

OPTICAL FIBER TELECOMMUNICATIONS SYSTEMS AND IMPAIRMENTS

Edited by

Ivan Kaminow / Tingye Liu

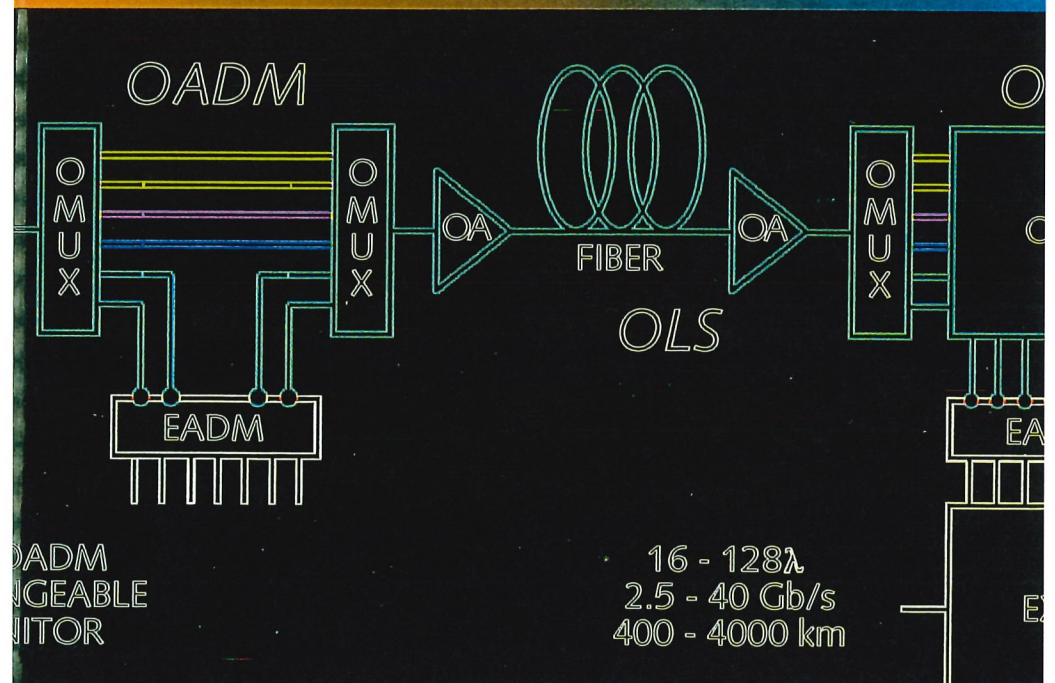


Exhibit 1015

Infinera Corporation v. Alexander Soto and Walter Soto

OPTICAL FIBER TELECOMMUNICATIONS IV B SYSTEMS AND IMPAIRMENTS

Edited by

IVAN P. KAMINOW

Bell Laboratories (retired)
Kaminow Lightwave Technology
Holmdel, New Jersey

TINGYE LI

AT&T Labs (retired)
Boulder, Colorado



ACADEMIC PRESS

An Elsevier Science Imprint

San Diego San Francisco New York
Boston London Sydney Tokyo

This book is printed on acid-free paper. (∞)

Copyright © 2002, Elsevier Science (USA). All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

The appearance of the code at the bottom of the first page of a chapter in this book indicates the Publisher's consent that copies of the chapter may be made for personal or internal use of specific clients. This consent is given on the condition, however, that the copier pay the stated per copy fee through the Copyright Clearance Center, Inc. (222 Rosewood Drive, Danvers, Massachusetts 01923), for copying beyond that permitted by Sections 107 or 108 of the U.S. Copyright Law. This consent does not extend to other kinds of copying, such as copying for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. Copy fees for pre-2001 chapters are as shown on the title pages. If no fee code appears on the title page, the copy fee is the same as for current chapters. \$35.00.

Academic Press

An Elsevier Science Imprint

525 B Street, Suite 1900, San Diego, California 92101-4495, USA

<http://www.academicpress.com>

Academic Press

Harcourt Place, 32 Jamestown Road, London NW1 7BY, UK

<http://www.academicpress.com>

Library of Congress Control Number: 2001098830

International Standard Book Number: 0-12-395173-9

PRINTED IN CHINA

02 03 04 05 06 07 RDC 9 8 7 6 5 4 3 2 1

Contents

<i>Contributors</i>	<i>xi</i>
Chapter 1 Overview <i>Ivan P. Kaminow</i>	1
Chapter 2 Growth of the Internet <i>Kerry G. Coffman and Andrew M. Odlyzko</i>	17
Chapter 3 Optical Network Architecture Evolution <i>John Strand</i>	57
Chapter 4 Undersea Communication Systems <i>Neal S. Bergano</i>	154
Chapter 5 High-Capacity, Ultra-Long-Haul Networks <i>John Zysskind, Rick Barry, Graeme Pendock, Michael Cahill, and Jinendra Ranka</i>	198
Chapter 6 Pseudo-Linear Transmission of High-Speed TDM Signals: 40 and 160 Gb/s <i>René-Jean Essiambre, Gregory Raybon, and Benny Mikkelsen</i>	232

Chapter 7	Dispersion-Managed Solitons and Chirped Return to Zero: What Is the Difference?	305
	<i>Curtis R. Menyuk, Gary M. Carter, William L. Kath, and Ruo-Mei Mu</i>	
Chapter 8	Metropolitan Optical Networks	329
	<i>Nasir Ghani, Jin-Yi Pan, and Xin Cheng</i>	
Chapter 9	The Evolution of Cable TV Networks	404
	<i>Xiaolin Lu and Oleh Sneizko</i>	
Chapter 10	Optical Access Networks	438
	<i>Edward Harstead and Pieter H. van Heyningen</i>	
Chapter 11	Beyond Gigabit: Application and Development of High-Speed Ethernet Technology	514
	<i>Cedric F. Lam</i>	
Chapter 12	Photonic Simulation Tools	564
	<i>Arthur J. Lowery</i>	
Chapter 13	Nonlinear Optical Effects in WDM Transmission	611
	<i>Polina Bayvel and Robert Killey</i>	
Chapter 14	Fixed and Tunable Management of Fiber Chromatic Dispersion	642
	<i>Alan E. Willner and Bogdan Hoanca</i>	
Chapter 15	Polarization-Mode Dispersion	725
	<i>Herwig Kogelnik, Robert M. Jopson, and Lynn E. Nelson</i>	

Chapter 16	Bandwidth-Efficient Modulation Formats for Digital Fiber Transmission Systems	862
	<i>Jan Conradi</i>	
Chapter 17	Error-Control Coding Techniques and Applications	902
	<i>P. Vijay Kumar, Moe Z. Win, Hsiao-Feng Lu, and Costas N. Georgiades</i>	
Chapter 18	Equalization Techniques for Mitigating Transmission Impairments	965
	<i>Moe Z. Win, Jack H. Winters, and Giorgio M. Vitetta</i>	
	Index to Volumes IVA and IVB	999

Chapter 17 | Error-Control Coding Techniques and Applications¹

P. Vijay Kumar

University of Southern California, Los Angeles, California and Scintera Networks, Inc., San Jose, California

Moe Z. Win

AT&T Labs-Research, Middletown, New Jersey

Hsiao-Feng Lu

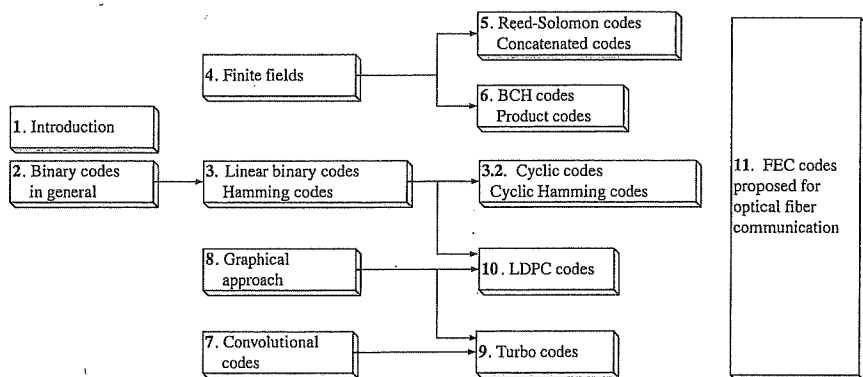
University of Southern California, Los Angeles, California

Costas N. Georghiades

Texas A&M University, College Station, Texas

17.1 Introduction

17.1.1 CHAPTER OVERVIEW



In this chapter, the forward error correction (FEC) codes relevant to the establishment of reliable communication over the fiber-optic communication channel are presented and discussed. Opinions expressed in this chapter are those of the authors and should not be construed as being universally held,

¹ This work was supported in part by the U.S. National Science Foundation under Grant NSF-CCR-0073555.

as the study of FEC design for optical communication has yet to mature. The accompanying flowchart provides an overview of the chapter and indicates the linkage between different sections. The reader wishing to obtain a quick overview of the topic of FEC for the fiber-optic channel can get by with just reading Sections 17.1 and 17.11. Sections 17.2 through 17.10 discuss the principles underlying the various FEC codes of interest.

Section 17.1.2 discusses how imperfections in the fiber-optic channel distort the transmitted signal. The role of FEC in overcoming some of these impairments is discussed in Section 17.1.3. The *coding gain* of an FEC code is a quantitative measure of the energy savings resulting from use of the code. The final section, Section 17.11, discusses the various FEC codes used or proposed for use on the optical channel, as well as the coding gains provided by these codes. Bounds on the maximum achievable coding gain are also provided here.

Sections 17.2 through 17.10 present the principles of error correction and provide background material on most of the FEC codes used or proposed for use for optical fiber communication. General binary error correcting codes are introduced in Section 17.2 followed in Section 17.3 by a discussion of a commonly used class of binary codes, namely, cyclic codes. The codes most often proposed for optical channel error correction are a class of nonbinary codes known as Reed-Solomon (RS) codes, and these codes are introduced in Section 17.5. RS codes are best described in the language of finite fields and Section 17.4 provides the relevant background. It is possible to combine a pair of codes to yield a third code through a process known as concatenation. In the typical construction of concatenated codes, one of the two constituent codes is a RS code. Concatenated codes are described in the latter part of Section 17.5. The largest and most well-known family of binary cyclic codes is the family of Bose, Ray-Chaudhuri, and Hocquenghem (BCH) codes. As with RS codes, the most natural description of these codes is in terms of finite fields, and Section 17.6 contains such a description. Product codes, like concatenated codes, are also constructed by combining a pair of codes. The product codes that appear in the optical communication literature are constructed from a pair of BCH codes. The discussion on product codes appears in Section 17.6.

The input to the optical fiber communication channel may be regarded as being binary, as the most common modulation scheme employed is on-off keying (OOK). It is convenient to break up the optical receiver into two blocks. The first block can either be a symbol estimator or else, a symbol detector. The symbol estimator puts out a positive real number x whose magnitude reflects the likelihood of a pulse having been transmitted at that time instant. The symbol detector consists of the symbol estimator described above followed by a threshold circuit that outputs one or zero depending upon whether the output of the symbol estimator exceeds or does not exceed an appropriate threshold. The choice of symbol estimator or symbol detector depends upon the type of FEC decoder used. A *hard-decision decoder* [44] is capable only of processing $\{0, 1\}$ inputs and therefore must be preceded by a symbol

detector. FEC decoders that are capable of processing the output of the symbol estimator without first quantizing this output to two levels, are called *soft-decision decoders*. Whereas most binary codes discussed in the literature possess a hard-decision decoding algorithm of reasonable complexity, the class of codes possessing a low-complexity, soft-decision decoding algorithm is much smaller. Hard-decision decoding is typically simpler to implement, but can result in an energy penalty on the order of one or two decibels (dB). Most decoders used or proposed for use on the optical fiber channel are hard-decision decoders, although recently, soft-decision decoders have begun to attract attention, see [1] for instance.

The three classes of codes discussed in Sections 17.7, 17.9, and 17.10, namely convolutional, turbo and low-density parity-check (LDPC) codes possess an efficient, low-complexity, soft-decision decoding algorithm. The low-complexity decoding algorithm comes about because these codes have a graphical representation that can be exploited by a decoder. Section 17.8 explains how one can obtain a graphical representation for these classes of codes and also shows how this representation leads to an efficient soft-decision decoding algorithm. The material presented in Sections 17.8–17.10 is of recent origin and is taken mostly from journal articles.

17.1.2 SIGNAL IMPAIRMENTS INTRODUCED BY THE OPTICAL CHANNEL

There are several causes for degradation of the optical signal as it progresses from transmitter to receiver. The three principal phenomena that affect the optical signal are attenuation, dispersion, and distortion due to noise [20].

Attenuation

Attenuation is the loss of optical energy, measured in dB/km, as the signal travels from transmitter to receiver.

Dispersion

Dispersion refers to the spreading of the optical signal in time during its passage down the fiber. There are two principal causes of dispersion in a single-mode fiber. *Chromatic* dispersion is pulse spreading caused by variation of the propagation constant of the electromagnetic light wave as a function of wavelength. The amount of dispersion increases linearly with distance and varies with wavelength. The dispersion is typically around 17 ps/km/nm at a wavelength of 1550 nm for conventional single-mode fibers, but is much lower (on the order of 2–6 ps/km/nm) in the case of recently installed non-zero-dispersion-shifted fibers. This dispersion can vary with time.

The second principal type of dispersion is *polarization-mode dispersion* (PMD) that arises because even in single-mode fibers, there are two modes of propagation distinguished by their polarization. The two modes travel with different group velocities because of fiber birefringence resulting once again, in pulse spreading. The amount of spreading in a commercial fiber varies as the square root of the length of the fiber. PMD also varies with time and wavelength.

A third cause of dispersion, *intermodal* dispersion, occurs only in multimode fibers. This is caused by the fact that the different modes of propagation are associated with different propagation constants. Multimode fibers are generally used to span short distances, such as those deployed in a local area network.

Dispersion of any of the types discussed above leads to inter-symbol interference (ISI), i.e., causes the channel output to depend not just on the corresponding input pulse, but also on adjacent pulses as well.

Noise

There are three principal sources of noise on an optical channel [18, 20]. *Shot noise* is the quantum noise due to the fact that the received signal at the output of a photodetector is a series of electrons with the number of electrons in a symbol interval having a Poisson distribution even when the incident light is of uniform intensity. *Thermal noise* is produced at the receiver and is typically modeled as an additive white Gaussian noise (AWGN). The third source of noise is *amplified spontaneous emission* (ASE) noise, which is generated when the signal is amplified using an optical amplifier. ASE noise can be modeled as being additive and Gaussian.

The most common form of detection is *direct detection*. Here, the incoming optical signal is converted to an electrical current proportional to the power of the incoming signal. In the absence of an optical amplifier, the major source of noise in a direct detection system is thermal noise. In the presence of an optical amplifier, the dominant source of noise is ASE [47].

In the case of *coherent detection*, the received signal is added to a local oscillator signal and the two signals are then converted to an electrical signal at a microwave frequency by a photodiode. This conversion makes it possible to process the converted signal using the array of signal-processing techniques developed for the microwave channel. However, coherent detection is harder to implement in practice and is not commonly employed.

The signal-to-noise-ratio (SNR) is the ratio of the average energy of the signal component of the input to the optical receiver to that of the noise present in the optical link. A precise definition of SNR appears in 17.11. The SNR has a direct relationship to the bit error rate (BER), with higher SNR leading to lower BER.

Other Sources of Distortion

There are several other sources of signal distortion. Chirping, self-phase modulation (SPM), stimulated Brillouin and Raman scattering are due to nonlinearities in the transmitting laser and fiber. A commonly-used technique to increase the rate of information transfer across an optical fiber is to use wavelength-division multiplexing (WDM). In WDM systems, fiber nonlinearities can give rise to additional sources of signal distortion known as cross-phase modulation and four-wave mixing.

17.1.3 THE ROLE OF FEC IN OPTICAL FIBER COMMUNICATION

There has been a recent upsurge in interest in using FEC [1, 17, 39, 46, 49] for communication over WDM fiber-optic channels. *Forward error correcting codes* tag extra redundant symbols to the message that can be exploited by the receiver to correct errors introduced by the optical channel. At the present time, FEC codes are designed and chosen primarily based on their ability to combat the additive ASE noise present on the WDM optical link. Although these codes can overcome to some extent the other sources of signal distortion discussed above, it is desirable that alternative means be used to combat these. For instance, equalization, either in the optical or electronic domain is an effective means of dispersion compensation. Equalization is discussed in greater detail elsewhere in this book.

In current optical WDM networks, bit error rates in the region 10^{-12} to 10^{-15} [17] are desirable, so when we speak of *reliable communication*, we will mean communication with BER in this region. The *bit-rate distance product*, i.e., the product of the length of fiber over which reliable transmission can be achieved times the data rate guaranteed on that length of fiber is an effective means of judging the impact of a technological innovation on the optical communication capability.

The error correction capability of an FEC code enables reliable communication at reduced energy levels, i.e., at reduced values of SNR. This SNR margin can be spent to improve the bit-rate distance product, for example by allowing the signal to reliably traverse a longer distance. Alternately, the SNR margin can be used to operate at a lower power level, thereby reducing nonlinear effects.

As mentioned previously, the *coding gain* of an FEC code is a quantitative measure of the savings in energy afforded by the code. A detailed discussion of coding gain appears in Section 17.11. The coding gains of some specific codes appear in this section, together with a discussion on the maximum-achievable coding gain.

We end with a few words on channel models. Consider an optical communication system in which the modulation is OOK and in which the first block in

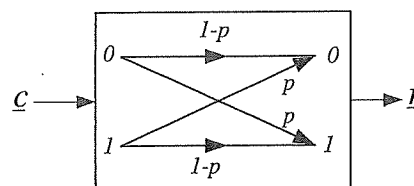


Fig. 17.1 The binary symmetric channel model.

the receiver is a symbol detector. The combination of optical fiber communication channel, and symbol detector effectively present to the hard-decision FEC decoder, a binary-input, binary-output channel (Fig. 17.1). It turns out that there is reasonable justification for assuming that the transition probabilities $0 \rightarrow 1$, $1 \rightarrow 0$ of this binary channel are equal. This topic is treated in greater detail in 17.11. As a result, the FEC decoder sees a symmetric binary-input binary-output channel known as the binary symmetric channel (BSC).

The next nine subsections are devoted to a discussion of the principles of error correction. Many of the codes discussed in these sections are designed to combat errors on the BSC.

17.2 Binary Codes in General

An excellent introduction to error correcting codes may be found in numerous textbooks [4, 6, 9, 24, 26, 31, 36, 41, 44]. A *block code* of length n is simply any subset of the set $\mathbb{F}_2^n$ of all binary n -tuples. Here we use $\mathbb{F}_2$ to denote the set $\{0, 1\}$. The elements of a code are called *codewords*. Each codeword is associated with a unique message. The *size* of a block code equals the number of codewords contained within the code. An example code is presented in Table 17.1.

To explain how the code accomplishes error correction we introduce the concept of Hamming weight and Hamming distance. The *Hamming weight*, $w_H(\underline{a})$ of a codeword $\underline{a}$ is the number of 1's in the n -tuple $\underline{a}$. The *weight distribution* $\{A_w | 0 \leq w \leq n\}$ of a code lists the number, A_w , of codewords of

Table 17.1 An Example Code

Codeword	Hamming Weight
(00000)	0
(01110)	3
(10101)	3
(11011)	4

Hamming weight w . Thus our example code has the weight distribution shown in Table 17.2.

The *Hamming distance*, $d_H(\underline{a}, \underline{b})$, between two n -tuples $\underline{a}$ and $\underline{b}$ is the number of symbols in which the two codewords differ. Table 17.3 lists the Hamming distances between all pairs of codewords in our length five code.

Given a received vector $\underline{r}$, a decoding algorithm known as the *minimum distance algorithm* attempts to find the codeword $\underline{c}$ closest in Hamming distance to $\underline{r}$ (see Fig. 17.2). This can be shown to be the optimal decoding algorithm from the point of view of minimizing the probability of incorrectly decoding a codeword when the channel is given to be the BSC.

Table 17.2 Hamming Weight Distribution of the Example Code

Weight w	0	1	2	3	4	5
A_w	1	0	0	2	1	0

Table 17.3 Hamming Distance Distribution of the Example Code

Codewords	00000	01110	10101	11011
00000	0	3	3	4
01110	3	0	4	3
10101	3	4	0	3
11011	4	3	3	0

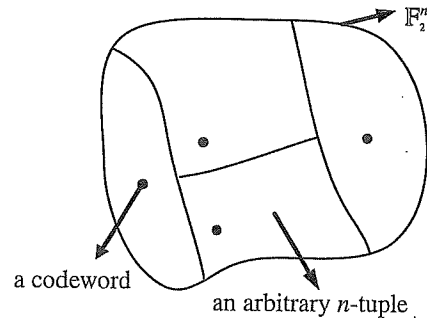


Fig. 17.2 Decoding of a binary code of length n . To decode a received n -tuple, one identifies the nearest codeword. This partitions the space $\mathbb{F}_2^n$ into regions as shown.

The *minimum distance*, $d_{\min}$, of a code is the smallest Hamming distance between a pair of distinct codewords within the code. Thus the minimum distance of our length five code equals 3. Whereas the exact error correction capability of a code depends generally upon the distance distribution between pairs of codewords, an estimate of this error correction capability can be obtained from knowledge of $d_{\min}$ as discussed below.

A code is said to be t_c error correcting and t_d error detecting, $t_d \geq t_c$, if the code can correct any error patterns that contain $\leq t_c$ errors, and in addition, is able to detect any pattern of t errors, $t_c < t \leq t_d$. For brevity we will refer to a t_c error correcting, t_d error detecting, $t_d \geq t_c$, code as a (t_c, t_d) -code. It turns out that a code having minimum distance $d_{\min}$ can be a (t_c, t_d) -code if and only if $d_{\min} \geq t_c + t_d + 1$. The decoding algorithm that makes this possible is called the *bounded-distance decoding algorithm* and runs as follows:

Given a received vector $\underline{r}$ the decoder searches to see if a codeword is present within Hamming distance t_c of the received vector. If this is the case, then the decoder declares that codeword to be the decoded codeword. If this is not the case, the decoder declares an uncorrectable error.

We explain why the decoding algorithm works. The Hamming distance function satisfies the triangle inequality which states that

$$d_H(\underline{x}, \underline{z}) \leq d_H(\underline{x}, \underline{y}) + d_H(\underline{y}, \underline{z}), \quad (17.1)$$

for any three binary n -tuples $\underline{x}, \underline{y}, \underline{z}$. The triangle inequality together with the condition $d_{\min} \geq t_c + t_d + 1$ can be used to show that the balls

$$B(\underline{c}, t_c) = \{n\text{-tuples } \underline{x} \mid d_H(\underline{x}, \underline{c}) \leq t_c\}, \quad (17.2)$$

of radius t_c centered around any codeword $\underline{c}$ are disjoint, i.e., if $\underline{c}_1, \underline{c}_2$ are distinct codewords,

$$B(\underline{c}_1, t_c) \cap B(\underline{c}_2, t_c) = \phi, \quad (17.3)$$

where ϕ denotes the empty set (see Fig. 17.3).

This explains how the code is able to recover if the number of errors is $\leq t_c$. If the number of errors t lies in the range $t_c < t \leq t_d$, then the received vector will not belong to any of the balls of radius t_c centered around codewords and the decoder will declare an uncorrectable error. Finally, when $t > t_d$, it is possible for the received vector $\underline{r}$ to lie in the ball of radius t_c surrounding a different codeword from that transmitted causing the decoder to decode incorrectly.

When a block code is used purely for the purpose of error correction, we have $t_c = t_d$, so the requirement on $d_{\min}$ changes to $d_{\min} \geq 2t_c + 1$. Given a block code of length n and minimum distance $d_{\min}$ satisfying $d_{\min} \geq 2t_c + 1$, the probability P_{cwe} of incorrectly decoding a codeword using the minimum distance decoding algorithm, over a BSC having crossover probability p , can

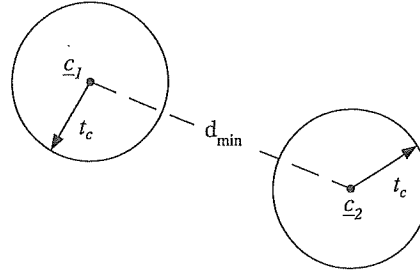


Fig. 17.3 Minimum distance $d_{\min}$ of the code causes balls of radius t_c centered around distinct codewords to be disjoint.

be upper bounded by

$$P_{\text{cwe}} \leq 1 - \sum_{i=0}^{t_c} \binom{n}{i} p^i (1-p)^{n-i}. \quad (17.4)$$

If the code is decoded using a bounded-distance decoding algorithm, then the right hand side of (17.4) gives the exact probability of decoding error.

17.3 Linear Binary Codes

Virtually all of the block codes commonly encountered in practice belong to the family of linear codes.

Modulo 2 Arithmetic

By modulo 2 arithmetic on the set $\{0, 1\}$, we mean addition and multiplication according to

$$0 + 0 = 0 \quad 0 + 1 = 1 + 0 = 1 \quad 1 + 1 = 0, \quad (17.5)$$

$$0 \cdot 0 = 0 \quad 0 \cdot 1 = 1 \cdot 0 = 0 \quad 1 \cdot 1 = 1. \quad (17.6)$$

The set $\mathbb{F}_2 = \{0, 1\}$ together with these operations forms an algebraic structure known as a field. A formal definition of field appears in the section on finite fields. Other more familiar examples of fields include $\mathbb{R}$, the set of all real numbers under the usual addition and multiplication rules, and $\mathbb{C}$, the set of all complex numbers under complex addition and multiplication.

The set $\mathbb{F}_2^n = \{0, 1\}^n$ of all binary n -tuples can be regarded as a vector space over the field $\mathbb{F}_2$. In this vector space, to add two vectors, $\underline{a}, \underline{b} \in \mathbb{F}_2^n$, one simply adds them component by component using modulo 2 addition. For example,

$$(011011) + (110111) = (101100). \quad (17.7)$$

Scalar multiplication is defined in the obvious way,

$$1 \cdot \underline{a} = \underline{a}, \quad 0 \cdot \underline{a} = \underline{0} = (0, 0, \dots, 0)^T. \quad (17.8)$$

Linear Block Codes

In the language of vector spaces, a linear block code $\mathcal{C}$ of length n is simply a subspace of $\mathbb{F}_2^n$. It turns out that a necessary and sufficient condition for a subset $\mathcal{S}$ of $\mathbb{F}_2^n$ to be a subspace is that

$$\underline{a} + \underline{b} \in \mathcal{S} \quad \text{whenever } \underline{a}, \underline{b} \in \mathcal{S}. \quad (17.9)$$

Our example code is linear. For instance, $(01110), (10101) \in \mathcal{C}$ and so is the modulo 2 sum $(01110) + (10101) = (11011)$.

In the case of linear codes, it is customary to speak of the *dimension* of a code in place of its size. An $[n, k]$ linear code $\mathcal{C}$ denotes a code $\mathcal{C}$ that is a subspace of $\mathbb{F}_2^n$ of dimension k . If $\mathcal{C}$ is an $[n, k]$ code, then there exist codewords $\{\underline{g}_0, \underline{g}_1, \dots, \underline{g}_{k-1}\}$ that form a basis for the code considered as a vector space over $\mathbb{F}_2$. Then every codeword $\underline{c} \in \mathcal{C}$ has a unique expression of the form

$$\underline{c} = \sum_{i=0}^{k-1} m_i \underline{g}_i, \quad m_i \in \mathbb{F}_2. \quad (17.10)$$

From this it follows that an $[n, k]$ linear code contains 2^k codewords and each codeword therefore represents one of 2^k possible messages. The coefficients $\{m_0, m_1, \dots, m_{k-1}\}$ are regarded as message bits. Equation (17.10) can be rewritten in matrix form as

$$\underline{c}^T = (c_0 c_1 \dots c_{n-1}) = \underline{m}^T \mathbf{G} = [m_0 m_1 \dots m_{k-1}] \begin{bmatrix} \underline{g}_0^T \\ \underline{g}_1^T \\ \vdots \\ \underline{g}_{k-1}^T \end{bmatrix}. \quad (17.11)$$

The $(k \times n)$ matrix $\mathbf{G}$ is called a *generator matrix* of the code $\mathcal{C}$. Generator matrices, in general, are not unique.

Example 1

Our example code of length 5 has a generator matrix:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (17.12)$$

and is clearly a $[5, 2]$ code.

Example 2

Consider the $[7, 1]$ code having generator matrix:

$$\mathbf{G} = [1111111]. \quad (17.13)$$

Here each codeword consists of seven repetitions of a single message symbol, and for this reason such a code is commonly called a repetition code.

Example 3

Our final example is the $[7, 4]$ code having generator matrix:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (17.14)$$

This code is capable of correcting a single error and is an example of a family of codes known as Hamming codes.

Minimum Distance

The third important parameter of a linear code is the *minimum distance*, $d_{\min}$. The Hamming distance $d_H(\underline{a}, \underline{b})$ between two binary n -tuples $\underline{a}, \underline{b}$ equals the Hamming weight of their modulo 2 difference, i.e.,

$$d_H(\underline{a}, \underline{b}) = w_H(\underline{a} - \underline{b}). \quad (17.15)$$

It follows that the minimum distance of a linear code $\mathcal{C}$ equals the minimum Hamming weight of a nonzero codeword in $\mathcal{C}$. Using this observation, it can be determined that the minimum distances of our example codes are as given in Table 17.4. Also shown in Table 17.4 are the (t_c, t_d) pairs for which the respective code can function as a t_c error correcting, t_d error detecting code. When it is desired to include the minimum distance $d_{\min}$ in the description of a linear code $\mathcal{C}$ of length n and dimension k , we will refer to the code as an $[n, k, d_{\min}]$ code.

Table 17.4 Parameters of the Example Codes

<i>Code</i>	$d_{\min}$	<i>Possible (t_c, t_d) Pair</i>
Length 5	3	(0, 2), (1, 1)
Repetition	7	(0, 6), (1, 5), (2, 4), (3, 3)
Hamming	3	(0, 2), (1, 1)

Parity Check Matrix

The *parity check* (PC) matrix $\mathbf{H}$ of an $[n, k]$ linear code $\mathcal{C}$ is any $((n - k) \times n)$ matrix of rank $(n - k)$ whose nullspace is precisely the k -dimensional vector space that is the code $\mathcal{C}$.

If the $[n, k]$ code $\mathcal{C}$ has generator matrix of the form

$$\mathbf{G} = \left[\mathbf{I}_k : \mathbf{P} \right], \quad (17.16)$$

for some $(k \times (n - k))$ matrix $\mathbf{P}$, the code is said to be *systematic*. It can be verified that in the case of systematic codes,

$$\mathbf{H} = \left[\mathbf{P}^T : \mathbf{I}_{n-k} \right] \quad (17.17)$$

is a valid PC matrix for the code.

The generators of all our three example codes are systematic. The corresponding PC matrices appear below:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

(Length 5 Code) (Repetition Code)

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (17.18)$$

(Hamming Code)

$d_{\min}$ from $\mathbf{H}$

The minimum distance $d_{\min}$ of a linear code $\mathcal{C}$ can also be determined from the parity check matrix. Consider for example the $[7, 4, 3]$ Hamming code. The seven-tuple $\underline{c}^T = (1011010)$ is a codeword since

$$\mathbf{H}\underline{c} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \underline{h}_0 + \underline{h}_2 + \underline{h}_3 + \underline{h}_5 = \underline{0}, \quad (17.19)$$

where $\underline{h}_i$, $0 \leq i \leq 6$, are the columns of $\mathbf{H}$. Thus the location of the nonzero components of a codeword $\underline{c}$ tells us that a linear dependence relation exists among the corresponding columns of the parity check matrix $\mathbf{H}$. It follows that if s is the *largest* integer such that any set of s columns drawn from $\mathbf{H}$ is linearly independent, we have $d_{\min} = s + 1$. In the case of the Hamming code, the columns of $\mathbf{H}$ are precisely the set of all nonzero binary three-tuples. It follows that in this case, $s = 2$ so that $d_{\min} = 3$, as was observed earlier.

Maximum Likelihood (ML) Decoding

We introduce here some terminology that will be used throughout the chapter. Let $\mathcal{C}$ be a linear code and let $\underline{c}$ be the codeword transmitted corresponding to message symbols $\{m_i\}_{i=0}^{k-1}$. (If the codeword is required to be fed to a modulator prior to transmission, we simply extend the channel model to include the modulator.) Let $\underline{r}$ be the received vector at the output of the channel.

In this setting, *ML codeword decoding* refers to decoding by a decoder that chooses the most likely codeword given the received vector $\underline{r}$, i.e., the ML codeword decoder decodes the codeword $\underline{c}$ that maximizes $P(\underline{r}|\underline{c})$ where P is the appropriate probability measure. Ties may be broken by tossing a coin for instance. It can be shown that under the assumption that all codewords $\underline{c}$ have an equal probability of being transmitted, ML codeword decoding minimizes the probability of codeword error. If the channel is a BSC with crossover probability $P < 1/2$, ML codeword decoding turns out to coincide with minimum distance decoding, i.e., decoding by choosing the codeword that is closest to $\underline{r}$ in Hamming distance.

Sometimes we will be interested in *ML message bit decoding*, denoted more simply at times as ML bit decoding. In this case, the receiver will decode each message bit separately. The receiver will decode the i^{th} message symbol m_i to be zero or one depending upon whether the likelihood ratio

$$\frac{P(m_i = 0|\underline{r})}{P(m_i = 1|\underline{r})} \quad (17.20)$$

is greater or less than one, respectively.

Standard Array Decoding

There is an efficient table-lookup procedure for carrying out ML codeword decoding of binary block codes having small redundancy over a BSC. This procedure, known as standard array decoding, can be found in most textbooks on coding theory.

Linear Codes with the Best Parameters

A list of the best-known binary block codes of length up to 512 may be found in [31], see also [7].

17.3.1 THE GENERAL HAMMING CODE

Let $m \geq 3$ be an integer and let $\mathbf{H}$ be an $(m \times (2^m - 1))$ matrix such that the columns of $\mathbf{H}$ precisely comprise the set of all nonzero m -tuples. Clearly this matrix has rank m . Arguing as in the case of the example Hamming code discussed earlier, we see that the code $\mathcal{C}$ having $\mathbf{H}$ as parity check matrix is a $[n = 2^m - 1, k = 2^m - 1 - m, d_{\min} = 3]$ linear block code. Any such code $\mathcal{C}$ is called a *Hamming code*.

Hamming codes are optimal with respect to a bound known appropriately enough as the *Hamming bound*, which states that the maximum size $A(n, d_{\min})$ of a code of block length n and minimum distance $d_{\min}$ satisfies:

$$A(n, d_{\min}) \leq \frac{2^n}{\sum_{i=0}^{t_c} \binom{n}{i}}. \quad (17.21)$$

Codes whose sizes achieve the Hamming bound with equality are known as *perfect codes*.

Hamming codes are perfect because:

$$2^k = \frac{2^n}{\sum_{i=0}^1 \binom{n}{i}} = \frac{2^{2^m-1}}{1 + 2^m - 1} = 2^{2^m-1-m}. \quad (17.22)$$

Hamming codes turn out to have the property of being cyclic, and for this reason they also appear in the next section.

17.3.2 CYCLIC CODES

The most commonly used linear block codes belong to a class of codes known as cyclic codes. Examples of cyclic binary codes include Hamming codes, the Golay code, and BCH codes. Reed-Solomon codes on the other hand, are cyclic but nonbinary. Cyclic binary codes are most naturally described in terms of polynomials.

Given a vector

$$\underline{c} = (c_0, c_1, \dots, c_{n-1}) \quad (17.23)$$

by *cyclic shift* of $\underline{c}$, we will mean any vector $\underline{c}'$ of the form

$$\underline{c}' = (c_\tau, c_{\tau+1}, \dots, c_{n-1}, c_0, \dots, c_{\tau-1}), \quad 0 < \tau \leq n-1. \quad (17.24)$$

A binary linear code $\mathcal{C}$ is said to be *cyclic* if whenever a codeword $\underline{c}$ belongs to $\mathcal{C}$, so does every cyclic shift $\underline{c}'$ of $\underline{c}$. Given a cyclic code $\mathcal{C}$ and a codeword $\underline{c} = (c_0, c_1, \dots, c_{n-1})$ we associate with the codeword, the *code polynomial*

$$c(x) = \sum_{i=0}^{n-1} c_i x^i. \quad (17.25)$$

Example 4

Consider the code $\mathcal{C}$ having generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \quad (17.26)$$

This code contains the eight codewords tabulated in Table 17.5.

It can be verified that the code is cyclic. We will alternatively regard a cyclic code $\mathcal{C}$ as a collection of code polynomials. It turns out that in every binary cyclic code the set of code polynomials has the property that it contains a unique nonzero polynomial $g(x)$ of minimum degree. This polynomial is called the *generating polynomial*. Our example cyclic code thus has the generating polynomial

$$g(x) = x^4 + x^3 + x^2 + 1. \quad (17.27)$$

The generating polynomial in any $[n, k]$ cyclic code has the following properties:

- (1) $g(x)$ has degree $= n - k$.
- (2) $g(x) \mid x^n + 1$ (i.e., $g(x)$ divides $x^n + 1$)
- (3) A polynomial over $\mathbb{F}_2$ of degree $\leq n - 1$ is a code polynomial if and only if it can be expressed in the form $c(x) = m(x)g(x)$ for some polynomial $m(x)$ of degree $\leq k - 1$.

Table 17.5 Codewords and Code Polynomial in an Example Cyclic Code

Codeword	Code		
	Polynomial	Degree	$m(x)$
0000000	0	$-\infty$	0
1011100	$x^4 + x^3 + x^2 + 1$	4	1
0101110	$x^5 + x^4 + x^3 + x$	5	x
0010111	$x^6 + x^5 + x^4 + x^2$	6	x^2
1110010	$x^5 + x^2 + x + 1$	5	$1 + x$
1001011	$x^6 + x^5 + x^3 + 1$	6	$1 + x^2$
0111001	$x^6 + x^3 + x^2 + x$	6	$x + x^2$
1100101	$x^6 + x^4 + x + 1$	6	$1 + x + x^2$

Example 5

Our example $[7, 3]$ code C has $g(x) = x^4 + x^3 + x^2 + 1$ of degree $= n - k = 7 - 3 = 4$. Also $g(x) \mid x^7 + 1$ since $x^7 + 1 = (x^4 + x^3 + x^2 + 1)(x^3 + x^2 + 1)$. Each code polynomial $c(x) \in C$ is expressible in the form $c(x) = m(x)g(x)$. For each code polynomial the associated polynomial $m(x)$ is displayed in Table 17.5.

From this it follows that a cyclic code of length n is completely specified once the generator polynomial $g(x)$ is given. Also, the $(k \times n)$ matrix $\mathbf{G}$, whose rows are code vectors in C associated to the code polynomials $\{x^i g(x) \mid 0 \leq i \leq k - 1\}$ can be verified to be a valid generator matrix for the code. The generator matrix $\mathbf{G}$ provided in (17.26) for our example $[7, 3]$ code is so constructed.

Cyclic Hamming Code

We have seen previously that the general Hamming code has parameters of the form:

$$[n, k, d_{\min}] = [2^m - 1, 2^m - 1 - m, 3], \quad m \geq 3 \quad (17.28)$$

and that these codes are most easily described in terms of a parity check matrix $\mathbf{H}$ of size $(m \times 2^m - 1)$ whose columns are precisely all nonzero binary m -tuples arranged in some order. It turns out that given an integer $m \geq 3$, setting $g(x)$ to be a primitive polynomial of degree m and $n = 2^m - 1$ results in a cyclic Hamming code. Primitive polynomials are discussed in Section 17.4.

Cyclic Golay Codes

Apart from Hamming codes, the only other interesting perfect binary code is the Golay code. The Golay code is a $[n = 23, k = 12, d_{\min} = 7]$ linear code. Moreover the code is cyclic corresponding to choice of generator polynomial:

$$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$$

or

$$g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1.$$

(17.29)

17.4 Finite Fields

The material in this section is essential for a good understanding of Reed-Solomon and BCH codes, discussed in Sections 17.5 and 17.6, respectively. Most textbooks on coding theory provide an introduction to finite fields.

Polynomials

A polynomial $f(x)$ over $\mathbb{F}_2$ is any expression in the indeterminate x of the form:

$$f(x) = \sum_{i=0}^n f_i x^i, \quad f_i \in \mathbb{F}_2. \quad (17.30)$$

The *degree* of a polynomial $f(x)$ is the largest exponent d of x whose associated coefficient $f_d \neq 0$. Addition of polynomials is carried out coefficient-wise, i.e.,

$$f(x) = \sum_{i=0}^n f_i x^i, \quad g(x) = \sum_{i=0}^n g_i x^i \Rightarrow f(x) + g(x) = \sum_{i=0}^n (f_i + g_i) x^i. \quad (17.31)$$

Multiplication is carried out in the usual way, i.e.,

$$\left(\sum_{i=0}^m f_i x^i \right) \cdot \left(\sum_{i=0}^n g_i x^i \right) = \sum_{k=0}^{m+n} h_k x^k, \quad (17.32)$$

where

$$h_k = \sum_{i=\max\{0, k-n\}}^{\min\{k, m\}} g_{k-i} f_i, \quad 0 \leq k \leq m+n. \quad (17.33)$$

Given two polynomials $f(x)$ and $g(x)$ of degree m, n with $m > n$, one can “divide” $f(x)$ by $g(x)$, using long division, to obtain a quotient $q(x)$ of degree $m - n$ and a remainder $r(x)$ of degree $< n$, i.e., we can write $f(x) = q(x)g(x) + r(x)$ with $\deg(q(x)) = m - n$, $\deg(r(x)) < n$.

Irreducible and Primitive Polynomials

A binary polynomial of degree m is said to be *irreducible* if it cannot be expressed as the product of two polynomials, each of lesser degree. The set of all irreducible polynomials of degree ≤ 4 is listed in Table 17.6. It turns out that every irreducible polynomial $f(x)$ of degree m divides $x^{2^m-1} + 1$.

Table 17.6 Irreducible Polynomials of Small Degree

<i>Degree</i>	<i>Irreducible Polynomials</i>
1	$x, x + 1$
2	$x^2 + x + 1$
3	$x^3 + x + 1, x^3 + x^2 + 1$
4	$x^4 + x + 1, x^4 + x^3 + 1, x^4 + x^3 + x^2 + x + 1$

Example 6

$$m = 2, f(x) = x^2 + x + 1$$

$$f(x) \mid x^3 + 1 \quad \text{since } (x^3 + 1) = (x + 1)(x^2 + x + 1). \quad (17.34)$$

$$m = 4, f(x) = x^4 + x^3 + x^2 + x + 1$$

$$\begin{aligned} f(x) \mid x^{15} + 1 \quad \text{since } (x^{15} + 1) &= (x^4 + x^3 + x^2 + x + 1) \\ &\times (x^{11} + x^{10} + x^6 + x^5 + x + 1). \end{aligned} \quad (17.35)$$

An irreducible polynomial $f(x)$ of degree m is said to be *primitive* if the smallest degree polynomial of the form $x^l + 1$ that $f(x)$ divides is $x^{2^m-1} + 1$.

Example 7

$f(x) = x^4 + x^3 + x^2 + x + 1$ is not primitive as $f(x) \mid x^5 + 1$ since

$$x^5 + 1 = (x + 1)(x^4 + x^3 + x^2 + x + 1). \quad (17.36)$$

However, the remaining degree 4 irreducible polynomials $x^4 + x + 1$ and $x^4 + x^3 + 1$ can be verified to be primitive.

A *field* is a set F of elements together with two operations, addition (+) and multiplication ($\cdot$), mapping pairs in F back to F , which satisfy the following three axioms:

- (1) The addition operation is associative and commutative and there is an additive identity element 0, called “zero” such that $a + 0 = a$ for all $a \in F$. Also, every element a has a unique additive inverse denoted by $(-a)$ that satisfies $a + (-a) = 0$.
- (2) The multiplication operation is associative and commutative and there is a multiplicative identity element called “1” such that $a \cdot 1 = a$ for all $a \in F$. Also, every nonzero element a has a unique multiplicative inverse, denoted by a^{-1} , that satisfies $a \cdot a^{-1} = 1$.
- (3) The distributive law holds, i.e.,

$$a \cdot (b + c) = a \cdot b + a \cdot c. \quad (17.37)$$

for all triples (a, b, c) in F .

For simplicity, we will often write ab in place of $a \cdot b$.

A *finite field* is simply a field containing a finite number of elements. Finite fields containing $\mathbb{F}_2$ relate to $\mathbb{F}_2$ in much the same way as the field $\mathbb{C}$ of complex numbers relates to the field $\mathbb{R}$ of real numbers. To obtain $\mathbb{C}$ from $\mathbb{R}$, one

introduces an imaginary element i that is a zero of the irreducible polynomial $f(x) = x^2 + 1$. Then $\mathbb{C}$ is simply the smallest field that contains $\mathbb{R}$ and i , and mathematically, we denote this by $\mathbb{C} = \mathbb{R}(i)$. Similarly, every finite field containing $\mathbb{F}_2$ can be described as the smallest field that contains $\mathbb{F}_2$ as well as the zero of a polynomial that is irreducible over $\mathbb{F}_2$. Moreover, this irreducible polynomial can be chosen to be primitive. Primitive polynomials are often chosen to construct finite fields since when constructed in this way, the finite field has a simpler description.

Example 8

The polynomial

$$p(x) = x^3 + x + 1 \quad (17.38)$$

is primitive over $\mathbb{F}_2$. Let α be a zero of $p(x)$, i.e., $\alpha^3 + \alpha + 1 = 0$. Set $F = \mathbb{F}_2(\alpha)$, i.e., F is the smallest field containing both $\mathbb{F}_2$ and α .

Every power of α can be expressed as a polynomial in α of degree ≤ 2 . For example, using $\alpha^3 + \alpha + 1 = 0$, we get $\alpha^3 = \alpha + 1$, $\alpha^4 = \alpha^2 + \alpha$, $\alpha^5 = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$, etc. Continuing in this fashion, we get Table 17.7. The eight distinct elements appearing in the table turn out to form the field $\mathbb{F}_2(\alpha)$. Thus we have two different representations of $\mathbb{F}_2(\alpha)$:

$$\mathbb{F}_2(\alpha) = \{0\} \cup \{\alpha^i | 0 \leq i \leq 6\} \quad (17.39)$$

and

$$\mathbb{F}_2(\alpha) = \{a_2\alpha^2 + a_1\alpha + a_0 | a_0, a_1, a_2 \in \mathbb{F}_2\}. \quad (17.40)$$

The exponential representation is convenient for the purposes of multiplication and division:

$$\alpha^3 \cdot \alpha^5 = \alpha^8 = \alpha \cdot \alpha^7 = \alpha \quad \text{and} \quad \frac{\alpha^3}{\alpha^4} = \frac{\alpha^3\alpha^7}{\alpha^4} = \alpha^6 \quad (17.41)$$

Table 17.7 Two Representations of Elements in the Finite Field of Eight Elements

<i>Exponential Representation</i>	<i>Polynomial Representation</i>	<i>Exponential Representation</i>	<i>Polynomial Representation</i>
0	0	α^4	$\alpha^2 + \alpha$
1	1	α^5	$\alpha^2 + \alpha + 1$
α	α	α^6	$\alpha^2 + 1$
α^2	α^2	α^7	1
α^3	$\alpha + 1$		

whereas the polynomial representation is convenient for addition:

$$(\alpha^2 + 1) + (\alpha + 1) = \alpha^2 + \alpha. \quad (17.42)$$

It turns out that every finite field containing $\mathbb{F}_2$ has size of the form 2^ℓ and that up to a relabeling of the elements, there is precisely one field of size 2^ℓ . We will use the notation $\mathbb{F}_{2^\ell}$ to refer to this field. The field $\mathbb{F}_{2^\ell}$ is also denoted by $GF(2^\ell)$ as finite fields are also called Galois fields in honor of the mathematician Evaristé Galois. All of the finite fields we consider here contain $\mathbb{F}_2$, and in the mathematical literature, such fields are referred to as finite fields of *characteristic two*.

As in the case of our example field, the general finite field $\mathbb{F}_{2^n}$ can be constructed by adjoining to $\mathbb{F}_2$, a zero α of a primitive binary polynomial $p(x)$ of degree n . This field contains 2^n elements and once again has two representations:

$$\mathbb{F}_{2^n} = \{0\} \cup \{\alpha^j \mid 0 \leq j \leq 2^n - 2\} \quad (17.43)$$

and

$$\mathbb{F}_{2^n} = \left\{ \sum_{j=0}^{n-1} a_j \alpha^j \mid a_j \in \mathbb{F}_2 \right\}. \quad (17.44)$$

The (multiplicative) *order* of a nonzero element θ in the finite field $GF(2^n)$ is the smallest exponent r such that $\theta^r = 1$. It can be shown that every nonzero element in $GF(2^n)$ has order dividing $2^n - 1$. Elements whose order equals $2^n - 1$ are said to be *primitive*. Thus the element α in our example finite field $GF(8)$ is primitive. It turns out that if $p(x)$ is a primitive binary polynomial of degree l and β is any zero of $p(x)$ in $GF(2^l)$, then β is a primitive element of $GF(2^l)$.

The commonly used representation of the finite field is the exponential representation. To facilitate addition under this representation, one sets up an add-1 table as shown in Table 17.8 in the case of our first example field $\mathbb{F}_8$.

Table 17.8 Add-1 Table for the Finite Field of Eight Elements

x	$x + 1$	x	$x + 1$
0	1	α^3	α
1	0	α^4	α^5
α	α^3	α^5	α^4
α^2	α^6	α^6	α^2

With the aid of the add-1 table, we can for instance, add

$$\alpha^3 + \alpha^5 + \alpha^6 = \alpha^3 (1 + \alpha^2 (1 + \alpha)) = \alpha^3 (1 + \alpha^2(\alpha^3)) \quad (17.45)$$

$$\alpha^3 + \alpha^5 + \alpha^6 = \alpha^3(1 + \alpha^5) = \alpha^3\alpha^4 = \alpha^7 = 1. \quad (17.46)$$

17.5 Reed-Solomon Codes

With this background in finite fields, we are now in a position to study the Reed-Solomon (RS) code. RS codes are in widespread use, for example in optical fiber communication [17], compact disks (CDs) [29] and deep-space communication [45]. In contrast to the codes discussed thus far, these codes are nonbinary.

Parallel to the binary case, a block code $\mathcal{C}$ of length n over the finite field $\mathbb{F}_q$, $q = 2^m$, $m \geq 1$, is said to be linear if whenever $\underline{c}_1, \underline{c}_2 \in \mathcal{C}$ implies that every linear combination

$$\theta_1 \underline{c}_1 + \theta_2 \underline{c}_2 \in \mathcal{C}, \quad \text{all } \theta_1, \theta_2 \in \mathbb{F}_q. \quad (17.47)$$

Again as in the binary case, a code $\mathcal{C}$ over $\mathbb{F}_q$ is said to be cyclic if $\mathcal{C}$ is linear and if in addition, whenever $(c_0, c_1, \dots, c_{n-1})$ is a codeword, so is $(c_\tau, c_{\tau+1}, \dots, c_{n-1}, c_0, c_1, \dots, c_{\tau-1})$, for $0 \leq \tau \leq n-1$.

Definition 1

Let $q = 2^m$, $m \geq 1$ and n be an integer dividing $q-1$ and α an element in $\mathbb{F}_q$ of order n . Then a RS code $\mathcal{C}$ having parameters $[n, k]$ over $\mathbb{F}_q$ can be defined as

$$\mathcal{C} = \left\{ (c_0, c_1, \dots, c_{n-1}) \mid \begin{array}{l} c_i = f(\alpha^i) \text{ for some polynomial } f(x) \\ \text{over } \mathbb{F}_q \text{ of degree } \leq k-1 \end{array} \right\}. \quad (17.48)$$

RS codes can be shown, using the definition given in Eq. (17.48), to be both linear and cyclic.

Minimum Distance

As in the binary case, we define the Hamming weight of an n -tuple over $\mathbb{F}_q$ to be the number of nonzero symbols in the n -tuple. Similarly, the Hamming distance between the pair of n -tuples $\underline{a}, \underline{b} \in \mathbb{F}_q^n$ equals the number of symbols in which the two vectors disagree. The Hamming distance $d_H(\underline{a}, \underline{b})$ between two n -tuples $\underline{a}, \underline{b}$ over $\mathbb{F}_q$, once again equals the Hamming weight of their difference, i.e.,

$$d_H(\underline{a}, \underline{b}) = w_H(\underline{a} - \underline{b}). \quad (17.49)$$

It follows that the minimum distance of a linear code $\mathcal{C}$ over $\mathbb{F}_q$ equals the minimum Hamming weight of a nonzero codeword in $\mathcal{C}$. Since a polynomial of degree $\leq k - 1$ can have at most $k - 1$ zeros in $\mathbb{F}_q$, the number of zeros in a nonzero codeword of an $[n, k]$ RS code cannot exceed $k - 1$. On the other hand, the polynomial

$$\prod_{i=0}^{k-1} (x - \alpha^i) \quad (17.50)$$

has exactly $k - 1$ zeros in $\mathbb{F}_q$. It follows that the minimal Hamming weight of a nonzero codeword in an $[n, k]$ Reed-Solomon code equals $n - k + 1$. By our previous observation, this must also be the minimum distance of the code. Thus, Reed-Solomon codes have parameters of the form $[n, k, n - k + 1]$. The Singleton bound states that a block code $\mathcal{C}$ of length n over an alphabet of size q having minimum distance $d_{\min}$ has size M upper bounded by

$$M \leq q^{n-d_{\min}+1}. \quad (17.51)$$

Codes achieving the Singleton bound with equality are said to be *maximum distance separable* (MDS). Thus RS codes are MDS.

Generator Matrix

A linear code of length n over $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$. The *dimension* of a linear code is the dimension of this subspace. As with binary codes, the notation $[n, k, d]$ refers to a code of length n , dimension k , and minimum distance d .

The notions of generator and parity-check matrices also carry over from the binary case. The generator matrix of any $[n, k, d]$ code over $\mathbb{F}_q$ is any $(k \times n)$ matrix whose rows form a basis for the code. Similarly, the parity-check matrix is any $((n - k) \times n)$ matrix whose nullspace is the $[n, k, d]$ code.

The generator matrix $\mathbf{G}$ shown in Eq. (17.52) follows directly from the definition of a RS code

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(n-1)} \end{bmatrix}. \quad (17.52)$$

If $g(x)$ is a polynomial of degree $\leq n - 1$ over $GF(q)$ satisfying $g(0) = 0$, then it can be shown that $\sum_{i=0}^{n-1} g(\alpha^i) = 0$. Setting $g(x) = x^i f(x)$, $1 \leq i \leq n - k$,

leads to the following parity-check matrix $\mathbf{H}$ of an RS code

$$\underbrace{\begin{bmatrix} 1 & \alpha & \cdots & \alpha^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha^{n-k} & \cdots & \alpha^{(n-k)(n-1)} \end{bmatrix}}_{\mathbf{H}}, \quad (17.53)$$

since $\mathbf{H}$ has the right rank and $\mathbf{H}\underline{c} = \underline{0}$ for any codeword $\underline{c}$ in the RS code.

Probability of Codeword Error

Let us assume that an $[n, k, d]$ RS code of length $n = 2^m - 1$ is used to transmit information over a BSC having crossover probability p . Let $d = 2t_c + 1$. To transmit information over the binary channel, each symbol $c_t \in \mathbb{F}_{2^m}$ of a codeword is converted into m binary digits. The probability P_s that any such symbol is received in error over the BSC is given by

$$P_s = 1 - (1 - p)^m. \quad (17.54)$$

Then following the argument used in the case of a general binary code in Section 17.2 to obtain an upper bound on the codeword error probability, we can upper bound the probability P_{cwe} of codeword error in the case of the RS code by

$$P_{\text{cwe}} \leq 1 - \sum_{i=0}^{t_c} \binom{n}{i} P_s^i (1 - P_s)^{n-i}. \quad (17.55)$$

Burst Error Correction

RS codes excel in burst error correction. Consider for example an $[255, 239, 17]$ RS code over $GF(256)$. Prior to transmission, each symbol in $GF(256)$ is converted to a string of eight binary digits, and these are transmitted in sequence. This effectively makes the RS code a binary code of length $8 * 255 = 2,040$. Since the RS code can correct up to eight symbol errors, the code will correctly decode even in the presence of a contiguous stream of up to $1 + 7 * 8 = 57$ erroneous binary digits, since such a string of errors will cause at most eight symbols over $GF(256)$ to be in error. Of course this burst error correction capability carries over to any nonbinary code with symbols in $GF(256)$. However, in comparison with other nonbinary codes of the same length over $GF(256)$, RS codes offer the largest minimum distance for given dimension.

Changing the Length of a RS Code

Given an $[n, k, d]$ RS code $\mathcal{C}$, it is possible to “shorten” the code to an $[n - a, k - a, d]$ code $\mathcal{C}_{sh}$ by considering the set $\mathcal{A}$ of all the codewords in $\mathcal{C}$ whose last a symbols equal zero and then deleting the last a symbols from each codeword to obtain the code $\mathcal{C}_{sh}$. We note that it is possible to define RS codes in a more general setting than what we have done here (see [26]).

Decoding

There are several efficient algorithms, such as the Berlekamp-Massey, Welch-Berlekamp, and Euclidean algorithms, for the decoding of RS codes, all with complexity on the order of the square of the length n of the code or less. Details can be found in most texts on coding theory, for example in [6, 26, 36, 41, 44], see also [4b, 22]. Under certain circumstances, it is possible to decode beyond the minimum distance of the RS code using a procedure known as list decoding, see [14b].

17.5.1 CONCATENATED CODES

Concatenated codes are linear binary codes that are constructed from two component codes called the *outer* and *inner* code. In the classical construction of concatenated codes as shown in Fig. 17.4, the outer code is typically a RS code and the inner binary code, a short block code. We illustrate with an example.

Example 9

Consider an $[255, 244, 12]$ RS code over $GF(256)$. Each symbol in this code corresponds to an 8-bit string. Replacing each 8-bit string by the corresponding 12-bit codeword in the $[12, 8, 3]$ single parity-check code yields a binary code with parameters $[12 \cdot 255, 244 \cdot 8, \geq 36]$. In general if the outer and inner codes have parameters $[N, K, D]$ and $[n, k, d]$, respectively, the concatenated code will have parameter $[Nn, Kk, \geq Dd]$ (Fig. 17.4).

Concatenated codes have large length and good minimum distance. The outer and inner codes have properties that complement each other. The RS code by itself, is vulnerable to isolated bit errors since each bit error can potentially cause a different code symbol to be in error. The inner code serves to

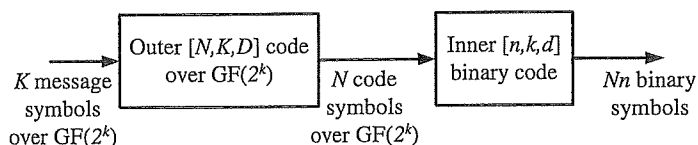


Fig. 17.4 An $[Nn, Kk, \geq Dd]$ concatenated code.

protect the RS code against isolated errors by providing an ability to either detect or correct them. Concatenated codes inherit from RS codes the ability to correct burst errors.

Whereas the complexity of decoding a block code of large length is high, concatenated codes are easily decoded using a suboptimal decoding algorithm (in relation to ML codeword decoding) in which the inner binary code is decoded first, followed by the decoding of the outer nonbinary code.

Although the outer code is typically a RS code, this is not essential and the RS code can be replaced by any efficient code with symbols lying in $GF(2^k)$, such as, for example, nonbinary BCH codes [26] or else algebraic-geometric codes [31]. Other means of concatenating two codes also exist, as we shall shortly see.

17.6 BCH Codes

BCH codes are a family of binary cyclic codes that are very flexible in terms of allowing a trade-off to be made between dimension and minimum distance. Also, for code lengths up to a few thousand, these codes are very efficient in terms of the performance trade-off that they offer.

Let n be a divisor of $2^p - 1$ for some integer $p \geq 3$. We assume that p is the smallest integer for which n divides $2^p - 1$. Let γ be a primitive element of $GF(2^p)$ and set $\alpha = \gamma^{(2^p-1)/n}$. Then α has order n .

Definition 2

Let n, p, α be as above. A t -error correcting BCH code $\mathcal{C}$ of length n is the set of all binary n -tuples that lie in the null-space of the $(t \times n)$ matrix

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \dots & \alpha^{(n-1)} \\ 1 & \alpha^3 & \alpha^6 & \dots & \dots & \alpha^{3(n-1)} \\ \vdots & \vdots & \vdots & \dots & \dots & \vdots \\ 1 & \alpha^{2l-1} & \alpha^{2(2l-1)} & \dots & \dots & \alpha^{(n-1)(2l-1)} \\ \vdots & \vdots & \vdots & \dots & \dots & \vdots \\ 1 & \alpha^{2t-1} & \alpha^{2(2t-1)} & \dots & \dots & \alpha^{(n-1)(2t-1)} \end{bmatrix}. \quad (17.56)$$

Note that if $\mathcal{N}$ denotes the nullspace of the above matrix over $GF(2^p)$, then the BCH code $\mathcal{C} = \mathcal{N} \cap \mathbb{F}_2^n$.

Example 10

Consider the single and double error correcting BCH codes of length $n = 15$. Here $t \in \{1, 2\}, p = 4, n = 15$. Let γ be a primitive element of $GF(16)$ satisfying

$\gamma^4 + \gamma + 1 = 0$. Here, since $n = 2^4 - 1$, we have $\alpha = \gamma$. The double-error-correcting BCH code is the set of binary 15-tuples that lie in the null-space of the matrix:

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \cdots & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \cdots & \alpha^{12} \end{bmatrix}. \quad (17.57)$$

For example, since

$$\begin{aligned} a(x) &= (1 + x + x^4)(1 + x + x^2 + x^3 + x^4) \\ &= 1 + x^4 + x^6 + x^7 + x^8 \end{aligned} \quad (17.58)$$

satisfies $a(\alpha) = a(\alpha^3) = 0$, we have that $[100010111000000]^T$ is a codeword in the BCH code. In the case of the single-error-correcting BCH code, the matrix $\mathbf{H}$ is given by

$$\mathbf{H} = [1 \ \alpha \ \alpha^2 \ \cdots \ \cdots \ \alpha^{14}]. \quad (17.59)$$

A more conventional (but less convenient) description of the BCH code can also be obtained. Each element α^l in the matrices has a unique representation in terms of the basis $\{1, \alpha, \alpha^2, \alpha^3\}$ for the four-dimensional vector space $\mathbb{F}_{16}$ over $\mathbb{F}_2$. Let the element α^l in the $\mathbf{H}$ matrix be replaced by its corresponding column vector. If this is done, in the case of the single-error-correcting BCH code, we will obtain the binary matrix

$$\mathbf{H}_{\text{bin}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \quad (17.60)$$

The fifth column of this matrix equals $[0011]^T$ since it corresponds to $\alpha^4 = 0 \cdot \alpha^3 + 0 \cdot \alpha^2 + 1 \cdot \alpha + 1 \cdot 1$. It can be shown that the BCH code can also be described as the set of all binary 15-tuples that belong to the null-space of this matrix. Since the columns of this (4×15) matrix comprise all distinct nonzero 4-tuples, this makes the single-error-correcting BCH code a cyclic Hamming code. This is always the case. For any length $n = 2^p - 1$, the single-error-correcting BCH code is a cyclic Hamming code.

It can be shown that the t -error correcting BCH code, as the name suggests, has minimum distance $d_{\min} \geq 2t + 1$. When the channel over which the codeword is transmitted is the BSC, BCH codes can be decoded using techniques very similar to those used to decode RS codes, see [6, 26, 36, 41, 44].

Because BCH codes are cyclic, they can be described in terms of a generator polynomial. It turns out that the generator polynomial of the BCH code is the smallest degree polynomial that has zeros

$$\alpha, \alpha^3, \dots, \alpha^{2t-1}, \dots, \alpha^{2t-1}. \quad (17.61)$$

Example 11

The single-error-correcting BCH code of length $n = 15$ has generator polynomial

$$g(x) = x^4 + x + 1, \quad (17.62)$$

since this is the smallest-degree polynomial of which the α in the example is a zero. The double-error-correcting BCH code of length $n = 15$ has, for the same reasons, the generator polynomial

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1. \quad (17.63)$$

Tables providing the parameters of BCH codes of lengths up to 1023 may be found, for example in [24].

17.6.1 PRODUCT CODES

Let C_1 and C_2 be $[n_1, k_1, d_1]$ and $[n_2, k_2, d_2]$ systematic codes respectively. The product code $C = C_1 \times C_2$ is a code having parameters $[n_1 n_2, k_1 k_2, d_1 d_2]$ and may be constructed as follows.

Let the $k_1 k_2$ information symbols be arranged in the form of a $(k_1 \times k_2)$ matrix A . Then construct an $(n_1 \times n_2)$ matrix B such that the matrix in the upper-left corner of B is precisely A (see Fig. 17.5). In addition, we require that the first k_1 rows of B are codewords in C_2 and that the first k_2 columns represent codewords in C_1 . Then let the remaining $((n_1 - k_1) \times (n_2 - k_2))$ submatrix in the lower-right corner of B be chosen such that the last $(n_1 - k_1)$ rows of B are also codewords in C_2 . (Alternately, the $((n_1 - k_1) \times (n_2 - k_2))$ matrix in the lower-right corner of B could be chosen in such a way that the last $(n_2 - k_2)$ columns in B are codewords in C_1 . It can be shown that either method yields the same $((n_1 - k_1) \times (n_2 - k_2))$ submatrix.) A straightforward argument shows that the minimum distance d of C satisfies $d = d_1 d_2$.

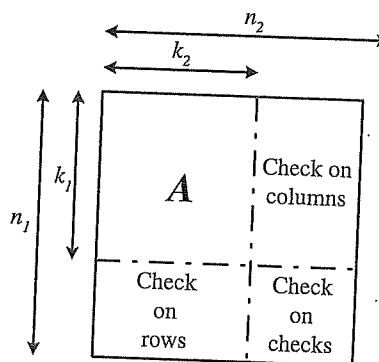


Fig. 17.5 An $[n_1 n_2, k_1 k_2]$ product code.

As with concatenated codes, product codes offer a means of constructing a block code of large length. Again there is a simple suboptimal decoding algorithm for product codes. One first decodes the rows of the code matrix $\mathbf{B}$ using a decoding algorithm for code C_2 to obtain the $(n_1 \times n_2)$ matrix $\mathbf{B}_1$. Next one decodes the columns of the matrix $\mathbf{B}_1$ using code C_1 , resulting in the $(n_1 \times n_2)$ matrix $\mathbf{B}_2$. The second decoding step thus attempts to correct errors resulting from incorrect decoding of rows by code C_2 . The $(k_1 \times k_2)$ matrix in the upper-left corner of $\mathbf{B}_2$ constitutes the decoded information bits. This procedure can be iterated to improve error correcting capability.

17.7 Convolutional Codes

Convolutional codes [9, 19, 24, 31, 36, 42–44] are examples of tree codes. The distinction between block and tree codes is that in the case of a block code it is possible to partition the input and output of the encoder into finite blocks in such a way that the i^{th} output block is a function only of the i^{th} input block. With tree codes, such a partition is, in general, not possible. Instead, it is possible for a particular output bit to be a function of all previous input bits. Convolutional codes are a special class of tree codes in which the input–output relation is given by a convolutional relation. We begin with an example.

Example 12

In the convolutional encoder shown in Fig. 17.6 there is one input stream $\{u_t\}_{t=0}^{\infty}$ and two output streams $\{v_t^{(1)}\}_{t=0}^{\infty}$ and $\{v_t^{(2)}\}_{t=0}^{\infty}$ related by

$$v_t^{(1)} = u_t + u_{t-1} + u_{t-2}, \quad t \geq 0 \quad (17.64)$$

$$v_t^{(2)} = u_t + u_{t-2}, \quad t \geq 0. \quad (17.65)$$

It is assumed that the convolutional encoder is initialized with $u_{-1} = u_{-2} = 0$. The final output of the encoder is the multiplexed stream $\{v_0^{(1)}, v_0^{(2)}, v_1^{(1)}, v_1^{(2)}, \dots\}$.

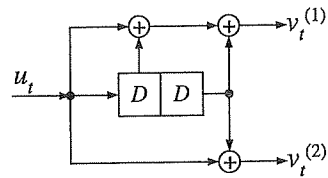


Fig. 17.6 An example convolutional encoder.

In the general case, there are k inputs and n outputs related by:

$$v_t^{(j)} = \sum_{i=1}^k \sum_{\tau=0}^{\nu} g_{\tau}^{(i,j)} u_{t-\tau}^{(i)}, \quad 1 \leq j \leq n, \quad (17.66)$$

where ν is the memory of the convolutional code. Thus our example encoder has memory $\nu = 2$. The convolutional code is said to have rate k/n , thus our example convolutional code is of rate $1/2$.

Consider the case when the inputs $\{u_t^{(i)}\}$ to the convolutional encoder are finite strings of length $N + \nu$ with the last ν bits on each string equal to 0. The outputs $\{v_t^{(j)}\}$ of the convolutional encoder are then finite strings as well. By multiplexing these output strings one obtains a vector $(v_0^{(1)}, \dots, v_0^{(n)}, v_1^{(1)}, \dots, v_1^{(n)}, \dots, v_{N-1+\nu}^{(1)}, \dots, v_{N-1+\nu}^{(n)})$. The collection of these vectors may be regarded as codewords in the convolutional code and in this way we may regard the convolutional code as a block code of large block length $n(N + \nu)$. The minimum distance between codewords as $N \rightarrow \infty$ is called the *minimum free distance*, d_{free} , of the convolutional code. Our example code turns out to have $d_{\text{free}} = 5$.

A simplified description of the input-output relationship in a convolutional code can be obtained through the use of power series. Let us define

$$u^{(i)}(D) = \sum_{t=0}^{\infty} u_t^{(i)} D^t, \quad 1 \leq i \leq k, \quad \text{and} \quad v^{(j)}(D) = \sum_{t=0}^{\infty} v_t^{(j)} D^t, \quad 1 \leq j \leq n. \quad (17.67)$$

Then the convolutional relation in (17.66) is replaced by

$$v^{(j)}(D) = \sum_{i=1}^k u^{(i)}(D) g^{(i,j)}(D), \quad 1 \leq j \leq n, \quad (17.68)$$

where $g^{(i,j)}(D) = \sum_{\tau=0}^{\nu} g_{\tau}^{(i,j)} D^{\tau}$.

Example 13

In the case of our example code we have

$$[v^{(1)}(D) \ v^{(2)}(D)] = [u(D)] [g^{(1)}(D) \ g^{(2)}(D)], \quad (17.69)$$

where $g^{(1)}(D) = 1 + D + D^2$ and $g^{(2)}(D) = 1 + D^2$. The matrix

$$G(D) = \begin{bmatrix} g^{(1,1)}(D) & g^{(1,2)}(D) & \dots & g^{(1,n)}(D) \\ \vdots & \vdots & & \vdots \\ g^{(k,1)}(D) & g^{(k,2)}(D) & \dots & g^{(k,n)}(D) \end{bmatrix} \quad (17.70)$$

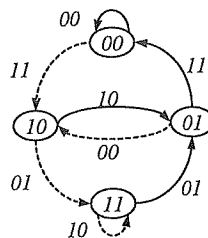


Fig. 17.7 State diagram of example code.

is called the *polynomial generator matrix* (PGM) of the code. In the case of the example code above, we have

$$G(D) = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 \end{bmatrix}. \quad (17.71)$$

An alternate description of the convolutional code is in terms of a *state diagram*. The state diagram of our example convolutional encoder is shown in Fig. 17.7. In the state diagram, the two symbols that form the label of each vertex are $(u_{i-1} \ u_{i-2})$. The solid edges correspond to input bits equal to 0 and the dotted edges represent input bits equal to 1. The label on each edge represents the output of the encoder corresponding to the input associated with the particular edge.

Recursive Systematic Convolutional Code

The conventional convolutional encoders considered thus far have finite impulse response. Consider a convolutional encoder with one input $\{u_i\}$ and two outputs $\{v_i^{(1)}\}$, $\{v_i^{(2)}\}$ whose associated power series $u(D) = \sum_{i=0}^{\infty} u_i D^i$, $v^{(1)}(D) = \sum_{i=0}^{\infty} v_i^{(1)} D^i$, $v^{(2)}(D) = \sum_{i=0}^{\infty} v_i^{(2)} D^i$, are related by

$$v^{(1)}(D) = u(D) \quad (17.72)$$

$$v^{(2)}(D) = \frac{u(D)[1 + D^4]}{1 + D + D^2 + D^3 + D^4}. \quad (17.73)$$

One method of implementing such an encoder is to introduce the auxiliary power series

$$s(D) = \sum_{i=0}^{\infty} s_i D^i = \frac{u(D)}{1 + D + D^2 + D^3 + D^4}. \quad (17.74)$$

We then have $u(D) = s(D)[1 + D + D^2 + D^3 + D^4]$, $v^{(1)}(D) = u(D)$ and $v^{(2)}(D) = s(D)[1 + D^4]$ leading to the time domain relations: $u_t = s_t + s_{t-1} + s_{t-2} + s_{t-3} + s_{t-4}$, $v_t^{(1)} = u_t$, $v_t^{(2)} = s_t + s_{t-4}$. This leads to the encoder shown in Fig. 17.8.

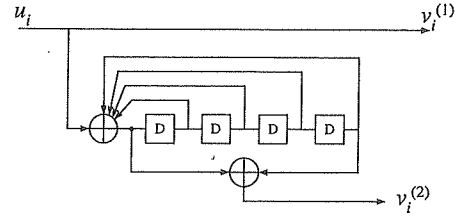


Fig. 17.8 A recursive systematic convolutional encoder.

In the figure, the contents of the four-bit shift register correspond to the 4-tuple $(s_{t-1}, s_{t-2}, s_{t-3}, s_{t-4})$. Note that this encoder is systematic since one of the two outputs coincides exactly with the input. Such convolutional encoders are referred to as *recursive systematic convolutional* encoders (RSC encoders) [5]. Here, in place of the PGM we have a matrix $\mathbf{G}(D)$ with rational polynomial entries:

$$\begin{bmatrix} v^{(1)}(D) & v^{(2)}(D) \end{bmatrix} = u(D) \underbrace{\begin{bmatrix} 1 & \frac{1+D^4}{1+D+D^2+D^3+D^4} \end{bmatrix}}_{\mathbf{G}(D)}. \quad (17.75)$$

RSC encoders form the building blocks of turbo codes, see Section 17.9.

Decoding of Convolutional Codes

The Viterbi algorithm [42, 43] is an algorithm for carrying out maximum-likelihood codeword decoding of a convolutional code. It is of low complexity whenever the number of states in the state diagram of the convolutional code is small. Maximum-likelihood decoding of the information bits of a convolutional code via the BCJR algorithm [3] is discussed in Section 17.8.1.

17.8 Graphical Approach to ML Decoding of Binary Codes

Claude Shannon has shown that given a communication channel, there is a quantity called *channel capacity*, C , [10] such that it is possible to transmit information at rate R across the channel reliably, provided $R < C$. Reliable communication means communication with bit-error probability arbitrarily close to zero. Reliable communication calls for the use of codes of large block length as long block codes are able to average out random distortions introduced by the channel, and hence make the channel more predictable.

If reliable communication is interpreted as communication at bit error rate on the order of 10^{-5} , then it has been shown that turbo and LDPC codes have

come extremely close (within fractions of a dB) of achieving capacity on the AWGN channel.

A second feature that both turbo and LDPC codes share in common is that in both cases, decoding is accomplished by passing messages iteratively around in a graph that represents the code. Efforts are currently underway at explaining the superlative performance of such a decoding scheme.

Here we content ourselves with showing how the distributive law, applied in a more general setting, can be used to provide a heuristic justification for the use of message passing in a graph to approximate ML decoding of a code.

17.8.1 ML DECODING VIA THE DISTRIBUTIVE LAW

In this rather long section, we lay down some principles that underline the decoding of turbo [5] and low-density parity-check codes [13]. The principal reference for the material in this section is the paper by Aji and McEliece [2], see also [8, 23].

The Distributive Law

The distributive law may be viewed as a rule that allows a savings in computation. For example, the distributive law

$$ab_1 + ab_2 = a(b_1 + b_2) \quad (17.76)$$

reduces three operations down to two. A second example is given in Example 14.

Example 14

Let A be a set of size q and let f and g be real-valued functions defined on 3-tuples and 2-tuples from A , respectively. Suppose it is desired to compute

$$F(x_1, x_4) = \sum_{x_2, x_3 \in A} f(x_1, x_2, x_4)g(x_1, x_3). \quad (17.77)$$

Direct computation would require $q^2[q^2(1) + (q^2 - 1)] = 2q^4 - q^2$ operations. However, the distributive law can be used to reorganize the computation as

$$F(x_1, x_4) = \sum_{x_2 \in A} f(x_1, x_2, x_4) \sum_{x_3 \in A} g(x_1, x_3) = \beta(x_1, x_4)\gamma(x_1), \quad (17.78)$$

where $\beta(x_1, x_4) = \sum_{x_2 \in A} f(x_1, x_2, x_4)$ and $\gamma(x_1) = \sum_{x_3 \in A} g(x_1, x_3)$.

Computing $\beta(x_1, x_4)$ for all (x_1, x_4) pairs requires $q^2(q - 1)$ operations. Computing $\gamma(x_1)$ requires $q(q - 1)$ operations. Thus a total of

$$q^2(q - 1) + q(q - 1) + q^2 = q^3 + q^2 - q \quad (17.79)$$

operations is now required, which could be a significant saving in computation.

The MPF Problem and ML Decoding

We next present the problem of marginalizing a product function (MPF) and show how the generalized distributive law (GDL) can be used to present an efficient solution.

Let $x_1, x_2, \dots, x_n$ be variables taking on values from the set $A_1, A_2, \dots, A_n$, respectively. Let q_i be the size of A_i . Let

$$S = \{x_1, x_2, \dots, x_n\} \quad (17.80)$$

and let $S_1, S_2, \dots, S_M$ be subsets of S , not necessarily distinct.

Each set S_i will be called a *local domain*. We associate to each local domain S_i a *local kernel* $\alpha_i : S_i \rightarrow \mathbb{R}$ where $\mathbb{R}$ denotes the set of all real numbers and introduce a *global kernel* $\beta(x_1, x_2, \dots, x_n)$ given by

$$\beta(x_1, x_2, \dots, x_n) = \prod_{i=1}^M \alpha_i(S_i). \quad (17.81)$$

By the S_i -marginalization of $\beta(x)$ we will mean the sum

$$\beta_i(x) = \sum_{S \setminus S_i} \beta(x) \quad (17.82)$$

and we will refer to this sum as the i^{th} *objective function*. (The notation $S \setminus S_i$ is a reference to the subset of the elements in S that do not belong to S_i .) Thus the MPF problem aims to efficiently compute the S_i -marginalization of the product function $\beta(\cdot)$. Our earlier example can be recast in this notation.

Example 15

The computation

$$F(x_1, x_4) = \sum_{x_2, x_3} f(x_1, x_2, x_4) g(x_1, x_3) \quad (17.83)$$

can be viewed as the marginalization of a global kernel as follows: Set

$$S = \{x_1, x_2, x_3, x_4\}, S_1 = \{x_1, x_2, x_4\}, S_2 = \{x_1, x_3\}, S_3 = \{x_1, x_4\} \quad (17.84)$$

and introduce the local kernels:

$$\alpha_1(S_1) = f(x_1, x_2, x_4), \alpha_2(S_2) = g(x_1, x_3), \alpha_3(S_3) = 1. \quad (17.85)$$

Then since

$$F(x_1, x_4) = \sum_{S \setminus S_3} \alpha_1(x_1, x_2, x_4) \alpha_2(x_1, x_3) \alpha_3(x_1, x_4) \quad (17.86)$$

it is the S_3 -marginalization of the global kernel

$$\beta(x_1, x_2, x_3, x_4) = \alpha_1(x_1, x_2, x_4) \alpha_2(x_1, x_3) \alpha_3(x_1, x_4). \quad (17.87)$$

For our second example, we consider the decoding of a simple binary linear code.

Example 16 (ML code symbol decoding of a block code)

Let C be the binary linear code having parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad (17.88)$$

Consider the problem of decoding the code over a BSC. Let $\underline{x}$ be the transmitted codeword, $\underline{e}$ the error pattern introduced by the BSC, and $\underline{y}$ the received vector given by

$$\underline{y} = \underline{x} + \underline{e}. \quad (17.89)$$

The goal under *ML code symbol decoding* is to find the i^{th} codeword symbol, $x_i \in \mathbb{F}_2$, $i = 1, 2, \dots, 7$, such that $P(x_i|\underline{y})$ is a maximum. Although this goal does not correspond to either ML codeword decoding or to ML message-bit decoding, it is sometimes easier to formulate, and it can be shown that under reasonable assumptions, this decoding technique will, with high probability, pick the code symbols of the transmitted codeword.

Assuming all the codewords are equally likely, we can rewrite

$$P(x_i = a|\underline{y}) \propto \sum_{\underline{x}: x_i = a} P(\underline{y}|\underline{x}) \chi_C(\underline{x}) = \sum_{\underline{x}: x_i = a} \prod_{j=1}^7 P(y_j|x_j) \chi_C(\underline{x}), \quad (17.90)$$

where the indicator function χ_C is given by:

$$\chi_C(\underline{x}) = \begin{cases} 1, & \underline{x} \in C, \\ 0, & \text{else,} \end{cases}$$

and where the $\propto$ sign indicates that we have ignored scale factors ($P(\underline{y})$ in this case) that are independent of the symbol a .

From the parity-check matrix, we have that

$$\chi_C(\underline{x}) = \chi_A(x_1, x_2, x_4) \chi_B(x_3, x_4, x_6) \chi_C(x_4, x_5, x_7) \quad (17.91)$$

where

$$\chi_A(x_1, x_2, x_4) = \begin{cases} 1, & x_1 + x_2 + x_4 = 0, \\ 0, & \text{else,} \end{cases} \quad (17.92)$$

$$\chi_B(x_3, x_4, x_6) = \begin{cases} 1, & x_3 + x_4 + x_6 = 0, \\ 0, & \text{else,} \end{cases} \quad (17.93)$$

and

$$\chi_C(x_4, x_5, x_7) = \begin{cases} 1, & x_4 + x_5 + x_7 = 0, \\ 0, & \text{else.} \end{cases} \quad (17.94)$$

We introduce the local kernel and domains in Table 17.9. Let $\sigma_i(x_i)$ be the S_i -marginalization, $1 \leq i \leq 7$,

$$\sigma_i(x_i) = \sum_{S \setminus S_i} \beta(x_1, x_2, \dots, x_7), \quad (17.95)$$

where

$$\beta(x_1, x_2, \dots, x_7) = \prod_{i=1}^7 P(y_i | x_i) \chi_A(S_8) \chi_B(S_9) \chi_C(S_{10}). \quad (17.96)$$

Then

$$\frac{\sigma_i(0)}{\sigma_i(1)} = \frac{P(x_i = 0 | y)}{P(x_i = 1 | y)}, \quad (17.97)$$

Table 17.9 Local Domains and Kernels for
ML Symbol Decoding Example

<i>Local Domain</i>	<i>Local Kernel</i>
$S_1 = \{x_1\}$	$P(y_1 x_1)$
$S_2 = \{x_2\}$	$P(y_2 x_2)$
$\vdots$	$\vdots$
$S_7 = \{x_7\}$	$P(y_7 x_7)$
$S_8 = \{x_1, x_2, x_4\}$	$\chi_A(x_1, x_2, x_4)$
$S_9 = \{x_3, x_4, x_6\}$	$\chi_B(x_3, x_4, x_6)$
$S_{10} = \{x_4, x_5, x_7\}$	$\chi_C(x_4, x_5, x_7)$

and thus the problem of ML code symbol decoding of the $[7, 4, 2]$ block code can be viewed as an instance of the MPF problem. If $\hat{\underline{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_7)$ is the decoded codeword, we set

$$\hat{x}_i = \begin{cases} 1, & \text{if } \sigma_i(1) \geq \sigma_i(0), \\ 0, & \text{else.} \end{cases}$$

Our next example considers the problem of ML decoding of a binary convolutional code.

Example 17 (ML message-bit decoding of a convolutional code)

Let $\{u_i\}_{i=0}^{N-1}$, $\{s_i\}_{i=0}^{N-1}$, and $\{y_i\}_{i=0}^{N-1}$ denote the sequence of inputs of a convolutional encoder, the sequence of states of the convolutional encoder and the sequence of channel outputs respectively. For simplicity, we assume that the convolutional encoder is of rate $R = 1/n$ and memory ν . Thus each u_i is binary, each state s_i represents a binary ν -tuple, and each output y_i is an n -tuple. For ML message-bit decisions, we need to compute

$$P(u_i = a | \{y_j\}_{j=0}^{N-1}) \propto \sum_{\substack{u_0, u_1, \dots, u_{i-1}, \\ u_{i+1}, \dots, u_{N-1}}} \sum_{s_0, s_1, \dots, s_{N-1}} P(\{u_i\}_{i=0}^{N-1}, \{s_j\}_{j=0}^{N-1}, \{y_j\}_{j=0}^{N-1}). \quad (17.98)$$

This can be rewritten in the form:

$$\begin{aligned} P(u_i = a | \{y_j\}_{j=0}^{N-1}) \\ \propto \sum_{\substack{\{u_j\}_{j=0}^{i-1} \\ \{u_j\}_{j=i+1}^{N-1}}} \sum_{\{s_j\}_{j=0}^{N-1}} \left(\prod_{i=0}^{N-1} P(u_i) \right) P(s_0) \prod_{j=1}^{N-1} P(s_j | s_{j-1}, u_{j-1}) \prod_{j=0}^{N-1} P(y_j | s_j, u_j). \end{aligned} \quad (17.99)$$

Figure 17.9 presents a graph known as a Bayesian network that depicts the statistical relationship between the various random variables.

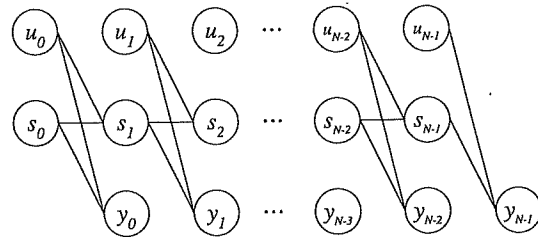


Fig. 17.9 Bayesian network of a convolutional code.

Table 17.10 Local Domains and Kernels for ML Decoding of the Convolutional Code

<i>Local Domain</i>	<i>Local Kernel</i>
u_j	$P(u_j), 0 \leq j \leq N-1$
s_0	$P(s_0)$
$\{s_j, s_{j-1}, u_{j-1}\}$	$P(s_j s_{j-1}, u_{j-1}), 1 \leq j \leq N-1$
$\{s_j, u_j, y_j\}$	$P(y_j s_j, u_j), 0 \leq j \leq N-1$

ML message-bit decisions are made according to

$$P(u_i = 0 | \{y_j\}) \underset{\substack{\text{say } u_i = 0}}{\overset{\substack{\text{say } u_i = 1}}{\leq}} P(u_i = 1 | \{y_j\}). \quad (17.100)$$

With local domains and kernels as shown in Table 17.10 and consequent global kernel

$$\beta(\{u_j, s_j, y_j\}_{j=0}^{N-1}) = P(\{u_j, s_j, y_j\}_{j=0}^{N-1}), \quad (17.101)$$

we see that the problem of ML message-bit decoding of a convolutional code can once again be recast as an MPF problem.

Junction Trees

The first step in applying the distributive law to solve the MPF problem is to set up a labeled graph known as a *junction tree*, in which each node in the graph is associated to a local domain. If it is not possible to organize the local domains into a junction tree then one has to modify this approach. When it is possible to organize the local domains into a junction tree, an algorithm known as Prim's greedy algorithm (see footnotes on p. 359 of [2]) may be employed to construct the junction tree. The label for a node in the graph is just the set of variables making up the local domain associated to that node.

A tree may be defined as a graph in which there is a unique path between any two vertices. A junction tree satisfies the additional constraint that the subgraph consisting of all vertices whose label includes a fixed variable, say x_1 , is connected. Alternately, a junction tree can be characterized by the property that if x_1 belongs to the label of vertices V_i and V_j it also belongs to the label of every vertex lying on the unique path between V_i and V_j .

Table 17.11 Local Domains and Kernels of
Example 18

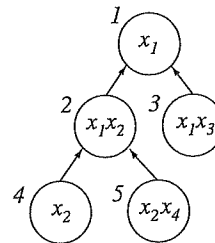
<i>Local Domain</i>	<i>Local Kernel</i>
$S_1 = \{x_1\}$	$\alpha_1(x_1)$
$S_2 = \{x_1, x_2\}$	$\alpha_2(x_1, x_2)$
$S_3 = \{x_1, x_3\}$	$\alpha_3(x_1, x_3)$
$S_4 = \{x_2\}$	$\alpha_4(x_2)$
$S_5 = \{x_2, x_4\}$	$\alpha_5(x_2, x_4)$

Example 18

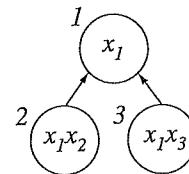
Consider the MPF problem of S_1 -marginalization of the global kernel

$$\beta(x_1, x_2, x_3, x_4) = \alpha_1(x_1) \alpha_2(x_1, x_2) \alpha_3(x_1, x_3) \alpha_4(x_2) \alpha_5(x_2, x_4) \quad (17.102)$$

given by $\beta(S_1) = \sum_{S \setminus S_1} \beta(x_1, x_2, x_3, x_4)$, where $S = \{x_1, x_2, x_3, x_4\}$ and the local domains and kernels are as shown in Table 17.11. The local domains can be organized into a junction tree as shown below:



Let us ignore for the moment the arrows along the edges. To verify that the graph is indeed a junction tree, note that the subgraph consisting of vertices containing the variable, x_i , $1 \leq i \leq 4$, are all connected, for instance, in case $i = 1$, we obtain the connected graph



Message Passing

Given the junction tree, application of the distributive law amounts to message passing between the vertices of a junction tree. Loosely speaking, the message passed from vertex V_i to vertex V_j in a junction tree is a *suitably marginalized* product of the local kernels of all the local domains that vertex V_i has either directly or indirectly been in communication with. More precisely, the message μ_{ij} passed from vertex V_i to vertex V_j is only a function of the variables in the intersection $S_i \cap S_j$ and is given by:

$$\mu_{ij}(S_i \cap S_j) = \left[\alpha_i(S_i) \prod_{\ell \in N_{ij}} \mu_{\ell i}(S_\ell \cap S_i) \right]_{S_j}, \quad (17.103)$$

where N_{ij} is the set of all vertices connected to V_i other than V_j , and where $[]_{S_j}$ denotes S_j -marginalization corresponding to summation over the variables in $S_i \setminus S_j$, i.e.,

$$\mu_{ij}(S_i \cap S_j) = \sum_{S_i \setminus S_j} \left[\alpha_i(S_i) \prod_{\ell \in N_{ij}} \mu_{\ell i}(S_\ell \cap S_i) \right]. \quad (17.104)$$

All messages $\mu_{ij}(S_i \cap S_j)$ are set equal to one initially, i.e., $\mu_{ij}(S_i \cap S_j) = 1$, all i, j .

A second operation carried out in the junction tree is *updating the state* $\sigma_i(S_i)$ of each vertex V_i . Initially, the state of a vertex V_i is simply its kernel $\alpha_i(S_i)$. The state is updated according to:

$$\sigma_i(S_i) = \alpha_i(S_i) \prod_{\ell \in N_i} \mu_{\ell i}(S_\ell \cap S_i), \quad (17.105)$$

where N_i is the collection of all vertices that are connected to vertex V_i .

To compute the desired objective function, one passes messages in accordance with a message schedule, and at the end of the message schedule, the state of a vertex is updated. The savings in computation comes about because under an appropriate message schedule, the marginalization inherent in message passing constitutes an efficient application of the distributive law.

In the present instance, we are interested only in computing the objective function at vertex V_1 . The next step in such cases is to direct all the edges in the graph so that they point toward vertex V_1 . Messages are passed only in the direction indicated by the arrows in accordance with the two rules below:

- (1) vertex V_i will pass a message to V_j only when it has received messages from all of its other neighbors,
- (2) vertex V_i will update its state only when it has received messages from all of its neighbors.

When these rules are applied to our example, the sequence of messages passed reads as follows:

Phase 1

$$\mu_{4,2}(S_4 \cap S_2) = \mu_{4,2}(x_2) = \alpha_4(S_4) = \alpha_4(x_2) \quad (17.106)$$

$$\mu_{5,2}(S_5 \cap S_2) = \mu_{5,2}(x_2) = \sum_{S_5 \setminus S_2} \alpha_5(S_5) = \sum_{x_4} \alpha_5(x_2, x_4) \quad (17.107)$$

$$\mu_{3,1}(S_3 \cap S_1) = \mu_{3,1}(x_1) = \sum_{S_3 \setminus S_1} \alpha_3(S_3) = \sum_{x_3} \alpha_3(x_1, x_3). \quad (17.108)$$

Phase 2

$$\mu_{2,1}(S_2 \cap S_1) = \mu_{2,1}(x_1) = [\alpha_2(S_2) \mu_{4,2}(S_4 \cap S_2) \mu_{5,2}(S_5 \cap S_2)]_{S_1} \quad (17.109)$$

$$\begin{aligned} &= \sum_{S_2 \setminus S_1} \alpha_2(S_2) \mu_{4,2}(S_4 \cap S_2) \mu_{5,2}(S_5 \cap S_2) \\ &= \sum_{x_2} \alpha_2(x_1, x_2) \mu_{4,2}(x_2) \mu_{5,2}(x_2). \end{aligned} \quad (17.110)$$

Phase 3

$$\sigma_1(S_1) = \alpha_1(S_1) \mu_{2,1}(S_2 \cap S_1) \mu_{3,1}(S_3 \cap S_1) = \alpha_1(x_1) \mu_{2,1}(x_1) \mu_{3,1}(x_1). \quad (17.111)$$

All the computations that comprise Phase 1 of the computation can be carried out simultaneously. However, the computations in Phase 1 must be completed prior to carrying out any computation that is part of Phase 2, etc.

The objective function computed given by $\sigma_1(S_1)$, when spelled out, reads as:

$$\sigma_1(x_1) = \alpha_1(x_1) \left[\sum_{x_2} \alpha_2(x_1, x_2) [\alpha_4(x_2)] \left[\sum_{x_4} \alpha_5(x_2, x_4) \right] \right] \left[\sum_{x_3} \alpha_3(x_1, x_3) \right]. \quad (17.112)$$

Clearly the computation when expressed in this fashion can be seen to make most efficient use of the distributive law.

Suppose next, that in the same example, we were interested in computing the objective functions at more than one, perhaps all of the vertices $V_1, V_2, \dots, V_5$. In this case it takes more effort to describe what constitutes an appropriate message schedule. We begin with some notation:

Let E denote the set of all edges in the junction tree. We will distinguish between edges (i, j) and (j, i) since reference to an edge (i, j) will

indicate our interest in passing a message from vertex V_i to vertex V_j . In our example:

$$E = \left\{ \begin{array}{cc} (1, 2) & (2, 1), \\ (1, 3) & (3, 1), \\ (2, 4) & (4, 2), \\ (5, 2) & (2, 5) \end{array} \right\}. \quad (17.113)$$

A *message schedule* is a sequence of subsets of E , such as $E_1 = \{(4, 2), (5, 2), (3, 1)\}$ and $E_2 = \{(2, 1)\}$.

With each message schedule we associate a *message trellis*. The trellis associated to message schedule $\{E_1, E_2\}$ is shown in Fig. 17.10. Each vertical column of vertices in the message trellis corresponds to the vertex V_i of the graph at successive time instants beginning with time $t = 0$. We use $V_i(t)$ to denote vertex V_i at time t . In the message trellis, we connect the pair of vertices $(V_i(t-1), V_j(t))$ if and only if (1) $i = j$ or (2) $(i, j) \in E_t$.

We say that vertex $V_i(t)$ *knows* $V_j(0)$ if there is a path in the message trellis connecting $V_j(0)$ to $V_i(t)$. In our example message trellis we see that $V_2(1)$ knows $V_4(0)$ and $V_5(0)$, but not $V_1(0)$. Also, $V_1(2)$ knows $V_i(0)$ for all i , $1 \leq i \leq 5$. It can be shown that a vertex $V_i(t)$ is ready to compute its objective function $\beta_i(S_i)$ if $V_i(t)$ knows $V_j(0)$, all j .

Thus given that it is desired to compute the objective functions $\{\beta_i(S_i) \mid 1 \leq i \leq M\}$, setting up an appropriate message schedule can be accomplished by using the message trellis to check that at the desired time instant t , the vertices $\{V_i(t), i = 1, 2, \dots, M\}$ all know $\{V_j(0), \text{all } j\}$.

We next show how the GDL can help solve the MPF problem associated to ML decoding of our example $[7, 4, 2]$ code.

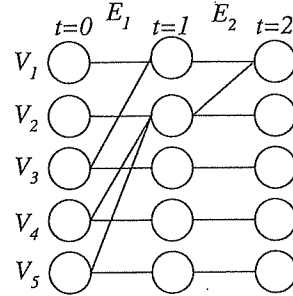


Fig. 17.10 Message trellis.

Example 19 (ML code symbol decoding of the $[7, 4, 2]$ code (continued))

Here we are interested in computing for all $i = 1, 2, \dots, 7$, and $x_i = 0, 1$ the following objective function:

$$\beta_i(x_i) = \sum_{\{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_7\}} \left\{ \prod_{j=1}^7 P(y_j|x_j) \chi_A(x_1, x_2, x_4) \chi_B(x_3, x_4, x_6) \chi_C(x_4, x_5, x_7) \right\}. \quad (17.114)$$

We assume that the communication channel is a BSC having crossover probability ϵ , $\epsilon < 1/2$. Thus for given $(y_1, y_2, \dots, y_7)$, $P(y_i|x_i) \in \{1 - \epsilon, \epsilon\}$. Our decisions will be based on the ratio $\beta_i(0)/\beta_i(1)$. The MPF nature of the problem is apparent. It will be found convenient to scale each local kernel $P(y_i|x_i)$ by the constant $(1 - \epsilon)$ (this will not affect the decision).

Let us assume that the received vector $\underline{y} = (0110100)$ in a instance of transmission across the BSC. The local kernel $\alpha_1(S_1)$ is then given by:

$$\alpha_1(S_1) = \frac{P(y_1|x_1)}{1 - \epsilon} = \begin{cases} 1, & x_1 = 0, \\ \epsilon/1 - \epsilon, & x_1 = 1. \end{cases} \quad (17.115)$$

Let $\theta = \epsilon/1 - \epsilon$, then $\theta < 1$. Set

$$\underline{\alpha}_1 = \begin{bmatrix} \alpha_1(0) \\ \alpha_1(1) \end{bmatrix} = \begin{bmatrix} 1 \\ \theta \end{bmatrix}.$$

We similarly obtain $\underline{\alpha}_i = \begin{bmatrix} 1 \\ \theta \end{bmatrix}$ for $i = 4, 6, 7$ and $\underline{\alpha}_i = \begin{bmatrix} \theta \\ 1 \end{bmatrix}$ for $i = 2, 3, 5$. We will use V_A, V_B, V_C to denote the vertices corresponding to the local domains $\{1, 2, 4\}$, $\{3, 4, 6\}$, and $\{4, 5, 7\}$, respectively. Figure 17.11 shows the local domains organized into a junction tree. We are interested here in computing the objective functions at all vertices V_i , $i = 1, \dots, 7$. With regard to the message schedule, we will adopt an inward-outward schedule in which the outlying vertices send their messages inward first. Once the innermost vertex V_4 has acquired knowledge of all the local kernels, the outward phase of message passing begins. The validity of this message schedule can be verified with the aid of a message trellis if desired. The messages passed are as follows:

Phase 1

$$\mu_{1,A}(S_1) = \alpha_1(x_1) \Rightarrow \underline{\mu}_{1,A} := \begin{bmatrix} \mu_{1,A}(0) \\ \mu_{1,A}(1) \end{bmatrix} = \begin{bmatrix} 1 \\ \theta \end{bmatrix} = \underline{\mu}_{6,B} = \underline{\mu}_{7,C}. \quad (17.116)$$

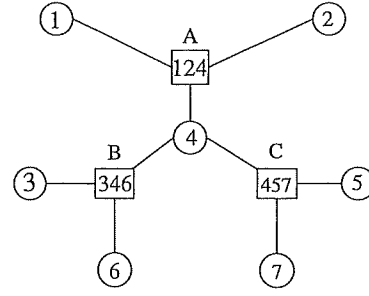


Fig. 17.11 Example of junction tree.

Similarly,

$$\underline{\mu}_{2,A} = \begin{bmatrix} \theta \\ 1 \end{bmatrix} = \underline{\mu}_{3,B} = \underline{\mu}_{5,C}. \quad (17.117)$$

Phase 2

$$\mu_{A,4}(x_4) = \sum_{x_1, x_2} \mu_{1,A}(x_1) \mu_{2,A}(x_2) \chi_A(x_1, x_2, x_4). \quad (17.118)$$

Since $\chi_A(x_1 + x_2 + x_4) = 0$ for $x_4 \neq x_1 + x_2$, we have

$$\begin{aligned} \mu_{A,4}(0) &= \sum_{x_1+x_2=0} \mu_{1,A}(x_1) \mu_{2,A}(x_2) \\ &= \mu_{1,A}(0) \mu_{2,A}(0) + \mu_{1,A}(1) \mu_{2,A}(1) = \theta + \theta = 2\theta. \end{aligned} \quad (17.119)$$

Similarly, $\mu_{A,4}(1) = 1 + \theta^2$, which implies

$$\underline{\mu}_{A,4} = \begin{bmatrix} 2\theta \\ 1 + \theta^2 \end{bmatrix}.$$

Also,

$$\underline{\mu}_{C,4} = \begin{bmatrix} 2\theta \\ 1 + \theta^2 \end{bmatrix} = \underline{\mu}_{B,4}.$$

Phase 3

$$\mu_{4,A}(x_4) = \mu_{B,4}(x_4) \mu_{C,4}(x_4) \alpha_4(x_4). \quad (17.120)$$

So, $\mu_{4,A}(0) = 2\theta \cdot 2\theta = 4\theta^2$, $\mu_{4,A}(1) = (1 + \theta^2)(1 + \theta^2)\theta = \theta(1 + \theta^2)^2$ and

$$\underline{\mu}_{4,A} = \begin{bmatrix} 4\theta^2 \\ \theta(1 + \theta^2)^2 \end{bmatrix}.$$

Similarly,

$$\underline{\mu}_{4,B} = \begin{bmatrix} 4\theta^2 \\ \theta(1 + \theta^2)^2 \end{bmatrix} = \underline{\mu}_{4,C}.$$

Phase 4

$$\mu_{A,1}(x_1) = \sum_{x_2, x_4} \mu_{4,A}(x_4) \mu_{2,A}(x_2) \chi_A(x_1, x_2, x_4). \quad (17.121)$$

$$\mu_{A,1}(0) = \mu_{4,A}(0) \mu_{2,A}(0) + \mu_{4,A}(1) \mu_{2,A}(1) = 4\theta^3 + \theta(1 + \theta^2)^2. \quad (17.122)$$

Similarly,

$$\mu_{A,1}(1) = \theta^2(1 + \theta^2)^2 + 4\theta^2, \underline{\mu}_{A,1} = \begin{bmatrix} 4\theta^3 + \theta(1 + \theta^2)^2 \\ \theta^2(1 + \theta^2)^2 + 4\theta^2 \end{bmatrix} = \underline{\mu}_{B,6} = \underline{\mu}_{C,7}$$

and

$$\underline{\mu}_{A,2} = \begin{bmatrix} \theta^2(1 + \theta^2)^2 + 4\theta^2 \\ 4\theta^3 + \theta(1 + \theta^2)^2 \end{bmatrix} = \underline{\mu}_{B,3} = \underline{\mu}_{C,5}.$$

Phase 5

$$\sigma_1(x_1) = \mu_{A,1}(x_1) \alpha_1(x_1) \quad (17.123)$$

So

$$\sigma_1(0) = \mu_{A,1}(0) \alpha_1(0) = 4\theta^3 + \theta(1 + \theta^2)^2. \quad (17.124)$$

Similarly, $\sigma_1(1) = \theta^3(1 + \theta^2)^2 + 4\theta^3$, so that

$$\underline{\sigma}_1 = \begin{bmatrix} 4\theta^3 + \theta(1 + \theta^2)^2 \\ \theta^3(1 + \theta^2)^2 + 4\theta^3 \end{bmatrix} = \underline{\sigma}_6 = \underline{\sigma}_7. \quad (17.125)$$

Similarly,

$$\underline{\sigma}_2 = \begin{bmatrix} 4\theta^3 + \theta^3(1 + \theta^2)^2 \\ 4\theta^3 + \theta(1 + \theta^2)^2 \end{bmatrix} = \underline{\sigma}_3 = \underline{\sigma}_5, \quad (17.126)$$

$$\underline{\sigma}_4 = \begin{bmatrix} 8\theta^3 \\ \theta(1 + \theta^2)^3 \end{bmatrix}. \quad (17.127)$$

Note that $\underline{\sigma}_4$ could have been computed in Phase 3.

Phase 6 (Decisions)

Since $\sigma_1(0) = 4\theta^3 + \theta(1 + \theta^2)^2 > \sigma_1(1) = \theta^3(1 + \theta^2)^2 + 4\theta^3$, we conclude that

$$P(x_1 = 0|y) > P(x_1 = 1|y). \quad (17.128)$$

Denoting the decoded codeword as before by $\hat{x} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_7)$, we therefore declare $\hat{x}_1 = 0$. Similarly, we get $\hat{x}_6 = \hat{x}_7 = 0$ and $\hat{x}_2 = \hat{x}_3 = \hat{x}_4 = \hat{x}_5 = 1$. It can be verified that $[0111100]^T$ is a valid codeword.

Example 20 (ML message bit decoding of convolutional codes (continued))

The local domains corresponding to the MPF problem in this case can be organized into a junction tree as shown in Fig. 17.12.

The linear nature of the junction tree coupled with the fact that it is desired to compute the objective functions at vertices u_i , $i = 0, 1, 2, 3$ results in a message passing schedule that could be viewed as consisting of a forward wave of message passing and a second backward propagating wave.

Part 1. Passing on A Priori Probabilities

$$\begin{aligned} \mu_{1,3}(u_0) &= P(u_0) & \mu_{2,3}(s_0) &= P(s_0) & \mu_{5,6}(u_1) &= P(u_1) \\ \mu_{8,9}(u_2) &= P(u_2) & \mu_{11,12}(u_3) &= P(u_3) \end{aligned} \quad (17.129)$$

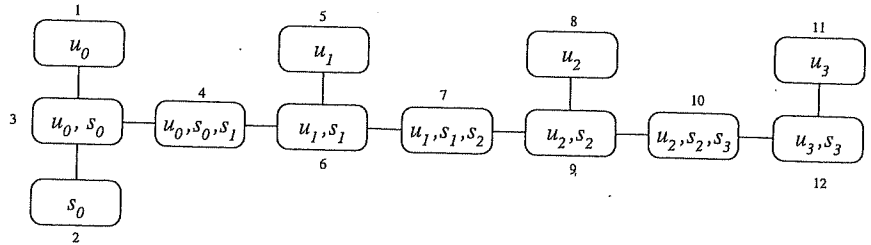


Fig. 17.12 Junction-tree representation of maximum-likelihood bit decoding of convolutional codes.

Part 2. Forward Wave

Phase I	$\mu_{3,4}(u_0, s_0) = \mu_{1,3}(u_0)\mu_{2,3}(s_0)P(y_0 u_0, s_0)$
Phase II	$\mu_{4,6}(s_1) = \sum_{u_0, s_0} \mu_{3,4}(u_0, s_0)P(s_1 u_0, s_0)$
Phase III	$\mu_{6,7}(u_1, s_1) = \mu_{5,6}(u_1)\mu_{4,6}(s_1)P(y_1 u_1, s_1)$
Phase IV	$\mu_{7,9}(s_2) = \sum_{u_1, s_1} \mu_{6,7}(u_1, s_1)P(s_2 u_1, s_1)$
Phase V	$\mu_{9,10}(u_2, s_2) = \mu_{7,9}(s_2)\mu_{8,9}(u_2)P(y_2 u_2, s_2)$
Phase VI	$\mu_{10,12}(s_3) = \sum_{u_2, s_2} \mu_{9,10}(u_2, s_2)P(s_3 u_2, s_2)$

Part 3. Backward Wave

Phase I	$\mu_{12,11}(u_3) = \sum_{s_3} \mu_{10,12}(s_3)P(y_3 u_3, s_3)$ $\mu_{12,10}(s_3) = \sum_{u_3} \mu_{11,12}(u_3)P(y_3 u_3, s_3)$
Phase II	$\mu_{10,9}(u_2, s_2) = \sum_{s_3} \mu_{12,10}(s_3)P(s_3 u_2, s_2)$
Phase III	$\mu_{9,7}(s_2) = \sum_{u_2} \mu_{8,9}(u_2)P(y_2 u_2, s_2)\mu_{10,9}(u_2, s_2)$ $\mu_{9,8}(u_2) = \sum_{s_2} \mu_{10,9}(u_2, s_2)P(y_2 u_2, s_2)\mu_{7,9}(s_2)$
Phase IV	$\mu_{7,6}(u_1, s_1) = \sum_{s_2} \mu_{9,7}(s_2)P(s_2 u_1, s_1)$
Phase V	$\mu_{6,4}(s_1) = \sum_{u_1} \mu_{5,6}(u_1)P(y_1 u_1, s_1)\mu_{7,6}(u_1, s_1)$ $\mu_{6,5}(u_1) = \sum_{s_1} \mu_{7,6}(u_1, s_1)P(y_1 u_1, s_1)\mu_{4,6}(s_1)$
Phase VI	$\mu_{4,3}(u_0, s_0) = \sum_{s_1} \mu_{6,4}(s_1)P(s_1 u_0, s_0)$
Phase VII	$\mu_{3,1}(u_0) = \sum_{s_0} \mu_{4,3}(u_0, s_0)P(y_0 u_0, s_0)\mu_{2,3}(s_0)$

Part 4. A Posteriori Probabilities

$$\begin{aligned} \sigma_1(u_0) &= \mu_{3,1}(u_0)P(u_0) & \sigma_5(u_1) &= \mu_{6,5}(u_1)P(u_1) \\ \sigma_8(u_2) &= \mu_{9,8}(u_2)P(u_2) & \sigma_{11}(u_3) &= \mu_{12,11}(u_3)P(u_3) \end{aligned} \quad (17.130)$$

Finally, since

$$\frac{\sigma_1(0)}{\sigma_1(1)} = \frac{P(u_0 = 0|y_0, y_1, y_2, y_3)}{P(u_0 = 1|y_0, y_1, y_2, y_3)}, \quad (17.131)$$

we decide $u_0 = 0$ if $\sigma_1(0) > \sigma_1(1)$, and similarly in the case of the other σ_i . The scheme just given for convolutional codes in which Parts 1, 2, 3, 4 are executed in sequence is clearly not optimized to yield the fastest decisions. The reader can fashion a message passing schedule that minimizes time. However, the schedule given does explain the commonly used terminology: forward-backward algorithm. This algorithm was discovered in a different

setting by several researchers, independently, including Baum, Welch, Bahl, Cooke, Jelinek, and Raviv (BCJR), for details see [2]. It is also often referred to as the BCJR or the Baum-Welch algorithm.

17.9 Turbo Codes

Turbo codes, also known as parallel concatenated convolutional codes, were discovered in 1993 by Berrou et al. [5], see also [15]. These codes presented a dramatic improvement in performance at low signal-to-noise ratios over an AWGN channel when compared with other codes existing at the time. An example turbo encoder is shown in Fig. 17.13. The encoder shown is composed of two identical rate 1/2 RSC encoders. The input to the first (top) encoder is a sequence $\{u_i\}_{i=0}^{N-1}$ of binary symbols. It is assumed that the top shift register is initially in the all-zero state and that the last v input bits $\{u_{N-v}, \dots, u_{N-1}\}$ are designed to restore the encoder to the all-zero state. The input $\{w_i\}_{i=0}^{N-1}$ to the second (bottom) encoder is the same set of input symbols but in a different order. Thus we may write:

$$(w_0 \dots w_{N-1}) = (u_0 u_1 \dots u_{N-1}) \mathbf{P} \quad (17.132)$$

for some $(N \times N)$ permutation matrix $\mathbf{P}$. The second encoder is also initialized to the all zero state.

The turbo encoder outputs three symbols $(v_i^{(1)}, v_i^{(2)}, v_i^{(3)})$ per input symbol u_i , $0 \leq i \leq N-1$. Whereas $v_i^{(1)} = u_i$, the remaining two symbols correspond

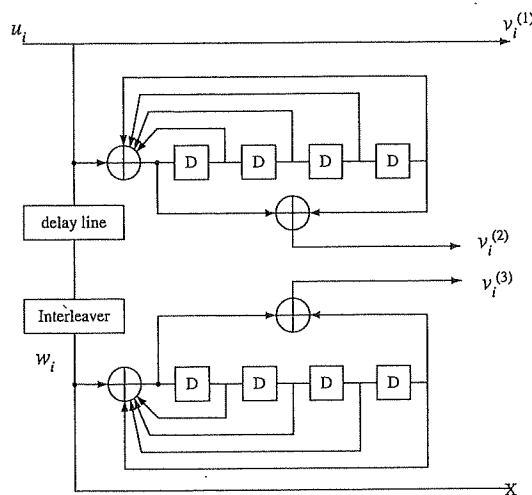


Fig. 17.13 Example of turbo encoder.

to outputs of the top and bottom encoder. Thus the overall rate of the code is one-third.

The input-output power series relationship of the two encoders may be expressed in the form

$$[v^{(1)}(D) v^{(2)}(D)] = u(D) \left[1 \frac{1 + D^4}{1 + D + D^2 + D^3 + D^4} \right], \quad (17.133)$$

$$[v^{(1)}(D) v^{(3)}(D)] = w(D) \left[1 \frac{1 + D^4}{1 + D + D^2 + D^3 + D^4} \right]. \quad (17.134)$$

The turbo encoder incorporates three novel features: (1) the use of two convolutional encoders to encode scrambled versions of the same input data, (2) the use of RSC encoders in place of conventional convolutional encoders and, (3) the use of a message passing decoding algorithm [23, 28] that employs an iterative message-passing schedule. It is this decoder that gives the code its name.

We explain with a simple example where $N = 4$, $w_0 = u_2$, $w_1 = u_1$, $w_2 = u_3$, and $w_3 = u_0$. Consider ML message-bit decoding of u_2 . Let $\{x_i, y_i, z_i\}_{i=0}^3$ denote the received symbols corresponding to the output streams $\{v_i^{(1)}, v_i^{(2)}, v_i^{(3)}\}_{i=0}^3$, respectively. Then

$$P(u_2 | \{x_i, y_i, z_i\}_{i=0}^3) \propto \sum_{u_0, u_1, u_3} \sum_{\{s_i\}_{i=0}^3} \sum_{\{q_i\}_{i=0}^3} P(\{u_i, s_i, q_i, x_i, y_i, z_i\}_{i=0}^3) \quad (17.135)$$

$$= \sum_{u_0, u_1, u_3} \sum_{\{s_i\}_{i=0}^3} \sum_{\{q_i\}_{i=0}^3} \prod_{i=0}^3 P(u_i) \left[\prod_{j=0}^3 P(x_j | u_j) \right] P(s_0) P(q_0) \quad (17.136)$$

$$\times \prod_{i=1}^3 [P(s_i | s_{i-1} u_{i-1}) P(q_i | q_{i-1} w_{i-1})] \left[\prod_{i=0}^3 P(y_i | u_i s_i) P(z_i | w_i q_i) \right], \quad (17.137)$$

where s_i, q_i denote respectively, the state of the top and bottom encoder.

At this point, one should, strictly speaking, identify the local domains and kernels associated with this computation and then set up a junction tree. However, the turbo decoder operates using the graph shown in Fig. 17.14. The reader will observe that this graph is not a tree as it is possible to identify several pairs of nodes with each pair connected by two or more paths.

Despite this, however, message passing is used for decoding, and the schedule runs as follows. It is evident that the graph is composed of two sections. The top section is the junction tree associated with the top encoder and similarly with the bottom section. The two sections are linked by edges designated by a dotted line.

To begin with, let us pretend that the dotted edges are absent. The upper section employs message passing using a forward-backward schedule and concludes by updating the states $\sigma_i(u_i)$ of the vertices labeled u_0, u_1, u_2 and u_3 .

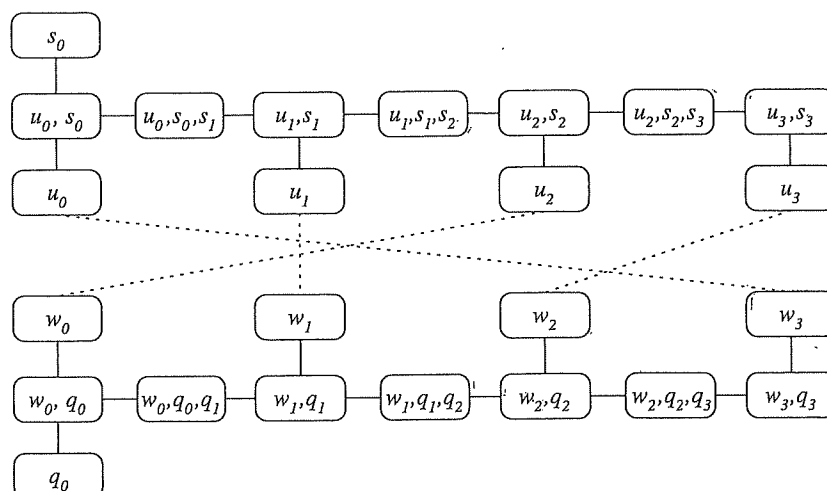


Fig. 17.14 Junction-tree representation of turbo codes.

These states $\sigma_i(u_i)$ reflect the a-posteriori probabilities of the symbols. The dotted edges then come into play. Each dotted edge indicates that the vertices at either end of the edge (such as, for example, vertices u_3 and w_2) should be viewed as being the same. Thus vertices u_3 and w_2 share the same state $\sigma_3(u_3) = \sigma_2(w_2)$. The bottom section of the graph is now activated. Messages are passed into the bottom section from the vertices w_i , $i = 0, 1, 2, 3$. Once again, we pretend that the dotted edges are absent and carry out forward-backward message passing, this time in the bottom section, concluding by updating the states of vertices w_i , $i = 0, 1, 2, 3$.

At this point, we repeat the earlier procedure of identifying the vertices w_i with the vertices u_i . The entire procedure described thus far constitutes one iteration. This procedure is iterated several times. It has experimentally been observed that with high probability, after several iterations, the probability ratio $\sigma_i(0)/\sigma_i(1)$ can be taken to be a good estimate of $P(u_i = 0|y)/P(u_i = 1|y)$ and decisions on the u_i made accordingly.

Performance of Turbo Codes

At low signal-to-noise ratios, turbo codes have probability of bit error that is significantly lower than that of a comparable convolutional code. The explanation for this excellent performance lies in the difference between the weight distribution of a turbo code and that of a comparable convolutional code. For details, the reader is referred to [30].

17.10 Low-Density Parity-Check Codes

The junction tree of our earlier example $[7, 4, 2]$ code can be redrawn as shown in Fig. 17.15. Such graphs are called *bipartite* graphs since the graph contains two classes of nodes. Each edge in the graph links a node with a second node belonging to the other class. In this particular bipartite graph, the two classes of nodes correspond to the code symbols and the parity checks on the code symbols, respectively. Such a graph is termed a *Tanner* graph [38]. Although the Tanner graph of our $[7, 4, 2]$ code is a junction tree, in general, Tanner graphs are not junction trees. Clearly it is possible to associate every linear, binary block code with a Tanner graph. The Tanner graph of a $[10, 5]$ code along with its parity check matrix $\mathbf{H}$ is shown in Fig. 17.16. It can be verified that this Tanner graph is not a tree. Note that each row of $\mathbf{H}$ has six 1's and each column has three 1's. Codes of large length, say of length 1000, with a small fixed number of 1's in each column and each row, were termed *low-density parity-check codes* (LDPC codes) by Gallager [13, 23, 35] in the 1960s. Gallager showed how these codes could achieve reliable information transmission over a BSC with probability of error approaching zero as the length of the code approached infinity. The decoding algorithm employed by Gallager was a form of message passing and was of complexity linear in the length of the code.

In recent years, following the success of turbo codes, there has been renewed interest in the decoding of these codes using various message passing algorithms, including those that correspond to message passing of the type carried out in our junction tree based decoding of the $[7, 4, 2]$ code. The approach here, in essence, is to construct LDPC codes in which the cycles in the graph are not too short in terms of the number of edges along the cycles, ignore the fact that the graph is not a junction tree, and iteratively continue message passing from every node to its neighbor until at some point, hopefully, each node attains its associated objective function. For more details, we refer the reader to [25, 35].

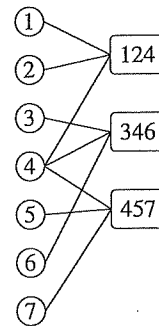


Fig. 17.15 The Tanner graph of $[7, 4, 2]$ code.

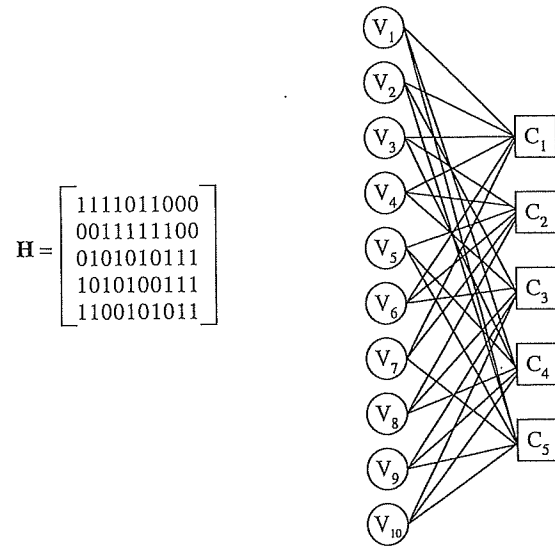


Fig. 17.16 The parity-check matrix of a $[10, 5]$ code and the corresponding Tanner graph.

17.11 FEC Codes Proposed for Optical Fiber Communication

In this section, we discuss the applicability of the codes described in Sections 17.2 through 17.10 to the optical channel. We begin with some preliminaries.

The rate R of an error correction code $\mathcal{C}$ is the ratio of the number of information-bearing message bits to the total number of bits transmitted. A code of rate R has *redundancy* $\rho = 1 - R$. The overhead ω of a code of rate R is given by $\omega = (1 - R)/R$. The overhead is often presented as a percentage. In the case of an $[n, k]$ block code, the rate, redundancy, and overhead of the code are given respectively by $R = k/n$, $\rho = (n - k)/n$, and $\omega = (n - k)/k$. For example, a $[255, 239]$ block code has $R = 0.937$, $\rho = 0.063$, and an overhead of 6.7%.

The complexity of a decoding algorithm is often measured in terms of the number of operations (addition, multiplication and division) over the relevant finite field. The fastest means of implementing finite field operations is via table lookup, which is feasible when the exponent m of the finite field $\mathbb{F}_{2^m}$ is not too large.

Coding Gain

Discussions of coding gain in the current optical-FEC literature usually take place in the setting of a WDM optical communication system in which the

dominant source of noise is the ASE noise introduced by the optical amplifiers present along the optical fiber. The modulation is assumed to be OOK and the receiver can be modeled as a symbol detector (see 17.1.3) followed by a hard-decision FEC decoder. The symbol detector consists of a direct-detection photodetector followed by an integrate-and-dump filter with integration-time interval T chosen to capture most of the energy of the incoming pulse. The output x of the integrate-and-dump filter is fed to a threshold circuit which declares the transmitted message bit to be a “1” or a “0” depending upon whether x exceeds or does not exceed, a threshold x_{th} . The output of the threshold circuit is fed to the input of the FEC decoder.

All distortions of the optical signal except that caused by the ASE noise are ignored. Thus an implicit assumption is made that the intersymbol interference caused by the various sources of pulse dispersion has been reduced to negligible levels through some form of optical or electronic equalization.

The ASE noise can be assumed to be modeled as colored Gaussian noise obtained by passing additive white Gaussian noise through a bandpass filter whose single-sided bandwidth equals B_o , the bandwidth of the optical amplifier. Using this fact it is shown in [16] that statistically, x is a random variable having a probability distribution known as the chi-square distribution [32] with the exact distribution depending upon whether the transmitted bit is a “1” or a “0”. The authors of [16] then go on to show how one can determine the optimal threshold setting x_{th} that minimizes bit error probability (BEP) p and how this minimum BEP p can be computed.

It is common in practice however, to assume that x has a Gaussian distribution. It is shown in [16], that while this assumption leads to an incorrect determination of the threshold x_{th} , it does yield an expression for a BEP $\hat{p}$ that is close to the true value p . Since it is the BEP that is of primary interest in this chapter, we will throughout use the approximation $\hat{p}$ for p given by the Gaussian assumption, i.e.,

$$p \approx \hat{p} = \text{erfc}_*(Q) := \frac{1}{\sqrt{2\pi}} \int_Q^\infty e^{-y^2/2} dy, \quad (17.138)$$

where Q is called the Q -factor and turns to be given by [16]

$$Q = \frac{2\gamma\sqrt{B_e/B_o}}{1 + \sqrt{1 + 4\gamma B_e/B_o}}, \quad (17.139)$$

where $\gamma = E/N_0$ is the SNR, i.e., the ratio of average signal energy E (per message bit) to the single-sided power density N_0 of the colored noise within the optical bandwidth B_o , and where $B_e = 1/T$ is the *electrical bandwidth* of the signal. The Q -factor is often expressed in terms of a quantity termed as

the optical-signal-to-noise ratio (OSNR) ω given by

$$\omega = \gamma \frac{B_e}{B_o} \quad (17.140)$$

so that in terms of the OSNR ω , the Q -factor is given by

$$Q = \frac{2\omega\sqrt{B_o/B_e}}{1 + \sqrt{1 + 4\omega}}. \quad (17.141)$$

For large values of OSNR, i.e., $\omega \gg 1$, the Q -factor is approximately given by

$$Q = \sqrt{\frac{E}{N_0}}. \quad (17.142)$$

We will from now on make the approximation

$$p \approx \text{erfc}_* \left(\sqrt{\frac{E}{N_0}} \right). \quad (17.143)$$

The combination of physical optical fiber communication channel and the receiver described above, has resulted in the creation of a binary-input, binary-output channel. It turns out [16] that without significant loss in accuracy, we may assume the transition probabilities in this binary-input, binary-output channel to be equal. This leads to the binary-symmetric-channel (BSC) model shown in Fig. 17.1. In the context of a BSC, the BEP parameter p given by (17.143) is called the *crossover probability*.

The coding gain delivered by an FEC code is measured as follows. We assume in what follows, that the goal is to achieve a message-bit error probability of 10^{-15} or less. Let E_{uncoded} denote the average energy per message bit required in the absence of coding to achieve the target 10^{-15} error rate. Note that in the uncoded case, there is no distinction between transmitted and message bits and hence E_{uncoded} also represents the average energy per transmitted bit.

Next, let p_{coded} be the crossover probability of the BSC at which the code $\mathcal{C}$ is able to deliver the target message bit error rate of 10^{-15} . Let R be the rate of the code. As in the uncoded case, let E_{coded} denote the average received energy per message bit. Since there are R message bits per transmitted bit in the coded case, we have that

$$p_{\text{coded}} = \text{erfc}_* \left(\sqrt{\frac{E_{\text{coded}}R}{N_0}} \right). \quad (17.144)$$

The coding gain η of the code C of rate R is then given by

$$\eta = 10 \log_{10} \frac{E_{\text{uncoded}}}{E_{\text{coded}}}. \quad (17.145)$$

In the optical FEC literature, the coding gain is also referred to as the net effective coding gain.

Let γ_{uncoded} , Q_{uncoded} and γ_{coded} , Q_{coded} denote the uncoded and coded SNR and Q -factor, respectively, defined via

$$\gamma_{\text{uncoded}} = \frac{E_{\text{uncoded}}}{N_0} \quad \gamma_{\text{coded}} = \frac{E_{\text{coded}}}{N_0} \quad (17.146)$$

$$Q_{\text{uncoded}} = \sqrt{\gamma_{\text{uncoded}}} \quad Q_{\text{coded}} = \sqrt{R\gamma_{\text{coded}}}. \quad (17.147)$$

The equations below express the coding gain in terms of SNR and Q -factor:

$$\eta = 10 \log_{10} \frac{\gamma_{\text{uncoded}}}{\gamma_{\text{coded}}} \quad (17.148)$$

$$\eta = 20 \log_{10} \frac{Q_{\text{uncoded}} \sqrt{R}}{Q_{\text{coded}}}. \quad (17.149)$$

Example 21

Let the target BER = 10^{-15} . Let the code C have rate $R = 8/9$ and let $p_{\text{coded}} = 10^{-5}$. Then since

$$\text{erfc}_*(7.942) = 10^{-15} \quad (17.150)$$

$$\text{erfc}_*(4.265) = 10^{-5}, \quad (17.151)$$

we have that $Q_{\text{uncoded}} = 7.942$ and $Q_{\text{coded}} = 4.265$ so that the (net effective) coding gain of the code is given by

$$\eta = 20 \log_{10} \left(\sqrt{\frac{8}{9} \frac{7.942}{4.265}} \right) = 4.89 \text{ dB}. \quad (17.152)$$

We next present an upper bound to the maximum achievable coding gain under hard-decision decoding, of the optical channel described above. As noted, the optical channel can be viewed as a BSC with crossover probability p given by

$$p = \text{erfc}_* \left(\sqrt{R\gamma_{\text{coded}}} \right) \quad (17.153)$$

Table 17.12 Maximum Achievable Coding Gain on the BSC

<i>Maximum Achievable</i>	
<i>Rate R</i>	<i>Coding Gain (BSC) (dB)</i>
0.85	11.17
0.90	10.59
0.95	9.69

where R is the rate of the code. It is possible using information theory [10] to determine the smallest value $\gamma_{coded}^{\min}$ of SNR required to guarantee reliable communication at rate R . This value is obtained by solving the equation

$$R = -p \log_2 p - (1-p) \log_2 (1-p), \quad \text{where } p = \text{erfc}_* \left(\sqrt{R \gamma_{coded}^{\min}} \right). \quad (17.154)$$

Let as before, $\gamma_{uncoded}$ denote the SNR needed to achieve reliable communication in the absence of coding. Interpreting reliable communication as communicating with BEP 10^{-15} , we can determine $\gamma_{uncoded}$ by solving

$$\text{erfc}_* \left(\sqrt{\gamma_{uncoded}} \right) = 10^{-15} \quad (17.155)$$

Then $10 \log_{10} (\gamma_{uncoded} / \gamma_{\min})$ is the maximum achievable coding gain at rate R . The maximum achievable coding gains at rates $R = 0.85, 0.9$ and 0.95 are tabulated in Table 17.12.

By replacing hard-decision decoding with soft-decision decoding, it may be possible to gain something on the order of an additional 1–2 dB depending upon the rate R of the code.

BCH Codes (Section 17.6)

BCH codes fall into the class of cyclic binary codes. From an optical standpoint, these codes offer certain advantages. They are efficient in having small overhead for given error correction capability, and are flexible in that they offer a trade-off to be made between overhead and error correction capability. When operated over a BSC, the codes can be decoded by a hard-decision decoding algorithm of reasonable complexity. If the BCH code has length n and is capable of correcting t errors, then decoding can be accomplished using roughly $4nt + 4t^2$ finite field operations, for details see [24].

A tabular listing of the coding overhead/coding gain trade-off offered by BCH codes is provided in Table 17.13. In deriving the coding gain of the BCH

Table 17.13 Lower Bound on Coding Gain of Some BCH Codes

<i>Length n</i>	<i>t</i>	<i>Rate R</i>	<i>Overhead ω (%)</i>	<i>Coding Gain (d3B)</i>
2047	27	0.85	17.64	8.02
2047	18	0.90	11.11	7.41
2047	9	0.95	5.26	6.17
4095	51	0.85	17.64	8.60
4095	34	0.90	11.11	8.09
4095	17	0.95	5.26	7.03
2047	1	0.99	0.60	1.59

codes, it was assumed that the bit error probability at the output of the BCH decoder is equal to the probability given in (17.4) of incorrectly decoding the transmitted codeword using the bounded-distance decoding algorithm. This assumption is equivalent to saying that whenever a codeword is incorrectly decoded, all the message bits are erroneously decoded. For this reason, the “coding gain” entries in Table 17.13 are lower bounds on the actual coding gain realized by the respective BCH code. Application of the bounded-distance decoding algorithm requires knowledge of the minimum distance of the code. In this case, the minimum distance was set equal to the value given by the BCH bound, see [26].

The lengths of the BCH codes were chosen to be 2047 and 4095. The first length is comparable to that of the binary code in the ITU G.975 Recommendation, which involves a length 255 RS code, which upon conversion to bits, results in a binary code of overall length $8 * 255 = 2040$. At both lengths, three sample codes are presented, corresponding to the code rates in the range 0.85–0.95. The bottom entry in the table corresponds to a BCH code with parameter $t = 1$, which corresponds as indicated in Section 17.6, to a single-error-correcting, cyclic Hamming code of length 2047.

In the optical communication literature, BCH codes have appeared as constituent codes in the construction of product codes (see below). Hamming codes were proposed for use on the fiber-optic channel in an early paper by Grover [14].

Product Codes (Section 17.6.1)

Product codes offer the benefits of large block length combined with ease of decoding. However, if it is desired to construct a product code C of high rate R , this forces the constituent codes C_1 and C_2 to have even higher rates. Typically in such cases the minimum distance $d = d_1 d_2$ of the resulting product code will be low.

A product code construction for optical communication, based on a pair of linear binary codes having parameters $[128, 113, 6]$ is described in [1]. These binary codes are obtained by appending an extra parity symbol to the $[127, 113, 5]$ double-error-correcting BCH code. An iterative soft-decision decoding algorithm is used to decode the code, resulting in a coding gain estimated at 9.7dB. The rate R of this product BCH code equals 0.78.

Reed-Solomon Codes (Section 17.5)

RS Codes in Practice

For optical fiber submarine communication systems, ITU Recommendation G.975 [17] recommends the use of a $[255, 239]$ Reed-Solomon code. In the absence of interleaving, this code is capable of correcting eight random byte errors within each codeword. As noted earlier, this translates into a maximum guaranteed burst error correction capability of 57 bits.

Interleaving of code bits across codewords can be used to increase the burst error correction capability at the expense of an increase in delay in decoding. ITU G.975 allows for interleaving up to depth 16, which would result in a maximal guaranteed burst error correction capability of $8 \times (7 \times 16 + 15) + 1 = 1017$ bits.

Decoding RS Codes

As discussed in Section 17.5, an $[n = 2^m - 1, k]$ RS code with code symbols lying in $\mathbb{F}_{2^m}$, has minimum distance $d_{\min} = n - k + 1$ and is capable of correcting $t = \lfloor d_{\min} - 1/2 \rfloor$ errors. The Berlekamp-Massey algorithm [4] can be used to decode the RS code with a running time on the order of n^2 .

Performance of RS Codes

We provide a tabular listing in Table 17.14 of the error-correction capability, rate, overhead and corresponding coding gain of length 255 and 511 RS codes in Table 17.14. The coding gains shown in the table assume a target BER of 10^{-15} . In deriving the value of coding gain, it was assumed that the bit-error probability at the output of the RS decoder was equal to the probability of incorrectly decoding the RS codeword under a bounded-distance decoding algorithm. This is equivalent to assuming that a decoding error causes all the corresponding message bits to be incorrectly decoded. Thus the value appearing in the table under “coding gain” is in actuality, a lower bound to the coding gain. The probability of decoding error under the bounded-distance decoding algorithm was determined using (17.4).

The last entry in the table corresponds to the RS $[255, 239, 17]$ code appearing in the ITU G.975 Recommendation [17].

Table 17.14 Lower Bound on the Coding Gain of Some RS Codes

<i>Length n</i>	<i>t</i>	<i>Rate R</i>	<i>Overhead ω (%)</i>	<i>Coding Gain (dB)</i>
255	19	0.85	17.64	7.42
255	13	0.90	11.11	6.70
255	6	0.95	5.26	5.40
511	38	0.85	17.64	8.04
511	25	0.90	11.11	7.53
511	13	0.95	5.26	6.56
255	17	0.937	6.72	5.99

Concatenated Codes (Section 17.5.1)

The discussion here is restricted to the technique of concatenating codes presented in Section 17.5.1. Concatenated codes offer the benefits of large length and the ability to handle isolated errors as well as error bursts. Large code lengths improve performance as these tend to average out distortions introduced by the channel, making them more predictable, and hence more correctable. As pointed out earlier, the overall rate of the concatenated codes is the product of the rates of the outer and inner codes.

There is a downside to the use of concatenated codes. With concatenated codes, the dimension of the inner code equals the number of bits in one bit of the outer code and thus is typically around the value 8. At dimension 8, the maximum rate of a block code equals 8/9. To construct a concatenated code of overall high rate, one is forced to use a very high-rate RS code, which drives down the overall error correction capability.

RS Codes in the Literature

It is shown in [21] that an increase in coding gain of about 1.2dB is observed when the ITU-recommended [255, 239] RS code is replaced by the more powerful [255, 223] RS code having 14.3% overhead.

Most of the error correction schemes discussed in OFC 2001 were based on the Reed-Solomon code. In [1], an FEC scheme featuring a pair of concatenated RS codes with an interleaver in between is examined (see Fig. 17.17). Note that the method of code concatenation described in [1] is different from that discussed in Section 17.5.

Two different example RS code pairs are considered. The first pair consists of two identical [255, 239] RS codes. The overhead of the concatenated code in this case is 13.8% and a 7.2dB coding gain was estimated at a BER of 10^{-13} when the interleaver depth equaled 32 bytes. In the second example, the RS codes have parameters [255, 239] and [255, 223], resulting in an overall

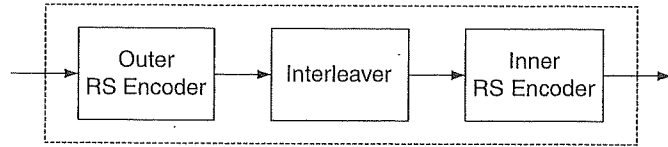


Fig. 17.17 Concatenated Reed-Solomon encoder.

overhead of 22%. A 7.7 dB coding gain was estimated when an interleaver of depth 32 bytes was used. The author also points out that an increase in coding gain results when the concatenated code is iteratively decoded.

A third method of concatenation involving RS codes is discussed in [39]. Here the constituent RS codes have parameters [255, 239] and [239, 223] for an overall overhead of 14%. The authors of [39] point out that when operating at high power levels, the increase in symbol rate arising from the use of FEC can result in increased nonlinear effects which tend to reduce the amount of coding gain. If this nonlinearity penalty is ignored, then the serial concatenation scheme is able to produce (roughly) 3 dB gain over the ITU recommendation. However, since the nonlinear penalty observed by the authors of [39] is about 1 dB, the net coding gain of approximately 7.5 dB observed at a BER of 10^{-13} , is roughly 2 dB higher than that afforded by the ITU-recommended code.

In [46], a new approach for enhanced PMD mitigation using a combination of polarization scrambling and FEC is presented. Due to the slow temporal dynamics of PMD, current systems have to be designed for the worst-case PMD constellation. By introducing polarization scrambling, these dynamics are accelerated such that bad PMD constellations can affect only a limited number of bits per FEC frame. These erroneous bits can be corrected by the FEC scheme. The FEC scheme discussed in the paper is a RS [255, 239, 17] code.

In [49], an alternative scheme for mitigating PMD using FEC coding and a first-order compensator is presented. A [255, 241] RS code is used as FEC code. The measured OSNR gain due to FEC (as compared to using only a first-order compensator), in the presence of only PMD and noise, at an outage probability of 10^{-3} , was 5.7 dB at 31 ps average PMD and increased to 7.5 dB at 43 ps average PMD.

Convolutional Codes (Section 17.7)

When the memory of the convolutional encoder is not too large, the Viterbi and BCJR algorithms present efficient means of accomplishing either hard- or soft-decision decoding of these codes. However, convolutional codes are typically low-rate codes. It is possible to puncture a low-rate convolutional code [44] to obtain a high-rate code that is also easily decodable, but this puncturing tends to decrease the Hamming distance between codewords in the code.

A discussion on the use of a convolutional code as an FEC scheme for a pulse-position modulation-based optical fiber communication system may be found in [12].

Turbo Codes (Section 17.9)

Turbo codes are also known as parallel concatenated convolutional codes. These codes tend to have low rates and also low values of minimum distance. Although these codes perform excellently at low SNR, at high SNR, the low minimum distance of these codes tends to degrade performance. It is also possible to iteratively decode convolutional codes that are serially concatenated [8,15] and these tend to have larger minimum distance and may therefore be better suited to the optical channel where very low bit errors are desired. Thus, it is desirable to identify serially concatenated convolutional codes, that in addition to having large minimum distance, also offer high rate.

Low-Density Parity-Check Codes (Section 17.10)

LDPC codes are potentially of interest in optical channels. Of course, one does need to identify suitable high-rate LDPC codes that also possess large minimum distance. Nonbinary LDPC codes [25] could also be of interest.

Acknowledgment

The authors would like to thank Habong Chung, Manini Shah, Ted Darcie, and Jack Winters for some very useful discussions.

References

- [1] O. Ait Sab, "FEC techniques in submarine transmission systems," *OFC 2001*, vol. 2, pp. TuF1.1–TuF1.3, 2001.
- [2] S. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 325–343, March 2000.
- [3] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.
- [4] E. R. Berlekamp, *Algebraic Coding Theory*, Aegean Park Press, Laguna Hills, 1984.
- [4b] E. R. Berlekamp, "Bounded distance +1 soft-decision Reed–Solomon decoding," *IEEE Trans. Inform. Theory*, vol. 42, pp. 704–720, May 1996.
- [5] G. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding: Turbo codes," in *Proc. 1993 Int. Conf. Commun.*, Geneva, Switzerland, pp. 1064–1070, May 1993.

- [6] R. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, Reading, Massachusetts, 1983.
- [7] A. E. Brouwer and T. Verhoeff, "An updated table of minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 2, pp. 662–676, March 1993.
- [8] K. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection*, Kluwer Academic Publishers, Boston, 2001.
- [9] G. Clark and J. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, New York, 1981.
- [10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley, New York, 1991.
- [11] E. Desurvire, *Erbium-Doped Fiber Amplifiers-Principles and Applications*, John Wiley & Sons, Inc., New York, 1994.
- [12] E. Forestieri, R. Ganaopadhyay, and G. Prati, "Performance of convolutional codes in a direct-detection optical PPM channel," *IEEE Trans. Comm.*, vol. 37, no. 12, pp. 1303–1317, December 1989.
- [13] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, January 1962.
- [14] W. D. Grover, "Forward error correction in dispersion-limited lightwave systems," *J. Lightwave Tech.*, vol. 6, pp. 643–654, May 1988.
- [14b] V. Guruswami and M. Sadan, "Improved decoding of Reed–Solomon and algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, September, 1999.
- [15] C. Heegard and S. Wicker, *Turbo Coding*, Kluwer Academic Publishers, Boston, 1998.
- [16] P. A. Humblet and M. Azizoglu, "On the bit error rate of lightwave systems with optical amplifiers," *J. Lightwave Tech.*, vol. 9, no. 11, pp. 1576–1582, November 1991.
- [17] International Telecommunication Union Telecommunication Standardization Sector (ITU-T), Series G: Transmission Systems and Media, Digital Systems and Networks, G.975.
- [18] G. Jacobsen, *Noise in Digital Optical Transmission Systems*, Artech House, Norwood, MA, 1994.
- [19] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Press, New York, 1998.
- [20] L. G. Kazovsky, S. Benedetto, and A. Willner, *Optical Fiber Communication Systems*, Artech House Publishers, Boston, October 1996.
- [21] H. Kidorf, N. Ramanujam, I. Hayee, M. Nissov, J. X. Cai, B. Pedersen, A. Puc and C. Rivers, "Performance improvement in high capacity, ultra-long distance, WDM systems using forward error correction codes," *OFC '00*, vol. 3, pp. 274–276, 2000.
- [22] R. Kötter, "A fast parallel implementation of a Berlekamp-Massey algorithm for algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 4, pp. 1353–1368, July 1998.

- [23] F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, February 2001.
- [24] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
- [25] D. J. C. Mackay and M. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *IMA Workshop on Codes, Systems and Graphical Models*, 1999.
- [26] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, North Holland, 1977.
- [27] D. Marcuse, "Derivation of analytical expression for the bit-error probability in lightwave systems with optical amplifiers," *J. Lightwave Tech.*, vol. 8, no. 12, pp. 1816–1823, December 1990.
- [28] R. J. McEliece, D. J. C. Mackay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithms," *IEEE J. Select Areas Comm.*, vol. 16, pp. 140–152, February 1998.
- [29] J. B. H. Peek, "Communications aspects of the compact disc digital audio system," *IEEE Comm. Mag.*, vol. 23, no. 2, pp. 7–15, February 1985.
- [30] L. C. Perez, J. Seghers, and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1698–1709, November 1996.
- [31] V. S. Pless and W. C. Huffman, *Handbook of Coding Theory*, Elsevier, New York, 1998.
- [32] J. K. Proakis, *Digital Communications*, 4th edn., McGraw-Hill, Boston, 2000.
- [33] A. Puc, F. Kerfoot, A. Simons, and D. Wilson, "Concatenated FEC experiment over 5000 km long straight line WDM test bed," *OFC '99*, vol. 3, pp. 255–258, 1999.
- [34] R. Ramaswami and N. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd edn., Morgan Kaufmann Publishers, San Francisco, 2002.
- [35] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, February 2001.
- [36] I. S. Reed and X. Chen, *Error-Control Coding for Data Networks*, Kluwer Academic Publishers, Amsterdam, Netherlands, 2000.
- [37] O. Sab and J. Fang, "Concatenated forward error correction schemes for long-haul DWDM optical transmission systems," in Tech. Dig., 25th European Conf. on Optical Comm. (ECOC'99), Nice, France, Paper Th C2.4.
- [38] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, September 1981.
- [39] H. Taga, H. Yamauchi, T. Inoue, K. Goto, N. Edagawa, and M. Suzuki, "Performance improvement of highly nonlinear long distance optical fiber transmission system using novel high gain forward error correcting code," *OFC 2001*, vol. 2, pp. TuF3.1–TuF3.3, 2001.

- [40] M. Tomizawa, Y. Yamabayashi, K. Murata, T. Ono, Y. Kobayashi, and K. Hagimoto, "Forward error correcting codes in synchronous fiber optic transmission systems," *J. Lightwave Tech.*, vol. 15, no. 1, pp. 43–52, January 1997.
- [41] S. A. Vanstone and P. C. Van Oorschot, *An Introduction to Error Correcting Codes with applications*, Kluwer Academic Publishers, Boston, 1989.
- [42] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Comm.*, vol. 19, pp. 751–772, Oct. 1971.
- [43] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, New York, 1979.
- [44] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.
- [45] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*, IEEE Press, New York, June 1994.
- [46] B. Wedding and C. N. Haslach, "Enhanced PMD mitigation by polarization scrambling and forward error correction," *OFC 2001*, vol. 2, pp. WAA1.1–WAA1.3, 2001.
- [47] J. Winters, R. Gitlin, and S. Kasturia, "Reducing the effects of transmission impairments in digital fiber optic systems," *IEEE Comm. Mag.*, pp. 68–76, June 1993.
- [48] J. -H. Wu and J. Wu, "Performance of Reed-Solomon codes in CPFSK coherent optical communications," *J. Opt. Comm.*, vol. 13, pp. 19–22, March 1992.
- [49] Y. Xie, O. Yu, L.-S. Yan, Q. H. Adamczyk, Z. Pan, S. Lee, A. E. Willner, and C. R. Menyuk, "Enhanced PMD mitigation using forward-error-correction coding and a first-order compensator," *OFC 2001*, vol. 2, pp. WAA2.1–WAA2.3, 2001.