchboard data and 18 hours of Callhome English data. Early experiments revealed that, after acoustic adaptation, no benefit was obtained by combining other sources of data (such as voicemail data, as described in Sect. 30.4.1) so no additional data was combined with the SWB (switchboard) data.

Two systems described in this work employ alternative acoustic models. One system models leaves using mixtures of diagonal-covariance Gaussian distributions that are discriminatively trained using maximum mutual information estimation (MMIE). In our MMIE training, we collect counts by running the forward–backward algorithm on a statically compiled decoding graph, using beam pruning to constrain the size of the search space [30.19]. This lets us exploit technology developed for fast decoding of conversational speech [30.20] for fast MMIE training as well. The second system models leaves with subspace precision and means (SPAM) models [30.21, 22]. SPAM models provide a framework for interpolating between diagonal-covariance and full-covariance Gaussian mixture models in terms of model complexity and model accuracy. Unlike the diagonal-covariance Gaussian models in this work, SPAM models do not directly use BIC-based model selection.

### Canonical Acoustic Models

We use two feature-space transformations, vocal-tract-length normalization (VTLN) [30.23] and maximum-likelihood feature-space regression (FMLLR) [30.24], in an adaptive training framework to train *canonical* acoustic models. The goal of canonical training is to reduce variability in the training data due to speaker- and channel-specific factors, thereby focusing the acoustic model on variability related to linguistic factors. At test time, the feature-space transforms are estimated in an unsupervised fashion, using results from earlier decoding passes.

Our implementation of VTLN uses a set of 21 warp factors that cover a ±20% linear rescaling of the frequency axis. The VTLN frequency warping is applied prior to mel binning in the feature computation. The VTLN warp factor for a speaker is chosen to maximize the likelihood of frames that align to vowels and semivowels under a voicing model that uses a single, full-covariance Gaussian per context-dependent state. Approximate Jacobian compensation of the likelihoods is performed by adding the log determinant of the sum of the outer products of the warped cepstra to the average frame log likelihood.

The FMLLR transformation is an affine transformation of the features in the final, 60-D recognition feature space that maximizes the likelihood of a speaker's data under an acoustic model. FMLLR is equivalent to constrained maximum-likelihood linear regression (MLLR) [30.24], where the MLLR transform is applied to both the means and covariances of the acoustic model. In the remainder of the paper, we will refer to canon-



Amazon EX1010 Page 618

ical models that use VTLN features as VTLN models and to canonical models that use VTLN features and an FMLLR transformation as SAT (speaker-adaptive trained) models.

### Acoustic Model Adaptation

At test time, we also use MLLR adaptation [30.25] of model means to further adapt the recognition system to the specific speaker and environment. Systems that use diagonal Gaussian mixture acoustic models perform two rounds of MLLR. The first round estimates one MLLR transform for all speech models and one MLLR transform for all nonspeech models, and new recognition hypotheses are generated with the adapted models. In the second round, multiple MLLR transforms are estimated using a regression tree and a count threshold of 5000 to create a transform for a regression class. The system using SPAM models performs a single round of adaptation in which a single MLLR transform for all models and a new FMLLR transform are estimated.

### Language Modeling and Recognition Lexicon Design

The data used to train the language models consist of 3 million switchboard words, 16 million broadcast news words, 1 million voicemail words and 600,000 call center words. For the initial rescoring of the word internal lattices we used a four-way interpolated language model, each of the components being a back-off 3-gram LM (language model) using modified Kneser–Ney smoothing [30.26]. The mixture weights ($0.45 \cdot \text{Swb} + 0.25 \cdot \text{BN} + 0.15 \cdot \text{VM} + 0.2 \cdot \text{CC}$) are optimized on a held-out set containing 5% of each of the training corpora. For the final rescoring of the left-context lattices the 3-gram mixture components are replaced with 4-gram language models, keeping the mixture weights the same. The 34,000-word vocabulary used in our experiments consists of all the high-count words from our training corpora. The pronunciation dictionary consists of 37,000 entries, yielding a ratio of 1.09 pronunciations per word in the vocabulary. Table 30.9 shows the perplexities and the out-of-vocabulary (OOV) rates for each of the five test sets.

**Table 30.9** Perplexities and OOV rates across different test sets

| Test set | Perplexity 3gm LM | Perplexity 4gm LM | OOV rate (%) |
|---|---|---|---|
| swb98 | 94.16 | 90.08 | 0.3 |
| mtg | 146.45 | 142.58 | 0.7 |
| cc1 | 111.69 | 106.42 | 0.3 |
| cc2 | 52.95 | 49.30 | 0.1 |
| vm | 94.66 | 89.32 | 1.1 |

### Recognition Process and Performance

Recognition of data system proceeded as follows:

P1 Speaker-independent decoding. The system uses mean-normalized MFCC features and an acoustic model consisting of 4078 left context-dependent states and 171 000 mixture components. Decoding is performed using IBM's rank-based stack decoding technology [30.27].

P2 VTLN decoding. VTLN warp factors are estimated for each speaker using forced alignments of the data to the recognition hypotheses from P1, then recognition is performed with a VTLN system that uses mean-normalized PLP features and an acoustic model consisting of 4440 left context-dependent states and 163 000 mixture components. Decoding is performed using IBM's rank-based stack decoder. In the cc2 and vm test sets, which have no speaker information, VTLN warp factors are estimated for individual utterances.

P3 Lattice generation. Initial word lattices are generated with a SAT system that uses mean-normalized PLP features and an acoustic model consisting of 3688 word-internal context-dependent states and 151 000 mixture components. FMLLR transforms are computed using recognition hypotheses from P2. The lattices are generated with a Viterbi decoder. The lattices are then expanded to trigram context, rescored with a trigram language model and pruned. In the cc2 and vm test sets, which have no speaker information, FMLLR transforms are estimated for individual utterances.

P4 Acoustic rescoring with large SAT models. The lattices from P3 are rescored with five different SAT acoustic models and pruned. The acoustic models are as follows:

- [A] An MMIE PLP system consisting of 10 437 left context-dependent states and 623 000 mixture components. This system uses maximum-normalization of c0 and side-based mean and variance normalization of all other raw features.
- [B] An MLE PLP system identical to the system of P4A, except for the use of MLE training of the acoustic model.
- [C] An MLE PLP system consisting of 10 450 left context-dependent states and 589 000 mixture components. This system uses mean normalization of all raw features.



Amazon EX1010 Page 619

**Table 30.10** Word error rates (percentage) for the components of the multidomain test set and the overall, average error rate for the corpus. For passes where multiple systems are used (P4–6), the best error rate for a test component is highlighted

| Pass | swb98 | mtg | cc1 | cc2 | vm | All |
|---|---|---|---|---|---|---|
| P1 | 42.5 | 62.2 | 67.8 | 47.6 | 35.4 | 51.1 |
| P2 | 38.7 | 53.7 | 56.9 | 44.1 | 31.7 | 45.0 |
| P3 | 36.0 | 44.6 | 46.6 | 40.1 | 28.0 | 39.1 |
| P4A | **31.5** | **39.4** | 41.7 | 38.2 | 26.7 | 35.5 |
| P4B | 32.3 | 40.0 | **41.3** | 39.0 | 26.7 | 35.9 |
| P4C | 32.5 | 40.2 | 42.1 | 39.9 | 27.0 | 36.3 |
| P4D | 31.7 | 40.3 | 42.6 | **37.6** | **25.8** | 35.6 |
| P4E | 33.0 | 40.5 | 43.4 | 38.8 | 26.9 | 36.5 |
| P5A | 30.9 | **38.3** | **39.4** | 36.9 | 26.1 | 34.3 |
| P5B | 31.5 | 38.5 | **39.4** | 37.0 | 26.5 | 34.6 |
| P5C | 31.6 | 38.7 | 41.0 | 39.4 | 26.8 | 35.5 |
| P5D | **30.8** | 39.0 | 41.1 | **36.7** | **25.6** | 34.6 |
| P5E | 32.1 | 38.9 | 41.8 | 36.8 | 26.4 | 35.2 |
| P6A | **30.4** | **38.0** | **38.9** | 36.5 | 25.7 | 33.9 |
| P6B | 31.0 | 38.3 | **38.9** | 36.4 | 25.8 | 34.1 |
| P6C | 31.2 | 38.4 | 40.1 | 38.9 | 26.3 | 35.0 |
| P6D | **30.4** | 38.6 | 40.8 | 36.3 | **25.5** | 34.3 |
| P6E | 31.5 | 38.5 | 41.6 | **35.9** | 25.7 | 34.6 |
| P7 | 29.0 | 35.0 | 37.9 | 33.6 | 24.5 | 32.0 |

- [D] A SPAM MFCC system consisting of 10 133 left context-dependent states and 217 000 mixture components. The SPAM models use a 120-dimensional basis for the precision matrices. This system uses mean normalization of all raw features.
- [E] An MLE MFCC system consisting of 10 441 left context-dependent states and 600 000 mixture components. This system uses maximum-normalization of c0 and mean normalization of all other raw features.

The FMLLR transforms for each of the five acoustic models are computed from the one-best hypotheses in the lattices from P3. FMLLR transforms are estimated for individual utterances in the vm test set, but on the cc2 test set a single FMLLR transform is estimated from all utterances. The vm test set contains many long utterances [30.28], and the FMLLR estimation procedure has sufficient data, even with very large acoustic models. We found that the cc2 test set contained only very short utterances, and the FMLLR procedure failed to converge on many utterances with the large acoustic models.

P5 Acoustic model adaptation. Each of the five acoustic models are adapted using one-best hypotheses from their respective lattices generated in P4; no cross-system adaptation is performed. As described above, the systems using Gaussian mixture acoustic models are adapted using two sets of MLLR transforms, while the SPAM acoustic model is adapted using an FMLLR transform and an MLLR transform. The lattices from P3 are rescored using the adapted acoustic models and pruned. As in P4, transforms are estimated for individual utterances in the vm test set, but are estimated globally for the cc2 test set.

P6 4-gram rescoring. Each of the five sets of lattices from P5 are rescored and pruned using a 4-gram language model.

P7 Confusion network combination. Each of the five sets of lattices from P6 are processed to generate confusion networks [30.29], then a final recognition hypothesis is generated by combining the confusion networks for each utterance.

The performance of the various recognition passes on the test set is summarized in Table 30.10.

### Conclusions

Reasonable recognition performance can be obtained on a broad sample of conversational American English tasks using acoustic models trained only on switchboard

and Callhome data. The results on the mtg set illustrate this point most strongly, for neither the acoustic models nor the language models are trained on meeting data. This supports the observation that the switchboard corpus is representative of the acoustic–phonetic and stylistic properties of conversational American English [30.30].

Multipass decoding with unsupervised adaptation and a combination of disparate systems are effective techniques for achieving good recognition performance on diverse data sources. On this test set, they can reduce the overall error rate from 51.1 to 32.0%.

While system combination can provide consistent gains in recognition performance, they are relatively small for the rather substantial amount of computation incurred. Had we used only the MMIE PLP system and performed consensus decoding instead of the confusion network combination in P7, the overall error rate on the test would have increased to 33.1%. This rather intensive amount of computation is a clear impediment to research and resulted in a major redesign of the system as described in the next section.

### 30.4.3 System Redesign

As described above, the main problem with the above system was its high level of complexity. Too many passes over too many systems are required – the above system ran in about 5000×RT (real-time), which is really outrageous. If one looks at the culprits in the computation, it is primarily the need to generate many sequences of lattices inefficiently because of an inability to handle all the recognition sources of knowledge in a single pass. For this reason a major redesign of the recognition system was performed to allow for all the knowledge sources needed for recognition to be compiled into a single static network, and many of the multiple passes used to determine warping factors and FMLLR transforms were combined into a single pass or eliminated altogether [30.20]. It was found that this combination not only sped up decoding enormously but also resulted in much better accuracy. Essentially, each pass of lattice generation and pruning was introducing search errors that were eliminated by migrating to a single-pass decoding process. Another innovation that was added was incorporation of a dynamic language model that could automatically modify the interpolation weights in a two-pass recognition process. Each utterance was decoded with a general language model and a lattice was produced. The interpolation weights of the LM components were then adjusted to minimize perplexity of the decoded utterance text, and the new LM that resulted was used to rescore the lattice to produce the final recognition result.

**Table 30.11** Recognition results on the multidomain test set after system redesign

| Corpus | Previous | After redesign |
|---|---|---|
| swb | 29.0 | 27.0 |
| mtg | 35.0 | 33.6 |
| cc1 | 33.6 | 24.7 |
| cc2 | 37.9 | 36.9 |
| vm | 24.5 | 20.9 |
| Average | 32.0 | 28.6 |
| Number of recognizers | 5 | 1 |
| Speed | 5000×RT | 20×RT |

The results are shown in Table 30.11. As can be seen, not only has the recognition accuracy significantly improved, but the amount of computation dramatically has also decreased. The need for multiple recognizers is essentially eliminated and even the speed of a single recognition pass significantly dropped. Analysis indicated that approximately 2% absolute of the performance gain was attributable to the presence of the adaptive language model.

### 30.4.4 Coda

After the above set of experiments were completed, a shift in emphasis to focus on public data, such as those present in the DARPA EARS (effective, affordable, reusable speech-to-text) and GALE (global autonomous language exploitation) programs, took place. The 2004 EARS system [30.14] was basically an extension of the system described in Sect. 30.4.3 to three systems (speaker independent, speaker adaptive with diagonal covariances, and speaker adaptive with full covariances), all trained with MPE (minimum phone error) [30.31] and fMPE [30.32]. In addition, significantly more data (2000 h) were used to train the models. Unfortunately, as described in Sect. 30.4.3 the signal processing needed to process the superhuman test set was a modified version of the EARS system signal processing to improve robustness, so it was only possible to decode the swb data component with the EARS system. The result was a 19% word error rate compared to a 27% word error rate on the swb component of the system described in Sect. 30.4.3. Approximately half of the improvement could be attributed to increased data and the rest divided across the new acoustic modeling (MPE and fMPE) and



Amazon EX1010 Page 621

the use of multiple systems. Assuming that at least some of these improvements would have held up across the other data components, it is safe to say we would have been looking at error rates between 20% and 25% on the overall superhuman corpus. Given the results of the human listening tests (Sect. 30.3.2), at least on short segments of speech, one could almost say that for telephony speech transcription and limited acoustic context, superhuman performance no longer seems completely out of reach.

## 30.5 Speculation

Although the above sections suggest some progress has been made in the direction of achieving superhuman speech recognition, it is clear that there is still a long path ahead. First, system performance even on a subset of domains (telephony transcription) – say at best 20% WER (word error rate) – is still a good factor of three or four away from human performance on this task – best reported results being about 5%. Second, except for the preliminary results described in Sect. 30.4.1, no serious attempts have yet been made to develop a single system simultaneously capable of handling transactional and transcription data. Lastly, the robustness of today's speech systems with respect to varying channels and noise is still weak. Is it possible, then, to achieve superhuman speech recognition by plugging away at the essentially incremental approaches that have typically resulted in incremental gains on the order of 10–20% a year that have been described in the rest of the paper, with no major insights from human performance?

Today's speech recognition systems have all evolved to have the same basic structure: spectral features are periodically sampled at the frame level from the speech signal, and the probability distribution of these feature vectors is captured by hidden Markov models (HMMs). The HMMs are combined with a language model of the short-term dependencies of word sequences, and the word hypotheses are produced by various efficient dynamic-programming-based search mechanisms [30.33]. Many alternatives and enhancements have been tried over the years: articulatory models, neural networks, segment models, trigger language models, structured language models, but all have either failed or not shown significant advantages over this basic structure. In addition, regular significant improvements have always resulted over the years via refinements to the basic structure, placing a successively increasing barrier to entry on alternative formulations.

There is no question that advances in speech recognition will still continue via incremental techniques – one would be foolish in the face of essentially 30 years of incremental improvements to deny that progress has not been continually made. However, this does not rule out exploration of alternative methodologies in parallel or on top of existing high-performance recognition systems. In the remainder of this chapter, some speculation is presented both about some promising incremental approaches with good payoffs, as well as some more-dramatic changes in speech recognition methodologies.

### 30.5.1 Proposed Human Listening Experiments

The listening experiments described above only scratched the surface in terms of trying to tease out what the most useful research directions in speech recognition might be. An extension of the above described experiments is the following.

1. Word sequences are generated automatically from a trigram LM, and then read aloud.
2. The sentences are decoded with an ASR (automatic speech recognition) system, and the WER is computed.
3. Noise is added to the to the recordings until human listeners have the same WER, i. e., the acoustics are calibrated to equalize human and machine performance.
4. Meaningful sentences are recorded, read, and decoded by the ASR system.
5. The same level of noise is added to the same meaningful sentences, which are then transcribed by human listeners.

When meaning is added to the sentences, there will be a differential impact on human and machine performance, presumably with humans benefiting more. This difference will provide a meaningful measure of the amount that can be gained by moving beyond trigram LMs and using broad syntactic, semantic, and pragmatic knowledge sources.

Yet another problem is that there is a mismatch between the metrics that are used to measure ASR performance, and the intended use of these systems. Performance is measured with word error rate, which is dominated by errors in function words. For example, the



Amazon EX1010 Page 622

five most common errors in a typical large-vocabulary recognition system are:

1. and/in
2. it/that
3. was/is
4. that/the
5. the/a

While these words are very common, it is not clear that recognizing them is always critical to the performance of a real application. For example, in a typical call-center application, a user might call to make a travel reservation and say something like *I want to fly to Boston on Tuesday.* What the system needs to produce is a representation something like (what = reservation; to = Boston; date = Tuesday; from = ????) and realize that an appropriate response is something like *Where are you flying from?* It does not matter, however, if there are minor recognition errors like *I want to fly to Boston at Tuesday,* or even more serious ones like *I went to fly to Boston on Tuesday,* as long as the system is able to recover the underlying meaning with confidence. In order to understand the relationship between word error and concept error, we propose to conduct a final set of experiments to flesh out this relationship.

One such experiment involves revisiting the confusion network experiments described earlier, but in the context of question-answering. In this experiment, subjects are shown confusion networks and then asked to answer questions. If they are able to do this on the basis of the sausages, then we are warranted in taking confusion networks – without further acoustic processing – as the basis for high-level linguistic processing, regardless of WER.

A second experiment revisits the notion of *calibrated acoustics*, but again in the context of question-answering. Here, increasing noise is added to utterances, and humans are asked to both transcribe the utterances and answer questions. Presumably, people will be able to answer questions even after the WER of their transcripts has degraded considerably, indicating that WER is not a good measure of the information transmitted, and suggesting the use of concept error or slot-filling error as a better metric.

In summary, by conducting a series of human listening and comprehension experiments, one can gain a better understanding of how to improve our current systems, what the fundamental limitations of different knowledge sources are, and how to measure system performance.

In addition to human listening experiments, it might be possible to establish other fundamental bounds, e.g., of orders of magnitude extensions of data sets for current basic structure language models. For instance, it is possible to use a Google query interface [30.34] to check the frequency of word sequences appearing in the lattice or *sausage*, i. e., to see if and how often corresponding 5-grams, 4-grams, 3-grams and 2-grams appear in the Google index. Such experiments would help to answer questions about the value of having four orders of magnitude more data for language modeling.

### 30.5.2 Promising Incremental Approaches

#### Modeling Spontaneous Speech

The main problem with spontaneous speech is that it contains substantial segmental and suprasegmental variations not present in read speech (which, of course, is much easier to collect in bulk), such as pitch and duration variations, hesitations, false starts, ungrammatical constructs, emotionally expressive speech (laughter, etc.). The high levels of human performance that can be obtained from just auditory presentation of speech out of context [30.4] suggest that the only clues that are needed to decipher speech lie in the local speech signal. In extremely noisy environments, other cues, such as visual ones, may be required. Consequently, if the WER performance is bad, it must be the case that

1. the stochastic models used to model the observation are not accurate;
2. current feature-extraction techniques (mel cepstra) do not extract all the necessary information;
3. the language model is inadequate; or most likely,
4. all of the above.

Most of the speech recognition systems today use frame-level observations. Though some work has been done on suprasegmental feature extraction [30.35], it has enjoyed only a limited amount of success. One possible reason is because neither frame-level observations nor suprasegmental observations are by themselves sufficient to do an adequate job of modeling (spontaneous) speech. Segmental models [30.36, 37] make fewer assumptions than HMMs and provide a better modeling framework by modeling sequences of observations rather than individual observations; however, the goal of multiscale modeling requires a framework that is more powerful than segmental models. The end objective of any model is to compute the joint probability of all observations. This joint probability can be expressed as a product of several conditional probabilities



Amazon EX1010 Page 623

– however our goal is to make this factorization different for different observations. Graphical models provide a mechanism whereby different factorizations of a joint distribution can be specified by means of a directed graph [30.38].

To model the statistical dependencies in the extracted features a graphical model that specifies the factorization of a joint distribution by means of a graph can be used. This framework can be used to model arbitrary dependencies, including temporal dependencies in different observation streams. Appropriate dependencies can be selected via correlation or mutual information techniques [30.39, 40]. In addition to these statistically identified dependencies, linguistic knowledge related to the speech production process can also easily be incorporated into the graphical model – for instance, the formant frequency estimates at a given time frame could provide valuable information about the identity of the phone at the current and adjacent time frames. Also, constraints on the formant trajectories between adjacent time frames (for continuity reasons) can easily be incorporated into graphical models.

Such a scheme was recently tried [30.41] and demonstrated improvements on the switchboard corpus. A particularly attractive aspect of such a model is that it can easily be augmented with additional suprasegmental features, such as say pitch, which might prove valuable for modeling syllables and words in tonal languages such as Mandarin, or for better spotting of disfluencies.

### Multi-Environment Systems

Another objective in achieving human performance is to develop a system that will work on several different types of speech: full bandwidth, telephone bandwidth, or recorded with close-talking or far-field microphones. One possibility is to bandlimit all sources of speech data and train a system using this data, however, this would provide inferior performance for cases where full-bandwidth or low-noise speech is available. Another possibility is just to run multiple models in parallel with a switch that detects bandwidth changes, as is done in many of today's systems that process broadcast news, but that approach completely fragments data into small pieces, generating a complex multicomponent system that is very hard to manage. Consequently, the goal is to either: (i) develop a single modeling framework that can make use of all the information in full-bandwidth speech when it is available, and can also deal with bandlimited speech, or (ii) investigate features that are insensitive to bandwidth-related distortions.

A possible modeling framework for (i) is to partition the observation vector into two components, one of which is always available, $o_{1,t}$, while the other may be hidden, $o_{2,t}$. The formulation is based on the fact that, if the global joint statistics of $o_1$ and $o_2$ are known, and for each class the probability density of $o_1$ and $o_2$ are known, then it is possible to predict the missing observation from the joint statistics.

The basic speech recognition problem is to find the joint probability of the observation sequence $o_{1,1} \cdots o_{1,T}$, and, when it is available, $o_{2,1} \cdots o_{2,T}$. As the observation probability is conditioned on a state sequence $s_1^T$ that is hidden, this may be written as

$$\begin{aligned} p\left(o_{1,1}^T o_{2,1}^T\right) &= \sum_{s_1^T} p\left(o_{1,1}^T o_{2,1}^T, s_1^T\right) \\ &= \sum_{s_1^T} p\left(o_{2,1}^T / o_{1,1}^T, s_1^T\right) p\left(o_{1,1}^T / s_1^T\right) p\left(s_1^T\right) \end{aligned} \tag{30.2}$$

If $o_{2,1}^T$ is observable, we can approximate the term $p(o_{2,1}^T / o_{1,1}^T, s_1^T)$ by $p(o_{2,1}^T / s_1^T)$, and if $o_{2,1}^T$ is not observable, we can approximate it by $\hat{o}_{2,1}^T$, where $\hat{o}_{2,1}^T = p(o_{2,1}^T / o_{1,1}^T)$, and the joint statistics of $o_{1,1}^T, o_{2,1}^T$ are used to compute the latter term. (For a Gaussian joint distribution, $\hat{o}_2$ turns out to be a linear operation on $o_1$.) Hence, the global joint statistics of $o_1, o_2$ are used to predict the value of $o_2$, making it possible to use information related to the probability distribution function (PDF) of $o_2$ for the class $s$, even when $o_2$ is not observed.

A generalization of this approach was successfully used in [30.42] to improve robustness for speech in very high-noise environments. In such noise, large sections of the time–frequency display for a speech signal are often masked by the noise, not just the higher frequencies as in telephony speech. It was shown that variations on the above technique could improve the effective signal-to-noise (S/N) ratio by more than 10 dB in certain cases. However, the technique has not yet been applied to bandlimited speech.

Possible frameworks for (ii) include the following. [30.40] investigated the utility of a *spectral-peak* feature that is closely related to the formant frequencies, and initial experimental results indicate that the spectral peak features definitely carry information that can help speech recognition even in clean conditions. These initial experiments were based on simple-minded *feature fusion* of the spectral peak features with the cepstra, and going beyond this simple approach to the more-sophisticated multiscale graphical models described above, which en-



Amazon EX1010 Page 624

able trajectories to be modeled in the formant space, will lead to powerful and robust models for speech recognition.

Another approach to (ii) is to adapt the models or features to the new environmental conditions. In addition to popular techniques such as MLLR and FMLLR, and more recently, fMPE, another approach is to use a nonparametric mapping approach for the cumulative probability distribution function [30.43]. Assume that it is possible to transform the adaptation features $x' = g(x)$ in such a way that the cumulative distribution function (CDF) of the transformed adaptation data is made identical to the CDF of the training data. This criterion is used to guide the design of the transformation. Falling as it does into the category of *data transformation* techniques, this method is independent of the modeling framework that is used; consequently, it can be easily incorporated into the segmental graphical model framework suggested earlier. In [30.44], this method was applied to the speech samples for the application of speaker identification; here this method is extended to deal with multidimensional observations that are used in speech recognition.

This approach is motivated by the following reasoning. It is well known that the assumed parameterization of the true PDF of the data is often inaccurate. Consequently, if we could deal with the empirically observed CDF of the data, there would be no need to make any modeling assumptions. Secondly, the CDF is a well-behaved function (monotonic and lying between 0 and 1), consequently it is relatively easy to define a mapping of the feature dimension that equates two CDFs. Thirdly, this method is computationally much simpler than most other adaptation schemes. Most adaptation schemes require the adaptation data from the test environment to be transcribed, with an associated overhead. The CDF matching scheme on the other hand does not require any prior transcription, and the process of computing the nonlinear transformation is also relatively inexpensive.

For the case of multidimensional features, if the dimensions are independent, then the CDF-matching technique can be applied to each dimension independently. However, as this is not generally the case, it is necessary first to transform the feature into a space where the dimensions are uncorrelated. This needs to be done simultaneously for both the training and adaptation data. The solution to this problem is a transformation consisting of the generalized eigenvectors of the covariance matrices of the training and test data.

This technique has been been used to compensate for the mismatch between landline and speakerphone telephone data. In Fig. 30.4, the CDF for the first cepstral dimensions for landline and speakerphone data is shown, with the corresponding mapping. In preliminary experiments, the CDF mapping technique improved performance on speakerphone data by 30% relative [30.43].

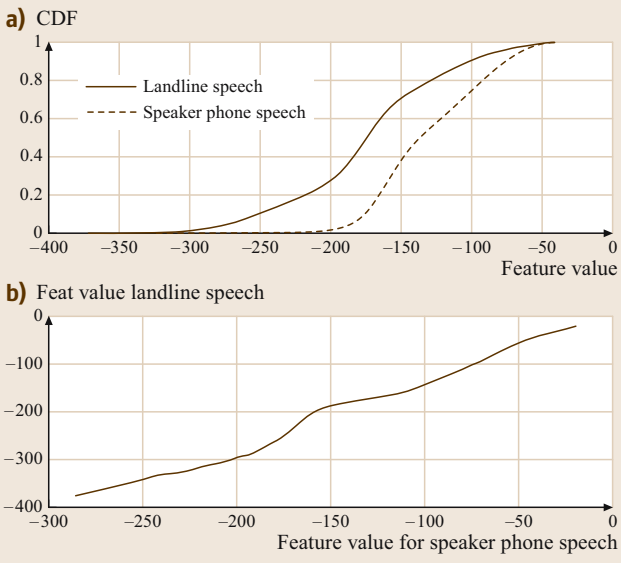



**Fig. 30.4a,b** CDF matching for landline and speakerphone features. **(a)** CDF functions for landline and speakerphone data, **(b)** the input–output mapping

### Improved Language Modeling

In the last few years, a number of research results by IBM [30.45, 46] and other research groups [30.47–49] have appeared that strongly suggest that linguistic information, such as meaning and language structure, can be used to improve speech recognition performance significantly. The scale of these research activities and experiments is limited and recognition experiments have tended to be performed only on narrow domains. However, the improvements are encouraging. For example, the semantic structured language models [30.45, 46] shown in Fig. 30.5 have reduced WER by 20%, and increased confidence by 30–60% for multiple ASR task domains, such as air travel, finance, medical and military domains, compared to when no semantic structure information was used. In such a language model, the $N$-best hypotheses are first produced by the recognizer. For each hypothesis, the recognized words are then mapped onto a set of *classes*. Then, a semantic parse of the classed words in each hypothesis is produced, and a maximum-entropy model is constructed in which the word $w_j$ is



Amazon EX1010 Page 625

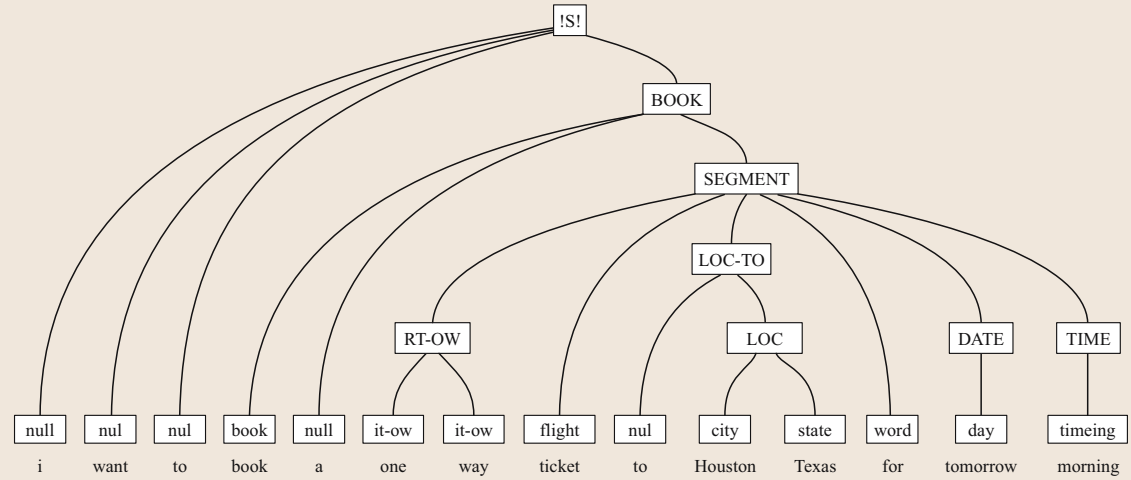



**Fig. 30.5** A semantic structured language model. The words are shown at the *bottom* of the parse tree. The next level up represents the output of the word classer, and the higher levels represent the parse tree constituents. At the bottom is shown a text-based bracketed representation useful for determining language model dependencies (see text)

predicted as

$$p\left(w_j|W_1^{j-1}\right) = p(w_j|w_{j-1}, w_{j-2}, p_j, g_j, c_j)\,, \tag{30.3}$$

where $p_j$ is the parent constituent of the current word, $c_j$ is the last complete constituent, and $g_j$ is the constituent identity of the highest-level concept below the sentence level in the semantic parse tree. For example, in Fig. 30.5 the parent constituent of *tomorrow* is *DATE*, the last complete constituent for *morning* is *DATE*, and the highest level concept for *morning* is *BOOK* (note that the lower boxes are the class assignments and not the parse tree constituents). The resultant language model is then used to rescore the $N$-best hypotheses.

In these experiments, only a small number of linguistic features, among the almost unlimited number of them, have been investigated for speech recognition. Typically, they are restricted to extracting syntactic or semantic attributes, and fitting them into the basic structure of speech recognition, i. e., adding semantic or structural attributes into a trigram or bigram language model and computing the new perplexity or measuring the resulting WER. These research activities are mainly in the area of implementation or engineering, rather than in developing new computational models that can naturally marry automatic speech recognition and computational linguistics. Only the latter can be expected to result in dramatic ASR performance improvements.

Such efforts should come naturally, as the types of information, i. e., acoustic (mainly statistical) and linguistic (mainly structural), are complementary. Even though creating complete new models for speech recognition is difficult, there are examples of leveraging the combined power of statistical and structural information – namely, in the field of natural language processing. Taking insights from such areas of computational linguistics such as word sense disambiguation, text meaning representation, and syntactic parsing, and combining them with the power of statistical techniques, new models have been created and successfully applied in text analytics, search and classification. Often these models use new knowledge resources, such as the Penn TreeBank, the Brown Corpus, WordNet, and the extended WordNet.



Amazon EX1010 Page 626

Based on these observations, two parallel lines of research are suggested in order to incorporate semantic and syntactic knowledge into ASR:

1. Investigate the use of a larger number of linguistic features, such as lexical features, semantic and syntactic relation and structural features, morphological features, part–whole relationship features, and experiment with more task domains of ASR. Gradually extend to broad domains of ASR. These will test the broad validity of the claims about the importance of semantic and structural information as well as help to find the set of useful features.
2. Investigate new computational models for speech recognition in which natural language structures and linguistic features are naturally integrated in one unified framework. The direct model proposed by IBM [30.50] is one possible approach. In this framework, a model is constructed in which the probability of a phonetic state, word, or sentence is computed as a direct function of the underlying knowledge sources including both linguistic and acoustic information. Figure 30.6 depicts such a model in which the probability of the next state of the model is a function of the word, the sentence, and the acoustic observation. This is contrasted with current practice, which uses a generative model to compute the probability of a set of features from an underlying word or sentence in conjunction with an independent language model. The new models could be used as one of the specialists in the doctor/specialist framework described below.

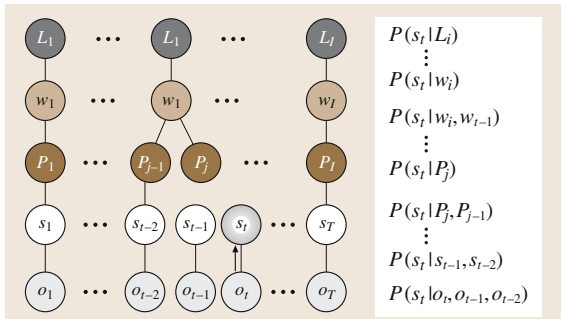



**Fig. 30.6** A direct model for automatic speech recognition. Here, the probability of the state is directly conditioned on the associated phoneme, the word, the language, and the observations

## 30.5.3 Promising Disruptive Approaches

### The Doctor/Specialists Paradigm

State-of-the-art ASR systems adopt a monolithic approach to acoustic and linguistic knowledge sources. For example, the same acoustic features and model are used to distinguish all acoustic confusions [30.33]. It is quite remarkable that this approach to ASR, chosen for its simplicity and lack of a better alternative, works so well in practice. However, it is also clear that for distinguishing particular acoustically confusable units, e.g., *a* versus *the*, a specialist classifier trained only to distinguish between the confusable units will perform significantly better. One solution is the design and development of a *doctor/specialists* framework for ASR. The basic idea is as follows. The doctor, who is responsible for the overall recognition process, will partition the speech into regions of certainty and regions of confusion, which can be done with the current state of the art. In regions of confusion, inputs from specialists (acoustic, linguistic, and visual) who are experts at distinguishing between particular sets of words or phonemes will be invoked and integrated in a systematic manner.

A key ingredient in ASR improvements over the past decade has been the inclusion of increasing amounts of contextual information – both acoustic and linguistic, and the framework above can be viewed as a significant step towards exploiting larger contextual information. The doctor uses all the contextual information available at a given point to decide on what the next local problem to be resolved is (e.g., is it *f* as in *fine* or *sh* as in *shine*) and then consults the best specialist for this problem. This approach to ASR may be significantly better than the current approach for the following reasons. First, specialists can use more-complex and discriminative models than those in use today [30.51, 52]. Specifically, for the specialists one can replace the generative (HMM-based) models in use today with the discriminative models that have been hard to use in ASR because of the large number of acoustic classes being modeled simultaneously. Secondly, specialists need only be trained in regions difficult for the doctor. Thirdly, specialists can focus on novel acoustic, linguistic and visual (when available) features specific to the confusions they have to resolve. Finally, because of the smaller size of the problems they tackle, specialists can use computationally and memory intensive template-matching approaches for resolving confusions. This methodology may also entail using orders of magnitude more acoustic training data than used by current systems (e.g., 100 000 h of speech). Some initial promising results on such a technique has been obtained in [30.53], where



Amazon EX1010 Page 627

the term *acoustic codebreaking* is used to describe the doctor/specialists procedure.

In principle this doctor/specialists approach outlined requires a completely new search strategy (doctor) and associated modeling strategies (specialists). In practice, we believe this strategy is well suited for implementation in the context of the current state of the art. Current ASR systems can generate confidence-weighted word lattices or sequences of word confusion sets (or sausages, see Fig. 30.1) [30.29]. The oracle word error rate on lattices and/or sausages can be significantly lower than the one-best word error rate, e.g., 10% versus 30%. This implies that resolving confusions in lattices or sausages will result in a dramatic gain in accuracy. To exploit this one would require

- research on the construction of acoustic, linguistic, and visual specialists to distinguish between phonemes and/or words that typically co-occur in sausages
- research on when to invoke and how to integrate input from the specialists

This manifestation of this paradigm naturally decouples the recognition process into two tasks, each of which can be the thrust of independent research programs:

- the generation of minimal size lattices and/or sausages with zero oracle word error rate, and
- the choice of the one-best word hypothesis in lattices and/or sausages with the help of specialists adapted to the particular confusions present in a given lattice or sausage.

### Large-Scale Information Fusion

The current state of the art in speech analysis operates in a highly stylized way: the waveform is processed in 25 millisecond chunks – one every 10 milliseconds – regardless of the acoustic conditions, and linguistic knowledge is captured at the level of word $N$-grams only. It is known, however, that in animal auditory processing there are hundreds of cells and neural regions that are specially adapted to respond to specific acoustic and linguistic stimuli [30.54], and that somehow these features are combined on an extremely large scale to produce speech perception. Drawing loosely on this analogy, another disruptive approach to speech recognition is to develop methods for extracting and combining extremely large numbers of complimentary acoustic and linguistic features. Figure 30.7 illustrates some of the acoustic aspects of this paradigm.

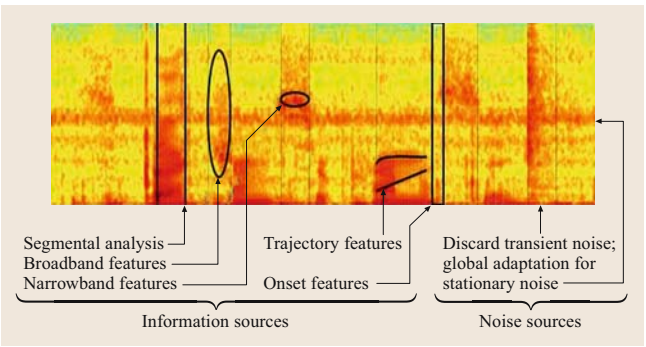



**Fig. 30.7** Multiscale and variable-rate features

Here, the spectrum corresponding to a word has been segmented into variable-length components, both vertically and horizontally. The first oval represents a broadband feature measuring the energy across a wide range of frequencies. The second oval, in contrast, is a narrowband feature, while the dark rectangle at the right measures the rate of change of energy across all frequencies, resulting in an onset detector. The individual curved lines track the formants of one syllable, and their temporal evolution results in two trajectory features. The long horizontal band of orange in the middle of the figure is hypothesized to be noise, and adaptation is used to remove its effects. Once a large number of features has been defined and evaluated, it still remains to combine them in a coherent probabilistic model, and here the maximum-entropy approach is ideally suited. The values of arbitrary numbers of both acoustic and linguistic features can be combined in a theoretically sound fashion to produce posterior estimates of word probabilities.

Although conceived as using far more features, and combining them in a less-supervised way, this model is closely related to the doctor/specialists model. The integrating probabilistic model, e.g., maximum entropy, acts essentially as a word-level doctor and combines features to determine word identity on a word-by-word basis. The feature values represent specialist outputs, and maximum entropy provides a framework for weighting them.

### Large-Scale Decoder Combination

Recent competition-grade ASR systems have begun to combine the outputs of several systems in a voting process [30.55]. In this strategy, a system actually consists of four or five independently constructed ASR system. Each of these is handcrafted to differ from the others in some minor details, for example the use of MFCC as opposed to PLP features, or the set of phonetic units



Amazon EX1010 Page 628

that is used. Typically these systems make somewhat uncorrelated errors, and therefore a voting strategy tends to converge on the correct answer. Often, the systems represent a failed attempt at building a better single decoder that results simply in something that makes equally many but different errors.

This basic strategy has been proven to create gains, but ignores one of the major developments in machine learning – the idea, termed boosting, that hundreds or thousands of systems can actually be manufactured and combined in a fully automated way so as to create mathematically provable bounds on the error rate of the resulting composite [30.56]. In this framework, a sequence of classifiers is constructed such that at each iteration the latest classifier tends to correct the errors of the previous ones. This is done by maintaining a carefully selected weight on each of the training examples so that it is possible to prove that, as long as a better-than-chance classifier can be built at each iteration, the error rate will drop to zero in an exponentially decreasing fashion. On the surface this may appear to generate a system of extreme complexity: a 40 dB complexity increase relative to the system described in Sect. 30.4.2. The key challenge is to develop a set of systems automatically that all have a similar structure whose overall management is similar. An initial start in this direction was described in [30.57], in which a set of similar systems were generated from the same data by randomizing the data used to initialize the state context acoustic decision trees. Significant recognition improvements were demonstrated on the switchboard task by combining up to six similar randomized systems. This can be generalized to other components of the system – say the signal processing and language models – to generate many parallel systems. Of course, boosting is not a panacea and a major issue will be how to generate appropriate sets of complementary systems to be combined.

In order to achieve such goals, the computing power of large clusters of workstations – using new architectures such as Blue Gene [30.58] and Cell [30.59] – can be used to build and combine decoders on a scale that has not been conceived before. Doing this will require full parameterization of the process of building a decoder (within the basic structure) so that system builds that utilize different subsets of the training data will be able to span the necessary space of acoustic and language models. Perhaps in the end this is the most promising approach of all, insofar as it begins to address building systems that even approach the complexity and processing power of the human brain.



## References


30.1 J.G. Fiscus, W.M. Fisher, A.F. Martin, M.A. Przybocki, D.S. Pallett: 2000 NIST evaluation of conversational speech recognition over the telephone, Proc. 2000 Speech Transcription Workshop (2000)

30.2 A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, C. Wooters: The ICSI Meeting corpus, Proc. ICASSP, Vol. I (2003) pp. 364–367

30.3 M. Padmanabhan, G. Saon, J. Huang, B. Kingsbury, L. Mangu: Automatic speech recognition performance on a voicemail transcription task, IEEE Trans. Speech Audio Process. **10**(7), 433–442(2002)

30.4 R.P. Lippmann: Speech recognition by machines and humans, Speech Commun. **22**(1), 1–15 (1997)

30.5 I. Pollack, J.M. Pickett: The intelligibility of excerpts from conversation, Lang. Speech **6**, 165–171 (1963)

30.6 E. Chang, R. Lippmann: Improving wordspotting performance with artificially generated data, Proc. ICASSP, Vol. 1 (1996) pp. 526–529

30.7 J.B. Allen: How do humans process and recognize speech?, IEEE Trans. Speech Audio Process. **2**(4), 567–577 (1994)

30.8 C.E. Shannon: Prediction and entropy of printed English, Bell Syst. Tech. J. **30**, 50–64 (1950)

30.9 NIST Speech Group: The Rich Transcription Spring 2003 (RT-03S) Evaluation Plan, Version 4 (2003) http://www.nist.gov/speech/tests/rt/rt2003/spring/docs/rt03-spring-eval-plan-v4.pdf

30.10 W. Byrne, D. Doermann, M. Franz, S. Gustman, J. Hajič, D. Oard, M. Picheny, J. Psutka, B. Ramabhadran, D. Soergel, T. Ward, W.-J. Zhu: Automatic recognition of spontaneous speech for access to multilingual oral history archives, IEEE Trans. Speech Audio Process. **12**(4), 420–435 (2004)

30.11 I. Nastajus: http://en.wikipedia.org/wiki/Naturally Speeking (2007)

30.12 P. Woodland, H.Y. Chan, G. Evermann, M.J.F. Gales, D.Y. Kim, X.A. Liu, D. Mrva, K.C. Sim, L. Wang, K. Yu, J. Makhoul, R. Schwartz, L. Nguyen, S. Matsoukas, B. Xiang, M. Afify, S. Abdou, J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, F. Lefevre, D. Vergyri, W. Wang, J. Zheng, A. Venkataraman, R.R. Gadde, A. Stolcke: SuperEARS: Multi-Site Broadcast News System, DARPA EARS 2004 Workshop (2007), http://www.sainc.com/richtrans2004/uploads/monday/EARS BN Super team.pdf

30.13 A. Aaron, S. Chen, P. Cohen, S. Dharanipragada, E. Eide, M. Franz, J.-M. Leroux, X. Luo, B. Maison, L. Mangu, T. Mathes, M. Novak, P. Olsen, M. Picheny, H. Printz, B. Ramabhadran, A. Sakra-

jda, G. Saon, B. Tydlitat, K. Visweswariah, D. Yuk: Speech recognition for DARPA Communicator, Proc. ICASSP, Vol. 1 (2001) pp. 489–492
30.14 H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, G. Zweig: The IBM 2004 conversational telephony system for rich transcription, Proc. ICASSP, Vol. 1 (2005) pp. 205–208
30.15 J. Fiscus: The Rich Transcription Spring 2006 (RT-06S) Evaluation Results (NIST Speech Group, 2007) http://www.nist.gov/speech/tests/rt/rt2006/spring/pdfs/rt06s-STT-results-v7.pdf
30.16 G. Saon, M. Padmanbhan, R. Gopinath, S. Chen: Maximum likelihood discriminant feature spaces, Proc. ICASSP, Vol. II (2000) pp. 1129–1132
30.17 R.A. Gopinath: Maximum likelihood modeling with Gaussian distributions for classification, Proc. ICASSP, Vol. 2 (1998) pp. 661–664
30.18 M.J.F. Gales: *Semi-tied full-covariance matrices for hidden Markov models*, Vol. CUED/F-INFENG/TR287 (Cambridge Univ. Engineering Department, Cambridge 1997)
30.19 J. Huang, B. Kingsbury, L. Mangu, G. Saon, R. Sarikaya, G. Zweig: Improvements to the IBM hub 5e system, Proc. NIST RT-02 Workshop (2002)
30.20 G. Saon, G. Zweig, B. Kingsbury, L. Mangu, U. Chaudhari: An architecture for rapid decoding of large vocabulary conversational speech, Proc. Eurospeech, Vol. 3 (2003) pp. 1977–1981
30.21 S. Axelrod, V. Goel, B. Kingsbury, K. Visweswariah, R. Gopinath: Large vocabulary conversational speech recognition with a subspace constraint on inverse covariance matrices, Proc. Eurospeech, Vol. 3 (2003) pp. 1613–1616
30.22 S. Axelrod, R.A. Gopinath, P. Olsen: Modeling with a subspace constraint on inverse covariance matrices, Proc. Int. Conf. Spoken Lang. Process., Vol. 2 (2002) pp. 2177–2180
30.23 S. Wegmann, D. MacAllaster, J. Orloff, B. Peskin: Speaker normalization on conversational telephone speech, Proc. ICASSP, Vol. 1 (1996) pp. 339–342
30.24 M.J.F. Gales: *Maximum likelihood linear transformations for HMM-based speech recognition*, Vol. CUED/F-INFENG/TR291 (Cambridge Univ. Engineering Department, Cambridge 1997)
30.25 C.J. Leggetter, P.C. Woodland: Speaker adaptation of continuous density HMMs using multivariate linear regression, Proc. Int. Conf. Spoken Lang. Process., Vol. I (1994) pp. 451–454
30.26 S.F. Chen, J. Goodman: An empirical study of smoothing techniques for language modeling, Computer, Speech Lang. **13**(4), 359–393 (1999)
30.27 L.R. Bahl, P.V. deSouza, P.S. Gopalakrishnan, D. Nahamoo, M. Picheny: Robust methods for using context-dependent features and models in a continuous speech recognizer, Proc. ICASSP, Vol. I (1994) pp. 533–536
30.28 M. Padmanabhan, G. Ramaswamy, B. Ramabhadran, P.S. Gopalakrishnan, C. Dunn: Issues involved in voicemail data collection, Proc. DARPA Broadcast News Transcription and Understanding Workshop (1998)
30.29 L. Mangu, E. Brill, A. Stolcke: Finding consensus in speech recognition: Word error minimization and other applications of confusion networks, Computer, Speech Lang. **14**(4), 373–400 (2000)
30.30 E. Shriberg, A. Stolcke, D. Baron: Observations on overlap: Findings and implications for automatic processing of multi-party conversation, Proc. Eurospeech, Vol. 2 (2001) pp. 1359–1362
30.31 D. Povey, P. Woodland: Minimum phone error and I-smoothing for improved discriminative training, Proc. ICASSP, Vol. 1 (2002) pp. 105–108
30.32 D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig: FMPE: Discriminatively trained features for speech recognition, Proc. ICASSP, Vol. 1 (2005) pp. 961–964
30.33 M. Padmanabhan, M. Picheny: Large-vocabulary speech recognition algorithms, IEEE Comput. **35**(4), 42–50 (2002)
30.34 Google Desktop Developer Group: http://www.google.com/apis/ (2007)
30.35 B.E.D. Kingsbury, N. Morgan, S. Greenberg: Robust speech recognition using the modulation spectrogram, Speech Commun. **25**(1-3), 117–132 (1998)
30.36 M. Ostendorf, V.V. Digilakis, O.A. Kimball: From HMMs to segment models: A unified view of stochastic modeling for speech recognition, Proc. IEEE Trans. Speech Audio Process. **4**(5), 360–378 (1996)
30.37 J. Bridle, L. Deng, J. Picone, H. Richards, J. Ma, T. Kamm, M. Schuster, S. Pike, R. Regan: An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition, Final Workshop Report, Center for Language and Speech Processing (The Johns Hopkins University, Baltimore 1998)
30.38 G. Zweig, M. Padmanabhan: Dependency modeling with Bayesian networks in a voicemail transcription system, Proc. Eurospeech, Vol. 3 (1999) pp. 1335–1338
30.39 J. Bilmes: Buried Markov models, Proc. ICASSP, Vol. 2 (1999) pp. 713–716
30.40 M. Padmanabhan: Use of spectral peak information in speech recognition, Proc. NIST Speech Transcription Workshop (2000)
30.41 Ö. Çetin, M. Ostendorf: Multi-rate and variable-rate modeling of speech at phone and syllable time scales, Proc. Int. Conf. Acoust. Speech Signal Process., Vol. 1 (2005) pp. 665–668
30.42 M.P. Cooke, P.D. Green, L.B. Josifovski, A. Vizinho: Robust automatic speech recognition with missing



Amazon EX1010 Page 630

and uncertain acoustic data, Speech Commun. **34**, 267–285 (2001)
30.43 S. Dharanipragada, M. Padmanabhan: A nonlinear unsupervised adaptation technique for speech recognition, Proc. Int. Conf. Spoken Lang. Process., Vol. IV (2000) pp. 556–559
30.44 R. Balchandran, R. Mammone: Non-parametric estimation and correction of non-linear distortion in speech systems, Proc. ICASSP, Vol. II (1998) pp. 749–752
30.45 H. Erdogan, R. Sarikaya, Y. Gao, M. Picheny: Semantic structured language models, Proc. Int. Conf. Speech Lang. Process., Vol. II (2002) pp. 933–936
30.46 R. Sarikaya, Y. Gao, M. Picheny: Word level confidence measurement using semantic features, Proc. ICASSP, Vol. I (2003) pp. 604–607
30.47 J. Bellegarda: Exploiting latent semantic information in statistical language modeling, Proc. IEEE **88**(8), 1279–1296 (2000)
30.48 F. Jelinek, C. Chelba: Putting language into language modeling, Proc. Eurospeech, Vol. 1 (1999) pp. KN–1–KN–4
30.49 I. Gurevych, R. Malaka, R. Porzel, H.P. Zorn: Semantic coherence scoring using an ontology, Proc. HLT-NAACL (2003) pp. 88–95
30.50 A. Likhododev, Y. Gao: Direct models for phoneme recognition, Proc. ICASSP, Vol. 1 (2002) pp. 89–92
30.51 V. Vapnik: The support vector method, Proc. Int. Conf. Artif. Neural Networds (1997) pp. 263–271
30.52 S. Della Pietra, V. Della Pietra, J. Lafferty: Inducing features of random fields, IEEE Trans. Pattern Anal. Mach. Intell. **19**(4), 380–393 (1997)
30.53 V. Venkataramani, W. Byrne: Lattice segmentation and support vector machines for large vocabulary continuous speech recognition, Proc. ICASSP, Vol. 1 (2005) pp. 817–820
30.54 L. Miller, M. Escabi, H. Read, C. Schreiner: Spatiotemporal receptive fields in the lemniscal auditory thalamus and cortex, J. Neurophysiol. **87**, 516–527 (2001)
30.55 J.G. Fiscus: A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER), Proc. IEEE Workshop Autom. Speech Recognition Understanding, Santa Barbara (1997) pp. 347–355
30.56 Y. Freund, R.E. Schapire: Experiments with a new boosting algorithm, Proc. ICML (1996) pp. 148–156
30.57 O. Siohan, B. Ramabhadran, B. Kingsbury: Constructing ensembles of ASR systems using randomized decision trees, Proc. ICASSP, Vol. 1 (2005) pp. 197–200
30.58 IBM Research Communication Dept.: http://www.research.ibm.com/bluegene (2007)
30.59 IBM Research Communication Dept.: http://www.research.ibm.com/cell (2007)



Amazon EX1010 Page 631

# 31. Natural Language Understanding

S. Roukos

We describe several algorithms for developing natural language understanding (NLU) applications. The algorithms include a rule-based system and several statistical systems. We consider two major types of NLU applications: dialog systems and speech mining. For dialog systems, the NLU function aims to understand the full meaning of a user's request in the context of a human–machine interaction in a narrow domain such as travel reservation. For speech mining applications, the NLU function aims at detecting the presence of a limited set of concepts and some of their relations in unrestricted human–human conversations such as in a call center or an oral history digital library. We describe in more detail a statistical parsing algorithm using decision trees for dialog systems and two word-tagging algorithms for speech mining.

31.1 **Overview of NLU Applications** ..... 618
31.1.1 Context Dependence ..... 619
31.1.2 Semantic Representation ..... 619
31.2 **Natural Language Parsing** ..... 620
31.2.1 Decision Tree Parsers ..... 621
31.3 **Practical Implementation** ..... 623
31.3.1 Classing ..... 623
31.3.2 Labeling ..... 623
31.4 **Speech Mining** ..... 623
31.4.1 Word Tagging ..... 624
31.5 **Conclusion** ..... 625
**References** ..... 626

Human language is a system for translating thought into physical output (spoken or written) that enables humans to communicate. Natural language understanding (NLU) covers the area of building systems that recover the meaning of spoken or written utterances, thereby enabling human–machine communication. We define NLU as a transduction from text (including text obtained by automatic speech recognition systems) into a formal semantic representation. The formal representation of meaning (or semantics) has been a challenging problem. Crucially, to mitigate the semantic representation challenge, the domain of discourse in NLU applications is in limited tasks/domains (e.g., flight reservations) to allow for building a comprehensive model of the semantics that need to be modeled by the system to enable effective human–machine interaction. [The Defense Advanced Research Projects Agency (DARPA) has funded pioneering research in the air travel information system (ATIS) project in the early 1990s and a follow-on effort on dialog systems for travel reservation called the DARPA Communicator in the 1998–2002 period.]

The communication situation is not symmetric in human–machine dialog systems: the speaker, who may or may not have a good model of the system's capabilities, has a large range of semantic actions while the system has a much narrower and application-specific repertoire and definitely does not have a *good* model of the speaker's semantic repertoire. However, NLU has been used effectively to build dialog systems that are increasingly replacing human operators in call centers by improving the range of tasks that can be handled without the need of a call-center agent. The effectiveness of these solutions is measured by the automation rate, the fraction of calls that are fully handled by the dialog system; for many call-center applications, an increase of a few percentage points in the automation rate typically justifies a multimillion-dollar NLU development project. In this chapter, we will focus on spoken language systems, which have provided the most fertile grounds for recent research on NLU, even though there has been a few web-based chat interfaces powered by NLU technology that we will not discuss further.



Amazon EX1010 Page 632

## 31.1 Overview of NLU Applications

The art of building NLU systems and spoken dialog systems (the latter to be discussed in more detail in Chap. 35) is in defining a good representation of the semantics of the application that can be used to map a user's utterance into the application semantics. We describe several example applications to illustrate the range of complexity of the required semantics for current NLU systems.

- Password reset: the user calls a system to reset his password for a specific software application. The system prompts the user to give either his serial number or full name; once the system identifies the user, it asks for confirmation for resetting the password; once the user confirms by speaking phrases such as *yes*, *sure*, or *I guess so*, it resets the password. This application and other similar ones such as package tracking (by users providing a tracking number to the system) are fairly simple in the sense that an utterance has a simple meaning. For password reset, the semantics are a list of user names and the *Yes*/*No* values to confirmation questions.
- Call routing: the user may be asked *How may I help you* and the user response is classified into one of many classes (typically tens of classes) that are used to direct the call to various departments (or call-center operators) in a large business. For example, if the user says *I want to check my credit-card balance* the user call is transferred to the credit-card department.
- Mutual funds: the user calls the system to find the closing price of a mutual fund, find balances in his account, purchase or sell shares in a mutual fund, transfer funds from one account to another, and to get a list of the most recent transactions in his account. The mutual fund application is one of many applications in the financial domain that have been deployed such as stock trading and banking.
- Flight reservation: the user calls the system to find available flights, determine the best options, the price, make a reservation, and optionally make car and hotel reservations. This and the previous application represent the prototypical NLU applications that use natural language parsing technology to extract the meaning of a sentence.
- Speech-to-speech (S2S) translation in narrow domains (e.g., medical and travel domains): in the medical domain, the system mediates a conversation between a patient speaking one language and a doctor who speaks another language. The patient's speech is recognized and *understood* by filling in a template such as type of symptom, time of onset of symptom, etc. These templates represent the type of information that the system can translate between the two languages. The completed template is then used to generate a natural language sentence in the target language that is then synthesized to the doctor. A similar template is completed for the doctor's utterances. These speech-to-speech systems, many of which are currently being developed under DARPA's spoken language communication and translation system for tactical use (TRANSTAC) program, rely heavily on the narrow domain of semantics to achieve reasonable performance. (These speech-to-speech systems are to be contrasted with broad-domain S2S systems that rely on broad-domain machine translation from a source language to a target language as is being developed in the European Union TC-Star and the DARPA global autonomous language exploitation (GALE) projects.)
- Speech mining: these are information extraction systems that process human-to-human calls in a call center using large-vocabulary speech-to-text followed by information extraction systems that identify topics and/or specific concepts in customer calls. For example, the systems can monitor if an agent has been following guidelines on thanking customers at the end of transactions. Speech mining solutions are still in their infancy but they are starting to be used in call centers. They rely on new understanding technology based on the information extraction technology; we will give a brief overview of information extraction later in this chapter.

Typically, applications such as password reset and call routing are not referred to as NLU applications since they require a simple mapping of phrases to atomic semantic concepts; the former is a simple map of words and phrases to class labels while the latter uses classifiers that classify utterances using word and phrase features into one of $N$ classes. The term NLU is reserved for those applications that have a much larger range of semantics such as the mutual funds, travel, and S2S medical applications described above. In these typical NLU applications, each utterance may have multiple *concepts* that may be combined into a more-complex request. Natural language parsing technology is used to under-



Amazon EX1010 Page 633

stand the structure of users' requests. We will describe in more detail how to build statistical parsers to extract the meaning of an utterance. Finally, the information extraction applications for speech mining typically work on much longer user utterances (as expected when one is analyzing human–human telephone calls) and extract more local pieces of information. These applications use information extraction technology that does not attempt to understand the whole utterance but rather *smaller* pieces of it. We describe in Sect. 31.2 statistical parsing and in Sect. 31.4 statistical models for information extraction.

### 31.1.1 Context Dependence

The meaning of a speech utterance may depend on the context of the conversation between the user and the machine. For context-independent sentences, the meaning is completely specified by the speech utterance independently of the conversation context. For example, the meaning of the user's utterance, *give me the first nonstop flight from Boston to Austin on Wednesday March 3rd*, is wholly extracted from the actual spoken sentence without regard to context, while the meaning of fragments such as *the second one* or *do you have an earlier flight* or the one word utterance *Chicago* depends on the context of the conversation; in these latter cases, the meaning depends on what the system said to the user in the previous turn of the conversation. *Chicago* will be the *from* or *to* city depending on what the system asked earlier. One approach is to define an ambiguous meaning representation for these utterances that another component, typically called a dialog manager, would disambiguate using the conversational state of the system. Another approach is to provide the statistical parser with some features that describe the state of the conversation; this context can be used to identify the meaning of the next fragment. We will illustrate one example of the latter approach in Sect. 31.2.

### 31.1.2 Semantic Representation

Since we need to map a user's utterance to its meaning using the semantic model of the application, a key step in building an NLU system is to design the semantic language for representing the meaning of the user's utterances. There are many different representations, we discuss one example. This formal language for the possible semantics of the ATIS travel task is based on the NL-MENU language developed by Texas Instruments for the ATIS task. This is a canonical example that exemplifies NLU interfaces to relational database management systems (RDBMS). The formal language is somewhat related to the structured query language (SQL) formal language to RDBMS with specific names that are specified by the various fields in the tables of the application's RDBMS. For example, the NL-MENU language used in the ATIS task has the following example formal *sentences* (bold words are keywords, upper-case words are variables):

- **List Flights from_city** CITY **to_city** CITY
- **List Flights from_city** CITY **to_city** CITY **flying_on** DATE
- **List Flights from_city** CITY **to_city** CITY **flying_on** DATE **departing_after** TIME

The formal language has its own interpreter to map to actual application programming interface (API) calls of back-end systems (that provide flight information or mutual fund account information). In the case of NL-MENU the translation to SQL is straightforward. In some applications, there have been efforts at using predicate logic (in particular, first-order logic) to represent the application semantics but these have not been commercially successful.

In general, application developers define a formal language for the semantics that the backend system can support. The design of the formal language has to take into account the fact that it will be used to interpret the users' natural language sentences. Corresponding to the second formal expression above, the user may have said any one of the following utterances:

- I want to go from Boston to Austin tomorrow;
- Boston to Austin this Tuesday;
- What flights do you have that go to Austin; I am planning to leave from Boston next Saturday.

Therefore, the two ingredients for an NLU system are

- a formal language specification of the application semantics
- a system that maps natural language utterances to their corresponding semantic representation

This latter step is the NLU or interpretation step, and is the main focus of this chapter, where we will discuss how to build the NLU component.

## 31.2 Natural Language Parsing

**Table 31.1** Grammar for understanding sentences in the mutual fund domain

| | | |
|---|---|---|
| S[$1.$2.$3] | → | INTRO? ACTION [$1] NUMBER [$2] of FUND [$3] |
| INTRO | → | I would like to \| I want to \| Please |
| ACTION [$1] | → | get me [BUY] \| buy [BUY] \| sell [SELL] |
| NUMBER [$1] | → | (one thousand [1000] \|one hundred [100] \|ten [10]) shares of? |
| FUND [$1] | → | Vanguard 500 index [VFINX] \| Fidelity contrafund [FCNTX] |

Early approaches to understand a sentence were based on developing grammars to translate a sentence into its meaning. Table 31.1 shows a canonical example using finite-state transducers (with Backus–Nauer form (BNF) notation to define the grammar). We use upper-case words to denote variables (also known as nonterminals) that are the left-hand sides, which are rewritten as the right-hand side (RHS), where the vertical line (|) denotes rewriting alternatives; elements followed by a question mark are optional (zero or one occurrence); the elements between brackets are the output language of the transducer. $1 denotes the first variable in the RHS for the output language, $2 will denote the second output variable, and so on.

In the above sample grammar, the first rule indicates concatenation of the three variables that represent the meaning of the three RHS variables. The third rule states that the meaning of ACTION is the meaning of the matching right-hand side, either $1=BUY or $1=SELL depending on which matches the input. A valid sentence that can be parsed by this grammar is: *Please get me one thousand shares of Fidelity Contrafund*, which would be translated to *BUY.1000.FCNTX*, which can be used to execute the request in a backend system.

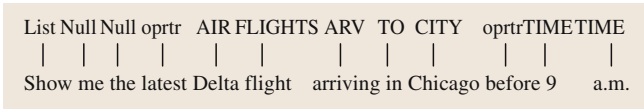



**Fig. 31.1** Example semantic labeling

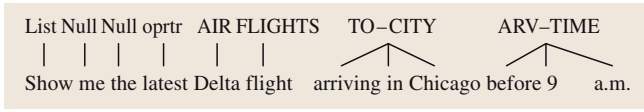



**Fig. 31.2** Example semantic parse tree

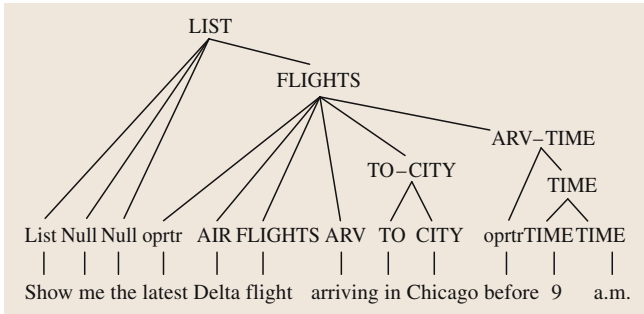



**Fig. 31.3** Semantic parse tree in the travel domain

In these transducers, the grammar developer has to capture both the range of input expressions of users and their corresponding output meaning. The above finite-state transducer is an example of the BNF-grammar-based parsers that can be developed. Other formalisms for grammar development have been pursued. However, these manual approaches to grammar development have since been replaced by statistical-based approaches. Instead of writing rules to cover the interpretation of input sentences, the statistical approach relies on the creation of a manually annotated corpus, called a treebank, of sentences, which are parsed by hand with their semantic meaning. The corpus is created by asking users to give example sentences on how they would talk to the system; the process is refined by actually collecting data in a pilot phase of the project. This corpus is then manually parsed. Figure 31.3 shows an example semantic parse tree of a sentence. To build a statistical parser, a corpus of 1000–10 000 sentences that are manually parsed is created and is used as a training corpus to build the parser. We now describe one method for developing a statistical parser based on decision trees.

Before we describe parser development, we need to discuss how the semantics of a sentence are captured in its parse tree. One has significant freedom in specifying the parsing style. Figures 31.1 and 31.2 give two options for the definition of an appropriate parse tree. Notice that these semantic parses may not agree with traditional syntactic structures such as those found in earlier *treebanks* [31.1]. Earlier attempts at statistical models for NLU relied on segmenting the sentence as a sequence of hidden Markov model (HMM)

states [31.2] (called *concepts*) and best illustrated in Fig. 31.2. This segmentation-based approach is reminiscent of the information extraction [31.3] approach to language understanding described in more detail in Sect. 31.3. We believe that more-effective models can be achieved with the hierarchical structure that a parse tree can produce for NLU applications. There have been attempts to use HMM models to capture recursive structure or nesting of concepts by designing a state space where state transitions allow one to form a hierarchy of states (and the corresponding transitions) as in the work on hidden understanding models [31.4], but a parsing-based solution is much more natural and effective.

It is a natural tendency to try to capture very detailed semantics when one is designing the guidelines for semantic annotation (i. e., parsing) for a new application. But we have found that the richness of the parse tree should be simplified for two reasons.

- To identify the minimal constituents needed to enable more-accurate models for parsing. We have found that highly nested structures tend to lead to lower parsing accuracy; while this is somewhat of an art, the nesting should be kept to the minimum needed to capture the semantic richness of the domain.
- The flatter the structure the easier it is to explain the annotation guidelines to human annotators and the faster they can do the manual parsing accurately and consistently. Consistency is an important requirement for building real systems since these annotations are application-specific, and consistent treebanks are needed to learn an accurate statistical parser.

Another set of understanding models are based on statistical translation models. In particular, the maximum-entropy (MaxEnt)-based approach described in [31.5] achieves very good results but requires a relatively large training corpus. The method also requires a comprehensive list of all possible formal language requests in the domain. While this method can be used when one has large amounts of data, it is more difficult to bootstrap a system with it and hence it has limited applicability in deploying new NLU applications.

### 31.2.1 Decision Tree Parsers

Some approaches to statistical parsing use a grammar, possibly extracted from a treebank, and a probabilistic model such as a probabilistic context-free grammar (PCFG) to determine the most probable parse for a sentence. In this work, we describe an approach that does not use a grammar to construct a probabilistic model $p(T|w_1^n)$, where $T$ is the parse tree of the word sequence $w_1^n$. We present a method for constructing a model for the conditional distribution of trees given a sentence for all possible trees for that sentence. To parse, we need to find the most probable tree given the model:

$$T^* = \underset{T \in \mathcal{T}(w_1^n)}{\arg\max}\ p(T|w_1^n)\,, \tag{31.1}$$

where the maximization is over all trees that span the $n$-word sentence.

The probabilistic models we explore depend on the *derivational* order of the parse tree. In [31.6], this type of model is referred to as a *history-based* grammar model where a (deterministic) leftmost derivation order is used to factor the probabilistic model. In this work, we use a set of bottom-up derivations of parse trees. Traditionally the derivation order identifies the order of application of grammar rules, i. e., it corresponds to the order of node expansion in a parse tree; in this work, we extend the notion to identify the order in which edges in a tree are created, which includes partial constituents or nodes. In [31.7], both a hidden derivational model as well as a deterministic derivational model are explored to assign the probability of a parse tree. In this work, we only discuss the deterministic bottom-up leftmost (BULM) derivation order.

We now discuss the derivation history model, the parsing model, the probabilistic models for node features, and the training algorithm.

Treebanks are a collection of $n$-ary branching trees, where each node in a tree is labeled by either a nonterminal label (also known as a constituent) or a pre-terminal label (called a tag) obtained by tagging a word (also known as a terminal). If a parse tree is interpreted as a geometric pattern, a constituent is no more than a set of edges that meet at the same node, each of which has a label. In Fig. 31.3, the nonterminal *ARV-TIME*, which spans the tags *oprtr time time* corresponding to the words *before 9 a.m.*, consists of an edge extending to the right from *oprtr* and an edge extending from the nonterminal *TIME* to the left.

We introduce a new definition for a derivation of a parse tree by using Fig. 31.4, which gives a partial subtree used in our parser. We associate with every node in the parse tree two features: a name which is either a tag or a no-terminal label, and an extension, which indicates whether the edge going to its parent is going right, left, up, or is unary. The unary result corresponds to a renaming of a nonterminal. By specifying the two

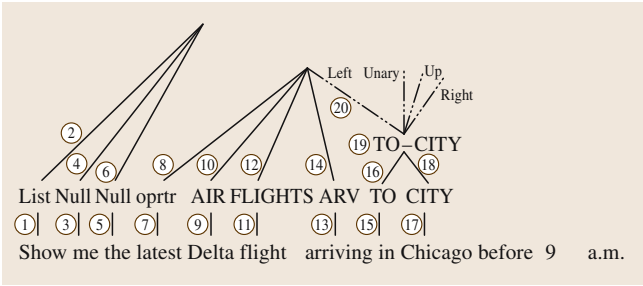



**Fig. 31.4** Partial semantic parse tree with bottom-up leftmost derivation

features (name and extension) for each node we can reconstruct the parse tree. The order of the nodes in which we specify these two features defines the derivation order. We only consider bottom-up leftmost derivations. In a BULM derivation, a node is named first; it may be extended only after it is named, and it is not named until all of the nodes beneath it are extended. When naming a node we refer to the derivation steps as either tagging (when the node is a pre-terminal) or labeling (when the node corresponds to a nonterminal).

We define a derivation as the sequence of actions needed to generate the parser tree. The actions come in three flavors: tagging, extending, and labeling. We illustrate these three action types by defining the derivation of the parse subtree shown in Fig. 31.4. We start by tagging the first word in a sentence; this is step 1. We then extend right (denoted as step 2), then we do step 3, which is to tag the second word as 'Null'. After step 19, where we have labeled the node $TO-CITY$, we now have four possible extension actions: left, unary, up, right, each with a probability that depends on the partial parse derivation so far. This corresponds to step 20, denoted by derivation step $d_{20}$, where node 19 is the active node.

At each decision point, there is an active node where the next action is performed. So the probability of a parse tree $T$ with derivation $d$ is:

$$p(T|w_1^n) = \prod_{1<j<|d|} p(d_j|d_1^{j-1}) \,, \tag{31.2}$$

where the probability of taking action $d_j$ given the previous derivation steps and the input sentence $w_1^n$ can use features from the partial *parse* constructed so far. In [31.7], a hidden derivational model is described that has a slightly better accuracy than the single derivation model discussed here; but in real-world deployments the speed benefit of single derivation models outweighs the small decrease in accuracy.

One can use a number of statistical models for the incremental action model, ranging from MaxEnt models such as that described in [31.8] or the linear models as described by [31.9]. However, we will describe a model using decision trees since it has been used with a commercial offering called IBM ViaVoice Telephony for building telephony-based NLU systems. We essentially have three models:

- Tag model: the tag feature value prediction is conditioned on a window of two words to the left, two words to the right, and two nodes to the left (the two nodes to the right are the same as the two words to the right in a BULM derivation). So if we denote the active node $N_k$ to tag word $w_i$, the two nodes to the left would be denoted by $N_k^{-2}$ and $N_k^{-1}$. We have:

  $$p(t_i|\text{context}) \approx p(t_i|w_{i-2}^{i+2}, t_{i-2}^{i-1}, N_k^{-2}, N_k^{-1}) \,. \tag{31.3}$$

  Note that to tag a word such as in step 15, the word *in*, the two nodes to the left are the partially created nodes that span *latest delta flight arriving* for $N_{15}^{-1}$ and *Show me the* for $N_{15}^{-2}$, respectively. The two words to the right are *Chicago* and *before*. The decision tree can use binary features (questions) that explore the substructure of the nodes. The words are represented by a binary bit vector that indicates membership of $M$ classes. Labels of nodes are also represented by a bit vector. These bit vectors define the binary questions used in building a decision tree model for the tagging decision.
- Extension model: similarly to the tag model, the extension model uses a window of two nodes to the left and two nodes (words in BULM) to the right and the current node $N_k$:

  $$p(e_k|\text{context}) \approx p(e_k|N_k^{-2}, N_k^{-1}, N_k, w_k^{+1}, w_k^{+2}) \,, \tag{31.4}$$

  where $w_k^{+i}$ denotes the $i$-th word to the right of the span of the current node $N_k$. A number of features are used to represent a node $N$: the label of the node, its rightmost child, its leftmost child, the rightmost word in its span, and the leftmost word. Each of these are represented by a bit vector of class memberships. One set of classes and corresponding bit vectors can be created by word clustering [31.10].
- Label model: we also use a window of size five centered on the node to label $N_k$. The model is given by:

  $$p(L_k|\text{context}) \approx p(L_k|Q_k, N_k^{-2}, N_k^{-1}, w_k^{+1}, w_k^{+2}) \,, \tag{31.5}$$

  where $Q_k$ is a set of questions that are defined, and explore the subtree under node $N_k$.



Amazon EX1010 Page 637

#### Statistical Decision Trees

These probability distributions are modeled by traditional decision trees with binary questions on the history. These binary questions rely heavily on defining classes by clustering words, tags, labels, or extension names. The clustering algorithms are described in [31.10]. One can also add head propagation rules as described in [31.7] to enrich the features used by the decision tree.

## 31.3 Practical Implementation

We have found that, the shallower the tree, the easier it is to build decision tree parsers with high accuracy. One approach to achieving shallower trees is to use a two-pass approach to semantic parsing.

### 31.3.1 Classing

For this first pass of semantic parsing the input sentence is marked by finding chunks. For example in the travel domain, time, location, and date expressions are marked. Figure 31.5 shows an example sentence. These trees are quite shallow and reminiscent of shallow parsing. The classed sentence becomes *Show me the latest AIR flight arriving in CITY before TIME.* This sentence is then passed to the second-pass parser.

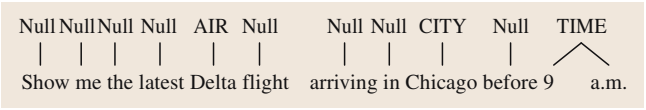



**Fig. 31.5** Classer output

### 31.3.2 Labeling

Once a sentence is tagged we replace the chunks by their classes and the resulting sentence is parsed for more-detailed semantics. The sentence in Fig. 31.5 becomes *Show me the latest AIR flight arriving in CITY before TIME*, which is now the input to the second-pass parser.

This cascade model for semantic parsing has been found to yield more-robust parsers for multiple NLU applications.

There has been a large amount of work on developing statistical parsers [31.11–14], but these have mostly been used with the Penn treebank for syntactic parsing and have a rather large training set of about 1 million words of parsed data. They have not been evaluated for NLU application where the available training sets are rather small, typically ranging from 10 000 thousand to 100 000 words of parsed data.

For practical NLU application development, a recipe for building a parser depends on the availability of sample sentences. This can be initiated by asking people familiar with the system design to write down 10–50 sentences each to obtain a starting set that can be used to bootstrap a first system. The system is then used to produce another round of data collection with employees or paid volunteers to get a few thousand sentences, which is used to build a more-robust semantic parser that can then be used in a pilot deployment, which after a one- or two-month pilot phase can become the production system. Unfortunately, this iterative cycle of data collection and system building is necessary for dialog applications since it is hard to anticipate what users will say when they interact with these application-specific systems. Also in this iterative approach, one can run the parse annotation tool in correction mode by parsing the next batch of data with the current parser and asking human annotators to correct the machine-produced parses. This correction mode speeds up the annotation process significantly. There has also been some work [31.15] on exploring active learning methods to select which sentences humans should manually annotate to reduce the annotation effort.

To handle context dependence, the parser can examine the input sentence and additional input that describes context, such as keywords summarizing the prompt that the system delivered to the user or other state variables. The context is used in the decision tree as additional features to disambiguate the parse of an input sentence.

## 31.4 Speech Mining

Another class of applications for natural language understanding is for systems that analyze human–human dialogs. These are referred to as speech mining applications and are starting to be deployed in call centers.



Amazon EX1010 Page 638

Speech mining covers a broad range of applications from early discovery of trends in a call center to monitoring and training call-center representatives (CSRs). In addition, speech mining of other types of conversations such as in broadcast news, talk shows, and oral history archives are currently a research focus.

In these applications the range of concepts discussed in the conversations is typically much larger than the dialog applications we have discussed so far. So instead of building a system that understands the meaning of all that is discussed, these speech mining applications focus on a specific subset of concepts and determine whether these concepts occur in the conversation. These information extraction systems can be viewed as word-tagging systems where concepts are spotted in the running speech (or text) and that possibly also detect relations between the spotted concepts. We give two example applications.

- *Speech mining for call centers:* one subset of application is for monitoring and training CSRs in a call center; the system is set up to detect that certain phrases are used by the CSR, such as did the CSR use the appropriate phrases in the closing script (e.g., *Thank you for calling* or *Anything else*) or for incipient problem detection by finding certain phrases that indicate a problem and its type from a customer's comments. The shallow semantics depend on the application but they may be as simple as the following two concepts indicated in the following customer's utterance: *I do not know how to install an additional [HARDWARE hard disk] I [REQUEST need your help];* basically this is about a hard disk and the customer is asking for support.

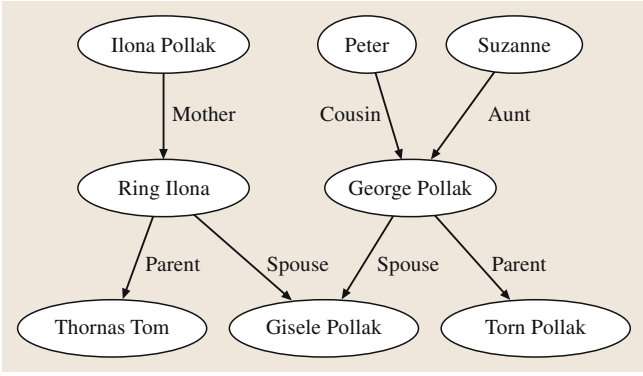



**Fig. 31.6** Personal network extracted from human-transcribed interview

- *Speech mining for discovery:* a significant number of audio databases are becoming available and searchable. Examples are oral history collections such as the collection used in the multilingual access to large spoken archieves (MALACH) project [31.16], which consists of a collection of two-hour interviews with holocaust survivors (there are 50 000 interviews in about 30 languages). Information extraction can be used to analyze these recordings to find mentions of people and find relationships between the people mentioned. One example is to extract all the people that are part of the extended family that are mentioned in the interview by doing entity extraction and *family* relation extraction between the spotted entities. Figure 31.6 shows an example network extracted from human-transcribed speech, and Fig. 31.7 shows the corresponding reference network. In Fig. 31.6, we see entities extracted and relations between them such as: *George* is the spouse of *Gisele*. We also notice errors due to incorrect entity extraction and relation extraction such as *Ring Iona* is a spouse of *Gisele Pollack*.

### 31.4.1 Word Tagging

We now describe two approaches for tagging words with semantics concepts: a linear classifier – the robust risk minimization (RRM) classifier – and a log-linear classifier – the maximum-entropy (MaxEnt) classifier. Both methods can integrate arbitrary types of information and make a classification decision by aggregating all information available for a given classification.

Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be the set of predicted classes, and $\mathcal{X}$ be the example space. Each example $x \in \mathcal{X}$ is associated with a vector of binary features $f(x) = (f_1(x), \ldots, f_m(x))$. We also assume the existence of a training data set $\mathcal{T} \subset \mathcal{X}$ and a test set $\mathcal{E} \subset \mathcal{X}$.

The RRM algorithm [31.9] constructs $n$ linear classifiers $(C_i)_{i=1,\ldots,n}$ (one for each predicted class), each predicting whether the current example belongs to the class or not. Every such classifier $C_i$ has an associated feature weight vector, $(w_{ij})_{j=1,\ldots,m}$, which is learnt during the training phase so as to minimize the classification error rate.

At test time, for each example $x \in \mathcal{E}$, the model computes a score

$$a_i(x) = \sum_{j=1}^{m} w_{ij} \cdot f_j(x) \,,$$



Amazon EX1010 Page 639

and labels the example with either the class corresponding to the classifier with the highest score, if above 0, or 'outside', otherwise. The RRM classifier is:

**foreach** $x \in \mathcal{E}$
  **foreach** $i = 1 \ldots n$
    $a_i[x] = \sum_{j=1}^{m} w_{ij} \cdot f_j(x)$
  $\text{class}(x) \leftarrow \arg \max_i a_i[x]$ .

This linear classifier is used for *sequence classification* by converting the activation scores into probabilities (through the *soft-max* function, for instance) and using the standard dynamic programming search algorithm (also known as a Viterbi search) to find the most likely tag sequence for the word sequence.

Somewhat similarly, the MaxEnt algorithm has an associated set of weights $(\alpha_{ij})_{j=1,\ldots,m}^{i=1,\ldots,n}$, which are estimated during the training phase so as to maximize the likelihood of the data [31.8]. Given these weights, the model computes the probability distribution of a particular example $x$ as follows:

$$P(c_i|x) = \frac{1}{Z} \prod_{j=1}^{m} \alpha_{ij}^{f_j(x)} , \quad Z = \sum_i \prod_j \alpha_{ij}^{f_j(x)} ,$$

where $Z$ is a normalization factor.

After computing the class probability distribution, the assigned class is the most probable one a posteriori. A schematic for the application of the MaxEnt approach to the test data is:

**foreach** $x \in \mathcal{E}$
  $Z \leftarrow 0$
  **foreach** $i = 1 \ldots n$
    $p_i[x] = \prod_{j=1}^{m} \alpha_{ij}^{f_j(x)}$
  Normalize($p$)
  $\text{class}(x) \leftarrow \arg \max_i p_i[x]$ .

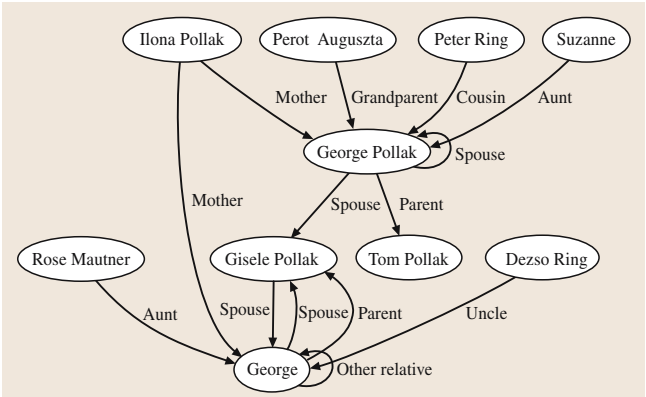



**Fig. 31.7** Correct personal network extracted from human-transcribed interview

Similarly to the RRM model, we use the model to perform sequence classification, through dynamic programming. Within this framework, any type of feature can be used, enabling the system designer to experiment with interesting feature types, rather than worry about specific feature interactions. We have found both methods to be equally as effective in a number of situations. The selection of features depends on the problem and the available linguistic resources for the language at hand. Example features are the words in a five-word window, the stems of these words, and prefixes and suffixes of various lengths. Also, one can include part-of-speech tags and WordNet synsets [31.17], if available for the language of interest. Domain-specific lexicons such as a list of product names can also be useful.

One uses the above classifiers with a Viterbi search to build a mention detection system for all the entities/concepts of interest. Following mention detection, we run a second classifier to find relations between concepts or every pair of mentions within a unit (e.g., a user turn). Similarly, the features used to detect whether a particular relation exist uses the type of mentions, the words between the mentions, and the parse tree of the sentence. A more-detailed description of one particular relation extraction system is given in [31.18].



## 31.5 Conclusion

Natural language understanding systems are increasingly being used in telephony-based dialog systems, as speech recognition systems that support large-vocabulary $n$-gram language models become more accurate. The ability to handle freeform user requests enables the automation of more-complex applications, resulting in significant cost reduction in call centers. The statistical modeling approach described in this chapter has proven

to be effective for high-accuracy systems. In addition, it enables an efficient solution for deploying an application in many languages. By using the same semantics and translating the training data from a system developed in one language, one can efficiently create equivalent systems in other languages. A bilingual system that allows a speaker to speak in either of two languages on a turn-by-turn basis can also be built. These bilingual systems may prove useful for many customer segments. In addition, these data-driven methods allow continuous tuning of a deployed system as more usage data is acquired in order to achieve higher levels of accuracy.

## References


31.1 M.P. Marcus, B. Santorini, M.A. Marcinkiewicz: Building a large annotated corpus of English: The Penn Treebank, Comput. Linguist. **19**, 313–330 (1993)

31.2 E. Levin, R. Pieraccini: CHRONUS: *The Next Generation, ARPA Spoken Language Systems Technology Workshop* (1995) pp. 269–271

31.3 R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, S Roukos: A statistical model for multilingual entity detection and tracking, Proc. of HLT-NAACL 2004 (2004) pp. 1–8

31.4 R. Schwartz, S. Miller, D. Stallard, J. Makhoul: Language understanding using hidden understanding models, Proc. of ICSLP-96 (1996)

31.5 K. A. Papineni, S. Roukos, R. T. Ward: Feature-based language understanding, Proc. of the 5th European Conference on Speech Communication and Technology (1997)

31.6 E. Black, F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, S. Roukos: Towards history-based grammars: Using richer models for probabilistic parsing, Proc. of the Association for Computational Linguistics (1993) pp. 31–37

31.7 F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, A. Ratnaparkhi, S. Roukos: Decision tree parsing using a hidden derivation model, Proc. of the 1994 Human Language Technology Workshop (1994) pp. 272–277

31.8 A. Berger, S. Della Pietra, V. Della Pietra: A maximum entropy approach to natural language processing, Comput. Linguist. **22**(1), 39–71 (1996)

31.9 T. Zhang, F. Damerau, D.E. Johnson: Text chunking based on a generalization of winnow, J. Machine Learning Res. **2**, 615–663 (2002)

31.10 P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, R. L. Mercer: Class-based *n*-gram models of natural language, Proc. of the IBM Natural Language ITL (1990)

31.11 A. Ratnaparkhi: A linear observed time statistical parser based on maximum entropy models, Proc. of the Second Conference On Empirical Methods in Natural Language Processing (EMNLP-2) (1997)

31.12 M. Collins: Head-driven statistical models for natural language parsing, Comput. Linguist. **29**, 589–637 (2003)

31.13 D. M. Bikel: On the parameter space of generative lexicalized statistical parsing models, PhD Dissertation, University of Pennsylvania (2004)

31.14 E. Charniak, M. Johnson: Coarse-to-fine n-best parsing and MaxEnt discriminative reranking, Proc. of Association for Computational Linguistics ACL (2005) pp. 173–180

31.15 M. Tang, X. Luo, S. Roukos: Active learning for statistical natural language parsing, Proc. of the Association for Computational Linguistics (2002)

31.16 D. Oard, D. Soergel, D. Doermann, X. Huang, G. C. Murray, J. Wang, B. Ramabhadran, M. Franz, S. Gustman, J. Mayfield, L. Kharevych, S. Strassel: Building an information retrieval test collection for spontaneous conversational speech, Proc. of SIGIR'04 (2004)

31.17 G. A. Miller: WordNet: A lexical database, Commun. ACM, 38(11) (1995)

31.18 N. Kambhatla: Minority vote: At-least-N voting improves recall for extracting relations, Proc. of COLING/ACL 2006 (2006) pp. 460–466




Amazon EX1010 Page 641

# 32. Transcription and Distillation of Spontaneous Speech

S. Furui, T. Kawahara

Automatic transcription of spontaneous human-to-human speech is expected to expand the applications of speech technology, enabling efficient access to audio archives such as broadcast programs, lectures, and meetings. Compared with utterances in human–machine interfaces, which have been the focus of most conventional speech recognition research, spontaneous speech has greater variation in both its acoustic and linguistic characteristics. Therefore, it is necessary to explore more elaborate and flexible modeling techniques for spontaneous speech recognition. Moreover, spontaneous speech processing requires the development of a different paradigm, in that faithful transcription is not necessarily useful because of the existence of disfluencies and the lack of sentence and paragraph markers. Studies on automatic detection of sentence/discourse boundaries and disfluencies are also needed. Speech summarization is an approach that generates effective output from a transcript. This chapter gives an overview of major research activities and a number of recent findings on these topics.

32.1 **Background** ..... 627
32.2 **Overview of Research Activities on Spontaneous Speech** ..... 628
32.2.1 Classification of Spontaneous Speech ..... 628
32.2.2 Major Projects and Corpora of Spontaneous Speech ..... 629
32.2.3 Issues in Design of Spontaneous Speech Corpora .... 630
32.2.4 Corpus of Spontaneous Japanese (CSJ) ..... 631
32.3 **Analysis for Spontaneous Speech Recognition** ..... 632
32.3.1 Observation in Spectral Analysis .... 632
32.3.2 Analysis of Speaking Rate ..... 634
32.3.3 Analysis of Factors Affecting ASR Accuracy ..... 634
32.4 **Approaches to Spontaneous Speech Recognition** ..... 635
32.4.1 Effect of Corpus Size ..... 635
32.4.2 Acoustic Modeling ..... 636
32.4.3 Models Considering Speaking Rate 637
32.4.4 Pronunciation Variation Modeling . 637
32.4.5 Language Model ..... 638
32.4.6 Adaptation of Acoustic Model ..... 638
32.4.7 Adaptation of Language Model ..... 639
32.5 **Metadata and Structure Extraction of Spontaneous Speech** ..... 640
32.5.1 Sentence Boundary Detection ..... 640
32.5.2 Disfluency Detection ..... 642
32.5.3 Detection of Topic and Discourse Boundaries ..... 643
32.6 **Speech Summarization** ..... 644
32.6.1 Categories of Speech Summarization ..... 644
32.6.2 Key Sentence Extraction ..... 645
32.6.3 Summary Generation ..... 646
32.7 **Conclusions** ..... 647
**References** ..... 647



## 32.1 Background

Automatic speech recognition (ASR) technology has achieved accuracy levels exceeding 90% for read speech and similar styles of speech, including news broadcasts given by anchors. However, the performance of ASR drops off dramatically for spontaneous speech such as human-to-human conversations. Spontaneous speech is significantly different from read speech, both acoustically and linguistically. Spontaneous speech is not clearly articulated with orthodox pronunciation, and contains numerous disfluencies and colloquial expressions. Although human beings face similar problems in hearing foreign languages, current ASR systems se-

riously lack the coverage and flexibility required to cope with spontaneous speech. Thus, in addition to noise robustness, spontaneous speech presents one of the greatest challenges in ASR research.

It is quite interesting to note that, although speech is almost always spontaneous in daily life, spontaneous speech recognition emerged as a visible research theme only about 10 years ago. Most conventional ASR systems have been developed as human–machine interfaces, primarily for the purposes of information input and access. Although these will no doubt remain the primary applications of ASR, there is also a growing interest in viewing speech not simply as a means to access information, but as a source of information in itself. Since speech is the most natural and effective method of communication between human beings, a variety of speech documents, including lectures, meetings, and broadcast programs, are produced everyday. With the pervasion of digital media technologies, these audio materials are increasingly archived both personally and publicly. Even more personal recordings, such as voice messages and conversations, are also being digitally recorded. However, it is not easy to quickly review, retrieve, selectively disseminate, and reuse these speech archives, if they are simply stored as audio signals. Automatic speech transcription is essential for creating knowledge resources from huge speech archives, and thus improving ASR so that it can better cope with spontaneous speech is critical to broaden its potential applications.

For improved ASR performance, it is indispensable to build acoustic and language models for spontaneous speech. Since the characteristics of spontaneous speech are different from those of read-style speech, it may be inadequate to simply apply conventional modeling schemes such as hidden Markov models (HMMs) and $n$-gram language models. The variation of spontaneous speech is much larger according to the speaker and the speaking style. Thus, we need to investigate what kinds of factors actually affect the characteristics of spontaneous speech and how to parameterize and computationally model these factors.

Despite the wide variety of spontaneous speech, it is difficult and costly to prepare a large speech corpus, because it involves manual transcription of utterances with many disfluencies, compared to the reading of prepared materials. This motivated us to build the corpus of spontaneous Japanese (CSJ), which can be used as a shared resource for various fields of spontaneous speech research. Several works presented herein are based on this corpus.

Spontaneous speech recognition also requires a paradigm shift from conventional ASR, which decodes utterances into a word sequence, because simple transcription of spontaneous speech includes many disfluencies, and so is neither readable nor appropriate for documentation. In addition, sentence and paragraph endings are not explicitly given. Therefore, a number of research projects, including those sponsored by the Defense Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST), are oriented to *rich transcription*, which also involves annotation of end-of-sentence markers and disfluency phenomena. Furthermore, we explore semantic-level processing to extract messages from spontaneous speech such as key sentence indexing and speech summarization. These issues are also addressed herein.

## 32.2 Overview of Research Activities on Spontaneous Speech

### 32.2.1 Classification of Spontaneous Speech

Speech recognition tasks can be classified into four categories, as shown in Table 32.1, according to two criteria: whether the targeting utterances are human-to-human or human-to-machine, and whether they constitute a dialogue or a monologue. The table lists typical tasks for each category.

The majority of commercial application systems are classified as category III, recognizing utterances in human-to-machine dialogues. Unlike other categories, the category III systems are usually designed to complete a specific task in a limited domain, for example, quoting stock prices or making ticket reservations. Therefore, the ASR system typically consists of a medium-size vocabulary and a task-specific grammar.

The most typical task belonging to category IV, which targets recognition of monologues performed when people are talking to a computer, is dictation. This task was an initial attempt at achieving general-purpose large-vocabulary continuous speech recognition (LVCSR), and provides a technological basis for other ASR applications, including those of categories I and II. The dictation system, however, assumes that users



Amazon EX1010 Page 643

**Table 32.1** Categorization of speech recognition tasks

| | Dialogue | Monologue |
|---|---|---|
| Human to human | (Category I)<br>Telephone conversation<br>Panel discussion<br>Meeting | (Category II)<br>Broadcast news<br>Lecture<br>Voice mail |
| Human to machine | (Category III)<br>Database query<br>Information retrieval | (Category IV)<br>Dictation<br>WSJ task |

clearly utter grammatically correct sentences with orthodox pronunciation. This assumption generally holds in human–machine interfaces, especially when users are aware that they are talking to a computer and ASR results are fed back to them in real time.

Typical tasks belonging to categories I and II, recognizing human-to-human interactions, are plotted in Fig. 32.1 with respect to the number of participants and spontaneity. Major projects conducted thus far in these categories are briefly described in the following subsection.

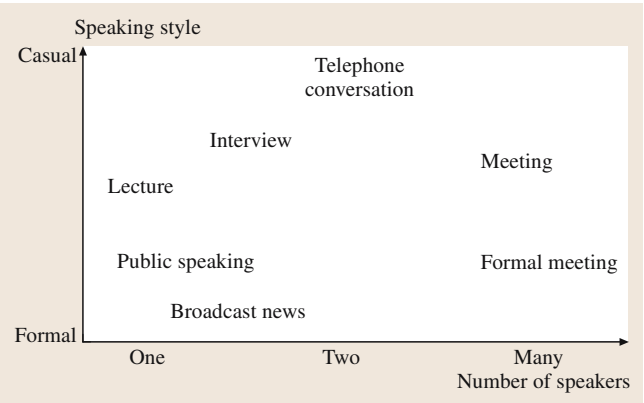



**Fig. 32.1** Spontaneous speech recognition tasks

## 32.2.2 Major Projects and Corpora of Spontaneous Speech

### Broadcast News

Broadcast news was initially and is still now intensively targeted by many projects, including DARPA Hub-4 [32.1] and the GALE (global autonomous language exploitation) programs. Most of the utterances in broadcast news programs are made by professional anchors, and language models can be well developed based on a large amount of closed captions. Thus, this particular ASR task is relatively easier. Because broadcast news is a rich information source, it is also used for research on content-based processing, such as topic detection and summarization [32.2, 3].

### Oral Presentations and Lectures

Public speaking and oral presentations are also appropriate targets for automatic transcription and archiving. Although the speakers are not necessarily professionals, they generally try their best to be understandable by their audience. Since the topic and vocabulary are often technical, it is necessary to adapt the language model as well as the acoustic model for every speaker. Although there were some previous efforts in this direction such as the TED (Translanguage English Database) corpus [32.4], a recorded collection of EUROSPEECH conference, the most comprehensive one is the corpus of spontaneous Japanese (CSJ) [32.5], which is a 660-hour speech collection of academic presentations and simulated public speeches, recorded using a close-talking microphone. The word error rate (WER) is around 20% with this corpus. The CSJ is thoroughly described in Sect. 32.2.4.

Lectures at universities are also being digitally archived. Their automatic transcription and indexing is also studied at MIT [32.6] and Microsoft under the iCampus project, using hundreds of hours of lectures and seminars at MIT. Similar efforts are being conducted at Kyoto University and the Tokyo Institute of Technology in Japan. The WER for these lectures is 40–50% [32.6]. Unlike the CSJ, the lecturers do not use headset microphones, and the speaking style is less formal in the classroom.

### Interviews and Telephone Conversations

Interviews and telephone conversations are basically forms of dialogue, but the interview is a less private style and is similar to the monologue. The MALACH (multilingual access to large spoken archives) project [32.7] was organized to advance the speech technology for access to large multilingual archives assembled by the Survivors of the Shoah Visual History Foundation (VHF). It is a 116 000-hour collection of interviews conducted in 32 languages with nearly 52 000 survivors of the Holocaust. Automatic transcription has been added to portions of the English and Czech speech using ASR with a WER below 40% [32.7]. This task is challenging because speech is heavily accented by elderly speakers who are often emotional.

Dialogue conversations over telephone channels have been most intensively studied in the DARPA-funded projects of HUB-5 [32.8] and EARS (effective affordable reusable speech-to-text) [32.9, 10]. The



Amazon EX1010 Page 644

Switchboard corpus [32.11], the CALLHOME corpus, and the Fisher corpus were set up for this purpose. Telephone conversations are generally the most spontaneous and casual. As a result, acoustic and pronunciation variations are very prominent. In terms of linguistic aspects, it is observed that a small vocabulary, characteristic of conversational style, accounts for large coverage of utterances, although these words have many pronunciation variants. The WER had been very large (around 40%), but was improved to 15% with the increase of the training data to 2100 hours and the combination of various training and decoding techniques [32.9, 10, 12].

#### Meetings

Meetings are also emerging as a major target of spontaneous speech recognition. Meeting recognition projects were initiated by NIST [32.13] and several European-funded projects such as AMI (augmented multi-party interaction) and CHIL (computers in the human interaction loop). Since meetings involve multiple participants, it is necessary to determine who spoke when. This task is referred to as speaker diarization. Overlapping of utterances is often observed and severely affects both speaker diarization and speech recognition. Moreover, the effect of the use of distant microphones, including microphone arrays, is also investigated. The WER is 20–30% with close-talking microphones and 30–40% with distant microphones [32.14]. Since the spontaneity of utterances is high, it was shown that the incorporation of telephone conversation corpora was effective when the size of the matched training data was small.

More formal meetings include those in Congress or Parliament. In Europe, the TC-STAR (technology and corpora for speech to speech translation) project [32.15], funded by the European Commission, primarily targets automatic speech transcription and translation of the meetings in European Parliament plenary sessions. The WER is close to 10% [32.16]. In Japan, the National Congress (Diet) is interested in the introduction of ASR technology into the next generation of the meeting transcription system. There, most of the meetings are not in plenary sessions, but in committees where the degree of spontaneity is higher. The WER without using matched training data was around 20% [32.17], and is improved to around 15% with a matched corpus.

### 32.2.3 Issues in Design of Spontaneous Speech Corpora

The appetite of today's statistical speech processing techniques for training material is well described by the aphorism, *There's no data like more data*. Large collections of speech and text, or corpora, are vital for research and development of ASR. Statistical methodology provides an automatic procedure to directly *learn* regularities in the speech data, but this means that the performance of the derived models and systems are significantly affected by the characteristics and the quality of the corpora. Therefore, there are several issues to be considered in the design of spontaneous speech corpora.

The first problem is how to collect speech by controlling variety and spontaneity. It is desirable to collect natural spontaneous speech without any constraints, but there are always privacy or corporate property issues encountered in collecting real data in public or in companies. Thus, we often ask collected subjects to speak naturally in a given situation or on a given topic. In this case, selection of the topic and the intimacy of the participants deeply affect the spontaneity. The variety of speakers and vocabulary must also be taken into account to make a general corpus.

The second issue is the quality of transcription and annotation. Manual transcription of spontaneous speech involves huge labor costs. Moreover, spontaneous speech includes many disfluency phenomena, such as self-repairs and partial words, and so specific guidelines for annotation are needed. For example, how do we transcribe when speakers make ambiguous pronunciation or apparent pronunciation errors? How do we handle overlapping utterances, especially when some speakers' voices are not easily recognizable? Recent progress in extensible mark-up languages (XML) is helpful with respect to structured annotation, but guidelines should be carefully designed and documented, and preferably shared among research communities [32.18, 19].

To reach large quantities, imperfect transcription can be used. In the case of broadcast news, closed caption may be useful, although it is not a faithful transcript of the utterances. In this context, lightly supervised training has been investigated [32.20, 21], and reported to be effective in reducing the transcription cost. However, this scheme is workable only when the baseline ASR performance is sufficiently high and reliable.

Another issue is the design of higher-level annotation. In relation to speech recognition, there are many topics in spontaneous speech processing, including sentence/topic boundary detection and key sentence indexing. Since spontaneous speech includes ill-formed phenomena, which are not well explained by conventional linguistic theory, the annotation of these events requires a good deal of analysis and experience us-



Amazon EX1010 Page 645

ing real data. This often becomes so subjective that agreement among different annotators is not high. Thus, the design of a reliable tag set is itself a topic for research.

### 32.2.4 Corpus of Spontaneous Japanese (CSJ)

After considering the above issues, we designed and compiled the corpus of spontaneous Japanese (CSJ) [32.22], which is one of the largest spontaneous speech corpora. The CSJ is a primary product of the science and technology agency priority program entitled *Spontaneous Speech: Corpus and Processing Technology*, which was conducted in Japan from 1999 to 2004 [32.5].

The CSJ consists of roughly seven million words with a total speech length of 660 hours. Mainly recorded are monologues of academic presentations (AP) and extemporaneous presentations (EP), as shown in Table 32.2, which amount to about 300 hours each. APs are live recordings made at nine different academic societies covering the fields of engineering, social science, and humanities. EPs are studio recordings of paid laymen's speeches on everyday topics, for example, "the most delightful memory of my life", which were presented in front of a small audience and in a relatively relaxed atmosphere. The topics of APs are related to expert fields, and the vocabularies are often technical, whereas the word distribution in EPs is expected to reflect ordinary Japanese language. While the majority of AP speakers are males aged 20–40 years old, the age and gender of EP speakers are more balanced. In APs, most of the speakers prepared or rehearsed the talk beforehand, but only a few read a manuscript. In summary, AP is real speech in front of a large audience, and EP is more balanced both acoustically and linguistically. The CSJ also includes a number of speech dialogues for the purpose of comparison with monologue speech.

The recordings were manually given both orthographic and phonetic transcription. Phonetic transcription adopts Japanese *kana* characters, but reflects faithful pronunciation rather than orthographic notation. Spontaneous speech-specific phenomena, such as filled pauses, word fragments, reduced articulation and mispronunciation, as well as non-speech events such as laughter and coughing, were also tagged following the carefully designed guidelines. The transcription was very costly, but provides a valuable information resource for statistical modeling of spontaneous speech, including pronunciation variation and disfluency phenomena.

**Table 32.2** Contents of the CSJ

| Type of speech | #speakers | #talks | Hours |
|---|---|---|---|
| Academic presentations (AP) | 838 | 1006 | 299.5 |
| Extemporaneous presentations (EP) | 580 | 1715 | 327.5 |
| Interview on AP | *(10) | 10 | 2.1 |
| Interview on EP | *(16) | 16 | 3.4 |
| Task-oriented dialogue | *(16) | 16 | 3.1 |
| Free dialogue | *(16) | 16 | 3.6 |
| Reading text | *(244) | 491 | 14.1 |
| Reading transcripts | *(16) | 16 | 5.5 |
| | | Total | 658.8 |

* counted as AP or EP speakers

One-tenth of the utterances, or half a million words, hereinafter referred to as the *core*, were tagged manually and used for training a morphological analysis and part-of-speech (POS) tagging program [32.23], which was used to automatically analyze the remaining portions of the corpus. The core consists of 70 APs, 107 EPs, 18 dialogues and six read speech files.

These were also tagged with prosodic information as well as higher-level linguistic information. For intonation labeling of spontaneous speech, the traditional J_ToBI was extended to X_JToBI [32.24], in which inventories of tonal events as well as break indices were considerably enriched to reflect spontaneous Japanese.

Linguistic annotation includes the following information:

- dependency structure of *bunsetsu*s, a minimal grammatical unit, consisting of one content word plus adjacent functional words
- clause and sentence boundaries
- discourse boundaries (for part of the core)
- key sentences selected by human subjects
- summaries made by human subjects
- subjective impression of the speech with several adjective-pair dimensions

These high-level annotations are useful for the study of *rich transcription*, and several works using them are explained later herein.



Amazon EX1010 Page 646

## 32.3 Analysis for Spontaneous Speech Recognition

Prior to the description of ASR studies, in this section, we present preliminary analyses of spontaneous speech using the CSJ.

### 32.3.1 Observation in Spectral Analysis

At first, the spectral characteristics of spontaneous speech were analyzed in comparison with those of read speech [32.25]. Utterances from four speech types in the CSJ, which are AP, EP, reading of the transcript of AP (read speech), and dialogues, were used in this analysis. In order to mitigate the effect of individual differences, utterances in different styles by the same five male and five female speakers were compared. Since not only the speakers but also the texts were identical for the reading of the transcribed speeches and the original AP utterances, it was possible to perform a very precise comparative analysis. Each speech had a duration of 10 minutes on average.

It is well known that a significant amount of phones are deleted in spontaneous speech [32.26]. In the CSJ, faithful transcription was manually given considering phone deletions and substitutions, although vowel deletion in consonant–vowel syllables was not taken into account with the transcription using the Japanese *kana* syllable characters.

As an acoustic analysis, 39-dimensional feature vectors, consisting of 12-dimensional mel-frequency cepstral coefficients (MFCC), log-energy, and their first and second derivatives, were extracted from utterances using a 25 ms-length window shifted every 10 ms. The cepstral mean subtraction (CMS) was applied to each utterance. For this experiment, a monophone HMM with three states, each having a single, diagonal Gaussian distribution, was trained for each phone, for each speaker, and for each speaking style. The mean and variance vectors of the 12-dimensional MFCC at the second state of the three-state HMM were extracted for each phone.

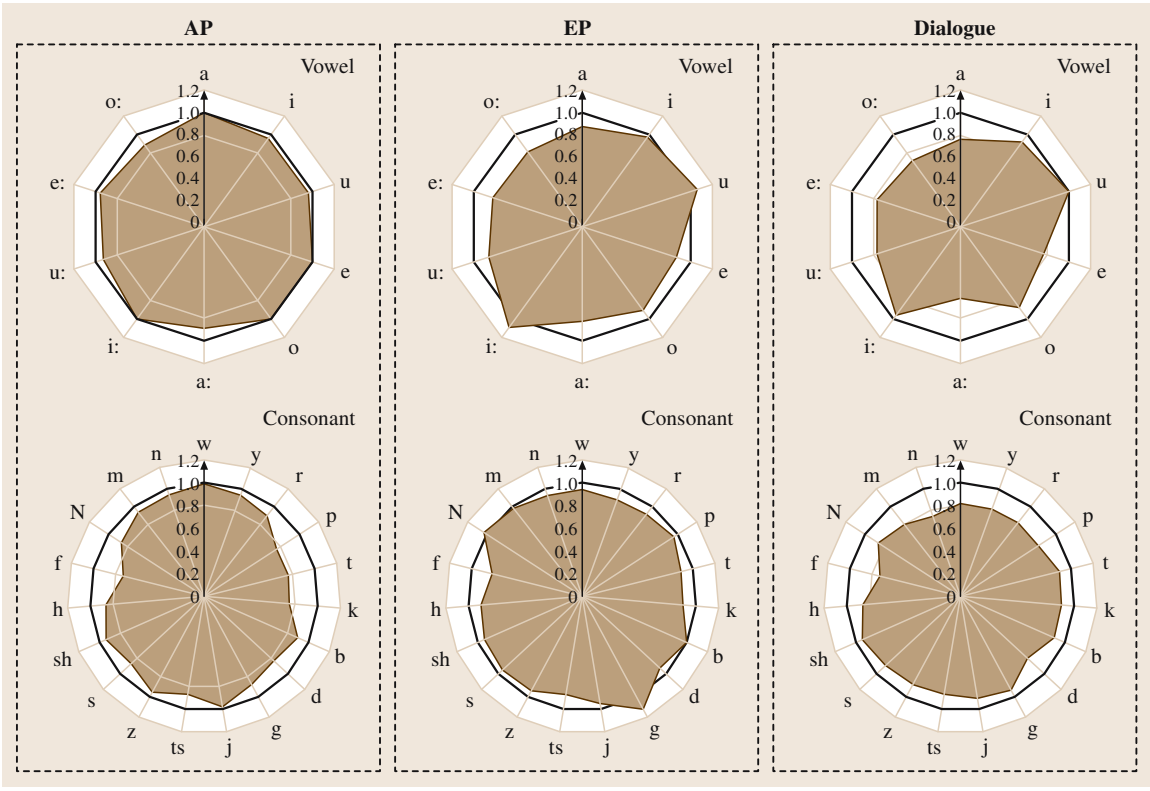



**Fig. 32.2** Reduction ratio of the vector norm between each phone and phone center in spontaneous speech to that in read speech



Amazon EX1010 Page 647

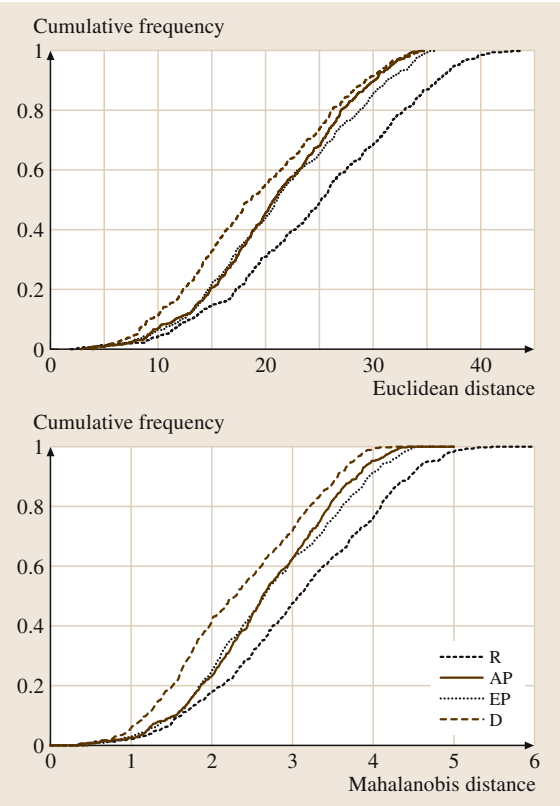



**Fig. 32.3** Distribution of distances between phones

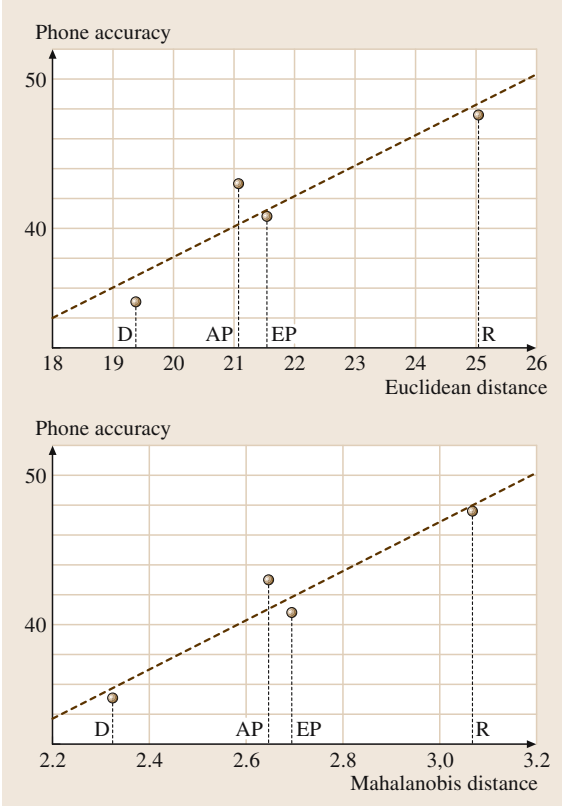



**Fig. 32.4** Relationship between phone distances and phone recognition accuracy

In order to analyze the phone reduction quantitatively, Euclidean distances between the mean vector of each phone and the center of the distribution of all phones (the vector averaged over all phones) were calculated, and the ratio of the distance for spontaneous speech (AP, EP, and dialogues) to that of read speech was calculated for each phone. Figure 32.2 shows the reduction ratios averaged over all speakers. The condition of no reduction (the reduction ratio equals to 1) is indicated by a thick line. The reduction of the MFCC space is observed for almost all of the phones in all three speaking styles, and it is most significant for dialogue utterances.

Next, the reduction of the cepstral distance between each phone pair is measured. The Euclidean distance using the mean MFCC vector of each phone and the Mahalanobis distance, which takes into account the variances, were measured. Figure 32.3 shows the cumulative frequency of distances between phones for each speaking style. Here, R and D denote read speech and dialogues, respectively. The distributions are statistically different from each other, except AP and EP. It is shown that the distance between phones decreases as the spontaneity of the utterances increases (D $\gg$ EP $>$ AP $\gg$ R).

Then, the relationship between these distances and ASR accuracy is investigated. Monophone HMMs with a single Gaussian distribution were prepared for each speaking style. A phone network with diphone probabilities was used as a language model for phone recognition. Figure 32.4 shows the relationship between the mean phone distance and the phone recognition accuracy. Correlation coefficients between them are 0.93 in the case of the Euclidean distance and 0.97 for the Mahalanobis distance. The lines in Fig. 32.4 indicate the regression over the four points. These results show that the phone recognition accuracy is strongly correlated with and can be predicted by the mean phone distance.



Amazon EX1010 Page 648

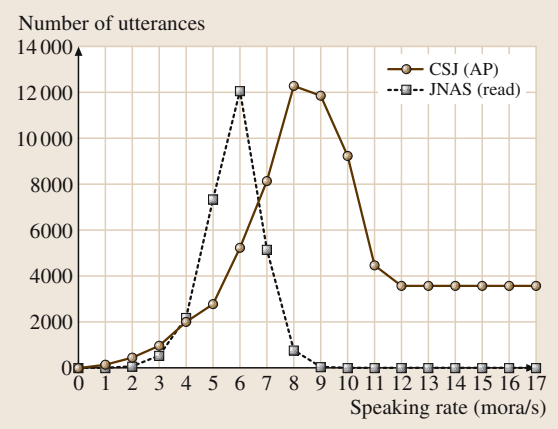



**Fig. 32.5** Speaking rate distribution of the CSJ and JNAS corpora

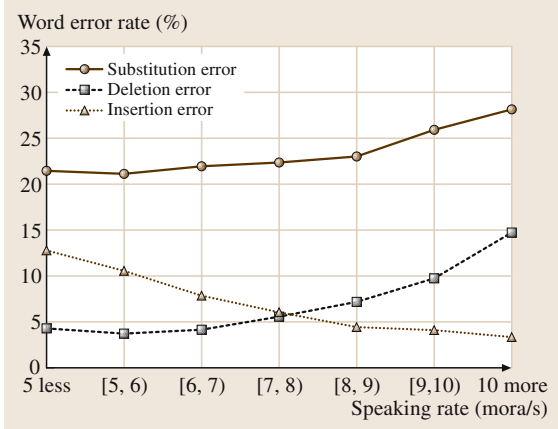



**Fig. 32.7** Percentage of substitution, deletion, and insertion errors for each speaking rate

### 32.3.2 Analysis of Speaking Rate

Next, the analysis results on the speaking rate, which is one of the most prominent characteristics in spontaneous speech, are presented.

In Fig. 32.5, the distributions of the speaking rate in the CSJ (AP of 35 hours) and the JNAS (Japanese newspaper article sentences) corpus (40 hours) are plotted. The speaking rate is estimated for every utterance and is defined as the mora count divided by the utterance duration (sec). For both corpora, manual phonetic transcription was used for defining morae. The mean and standard deviation of the speaking rate of the JNAS corpus were 6.27 and 0.97, and those of the CSJ/AP were 8.70 and 2.10, respectively. This confirms that spontaneous speech is faster than read speech and that the speaking rate variation of spontaneous speech is larger than that of read speech.

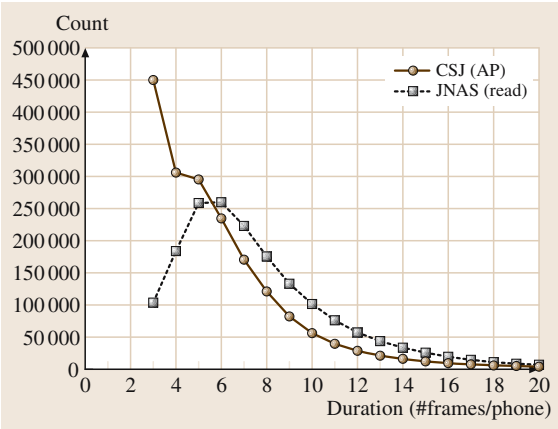



**Fig. 32.6** Phone duration distribution of the CSJ and JNAS corpora

The distributions of phone duration in spontaneous and read speech are also plotted in Fig. 32.6. The phone duration is estimated based on the Viterbi algorithm for a given phonetic transcript. Since we used three-state phone HMMs without state skipping, the minimum duration is three frames (30 ms). Many segments in the CSJ may have a shorter duration, but are forcibly aligned to three frames. This may have caused a serious mismatch. Moreover, a fast speaking rate suggests that these segments are poorly articulated and cause problems in ASR.

The relationship between the WER and the speaking rate is also investigated using 15 APs by male speakers. In Fig. 32.7, the breakdown of recognition errors is shown for each speaking rate. Faster utterances are confirmed to generally be more difficult for ASR. Moreover, we observe different tendencies in the errors according to the speaking rate, that is, more deletion errors in faster speech and more insertions in slower speech.

### 32.3.3 Analysis of Factors Affecting ASR Accuracy

Next, we present a preliminary analysis using a baseline speech recognition system. The acoustic model consists of gender-independent triphone HMMs that have 3000 shared states with 16 Gaussian mixture components. The language model is a standard trigram model of word-pronunciation entries, and the generated lexicon size is approximately 30 000.



Amazon EX1010 Page 649

Individual differences in ASR accuracy were analyzed using 10 minutes from the CSJ presentations by 51 male speakers [32.27]. Six types of attributes were considered in the analysis: averaged acoustic frame likelihood (AL), speaking rate (SR), word perplexity (PP), out-of-vocabulary rate (OR), filler rate (FR), and repair rate (RR). The speaking rate is defined as the average number of phones per second. The filler rate and the repair rate are the number of filled pauses and self-repairs divided by the number of words, respectively.

The analysis results indicate that the attributes exhibiting a real correlation with the word accuracy are the speaking rate (SR), the out-of-vocabulary rate (OR), and the repair rate (RR). Although the other attributes also have a correlation with the accuracy, the correlation is actually caused through these more fundamentally influential attributes. For example, the word perplexity (PP) is correlated with accuracy, but if its correlation with OR is removed, PP does not correlate well with accuracy.

The following equation was obtained as a result of a linear regression model of the word accuracy (Acc) with the six attributes:

$$\begin{aligned}\mathrm{Acc} = {} & 0.12\mathrm{AL} - 0.88\mathrm{SR} - 0.02\mathrm{PP} \\ & - 2.2\mathrm{OR} + 0.32\mathrm{FR} - 3.0\mathrm{RR} + 95\,. \end{aligned} \tag{32.1}$$

The regression coefficient of RR is −3.0 and that of OR is −2.2. This means that a 1% increase in the repair rate or the out-of-vocabulary rate corresponds, respectively, to a 3.0% or 2.2% decrease in the word accuracy. This is probably because a single recognition error caused by a repair or an out-of-vocabulary word triggers secondary errors due to linguistic constraints. The determination coefficient of the multiple linear regression is 0.48, meaning that roughly half of the variance of the word accuracy can be explained by the model. The normalized representation of the regression analysis, in which the variables are normalized with their means and variances, indicates that the coefficients of SR, OR, and RR are relatively large.

**Table 32.3** Statistics of the attributes in the CSJ

| | AP | EP |
|---|---|---|
| SR: speaking rate (mora/s) | 9.05 (1.09) | 7.97 (0.79) |
| PP: perplexity | 81.1 (25.3) | 82.4 (25.2) |
| OR: OOV rate (%) | 1.45 (0.66) | 1.71 (0.82) |
| FR: filler rate (%) | 6.80 (3.44) | 5.45 (3.25) |
| RR: repair rate (%) | 1.40 (0.79) | 1.25 (0.85) |

Upper row: mean, lower row: standard deviation

The statistics of these attributes are listed in Table 32.3 for the AP and EP parts of the CSJ. Here, the speaking rate (SR) is computed by dividing the number of morae by the total duration (sec) of the talk, excluding pauses. Note that, in this subsection, the speaking rate is computed as an average over the entire talk, whereas it was computed utterance by utterance in Sect. 32.3.2 for more precise analysis. This difference in the definition greatly influences the standard deviation. In Table 32.3, it is observed that speaking in AP is faster and more disfluent; more specifically it has more fillers and repairs. For reference, in the JNAS read speech corpus, the speaking rate is slower (7.36 mora/sec) and there are no fillers and repairs.

Based on these analyses, test sets for ASR evaluation were prepared for the AP and EP parts of the CSJ. These sets are designed by balancing three factors of speaking rate (SR), perplexity (PP), and repair rate (RR) that affect the ASR performance. Here, we use the perplexity (PP) instead of the out-of-vocabulary rate (OR), because OR intrinsically depends on vocabulary and easily varies when the lexicon is modified.

## 32.4 Approaches to Spontaneous Speech Recognition

Following the observation of several characteristics in spontaneous speech, in this section, major approaches to speech recognition are reviewed with respect to acoustic and language models.

### 32.4.1 Effect of Corpus Size

First, the effect of the corpus size was investigated using the CSJ. Fig. 32.8 shows the WER, adjusted test-set



Amazon EX1010 Page 650

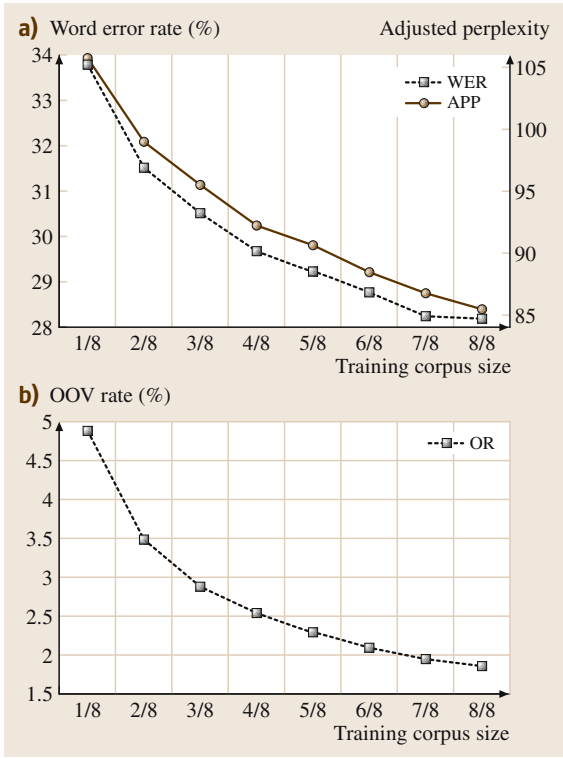



**Fig. 32.8** **(a)** WER, adjusted test-set perplexity (APP); **(b)** and out-of-vocabulary rate (OR) as a function of the size of language model training data

perplexity (APP), and out-of-vocabulary rate (OR) as a function of the size of the language model training data, with the condition that the acoustic model is fixed. The adjusted perplexity was used to normalize the effect of the increase in the vocabulary size on the perplexity according to the increase in the training data size. By increasing the language model training data size from 1/8 (0.86M words) to 8/8 (6.84M words), the WER, perplexity and OOV rate are reduced by 17%, 19%, and 62%, respectively.

On the other hand, Fig. 32.9 shows the WER as a function of the size of the acoustic model training data, when the language model is made using the entire training data set. By increasing the acoustic model training data from 1/8 (68 hours) to 8/8 (510 hours), the WER is reduced by 6.3%.

These results show that the WER is significantly reduced by the increase of the training data. It is confirmed that the size of the CSJ has a strong impact on modeling spontaneous speech. Comparing Fig. 32.8 and Fig. 32.9, the effect of WER reduction on the language

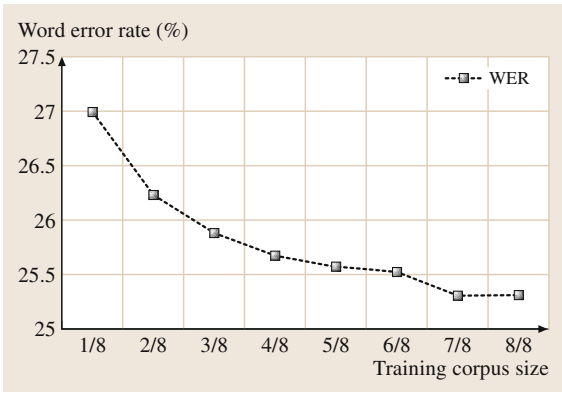



**Fig. 32.9** WER as a function of the size of acoustic model training data

model is much greater and appears not to saturate even when using the entire corpus (rightmost plot). This suggests the necessity for more language model training data.

## 32.4.2 Acoustic Modeling

As shown in the previous section, spontaneous speech has greater variation both in spectral and temporal structures than read-style speech. As the acoustic variation is largely dependent on speakers, feature normalization techniques such as vocal-tract-length normalization (VTLN) [32.28, 29] are effective. The speaker-adaptive training (SAT) [32.30, 31] scheme has also been extensively studied.

On the other hand, it should be questioned whether the conventional HMM is appropriate for modeling spontaneous speech. In principle, an HMM assumes piecewise transitions of states with respective distributions, even though it can incorporate dynamic spectral features such as ΔCepstrum. The segment model [32.32] and the trajectory model [32.33] have been proposed to capture the dynamics of speech. These models, however, have huge computational costs and often require the estimation of segmental boundaries. Thus, they are usually applied only in the rescoring stage after initial HMM-based decoding, and their potential may not have been fully exploited.

Another approach is to model syllables or functional words, instead of phones, in order to model several variants that are characteristic of frequent words such as 'that'. These types of models are often effective in context-independent models, but are difficult to extend to context-dependent models, which have enor-



Amazon EX1010 Page 651

mous numbers of combinations and encounter the data sparseness problem.

Many factors affect acoustic variation in spontaneous speech. A machine learning approach is to list and include these factors in the decision tree-based state clustering in context-dependent phone HMM. In this case, the context is extended from conventional phonetic context to any factors causing variations such as speaking rate [32.34], whether or not the phone concerned is included in a functional word, and whether or not the phone is stressed. This approach requires labeling of these kinds of information, which would be expensive if performed manually and unreliable if performed automatically. Moreover, the binary decision in the decision tree may not be appropriate, because the effects of these factors are not so apparent.

A more-generalized framework is dynamic Bayesian network (DBN) modeling [32.35] or graphical modeling [32.36,37], which includes these factors in the topology of an HMM-like network and estimates their influence in a probabilistic manner. One example that counts speaking rate is presented in the following subsection.

### 32.4.3 Models Considering Speaking Rate

As described in Sects. 32.3.2 and 32.3.3, one of the most important issues in spontaneous speech recognition is how to cope with the speaking rate fluctuation, especially fast speaking. As such, a number of studies have been conducted on this topic.

Initially, dedicated acoustic modeling to fast speech, where fast-speech models are prepared in parallel with the normal models, was investigated (e.g., [32.38–40]). However, this explicit approach is not adequate because it is not straightforward to classify the training data, especially frame by frame, to fast and poorly articulated samples.

Several studies have proposed dedicated acoustic analysis [32.41–43] for fast speech. These methods estimate the phone boundaries or speaking rates, and normalize the speaking rate by changing the analysis frame or acoustic model according to the estimated speaking rate.

*Okuda* et al. [32.44] proposed a speaking rate compensation method, which decodes an input utterance using several sets of frame period and frame length parameters for speech analysis. The method, then, selects the set with the highest likelihood normalized by the frame period. Furthermore, this approach was applied to the training phase of the acoustic model just like VTLN-based training.

*Shinozaki* et al. [32.45] proposed an acoustic model that adjusts mixture weights and transition probabilities of an HMM for each frame according to the local speaking rate. The proposed model, implemented based on a dynamic Bayesian network framework, has a hidden variable representing the variation of the *mode* of the speaking rate, and its value controls the parameters of the underlying HMM.

*Nanjo* et al. [32.46] proposed a hybrid approach, called speaking-rate-dependent decoding, which applies the most adequate acoustic analysis, phone models and decoding parameters according to the estimated speaking rate. Several methods were investigated and their selective application led to improved accuracy.

### 32.4.4 Pronunciation Variation Modeling

The phonetic variation caused by spontaneous utterance can also be modeled in a pronunciation dictionary, in which a list of possible phone sequences for each word is defined. While the orthodox pronunciation forms are referred to as baseforms, the variants observed in spontaneous speech are called *surface forms*. Several studies have also addressed a framework that jointly optimizes pronunciation entries and acoustic models [32.47], or the model unit itself [32.48].

These surface form entries are often derived from speech data by aligning them with phone models [32.26, 49]. In the CSJ, actual phonetic (*kana*) transcription is given manually, so the set of surface forms is easily defined. However, the simple addition of surface form entries results in the side-effect of false matching. Thus, effective but constrained use of these surface forms are necessary. One approach is context-dependent modeling, for example, dedicated surface forms are chosen according to the speaking rate [32.38, 50, 51].

Another approach is statistical modeling, which is similar to language modeling. Namely, the unigram probability of each pronunciation form is assigned and is multiplied by the language model probability in decoding [32.52–54]. In this case, the statistical framework of ASR is reformulated as:

$$w' = \arg\max_{w,p} P(x|p)P(p|w)P(w)\,. \qquad (32.2)$$

Here, $P(p|w)$ is the pronunciation probability of surface form $p$ for word $w$. *Nanjo* et al. [32.46] investigated the comparison of statistical models using the CSJ, and concluded that cutting off less frequent surface forms is crucial, and that the unigram model is effective, whereas the trigram model offered a marginal benefit.



Amazon EX1010 Page 652

When the surface form is derived for word units, it is dependent on the task and corpus and is not necessarily applicable to different tasks. Phone-based modeling of pronunciation variation is more general and portable to various lexicons. Surface forms are obtained by applying such a model to phone sequences of baseforms. As the modeling framework, the decision tree [32.55], the neural network [32.56], and the confusion matrix [32.57] have been proposed.

*Akita* et al. [32.58] proposed generalized modeling of subword-based mapping between baseforms and surface forms using variable-length phone context. The variation patterns of phone sequences are automatically extracted together with their contexts of up to two preceding and following phones, which are determined by their occurrence statistics. A set of rewrite rules are then derived with their probabilities and variable-length phone contexts. The model effectively predicts pronunciation variations depending on the phone context using a back-off scheme.

### 32.4.5 Language Model

Language model training for spontaneous speech is much more difficult than that for dictation systems, which can make use of huge language resources such as newspaper articles and Web pages. Most of the available language data are written text, and are mismatched with the spoken style. For language modeling of spontaneous speech, a great deal of transcription is essential, but has a huge cost.

Even the CSJ with a text size of 7M words, which is one of the largest spontaneous speech corpora, is comparable to or smaller than four months' worth of newspaper articles with respect to the text size. As shown in Fig. 32.8, increases in the amount of training data have a significant effect on ASR accuracy, and the improvement is not saturated even with the full CSJ.

Another problem in language modeling is how to deal with disfluency in spontaneous speech. Initially, insertion of fillers was assumed as an interruption of the $n$-gram chains, and so was often disregarded in $n$-gram training or prediction. Such a strategy may be effective when the disfluency is regarded as exceptional, as in dictation systems or when the size of the training corpus is small. When large corpora of spontaneous speech were built, however, the occurrence of fillers was confirmed to have such a regularity that the conventional $n$-gram model dealing with them as ordinary words was the most effective. Similarly, long pauses can be modeled as individual lexical entries in the $n$-gram model and are useful for improving the prediction of words.

The most widely used solution to enhance language model training data is to combine or interpolate with other existing text databases that are not necessarily spontaneous speech corpora but are related to the target task domain. These include proceedings of lectures, minutes of meetings, and closed captions for broadcast programs. Recently, the World Wide Web has become a major language resource [32.59], and not a few Web sites contain spoken-style documents, such as records of lectures and meetings. Several studies have addressed effective query generation to collect relevant Web pages [32.60, 61] and the selection of spoken-style texts [32.62].

*Akita* et al. [32.63] proposed a *translation* approach that estimates language model statistics ($n$-gram counts) of spontaneous speech from a document-style large corpus based on the framework of statistical machine translation (SMT). The translation is designed for modeling characteristic linguistic phenomena in spontaneous speech, such as insertion of fillers, and estimating their occurrence probabilities. These contextual patterns and probabilities are derived from a small parallel aligned corpus of faithful transcripts and their documented records. This method was successfully applied to the estimation of the language model for National Congress meetings from their minute archives.

### 32.4.6 Adaptation of Acoustic Model

Since the variation of acoustic features is very large in spontaneous speech, speaker adaptation of the acoustic model is effective and is almost essential. Although many factors affect the acoustic characteristics of spontaneous speech such as speaking rate and speaking styles, speaker adaptation is a simple solution to handle all of these factors in an implicit manner. The acoustic model adaptation also involves channel adaptation, that is, the characteristics of rooms and microphones are also normalized.

In particular, in lectures or meetings, each speaker makes many utterances in the same session. Thus, a considerable amount of data is available to conduct unsupervised adaptation in a batch mode, where the initial ASR result with the speaker-independent model is used for adaptation of the acoustic model, which is then used for rescoring or re-decoding. Standard adaptation techniques such as maximum-likelihood linear regression (MLLR) are used, and filtering the reliable



Amazon EX1010 Page 653

ASR hypotheses with confidence measures can also be incorporated.

### 32.4.7 Adaptation of Language Model

Adaptation of the language model is also important to deal with a variety of topics and speaking styles. In lectures and meetings, the topic is focused and consistent throughout the entire session. Therefore, language model adaptation is feasible even in an unsupervised or batch mode, as in the acoustic model adaptation, and computationally expensive methods can be allowed in offline transcription tasks.

The simplest methods are the cache model [32.64, 65], and the trigger model [32.66], which weigh the probability of words recently used in the utterances or talk, or those directly related to the previous topic words. *Troncoso* et al. [32.67] proposed a trigger-based language model adaptation method aimed at meeting transcription. The initial ASR result is used to extract task-dependent trigger pairs and to estimate their statistics. This method achieved a remarkable perplexity reduction of 28%. However, these cache or trigger models do not necessarily bring about a significant improvement in ASR accuracy because they were not robust against initial ASR errors, i. e., errors can propagate easily. Moreover, the methods are usually only applicable to the rescoring framework and are constrained by the quality of initial word graphs.

Thus, a potentially more-effective scheme is to adapt or reconfigure $n$-gram models for re-decoding, as in the acoustic model adaptation. *Yokoyama* et al. [32.68] constructed an $n$-gram model from the initial ASR result and interpolated it with the baseline model. They incorporated a class-based model for robust estimation from the small and erroneous ASR text [32.69]. *Lussier* et al. [32.70] extended the framework to optimize the interpolation weight using the EM (expectation-maximization) algorithm. *Nanjo* et al. [32.46] investigated methods to select the most relevant texts from the corpus [32.71] based on the initial ASR result. As a criterion for text selection, they used the tf-idf (term frequency and inverse document frequency) measure and perplexity by the $n$-gram model generated from the initial ASR result, and demonstrated that they have comparable and significant effects in reducing the WER. The results for the CSJ AP test set are summarized in Table 32.4.

Another approach is to model topics in an implicit manner, because the topics are not so definite and often become complex. One implementation of this concept is a mixture of multiple language models covering various topics [32.72]. Adaptation based on interpolation can be performed by weighting or emphasizing models relevant to the input speech.

Recently, latent semantic analysis (LSA), which maps documents into implicit topic subspaces using singular value decomposition (SVD), has been investigated extensively for language modeling [32.73]. A probabilistic formulation, PLSA (probabilistic latent semantic analysis) [32.74], is powerful for characterizing the topics and documents in a probabilistic space and predicting word probabilities. *Akita* et al. [32.75] proposed an adaptation method based on two subspaces of topics and speaker characteristics. Here, PLSA was performed on the initial ASR result to provide unigram probabilities conditioned on the input speech, and the baseline model is adapted by scaling $n$-gram probabilities with these unigram probabilities. The method was applied to automatic transcription of panel discussions and was shown to be effective in reducing both perplexity and WER.

A summary of the current ASR performance for the CSJ AP test set, using the baseline model enhanced with the complete corpus and a larger Gaussian mixture (160K in total), is given in Table 32.5. By combining the adaptation of acoustic and language models using the initial ASR result, the word accuracy was improved to 83%.

**Table 32.4** The effect of language model adaptation

| Method | WER | PP |
|---|---|---|
| Baseline* | 30.5 | 74.9 |
| Text selection by perplexity | 29.7 | 68.7 |
| Text selection by tf-idf | 29.1 | 70.2 |
| $n$-gram made from ASR result | 28.8 | 51.8 |
| Combination of all of the above | 27.6 | 46.7 |

* Baseline trained with an intermediate version of the CSJ

**Table 32.5** ASR results for CSJ/AP (academic presentations)

| Method | WER |
|---|---|
| Baseline | 22.2 |
| + MPE (minimum phone error) training | 19.9 |
| + AM adaptation (MLLR) | 18.1 |
| + LM adaptation ($n$-gram from ASR) | 17.1 |



Amazon EX1010 Page 654

## 32.5 Metadata and Structure Extraction of Spontaneous Speech

The transcript of spontaneous speech is not easy to read, even if it contains no recognition errors. The primary factor is disfluency, such as fillers, repetitions, self-repairs, and word fragments. These make spontaneous speech transcripts very different from written text. Actually, in the official meeting records of the Japanese Congress, we observe that approximately 11% of words are changed from the verbatim transcript (deletion 8.5%, insertion 1.0%, substitution 1.8%). Although there are a number of colloquial expressions corrected to more formal expressions, the great majority of the differences are caused by disfluencies. This means that even if we realize a perfect ASR system in the conventional criteria where speech is faithfully decoded into uttered word sequences, it has an edit distance (WER) of more than 10% from the text made by a human transcriber. This phenomenon is observed in any language. The ultimate intelligent transcription system should be able to clean these disfluencies automatically, but the first step is to model and detect these phenomena.

The second factor is segmentation of sentence and paragraph units, which is essential in written text. It seems that in most written languages the unit of the sentence is well defined, and thus can be objectively annotated, even though the end-of-sentence marks (the period in English) may differ. On the other hand, units smaller than the sentence (often marked by commas in English) and units larger than the sentence (marked by line breaks and indentation) are often subjective. Therefore, we focus on the sentence unit. Conventionally, punctuation marks are not counted when measuring ASR accuracy, and many ASR systems do not output these marks. In some dictation systems, users have to utter these marks explicitly, which is never expected in spontaneous speech. Thus, automatic segmentation of transcripts into sentences and paragraphs is another key factor in developing a *rich transcription* system.

Several studies in this direction have been performed for switchboard conversations [32.76]. In rich transcription (RT) evaluation of the DARPA EARS program, the metadata extraction (MDE) framework was designed for the following four tasks [32.77, 78].

- sentence unit (SU) detection
- edit word detection (repair detection)
- filler word detection
- interruption (disfluent) point (IP) detection

Here, the interruption point detection is, by definition, closely related to the detection of edit words and fillers.

Similarly in the CSJ, the following tags are given to the core of 0.5M words [32.22].

- sentence and clause boundaries
- filler tag (F tag)
- disfluency tag (D tag) for repairs and repetitions

We introduce a number of the key methods and results in the following subsections.

### 32.5.1 Sentence Boundary Detection

Detection of the sentence unit is vital for linguistic processing of spontaneous speech, since most of the conventional natural language processing (NLP) systems assume that the input is segmented by sentence units. Sentence segmentation is also an essential step to key sentence indexing and summary generation, which are described in the next section.

In spontaneous speech, especially in Japanese, in which subjects and verbs can be omitted, the unit of the sentence is not so evident. In the CSJ, therefore, the clause unit is first defined based on the morphological information of end-of-sentence or end-of-clause expressions. The sentence unit is then annotated by human judgment considering syntactic and semantic information.

Several approaches to automatic detection of sentence boundaries are described in the following subsections.

#### Statistical Language Model (SLM)

In fluent speech or read speech of well-formed sentences, it is possible to assume that long pauses can be interpreted as punctuation marks, and the insertion of periods (sentence boundaries) or commas can be determined by the neighboring word contexts.

Thus, the baseline method makes use of an $n$-gram statistical language model (SLM) that is trained using a text with punctuation symbols, in order to determine a pause to be converted to a period. Specifically, for a word sequence around a pause, $X = (w_{-2}, w_{-1}, \mathit{pause}, w_1, w_2)$, a period is inserted at the place of the pause if $P(W_1) = P(w_{-2}, w_{-1}, \text{period}, w_1, w_2)$ is larger than $P(W_2) = P(w_{-2}, w_{-1}, w_1, w_2)$ by some margin. Actually, this decoding is formulated as the maximization of a likelihood $\log P(W) + \beta \cdot |W|$, where $|W|$ denotes the number of words in $W$ and $\beta$ is the insertion penalty widely used in ASR.



Amazon EX1010 Page 655

In spontaneous speech, however, approaches that rely heavily on the pauses are not successful. Speakers put pauses in places other than the ends of sentences for certain discourse effects, and disfluency causes irregular pauses (interruption points), while consecutive sentences are often continuously uttered without a pause between them.

### Support Vector Machines (SVMs)

A simpler but more general approach is to treat the pause duration as one of the features in addition to the lexical features, and feed them into a machine learning framework. We adopted support vector machines (SVM) because there are a wide variety of cue expressions suggesting sentence endings in Japanese. In this case, sentence boundary detection is regarded as a text chunking problem [32.79], and we adopt the IE labeling scheme, where I and E denote the inside chunk and end of chunk, respectively. For every input word, a feature vector is composed of the preceding and the following three words, together with their POS tags and the durations of the subsequent pauses, if any. The pause duration is normalized by the average in a turn or a talk, because it is affected by the speaking rate and significantly different between speakers. Dynamic features or estimated results of preceding input parts can also be fed into SVM. The SVM is considered to be powerful for handling a very large number of features and finding the critical features called *support vectors*.

### Maximum-Entropy Method (MaxEnt) and Conditional Random Fields (CRF)

The maximum-entropy method (MaxEnt) [32.80, 81] is also widely used in NLP to combine multiple features $F$ in an exponential form and classify event $e$ given $F$:

$$p(e|F) = \frac{1}{Z(F)} \prod_i \mathrm{e}^{\lambda_i f_i(e,F)} , \qquad (32.3)$$

where $f_i(e, F)$ is binary depending on the match of the $i$-th feature in the current data $F$, and the weights $\lambda_i$ is trained to maximize $p(e|F)$ for training data. Similar features used for SVM can be fed into the model, although real-valued features such as pause duration are usually quantized into several bins.

Conditional random fields (CRFs) [32.82] are an extension of MaxEnt to handle sequential inputs. It is also an extension of HMM [32.83] so as to flexibly handle multiple features which may not be independent, and to be optimized for classification. Thus, sentence boundary detection is formulated as a chunking problem using the IOE labeling scheme or a similar scheme.

Unlike SVM, MaxEnt and CRF are statistical models, and they compute a posterior probability, which can be used as a confidence measure. This property is useful when we want to control the operating point of recall and precision, or in case that the result is combined with other processing using higher-level knowledge sources. On the other hand, SVM does not estimate probability distribution but focuses on classification boundaries, thus can be more robustly trained with sparse data of a huge dimension of features.

### Experimental Evaluations

First, we present the results evaluated in the CSJ [32.84]. The test set was that used for ASR evaluation and consists of 30 presentations or 71 000 words in total. Both SLM and SVM described in the previous subsections were trained with the core 168 presentations of 424 000 words, excluding the test set. In this experiment, we used ASR results without conducting speaker adaptation and the WER was approximately 30%. The results are summarized in Table 32.6, where the recall, precision, and F-measure are computed for sentence boundaries. Correct transcripts (text) are used for reference. SVM realizes higher performance in the text case, but significantly degraded it in the ASR case. On the other hand, SLM shows robustness for erroneous input. It is noteworthy that performance degradation by using ASR is much smaller than the WER.

In [32.85], *Liu* et al. reported the sentence boundary detection results in MDE tasks of rich transcription evaluation (RT-04). They trained HMM, MaxEnt, and CRF methods, which gave similar performance, and combined them by obtaining their majority vote for further improvement [32.86]. They used the boundary detection error rate, which counts deletion (miss) and insertion (false detection) errors. It is comparable to the summation of the complements of recall and precision if the number of detected boundaries is the same

**Table 32.6** Results of sentence unit (boundary) detection in the CSJ

| | Recall | Prec. | F-measure |
|---|---|---|---|
| SLM (text) | 79.2 | 84.6 | 81.8 |
| SLM (ASR) | 70.2 | 71.6 | 70.9 |
| SVM (text) | 82.7 | 88.0 | 85.3 |
| SVM (ASR) | 56.4 | 66.0 | 60.9 |



Amazon EX1010 Page 656

**Table 32.7** Summary of sentence unit detection

| | CTS | BN | CSJ |
|---|---|---|---|
| Training data size | 484K | 353K | 424K |
| Test-set size | 35K | 46K | 71K |
| WER of ASR | 14.9 | 11.7 | 30.2* |
| SU detection error (text) | 26.4 | 48.2 | 35.2 |
| SU detection error (ASR) | 36.3 | 57.2 | 57.7 |

* ASR result for the CSJ is not latest

as that of the correct boundaries. A summary comparison of the sentence unit detection for conversational telephone speech (CTS) and broadcast news (BN) is shown in Table 32.7, in which the results for the CSJ are also given for comparison. The performance for CTS is much better than for BN, because BN has more complex sentences, while in CTS, pronouns and back-channels work as good predictors for boundaries between shorter sentences. Moreover, speaker turn information, if available, will greatly help the system identify the end of sentences. Compared to these results, the performance for the CSJ monologue is not lower, despite the larger WER.

Liu et al. claimed that prosodic features are crucial to performance, working robustly against ASR errors. On the other hand, some works [32.87] have suggested that the most dominant prosodic feature is pause information, which is primarily used in the CSJ evaluation. Pitch and energy are apparently useful in the perception of sentence boundaries, but they are not easy to parameterize reliably. A typical solution is to introduce a separate classifier based on a decision tree or neural network that computes the likelihood integrating possible prosodic features [32.86, 88].

Another approach for further improvement is to incorporate higher-level linguistic information, such as syntactic dependency and caseframe structures. *Shitaoka* et al. [32.89] presented an interactive framework of parsing and sentence boundary detection, and showed that dependency structure analysis can help sentence boundary detection and vice versa. *Hamabe* et al. [32.90] addressed the detection of quotations, which is similar to sentence boundary detection, but involves analysis of very complex sentence structures.

### 32.5.2 Disfluency Detection

Disfluency is another prominent characteristic of spontaneous speech. Disfluency is inevitable because humans make utterances while thinking about what to say, and the pipeline processing is often clogged. Thus, the detection of disfluencies may be useful for analyzing the discourse structure or speaker's mental status. However, disfluencies should be removed for improving readability and applying conventional NLP systems including machine translation and summarization.

Disfluency is classified into the following two broad categories:

- fillers (such as 'um' and 'uh'), including discourse markers (such as 'well' and 'you know'), with which speakers try to fill pauses while thinking or to attract the attention of listeners;
- repairs, including repetitions and restarts, where speakers try to correct, modify, or abort earlier statements.

Note that fillers usually appear in the middle of repairs.

In the CSJ, specific tags are given for respective categories. The filler rate and the repair rate (ratios against the number of words) are listed in Table 32.3. Here, APs have more disfluencies because presenters speak much faster. In the RT-04 data, the filler rate is 9.2% for CTS and 2.1% for BN [32.85]. BN has fewer fillers because it is delivered mostly by professional anchors.

Lexical filler words are usually obtained as the output of the ASR system, and their recognition accuracy is much the same as that of ordinary words. However, there are a number of words that also functions as non-fillers such as 'well' in English and '*ano*' in Japanese. For these distinctions, prosodic features will be useful since we can recognize fillers even for unfamiliar languages. *Quimbo* et al. [32.91] investigated the difference in prosodic features in these words in Japanese.

On the other hand, the detection of self-repairs involves much more complex processes. The most conventional approach is to assume the repair interval model (RIM) [32.92], which consists of the following three parts in order: the reparandum (RPD), the portion to be prepared; the disfluency (DF), fillers or discourse markers; and the repair (RP), the portion to correct or replace the RPD, as shown below:

*(RPD) ! (DF) (RP)*
*(ex.) "I'm going {RPD: to Tokyo} ! {DF: no} {RP: to Kyoto}"*

The first step to the self-repair analysis based on this model is to detect the DF or interruption points (IP), denoted with '!' in the above, which seem rela-



Amazon EX1010 Page 657

**Table 32.8** Example of presumed discourse markers derived from academic presentations

| |
|---|
| *tsugi* (next), *sakihodo* (previously), *ima* (now), *jiQsai* (actually), *koNkai* (this time), *saigo* (lastly), *saisho* (firstly) |
| *keNkyuu* (study), *setsumei* (to explain), *haQpyou* (to present) |
| *keQka* (result), *jiQkeN* (experiment), *moNdai* (problem), *hyouka* (evaluation), *houhou* (method) |
| Listed in Japanese with corresponding English translations |

tively easy to spot. The DF usually consists of filler words, and IP detection can be formulated in much the same manner as the sentence boundary detection using neighboring lexical features together with prosodic features [32.93]. *Nakatani* et al. [32.92] pointed out the importance of prosodic features in this process. In rich transcription evaluation (RT-04), the patterns following the RIM are mainly focused, and the aforementioned approach was taken by *Liu* et al. [32.93]. For this purpose, they used the same classifiers based on HMM, MaxEnt, and CRF. After finding the DF or IP, the next step is to identify the starting point of the reparandum (RPD). Heuristic rule sets could be handcrafted for this purpose. A more elaborate statistical model was introduced to make alignment between the reparandum (RPD) and repair (RP) segments in [32.94]. As an even more general model, *Honal* et al. [32.95] introduced a statistical machine translation framework, which models self-repairs being caused through a noisy channel. These models work effectively if fillers are present in the DF position (following IP), and parts of RPD and RP segments are lexically matched, for example, 'on Monday, no, on Tuesday.' The latter assumption generally holds in English, in which phrases start with function words or pronouns and speakers repeat them in self-repairs. In this case, the removal of the RPD and DF is sufficient to clean the transcript.

In the CSJ or Japanese monologue, however, we observe many cases that do not satisfy this assumption or RIM. First, DF or filler words are often absent. Second, RPD and RP segments often have nothing in common on the surface level, although they may be semantically related, for example, "*ana* (hole) ! *mizo* (trench) *wa* . . . " This phenomenon makes it extremely difficult to perform machine learning using lexical features. Actually, using SVM, we obtained a detection accuracy (F-measure) of 77.1% for the cases in which the RPD and RP segments have some words in common, but only 20.0% otherwise. This result suggests that high-level semantic information is necessary for further improvement.

### 32.5.3 Detection of Topic and Discourse Boundaries

Topic and discourse boundary detection of spontaneous speech such as broadcast news and lectures has also been studied. It is formalized as a topic detection and tracking (TDT) task, where broadcast news is the major target and text-based techniques are applied and extended to ASR results. Incorporation of prosodic features has also been investigated. *Passonneau* and *Litman* [32.96] addressed discourse segmentation of narrative monologues using various combinations of prosodic features, cue phrase features, and noun-phrase reference features. They crafted decision rules combining these features both by hand and by machine learning. *Shriberg* et al. [32.83] applied machine learning techniques of decision tree learning and HMM, which were also used in the sentence boundary detection, to topic segmentation of broadcast news. *Haase* et al. [32.97] also tried discourse segmentation of monologue news reports using several prosodic features. *Kawahara* et al. [32.98] presented a method that automatically extracts discourse markers useful for the detection of topic boundaries in oral presentations of the CSJ. The presumed discourse markers were derived in a totally unsupervised manner based on word statistics and pause information. An example list of discourse markers derived from academic presentations (APs) is given in Table 32.8.

In dialogues or meetings involving multiple talkers, the detection of speaker turns is also critical. While the approach based on speaker identification techniques is commonly used, the use of linguistic information such as discourse information should be explored. Recently, speaker identification research itself is being extended to incorporate linguistic information.



Amazon EX1010 Page 658

## 32.6 Speech Summarization

As described in Sects. 32.3 and 32.4, ASR accuracy for spontaneous speech is not yet sufficient for quality automatic transcription. Recognition errors result in transcripts with irrelevant or incorrect information. In addition, spontaneous speech is ill-formed and usually includes redundant information caused by disfluencies. Direct transcription is therefore not always useful, and processes for extracting important information and removing irrelevant information are necessary for effective presentation of the transcript.

Automatic speech summarization is an approach aimed at accomplishing this goal. By condensing the important points of long audio materials or speech documents, such as broadcast programs and lectures, and presenting them in a concise form, the system will reduce the time needed for reviewing them and improve the effectiveness of their retrieval. Thus, speech summarization technology is expected to play an important role in building various speech archives, by providing a valuable means for efficiently accessing and absorbing more information in a much shorter time. It will also contribute to the advancement of question answering (QA) systems targeting speech documents, because both involve extracting the most salient information from such documents.

Speech summarization poses a number of significant challenges that distinguish it from general text summarization [32.99]. Applying text-based technologies [32.100] to speech is not always viable and systems are often not equipped to capture speech-specific phenomena. One fundamental problem with speech summarization is that it must deal with ASR errors and disfluencies [32.101, 102]. Another problem is that speech has no clear structure, whereas text contains not only sentences and paragraphs defined by punctuation and line breaking, but also titles for articles and individual sections, which provide useful information for summarization. These issues were addressed in the previous section, but automatic extraction of these kinds of information is inevitably error-prone.

### 32.6.1 Categories of Speech Summarization

Speech summarization can be classified into the following two categories based on its presentation form:

- speech-to-text summarization (output by text)
- speech-to-speech summarization (output by speech)

Speech-to-text summarization has advantages in that: (1) the documents can be easily looked through, (2) the parts of the documents that are of interest to the user can be easily spotted, and (3) information extraction and retrieval techniques can be easily applied to the documents. However, it also has disadvantages in that incorrect information caused by ASR errors is inevitable, and prosodic information such as nuance and emotion of the speaker, which is conveyed only in speech, cannot be presented. On the other hand, speech-to-speech summarization preserves the acoustic information included in the original speech.

Current approaches to speech summarization can be classified into two categories:

- sentence extraction (extracting important sentences or clauses as they are, and concatenating them in order)
- sentence compaction (generating summarized sentences by shortening, fusing, or even modifying the transcript)

The manual summarization process with text output by human editors corresponds to the latter, and the former can also be used as an intermediate step to the latter [32.103]. By applying sentence compaction, flexible output can be generated by concatenating key information and removing redundant expressions in speech. Users can then quickly browse and understand the content. Therefore, the sentence compaction approach is appropriate for speech-to-text summarization, and this method can be applied to any size of speech documents from headline news to long lectures.

In the case of speech-to-speech summarization, the extraction approach has an advantage in that the output can be easily made by concatenating corresponding segments from the original speech. On the other hand, the sentence compaction approach requires a text-to-speech (TTS) system, because concatenation would produce unnatural sounds with clipping noises and cannot be applied to units smaller than words. However, the quality of the synthesized speech, even by state-of-the-art TTS systems, is not as high as natural speech. In addition, the TTS-based method cannot mitigate the problems which inevitably result from ASR errors. Therefore, sentence extraction is more appropriate for speech-to-speech summarization, even though the reproduced speech includes redundant expressions and disfluencies. By indexing sentences, the system can provide users



Amazon EX1010 Page 659