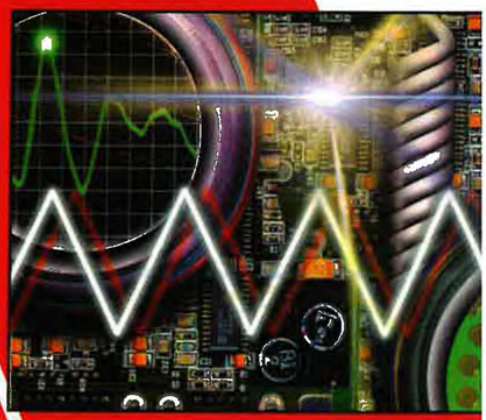
Understanding Digital Signal Processing



RICHARD G. LYONS

Sony v. Jawbone
U.S. Patent No. 7,246,058
Sony Ex. 1008 (Part 1 of 3)

Understanding Digital Signal Processing

Understanding Digital Signal Processing

Richard G. Lyons



ADDISON-WESLEY PUBLISHING COMPANY

An imprint of Addison Wesley Longman, Inc.
Reading, Massachusetts · Harlow, England · Menlo Park, California
Berkeley, California · Don Mills, Ontario · Sydney
Bonn · Amsterdam · Tokyo · Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters.

The publisher offers discounts on this book when ordered in quantity for special sales.

For more information, please contact:
Corporate & Professional Publishing Group
Addison Wesley Longman, Inc.
One Jacob Way
Reading, Massachusetts 02867

Library of Congress Cataloging-in-Publication Data

Lyons, Richard G., 1948–
Understanding digital signal processing / Richard G. Lyons.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-63467-8 (hc)
1. Signal processing--Digital techniques. I. Title.
TK5102.9.L96 1997
621.382'2--dc20 96-28818
CIP

Copyright 1997 By Addison Wesley Longman, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

0-201-63467-8

1 2 3 4 5 6 7 8 9—MA—00999897

First Printing, October 1996

D E

I dedicate this book to my
could go with you; to my
work; to my father Grady w
that workbench in the base
to my brother Ken who su
running interference for us;
Schlessinger for keeping us
the Iron Riders Motorcycle

ıd sellers to
ırks. Where
-Wesley was
ɔeen printed

ı ordered in

ta

d G. Lyons.

itle.

96-28818
CIP

Inc.

ıay be repro-
n any form or
ıg, recording,
n of the pub-
ɔlished simul-

DEDICATION

I dedicate this book to my two daughters Julie and Meredith, I wish I could go with you; to my mother Ruth for making me finish my homework; to my father Grady who didn't know what he started when he built that workbench in the basement; to my brother Ray for improving us all; to my brother Ken who succeeded where I failed; to my sister Nancy for running interference for us; to John Lennon for not giving up; to Dr. Laura Schlessinger for keeping us honest; to my advisor Glenn Caldwell and to the Iron Riders Motorcycle Club (Niles, CA) who keep me alive.

CONTENTS

Preface xi

1 DISCRETE SEQUENCES AND SYSTEMS 1

1.1 Discrete Sequences and Their Notation 2
1.2 Signal Amplitude, Magnitude, Power 8
1.3 Signal Processing Operational Symbols 10
1.4 Introduction to Discrete Linear Time-Invariant Systems 12
1.5 Discrete Linear Systems 13
1.6 Time-Invariant Systems 18
1.7 The Commutative Property of Linear Time-Invariant Systems 20
1.8 Analyzing Linear Time-Invariant Systems 20

2 PERIODIC SAMPLING 23

2.1 Aliasing: Signal Ambiguity in the Frequency Domain 23
2.2 Sampling Low-Pass Signals 29
2.3 Sampling Bandpass Signals 32
2.4 Spectral Inversion in Bandpass Sampling 43

3 THE DISCRETE FOURIER TRANSFORM 49

3.1 Understanding the DFT Equation 50
3.2 DFT Symmetry 63
3.3 DFT Linearity 65
3.4 DFT Magnitudes 66
3.5 DFT Frequency Axis 67
3.6 DFT Shifting Theorem 68
3.7 Inverse DFT 70
3.8 DFT Leakage 71
3.9 Windows 80
3.10 DFT Scalloping Loss 88
3.11 DFT Resolution, Zero Stuffing, and Frequency-Domain Sampling 89
3.12 DFT Processing Gain 93
3.13 The DFT of Rectangular Functions 97
3.14 The DFT Frequency Response to a Complex Input 119
3.15 The DFT Frequency Response to a Real Cosine Input 123
3.16 The DFT Single-Bin Frequency Response to a Real Cosine Input 125

4 THE FAST FOURIER TRANSFORM 129

4.1 Relationship of the FFT to the DFT 130
4.2 Hints on Using FFTs in Practice 131
4.3 FFT Software Programs 136
4.4 Derivation of the Radix-2 FFT Algorithm 136
4.5 FFT Input/Output Data Index Bit Reversal 145
4.6 Radix-2 FFT Butterfly Structures 146

5 FINITE IMPULSE RESPONSE FILTERS 157

5.1 An Introduction to Finite Impulse Response FIR Filters 158
5.2 Convolution in FIR Filters 163
5.3 Low-Pass FIR Filter Design 174
5.4 Bandpass FIR Filter Design 191
5.5 Highpass FIR Filter Design 193
5.6 Remez Exchange FIR Filter Design Method 194
5.7 Half-Band FIR Filters 197
5.8 Phase Response of FIR Filters 199
5.9 A Generic Description of Discrete Convolution 204

6 INFINITE IMPULSE RESPONSE FILTERS 219

6.1 An Introduction to Infinite Impulse Response Filters 220
6.2 The Laplace Transform 223
6.3 The z-Transform 238
6.4 Impulse Invariance IIR Filter Design Method 254
6.5 Bilinear Transform IIR Filter Design Method 272
6.6 Optimized IIR Filter Design Method 284
6.7 Pitfalls in Building IIR Digital Filters 286
6.8 Cascade and Parallel Combinations of Digital Filters 290
6.9 A Brief Comparison of IIR and FIR Filters 292

7 ADVANCED SAMPLING TECHNIQUES 297

7.1 Quadrature Sampling 297
7.2 Quadrature Sampling with Digital Mixing 301
7.3 Digital Resampling 303

8 SIGNAL AVERAGING 319

8.1 Coherent Averaging 320
8.2 Incoherent Averaging 327
8.3 Averaging Multiple Fast Fourier Transforms 330
8.4 Filtering Aspects of Time-Domain Averaging 340
8.5 Exponential Averaging 341

9 DIGITAL DATA FORMATS AND THEIR EFFECTS 349

9.1 Fixed-Point Binary Formats 349
9.2 Binary Number Precision and Dynamic Range 356
9.3 Effects of Finite Fixed-Point Binary Word Length 357
9.4 Floating-Point Binary Formats 375
9.5 Block Floating-Point Binary Format 381

10 DIGITAL SIGNAL PROCESSING TRICKS 385

10.1 Frequency Translation without Multiplication 385
10.2 High-Speed Vector-Magnitude Approximation 400
10.3 Data Windowing Tricks 406
10.4 Fast Multiplication of Complex Numbers 411
10.5 Efficiently Performing the FFT of Real Sequences 412
10.6 Calculating the Inverse FFT Using the Forward FFT 425
10.7 Fast FFT Averaging 429
10.8 Simplified FIR Filter Structure 430
10.9 Accurate A/D Converter Testing Technique 432
10.10 Fast FIR Filtering Using the FFT 435
10.11 Calculation of Sines and Cosines of Consecutive Angles 436
10.12 Generating Normally Distributed Random Data 438

APPENDIX A. THE ARITHMETIC OF COMPLEX NUMBERS 443

A.1 Graphical Representation of Real and Complex Numbers 443
A.2 Arithmetic Representation of Complex Numbers 444
A.3 Arithmetic Operations of Complex Numbers 446
A.4 Some Practical Implications of Using Complex Numbers 453

APPENDIX B. CLOSED FORM OF A GEOMETRIC SERIES 455

APPENDIX C. COMPLEX SIGNALS AND NEGATIVE FREQUENCY 458

C.1 Development of Imaginary Numbers 460
C.2 Representing Real Signals Using Complex Phasors 462
C.3 Representing Real Signals Using Negative Frequencies 467
C.4 Complex Signals and Quadrature Mixing 471

APPENDIX D. MEAN, VARIANCE, AND STANDARD DEVIATION 476

D.1 Statistical Measures 476
D.2 Standard Deviation, or RMS, of a Continuous Sinewave 480
D.3 The Mean and Variance of Random Functions 481
D.4 The Normal Probability Density Function 484

APPENDIX E. DECIBELS (dB AND dBm) 486

E.1 Using Logarithms to Determine Relative Signal Power 486
E.2 Some Useful Decibel Numbers 492
E.3 Absolute Power Using Decibels 493

APPENDIX F. DIGITAL FILTER TERMINOLOGY 494

Index 507

PREFACE

Learning Digital Signal Processing

Learning the fundamentals, and how to speak the language, of digital signal processing does not require profound analytical skills or an extensive background in mathematics. All you need is a little experience with elementary algebra, knowledge of what a sinewave is, this book, and enthusiasm. This may sound hard to believe, particularly if you've just flipped through the pages of this book and seen figures and equations that appear rather complicated. The content here, you say, looks suspiciously like the material in technical journals and textbooks, material that is difficult to understand. Well, this is not just another book on digital signal processing.

This book's goal is to gently provide explanation followed by illustration, not so that you may understand the material, but that you must understand the material.[†] Remember the first time you saw two people playing chess? The game probably appeared to be mysterious and confusing. As you now know, no individual chess move is complicated. Given a little patience, the various chess moves are easy to learn. The game's complexity comes from deciding what combinations of moves to make and when to make them. So it is with understanding digital signal processing. First we learn the fundamental rules and processes and, then, practice using them in combination.

If learning digital signal processing is so easy, then why does the subject have the reputation of being difficult to understand? The answer lies partially in how the material is typically presented in the literature. It's difficult to convey technical information, with its mathematical subtleties, in written form. It's one thing to write equations, but it's another matter altogether to explain what those equations really mean from a practical standpoint, and that's the goal of this book.

Too often, written explanation of digital signal processing theory appears in one of two forms: either mathematical miracles occur and you are simply given a short and sweet equation without further explanation, or you are engulfed in a flood of complex variable equations and phrases

[†] "Here we have the opportunity of expounding more clearly what has already been said" (Rene Descartes).

such as "it is obvious that," "such that $W(f) \geq 0 \; \forall \; f$," and "with judicious application of the homogeneity property." Authors usually do provide the needed information, but, too often, the reader must grab a pick and shovel, put on a miner's helmet, and try to dig the information out of a mountain of mathematical expressions. (This book presents the results of several fruitful mining expeditions.) How many times have you followed the derivation of an equation, after which the author states that he or she is going to illustrate that equation with a physical example—and this turns out to be another equation? Although mathematics is necessary to describe digital signal processing, I've tried to avoid overwhelming the reader because a recipe for technical writing that's too rich in equations is hard for the beginner to digest.[†]

The intent of this book is expressed in a popular quote from E. B. White in the introduction of his *Elements of Style* (New York: Macmillan Publishing, 1959):

> Will (Strunk) felt that the reader was in serious trouble most of the time, a man floundering in a swamp, and that it was the duty of anyone attempting to write English to drain the swamp quickly and get his man up on dry ground, or at least throw him a rope.

I've attempted to avoid the traditional instructor-student relationship, but, rather, to make reading this book like talking to a friend while walking in the park. I've used just enough mathematics to develop a fundamental understanding of the theory, and, then, illustrate that theory with examples.

The Journey

Learning digital signal processing is not something you accomplish; it's a journey you take. When you gain an understanding of some topic, questions arise that cause you to investigate some other facet of digital signal processing. Armed with more knowledge, you're likely to begin exploring further aspects of digital signal processing much like those shown in the following diagram. This book is your tour guide during the first steps of your journey.

You don't need a computer to learn the material in this book, but it would help. Digital signal processing software allows the beginner to ver-

[†] "We need elucidation of the obvious more than investigation of the obscure" (Oliver Wendell Holmes).

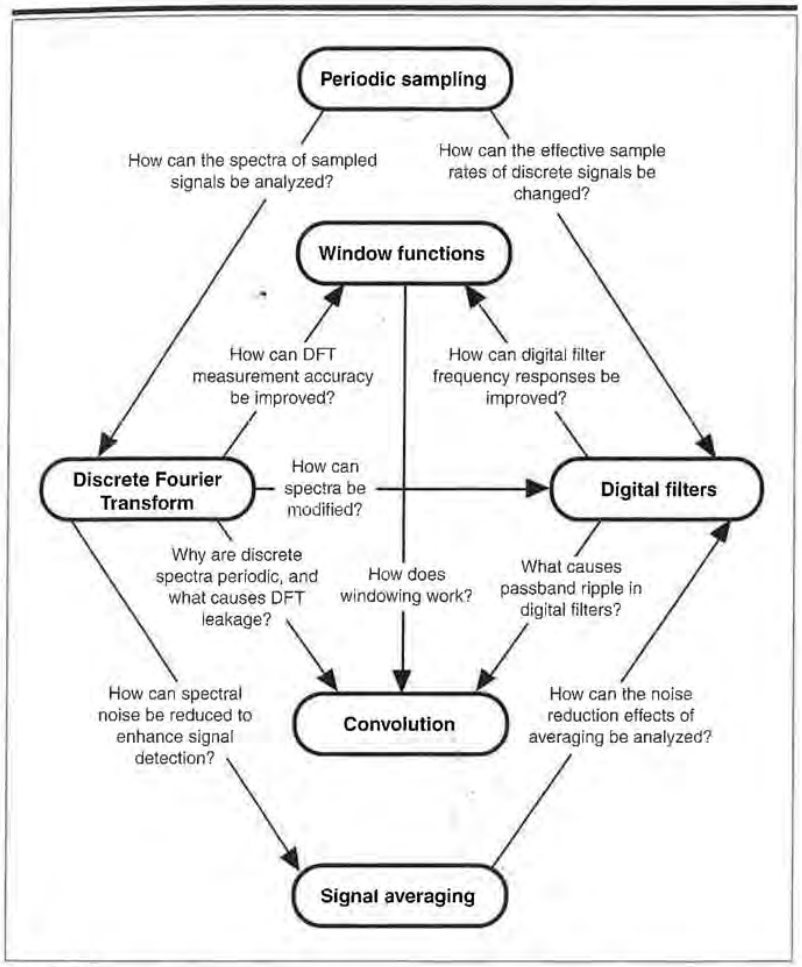


Figure P-1

ify signal processing theory through trial and error.[†] In particular, software routines that plot signal data, perform the fast Fourier transform, and analyze digital filters would be very useful.

As you go through the material in this book, don't be discouraged if your understanding comes slowly. As the Greek mathematician Menaechmus

[†] "One must learn by doing the thing; for though you think you know it, you have no certainty until you try it" (Sophocles).

curtly remarked to Alexander the Great, when asked for a quick explanation of mathematics, "There is no royal road to mathematics." Menaechmus was confident in telling Alexander that the only way to learn mathematics is through careful study. The same applies to digital signal processing. Also, don't worry if you have to read some of the material twice. While the concepts in this book are not as complicated as quantum physics, as mysterious as the lyrics of the song "Louie Louie," or as puzzling as the assembly instructions of a metal shed, they do get a little involved. They deserve your attention and thought. So go slow and read the material twice if you have to; you'll be glad you did. If you show persistence, to quote a phrase from Susan B. Anthony, "Failure is impossible."

Coming Attractions

Chapter 1 of this book begins by establishing the notation used throughout the remainder of our study. In that chapter, we introduce the concept of discrete signal sequences, show how they relate to continuous signals, and illustrate how those sequences can be depicted in both the time and frequency domains. In addition, Chapter 1 defines the operational symbols we'll use to build our signal processing system block diagrams. We conclude that chapter with a brief introduction to the idea of linear systems and see why linearity enables us to use a number of powerful mathematical tools in our analysis.

Chapter 2 introduces the most frequently misunderstood process in digital signal processing, periodic sampling. Although it's straightforward to grasp the concept of sampling a continuous signal, there are mathematical subtleties in the process that require thoughtful attention. Beginning gradually with simple examples of low-pass sampling and progressing to the interesting subject of bandpass sampling, Chapter 2 explains and quantifies the frequency-domain ambiguity (aliasing) associated with these important topics. The discussion there highlights the power and pitfalls of periodic sampling.

Chapter 3 is devoted to one of the foremost topics in digital signal processing, the discrete Fourier transform (DFT). Coverage begins with detailed examples illustrating the important properties of the DFT and how to interpret DFT spectral results, progresses to the topic of windows used to reduce DFT leakage, and discusses the processing gain afforded by the DFT. The chapter concludes with a detailed discussion of the various forms of the transform of rectangular functions that the beginner is likely to encounter in the literature. That last topic is included there to clarify and illustrate the DFT of both real and complex sinusoids.

Chapter 4 covers the innovation that made the most profound impact on the field of digital signal processing, the fast Fourier transform (FFT). There we show the relationship of the popular radix-2 FFT to the DFT, quantify the powerful processing advantages gained by using the FFT, demonstrate why the FFT functions as it does, and present various FFT implementation structures. Chapter 4 also includes a list of recommendations to help us when we use the FFT in practice.

Chapter 5 ushers in the subject of digital filtering. Beginning with a simple low-pass finite impulse response (FIR) filter example, we carefully progress through the analysis of that filter's frequency-domain magnitude and phase response. Next we learn how window functions affect and can be used to design FIR filters. The methods for converting low-pass FIR filter designs to bandpass and highpass digital filters are presented, and the popular Remez Exchange (Parks McClellan) FIR filter design technique is introduced and illustrated by example. In that chapter we acquaint the reader with, and take the mystery out of, the process called convolution. Proceeding through several simple convolution examples, we conclude Chapter 5 with a discussion of the powerful convolution theorem and show why it's so useful as a qualitative tool in understanding digital signal processing.

Chapter 6 introduces a second class of digital filters, infinite impulse response (IIR) filters. In discussing several methods for the design of IIR filters, the reader is introduced to the powerful digital signal processing analysis tool called the z-transform. Because the z-transform is so closely related to the continuous Laplace transform, Chapter 6 starts by gently guiding the reader from the origin, through the properties, and on to the utility of the Laplace transform in preparation for learning the z-transform. We'll see how IIR filters are designed and implemented, and why their performance is so different from FIR filters. To indicate under what conditions these filters should be used, the chapter concludes with a qualitative comparison of the key properties of FIR and IIR filters.

Chapter 7 discusses two important advanced sampling techniques prominent in digital signal processing, quadrature sampling and digital resampling. In the chapter we discover why quadrature sampling is so useful when signal phase must be analyzed and preserved, and how this special sampling process can circumvent some of the limitations of traditional periodic sampling techniques. Our introduction to digital resampling shows how we can, and when we should, change the effective sample rate of discrete data after the data has already been digitized. We've delayed the discussion of digital resampling to this chapter

because some knowledge of low-pass digital filters is necessary to understand how resampling schemes operate.

Chapter 8 covers the important topic of signal averaging. There we learn how averaging increases the accuracy of signal measurement schemes by reducing measurement background noise. This accuracy enhancement is called processing gain, and the chapter shows how to predict the processing gain associated with averaging signals in both the time and frequency domains. In addition, the key differences between coherent and incoherent averaging techniques are explained and demonstrated with examples. To complete the chapter, the popular scheme known as exponential averaging is covered in some detail.

Chapter 9 presents an introduction to the various binary number formats that the reader is likely to encounter in modern digital signal processing. We establish the precision and dynamic range afforded by these formats along with the inherent pitfalls associated with their use. Our exploration of the critical subject of binary data word width (in bits) naturally leads us to a discussion of the numerical resolution limitations of analog to digital (A/D) converters and how to determine the optimum A/D converter word size for a given application. The problems of data value overflow roundoff errors are covered along with a statistical introduction to the two most popular remedies for overflow, truncation and rounding. We end the chapter by covering the interesting subject of floating-point binary formats that allow us to overcome most of the limitations induced by fixed-point binary formats, particularly in reducing the ill effects of data overflow.

Chapter 10 provides a collection of *tricks of the trade* that the professionals often use to make their digital signal processing algorithms more efficient. Those techniques are compiled into a chapter at the end of the book for two reasons. First, it seems wise to keep our collection of tricks in one chapter so that we'll know where to find them in the future. Second, many of these schemes require an understanding of the material from the previous chapters, so the last chapter is an appropriate place to keep our collection of clever tricks. Exploring these techniques in detail verifies and reiterates many of the important ideas covered in previous chapters.

The appendices include a number of topics to help the beginner understand the mathematics of digital signal processing. A comprehensive description of the arithmetic of complex numbers is covered in Appendix A, while Appendix B derives the often used, but seldom explained, closed form of a geometric series. Appendix C strives to clarify the troubling topics of complex signals and negative frequency. The statistical concepts of

mean, variance, and standard deviation are introduced and illustrated in Appendix D, and Appendix E provides a discussion of the origin and utility of the logarithmic decibel scale used to improve the magnitude resolution of spectral representations. In a slightly different vein, Appendix F provides a glossary of the terminology used in the field of digital filters.

Acknowledgments

How do I sufficiently thank the people who helped me write this book? I do this by stating that any quality existing herein is due to the following talented people: for their patient efforts in the unpleasant task of reviewing early versions of the manuscript, I am grateful to Sean McCrory, Paul Chestnut, Paul Kane, John Winter, Terry Daubek, and Robin Wiprud. Special thanks go to Nancy Silva for her technical and literary guidance, and encouragement, without which this book would not have been written. For taking time to help me understand digital signal processing, I thank Frank Festini, Harry Glaze, and Dick Sanborn. I owe you people.

Gratitude goes to the reviewers, under the auspices of Addison-Wesley, whose suggestions improved much of the material. They are Mark Sullivan, David Goodman, Satyanarayan Namdhari, James Kresse, Ryerson Gewalt, David Cullen, Richard Herbert, Maggie Carr, and anonymous at Alcatel Bell. Finally, I acknowledge my good fortune in being able to work with those talented folks at Addison-Wesley: Rosa Aimée González, Simon Yates, and Tara Herries.

If you're still with me this far into the Preface, I end by saying that I had a ball writing this book and hope you get some value out of reading it.

CHAPTER ONE

Discrete Sequences and Systems

Digital signal processing has never been more prevalent or easier to perform. It wasn't that long ago when the fast Fourier transform (FFT), a topic we'll discuss in Chapter 4, was a mysterious mathematical process used only in industrial research centers and universities. Now, amazingly, the FFT is readily available to us all. It's even a built-in function provided by inexpensive spreadsheet software for home computers. The availability of more sophisticated commercial signal processing software now allows us to analyze and develop complicated signal processing applications rapidly and reliably. We can now perform spectral analysis, design digital filters, develop voice recognition, data communication, and image compression processes using software that's interactive in both the way algorithms are defined and how the resulting data are graphically displayed. Since the mid-1980s the same integrated circuit technology that led to affordable home computers has produced powerful and inexpensive hardware development systems on which to implement our digital signal processing designs.[†] Regardless, though, of the ease with which these new digital signal processing development systems and software can be applied, we still need a solid foundation in understanding the basics of digital signal processing. The purpose of this book is to build that foundation.

In this chapter we'll set the stage for the topics we'll study throughout the remainder of this book by defining the terminology used in digital signal

[†] During a television interview in the early 1990s, a leading computer scientist stated that had automobile technology made the same strides as the computer industry, we'd all have a car that would go a half million miles per hour and get a half million miles per gallon. The cost of that car would be so low that it would be cheaper to throw it away than pay for one day's parking in San Francisco.

processing, illustrating the various ways of graphically representing discrete signals, establishing the notation used to describe sequences of data values, presenting the symbols used to depict signal processing operations, and briefly introducing the concept of a linear discrete system.

1.1 Discrete Sequences and Their Notation

In general, the term *signal processing* refers to the science of analyzing time-varying physical processes. As such, signal processing is divided into two categories, analog signal processing and digital signal processing. The term *analog* is used to describe a waveform that's continuous in time and can take on a continuous range of amplitude values. An example of an analog signal is some voltage that can be applied to an oscilloscope resulting in a continuous display as a function of time. Analog signals can also be applied to a conventional spectrum analyzer to determine their frequency content. The term *analog* appears to have stemmed from the analog computers used prior to 1980. These computers solved linear differential equations by means of connecting physical (electronic) differentiators and integrators using old-style telephone operator patch cords. That way, a continuous voltage or current in the actual circuit was *analogous* to some variable in a differential equation, such as speed, temperature, air pressure, etc. (Although the flexibility and speed of modern-day digital computers have since made analog computers obsolete, a good description of the short-lived utility of analog computers can be found in reference [1].) Because present-day signal processing of continuous radio-type signals using resistors, capacitors, operational amplifiers, etc., has nothing to do with analogies, the term *analog* is actually a misnomer. The more correct term is *continuous signal processing* for what is today so commonly called analog signal processing. As such, in this book we'll minimize the use of the term *analog signals* and substitute the phrase *continuous signals* whenever appropriate.

The term *discrete-time signal* is used to describe a signal whose independent time variable is quantized so that we know only the value of the signal at discrete instants in time. Thus a discrete-time signal is not represented by a continuous waveform but, instead, a sequence of values. In addition to quantizing time, a discrete-time signal quantizes the signal amplitude. We can illustrate this concept with an example. Think of a continuous sinewave with a peak amplitude of 1 at a frequency f_o described by the equation

$$x(t) = \sin(2\pi f_o t) . \tag{1-1}$$

The frequency f_o is measured in hertz (Hz). (In physical systems, we usually measure frequency in units of hertz. One Hz is a single oscillation, or cycle, per second. One kilohertz (kHz) is a thousand Hz, and a megahertz (MHz) is one million Hz.[†]) With t in Eq. 1-1 representing time in seconds, the $f_o t$ factor has dimensions of cycles, and the complete $2\pi f_o t$ term is an angle measured in radians.

Plotting Eq. (1-1), we get the venerable continuous sinewave curve shown in Figure 1-1(a). If our continuous sinewave represents a physical voltage, we could *sample* it once every t_s seconds using an analog-to-digital converter and represent the sinewave as a sequence of discrete values. Plotting those individual values as dots would give us the discrete waveform in Figure 1-1(b). We say that Figure 1-1(b) is the "discrete-time" version of the continuous signal in Figure 1-1(a). The independent variable t in Eq. (1-1) and Figure 1-1(a) is continuous. The independent *index* variable n in Figure 1-1(b) is discrete and can have only integer values. That is, index n is used to identify the individual elements of the discrete sequence in Figure 1-1(b).

Do not be tempted to draw lines between the dots in Figure 1-1(b). For some reason, people (particularly those engineers experienced in working with continuous signals) want to connect the dots with straight lines, or the stairstep lines shown in Figure 1-1(c). Don't fall into this innocent-looking trap. Connecting the dots can mislead the beginner into forgetting that the $x(n)$ sequence is nothing more than a list of numbers. Remember, $x(n)$ is a discrete-time sequence of individual values, and each value in that sequence plots as a single dot. It's not that we're ignorant of what lies between the dots of $x(n)$; there *is* nothing between those dots.

We can reinforce this discrete-time sequence concept by listing those Figure 1-1(b) sampled values as follows:

$$\begin{aligned}
&x(0) = 0 && \text{(1st sequence value, index } n = 0\text{)}\\
&x(1) = 0.31 && \text{(2nd sequence value, index } n = 1\text{)}\\
&x(2) = 0.59 && \text{(3rd sequence value, index } n = 2\text{)}\\
&x(3) = 0.81 && \text{(4th sequence value, index } n = 3\text{)}\\
&\quad\ldots && \quad\ldots\\
& && \text{and so on,}
\end{aligned} \tag{1-2}$$

[†] The dimension for frequency used to be *cycles/second*; that's why the tuning dials of old radios indicate frequency as kilocycles/second (kcps) or megacycles/second (Mcps). In 1960 the scientific community adopted hertz as the unit of measure for frequency in honor of the German physicist, Heinrich Hertz, who first demonstrated radio wave transmission and reception in 1887.

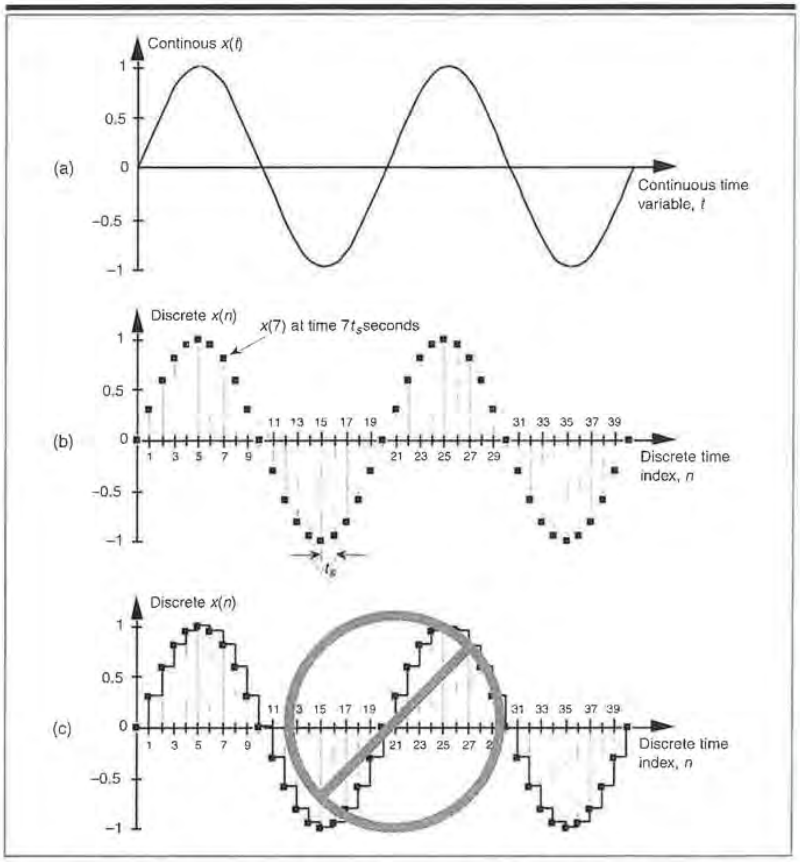


Figure 1-1 A time-domain sinewave: (a) continuous waveform representation; (b) discrete sample representation; (c) discrete samples with connecting lines.

where n represents the time index integer sequence 0, 1, 2, 3, etc., and t_s is some constant time period. Those sample values can be represented collectively, and concisely, by the discrete-time expression

$$x(n) = \sin(2\pi f_o n t_s) \,. \tag{1-3}$$

(Here again, the $2\pi f_o n t_s$ term is an angle measured in radians.) Notice that the index n in Eq. (1-2) started with a value of 0, instead of 1. There's nothing sacred about this; the first value of n could just as well have been 1, but we start the index n at zero out of habit because doing so allows us to

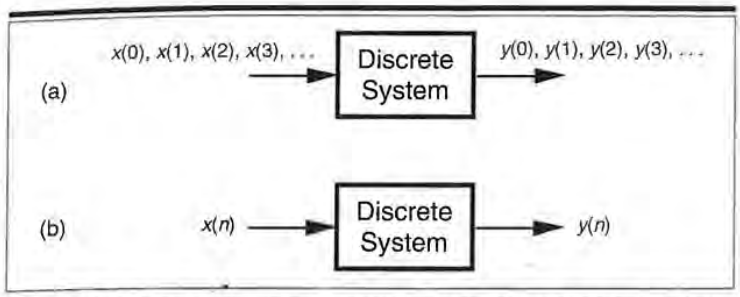


Figure 1-2 With an input applied, a discrete system provides an output: (a) the input and output are sequences of individual values; (b) input and output using the abbreviated notation of $x(n)$ and $y(n)$.

describe the sinewave starting at time zero. The variable $x(n)$ in Eq. (1-3) is read as "the sequence x of n." Equations (1-1) and (1-3) describe what are also referred to as *time-domain* signals because the independent variables, the continuous time t in Eq. (1-1), and the discrete-time nt_s values used in Eq. (1-3) are measures of time.

With this notion of a discrete-time signal in mind, let's say that a discrete system is a collection of hardware components, or software routines, that operate on a discrete-time signal sequence. For example, a discrete system could be a process that gives us a discrete output sequence $y(0)$, $y(1)$, $y(2)$, etc., when a discrete input sequence of $x(0)$, $x(1)$, $x(2)$, etc., is applied to the system input as shown in Figure 1-2(a). Again, to keep the notation concise and still keep track of individual elements of the input and output sequences, an abbreviated notation is used as shown in Figure 1-2(b) where n represents the integer sequence 0, 1, 2, 3, etc. Thus, $x(n)$ and $y(n)$ are general variables that represent two separate sequences of numbers. Figure 1-2(b) allows us to describe a system's output with a simple expression such as

$$y(n) = 2x(n) - 1 . \tag{1-4}$$

Illustrating Eq. (1-4), if $x(n)$ is the five-element sequence: $x(0) = 1$, $x(1) = 3$, $x(2) = 5$, $x(3) = 7$, and $x(4) = 9$, then $y(n)$ is the five-element sequence $y(0) = 1$, $y(1) = 5$, $y(2) = 9$, $y(3) = 13$, and $y(4) = 17$.

The fundamental difference between the way time is represented in continuous and discrete systems leads to a very important difference in how we characterize frequency in continuous and discrete systems. To illustrate, let's reconsider the continuous sinewave in Figure 1-1(a). If it represented a voltage at the end of a cable, we could measure its

frequency by applying it to an oscilloscope, a spectrum analyzer, or a frequency counter. We'd have a problem, however, if we were merely given the list of values from Eq. (1-2) and asked to determine the frequency of the waveform they represent. We'd graph those discrete values, and, sure enough, we'd recognize a single sinewave as in Figure 1-1(b). We can say that the sinewave repeats every 20 samples, but there's no way to determine the exact sinewave frequency from the discrete sequence values alone. You can probably see the point we're leading to here. If we knew the time between samples—the sample period t_s—we'd be able to determine the absolute frequency of the discrete sinewave. Given that the t_s sample period is, say, 0.05 milliseconds/sample, the period of the sinewave is

$$\text{sinewave period} = \frac{20 \text{ samples}}{\text{period}} \cdot \frac{0.05 \text{ milliseconds}}{\text{sample}} = 1 \text{ millisecond.} \quad (1\text{-}5)$$

Because the frequency of a sinewave is the reciprocal of its period, we now know that the sinewave's absolute frequency is 1/(1 ms), or 1 kHz. On the other hand, if we found that the sample period was, in fact, 2 milliseconds, the discrete samples in Figure 1-1(b) would represent a sinewave whose period is 40 milliseconds and whose frequency is 25 Hz. The point here is that, in discrete systems, absolute frequency determination in Hz is dependent on the sample frequency $f_s = 1/t_s$. We'll be reminded of this dependence throughout the rest of this book.

In digital signal processing, we often find it necessary to characterize the frequency content of discrete-time domain signals. When we do so, this frequency representation takes place in what's called the *frequency domain*. By way of example, let's say we have a discrete sinewave sequence $x_1(n)$ with an arbitrary frequency f_o Hz as shown on the left side of Figure 1-3(a). We can also describe $x_1(n)$ as shown on the right side of Figure 1-3(a) by indicating that it has a frequency of 1, measured in units of f_o, and no other frequency content. Although we won't dwell in it just now, notice that the frequency-domain representations in Figure 1-3 are themselves discrete.

To illustrate our time- and frequency-domain representations further, Figure 1-3(b) shows another discrete sinewave $x_2(n)$, whose peak amplitude is 0.4, with a frequency of $2f_o$. The discrete sample values of $x_2(n)$ are expressed by the equation

$$x_2(n) = 0.4 \cdot \sin(2\pi 2 f_o n t_s) . \quad (1\text{-}6)$$

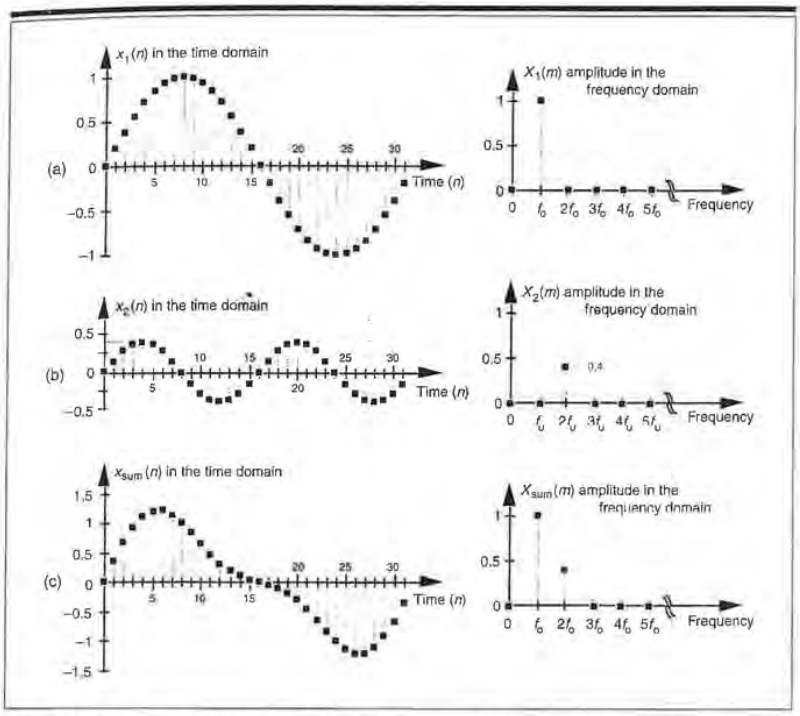


Figure 1-3 Time- and frequency-domain graphical representations: (a) sinewave of frequency f_o; (b) reduced amplitude sinewave of frequency $2f_o$; (c) sum of the two sinewaves.

When the two sinewaves, $x_1(n)$ and $x_2(n)$, are added to produce a new waveform $x_{sum}(n)$, its time-domain equation is

$$x_{sum}(n) = x_1(n) + x_2(n) = \sin(2\pi f_o n t_s) + 0.4 \cdot \sin(2\pi 2 f_o n t_s) , \qquad (1\text{-}7)$$

and its time- and frequency-domain representations are those given in Figure 1-3(c). We interpret the $X_{sum}(m)$ frequency-domain depiction, the *spectrum*, in Figure 1-3(c) to indicate that $X_{sum}(n)$ has a frequency component of f_o Hz and a reduced-amplitude frequency component of $2f_o$ Hz.

Notice three things in Figure 1-3. First, time sequences use lowercase variable names like the "x" in $x_1(n)$, and uppercase symbols for frequency-domain variables such as the "X" in $X_1(m)$. The term $X_1(m)$ is read as "the spectral sequence X sub one of m." Second, because the $X_1(m)$ frequency-domain representation of the $x_1(n)$ time sequence is itself a sequence (a list

of numbers), we use the index "m" to keep track of individual elements in $X_1(m)$. We can list frequency-domain sequences just as we did with the time sequence in Eq. (1-2). For example $X_{sum}(m)$ is listed as

$X_{sum}(0) = 0$ (1st $X_{sum}(m)$ value, index $m = 0$)
$X_{sum}(1) = 1.0$ (2nd $X_{sum}(m)$ value, index $m = 1$)
$X_{sum}(2) = 0.4$ (3rd $X_{sum}(m)$ value, index $m = 2$)
$X_{sum}(3) = 0$ (4th $X_{sum}(m)$ value, index $m = 3$)
. . . . . .
and so on,

where the frequency index m is the integer sequence 0, 1, 2, 3, etc. Third, because the $x_1(n) + x_2(n)$ sinewaves have a phase shift of zero degrees relative to each other, we didn't really need to bother depicting this phase relationship in $X_{sum}(m)$ in Figure 1-3(c). In general, however, phase relationships in frequency-domain sequences are important, and we'll cover that subject in Chapters 3 and 5.

A key point to keep in mind here is that we now know three equivalent ways to describe a discrete-time waveform. Mathematically, we can use a time-domain equation like Eq. (1-6) for example. We can also represent a time-domain waveform graphically as we did on the left side of Figure 1-3, and we can depict its corresponding, discrete, frequency-domain equivalent as that on the right side of Figure 1-3.

As it turns out, the discrete-time domain signals we're concerned with are not only quantized in time; their amplitude values are also quantized. Because we represent all digital quantities with binary numbers, there's a limit to the resolution, or granularity, that we have in representing the values of discrete numbers. Although signal amplitude quantization can be an important consideration—we cover that particular topic in Chapter 9—we won't worry about it just now.

1.2 Signal Amplitude, Magnitude, Power

Let's define two important terms that we'll be using throughout this book: amplitude and magnitude. It's not surprising that, to the layman, these terms are typically used interchangeably. When we check our thesaurus, we find that they are synonymous.[†] In engineering, however, they mean

[†] Of course, layman are "other people." To the engineer, the brain surgeon is the layman. To the brain surgeon, the engineer is the layman.

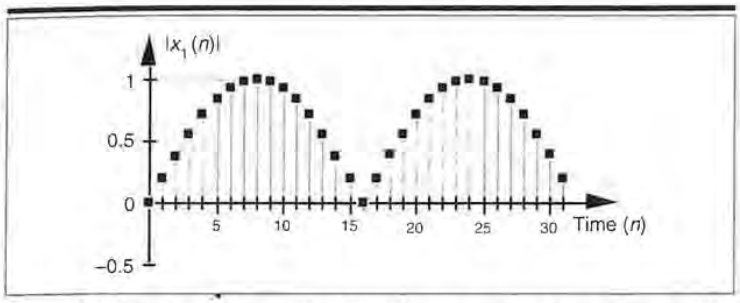


Figure 1-4 Magnitude samples, $|x_1(n)|$, of the time waveform in Figure 1-3(a).

two different things, and we must keep that difference clear in our discussions. The amplitude of a variable is the measure of how far, and in what direction, that variable differs from zero. Thus, signal amplitudes can be either positive or negative. The time-domain sequences in Figure 1-3 presented the sample value amplitudes of three different waveforms. Notice how some of the individual discrete amplitude values were positive and others were negative.

The magnitude of a variable, on the other hand, is the measure of how far, regardless of direction, its quantity differs from zero. So magnitudes are always positive values. Figure 1-4 illustrates how the magnitude of the $x_1(n)$ time sequence in Figure 1-3(a) is equal to the amplitude, but with the sign always being positive for the magnitude. We use the modulus symbol (| |) to represent the magnitude of $x_1(n)$. Occasionally, in the literature of digital signal processing, we'll find the term *magnitude* referred to as the *absolute value*.

When we examine signals in the frequency domain, we'll often be interested in the power level of those signals. The power of a signal is proportional to its amplitude (or magnitude) squared. If we assume that the proportionality constant is one, we can express the power of a sequence in the time or frequency domains as

$$x_{pwr}(n) = x(n)^2 = |x(n)|^2, \tag{1-8}$$

or

$$X_{pwr}(m) = X(m)^2 = |X(m)|^2. \tag{1-8'}$$

Very often we'll want to know the difference in power levels of two signals in the frequency domain. Because of the squared nature of power,

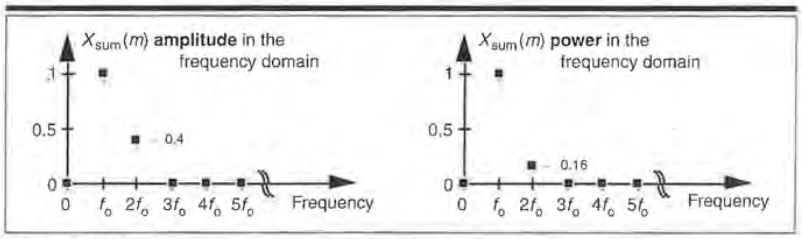


Figure 1-5 Frequency-domain amplitude and frequency-domain power of the $x_{sum}(n)$ time waveform in Figure 1-3(c).

two signals with moderately different amplitudes will have a much larger difference in their relative powers. In Figure 1-3, for example, signal $x_1(n)$'s amplitude is 2.5 times the amplitude of signal $x_2(n)$, but its power level is 6.25 that of $x_2(n)$'s power level. This is illustrated in Figure 1-5 where both the amplitude and power of $X_{sum}(m)$ are shown.

Because of their squared nature, plots of power values often involve showing both very large and very small values on the same graph. To make these plots easier to generate and evaluate, practitioners usually employ the decibel scale as described in Appendix E.

1.3 Signal Processing Operational Symbols

We'll be using block diagrams to graphically depict the way digital signal-processing operations are implemented. Those block diagrams will comprise an assortment of fundamental processing symbols, the most common of which are illustrated and mathematically defined in Figure 1-6.

Figure 1-6(a) shows the addition, element for element, of two discrete sequences to provide a new sequence. If our sequence index n begins at 0, we say that the first output sequence value is equal to the sum of the first element of the b sequence and the first element of the c sequence, or $a(0) = b(0) + c(0)$. Likewise, the second output sequence value is equal to the sum of the second element of the b sequence and the second element of the c sequence, or $a(1) = b(1) + c(1)$. Equation (1-7) is an example of adding two sequences. The subtraction process in Figure 1-6(b) generates an output sequence that's the element-for-element difference of the two input sequences. There are times when we must calculate a sequence whose elements are the sum of more than two values. This operation, illustrated in Figure 1-6(c), is called summation and is very common in

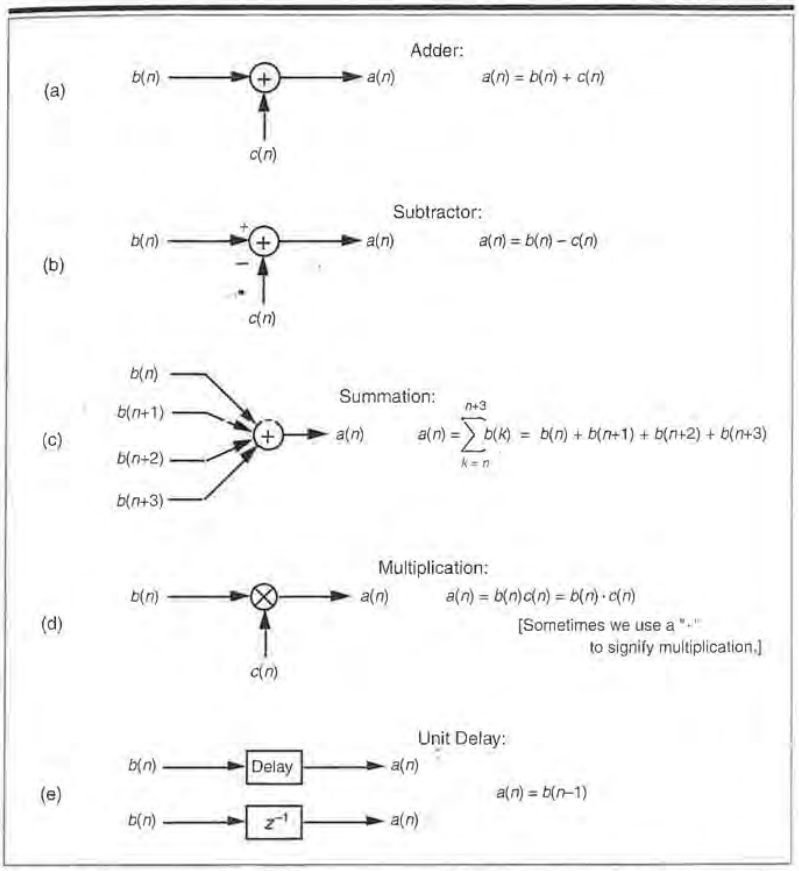


Figure 1-6 Terminology and symbols used in digital signal processing block diagrams.

digital signal processing. Notice how the lower and upper limits of the summation index k in the expression in Figure 1-6(c) tell us exactly which elements of the b sequence to sum to obtain a given $a(n)$ value. Because we'll encounter summation operations so often, let's make sure we understand their notation. If we repeat the summation equation from Figure 1-6(c) here we have

$$a(n) = \sum_{k=n}^{n+3} b(k) \,. \tag{1-9}$$

This means that

$$\begin{aligned}
&\text{when } n = 0, \text{ index } k \text{ goes from 0 to 3, so } && a(0) = b(0) + b(1) + b(2) + b(3)\\
&\text{when } n = 1, \text{ index } k \text{ goes from 1 to 4, so } && a(1) = b(1) + b(2) + b(3) + b(4)\\
&\text{when } n = 2, \text{ index } k \text{ goes from 2 to 5, so } && a(2) = b(2) + b(3) + b(4) + b(5)\\
&\text{when } n = 3, \text{ index } k \text{ goes from 3 to 6, so } && a(3) = b(3) + b(4) + b(5) + b(6)\\
&\quad \ldots && \ldots\\
&\qquad\qquad \text{and so on.} && \qquad (1\text{-}10)
\end{aligned}$$

We'll begin using summation operations in earnest when we discuss digital filters in Chapter 5.

The multiplication of two sequences is symbolized in Figure 1-6(d). Multiplication generates an output sequence that's the element-for-element product of two input sequences: $a(0) = b(0)c(0)$, $a(1) = b(1)c(1)$, and so on. The last fundamental operation that we'll be using is called the *unit delay* in Figure 1-6(e). While we don't need to appreciate its importance at this point, we'll merely state that the unit delay symbol signifies an operation where the output sequence $a(n)$ is equal to a delayed version of the $b(n)$ sequence. For example, $a(5) = b(4)$, $a(6) = b(5)$, $a(7) = b(6)$, etc. As we'll see in Chapter 6, due to the mathematical techniques used to analyze digital filters, the unit delay is very often depicted using the term z^{-1}.

The symbols in Figure 1-6 remind us of two important aspects of digital signal processing. First, our processing operations are always performed on sequences of individual discrete values, and second, the elementary operations themselves are very simple. It's interesting that, regardless of how complicated they appear to be, the vast majority of digital signal processing algorithms can be performed using combinations of these simple operations. If we think of a digital signal processing algorithm as a recipe, then the symbols in Figure 1-6 are the ingredients.

1.4 Introduction to Discrete Linear Time-Invariant Systems

In keeping with tradition, we'll introduce the subject of linear time-invariant (LTI) systems at this early point in our text. Although an appreciation for LTI systems is not essential in studying the next three chapters of this book, when we begin exploring digital filters, we'll build on the strict definitions of linearity and time invariance. We need to recognize and understand the notions of linearity and time invariance not just because the vast majority of discrete systems used in practice are LTI systems, but also

because LTI systems are very accommodating when it comes to their analysis. That's good news for us because we can use straightforward methods to predict the performance of any digital signal processing scheme as long as it's linear and time invariant. Because linearity and time invariance are two important system characteristics having very special properties, we'll discuss them now.

1.5 Discrete Linear Systems

The term *linear* defines a special class of systems where the output is the superposition, or sum, of the individual outputs had the individual inputs been applied separately to the system. For example, we can say that the application of an input $x_1(n)$ to a system results in an output $y_1(n)$. We symbolize this situation with the following expression:

$$x_1(n) \xrightarrow{\text{results in}} y_1(n) \tag{1-11}$$

Given a different input $x_2(n)$, the system has a $y_2(n)$ output as

$$x_2(n) \xrightarrow{\text{results in}} y_2(n) \ . \tag{1-12}$$

For the system to be linear, when its input is the sum $x_1(n) + x_2(n)$, its output must be the sum of the individual outputs so that

$$x_1(n) + x_2(n) \xrightarrow{\text{results in}} y_1(n) + y_2(n) \ . \tag{1-13}$$

One way to paraphrase expression (1-13) is to state that a linear system's output is the sum of the outputs of its parts. Also, part of this description of linearity is a proportionality characteristic. This means that if the inputs are scaled by constant factors c_1 and c_2 then the output sequence parts are also scaled by those factors as

$$c_1x_1(n) + c_2x_2(n) \xrightarrow{\text{results in}} c_1y_1(n) + c_2y_2(n) \ . \tag{1-14}$$

In the literature, this proportionality attribute of linear systems in expression (1-14) is sometimes called the *homogeneity property*. With these thoughts in mind, then, let's demonstrate the concept of system linearity.

1.5.1 Example of a Linear System

To illustrate system linearity, let's say we have the discrete system shown in Figure 1-7(a) whose output is defined as

$$y(n) = \frac{-x(n)}{2}, \tag{1-15}$$

that is, the output sequence is equal to the negative of the input sequence with the amplitude reduced by a factor of two. If we apply an $x_1(n)$ input sequence representing a 1-Hz sinewave sampled at a rate of 32 samples

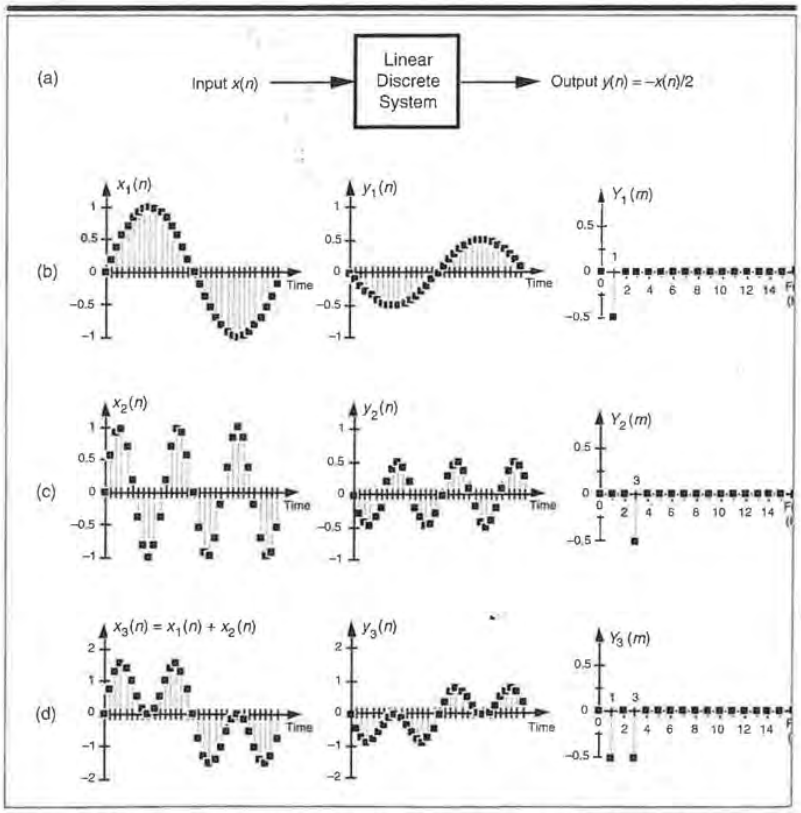


Figure 1-7 Linear system input-to-output relationships: (a) system block diagram where $y(n) = -x(n)/2$; (b) system input and output with a 1-Hz sinewave applied; (c) with a 3-Hz sinewave applied; (d) with the sum of 1-Hz and 3-Hz sinewaves applied.

per cycle, we'll have a $y_1(n)$ output as shown in the center of Figure 1-7(b). The frequency-domain spectral amplitude of the $y_1(n)$ output is the plot on the right side of Figure 1-7(b) indicating that the output comprises a single tone of peak amplitude equal to −0.5 whose frequency is 1 Hz. Next, applying an $x_2(n)$ input sequence representing a 3-Hz sinewave, the system provides a $y_2(n)$ output sequence, as shown in the center of Figure 1-7(c). The spectrum of the $y_2(n)$ output, $Y_2(m)$, confirming a single 3-Hz sinewave output is shown on the right side of Figure 1-7(c). Finally—here's where the linearity comes in—if we apply an $x_3(n)$ input sequence that's the sum of a 1-Hz sinewave and a 3-Hz sinewave, the $y_3(n)$ output is as shown in the center of Figure 1-7(d). Notice how $y_3(n)$ is the sample-for-sample sum of $y_1(n)$ and $y_2(n)$. Figure 1-7(d) also shows that the output spectrum $Y_3(m)$ is the sum of $Y_1(m)$ and $Y_2(m)$. That's linearity.

1.5.2 Example of a Nonlinear System

It's easy to demonstrate how a nonlinear system yields an output that is not equal to the sum of $y_1(n)$ and $y_2(n)$ when its input is $x_1(n) + x_2(n)$. A simple example of a nonlinear discrete system is that in Figure 1-8(a) where the output is the square of the input described by

$$y(n) = [x(n)]^2 . \tag{1-16}$$

We'll use a well-known trigonometric identity and a little algebra to predict the output of this nonlinear system when the input comprises simple sinewaves. Following the form of Eq. (1-3), let's describe a sinusoidal sequence, whose frequency $f_o = 1$ Hz, by

$$x_1(n) = \sin(2\pi f_o n t_s) = \sin(2\pi \cdot 1 \cdot n t_s) . \tag{1-17}$$

Equation (1-17) describes the $x_1(n)$ sequence on the left side of Figure 1-8(b). Given this $x_1(n)$ input sequence, the $y_1(n)$ output of the nonlinear system is the square of a 1-Hz sinewave, or

$$y_1(n) = [x_1(n)]^2 = \sin(2\pi \cdot 1 \cdot n t_s) \cdot \sin(2\pi \cdot 1 \cdot n t_s) . \tag{1-18}$$

We can simplify our expression for $y_1(n)$ in Eq. (1-18) by using the following trigonometric identity:

$$\sin(\alpha) \cdot \sin(\beta) = \frac{\cos(\alpha - \beta)}{2} - \frac{\cos(\alpha + \beta)}{2} . \tag{1-19}$$

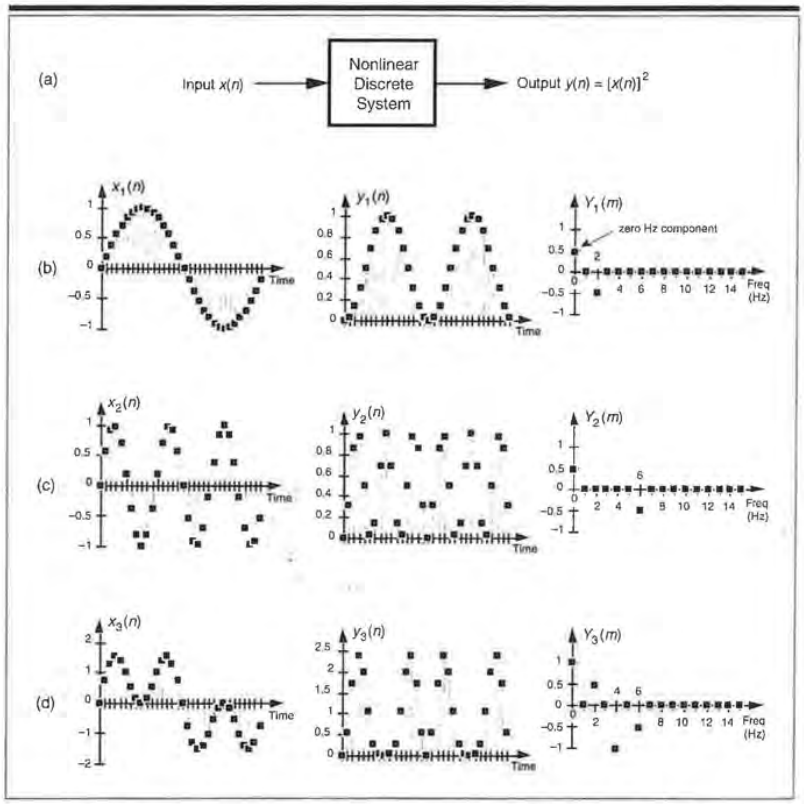


Figure 1-8 Nonlinear system input-to-output relationships: (a) system block diagram where $y(n) = (x(n))^2$; (b) system input and output with a 1-Hz sinewave applied; (c) with a 3-Hz sinewave applied; (d) with the sum of 1-Hz and 3-Hz sinewaves applied.

Using Eq. (1-19), we can express $y_1(n)$ as

$$y_1(n) = \frac{\cos(2\pi \cdot 1 \cdot nt_s - 2\pi \cdot 1 \cdot nt_s)}{2} - \frac{\cos(2\pi \cdot 1 \cdot nt_s + 2\pi \cdot 1 \cdot nt_s)}{2}$$

$$= \frac{\cos(0)}{2} - \frac{\cos(4\pi \cdot 1 \cdot nt_s)}{2} = \frac{1}{2} - \frac{\cos(2\pi \cdot 2 \cdot nt_s)}{2}, \qquad (1\text{-}20)$$

which is shown as the all positive sequence in the center of Figure 1-8(b). Because Eq. (1-19) results in a frequency sum ($\alpha + \beta$) and frequency difference ($\alpha - \beta$) effect when multiplying two sinusoids, the $y_1(n)$ output

sequence will be a cosine wave of 2 Hz and a peak amplitude of −0.5, added to a constant value of 1/2. The constant value of 1/2 in Eq. (1-20) is interpreted as a zero Hz frequency component, as shown in the $Y_1(m)$ spectrum in Figure 1-8(b). We could go through the same algebraic exercise to determine that, when a 3-Hz sinewave $x_2(n)$ sequence is applied to this nonlinear system, the output $y_2(n)$ would contain a zero Hz component and a 6 Hz component, as shown in Figure 1-8(c).

System nonlinearity is evident if we apply an $x_3(n)$ sequence comprising the sum of a 1-Hz and a 3-Hz sinewave as shown in Figure 1-8(d). We can predict the frequency content of the $y_3(n)$ output sequence by using the algebraic relationship

$$(a+b)^2 = a^2+2ab+b^2\,, \tag{1-21}$$

where a and b represent the 1-Hz and 3-Hz sinewaves, respectively. From Eq. (1-19), the a^2 term in Eq. (1-21) generates the zero-Hz and 2-Hz output sinusoids in Figure 1-8(b). Likewise, the b^2 term produces in $y_3(n)$ another zero-Hz and the 6-Hz sinusoid in Figure 1-8(c). However, the $2ab$ term yields additional 2-Hz and 4-Hz sinusoids in $y_3(n)$. We can show this algebraically by using Eq. (1-19) and expressing the $2ab$ term in Eq. (1-21) as

$$2ab = 2\cdot\sin(2\pi\cdot 1\cdot nt_s)\cdot\sin(2\pi\cdot 3\cdot nt_s)$$

$$= \frac{2\cos(2\pi\cdot 1\cdot nt_s - 2\pi\cdot 3\cdot nt_s)}{2} - \frac{2\cos(2\pi\cdot 1\cdot nt_s + 2\pi\cdot 3\cdot nt_s)}{2}$$

$$= \cos(2\pi\cdot 2\cdot nt_s) - \cos(2\pi\cdot 4\cdot nt_s)\ .^{\dagger} \tag{1-22}$$

Equation (1-22) tells us that two additional sinusoidal components will be present in $y_3(n)$ because of the system's nonlinearity, a 2-Hz cosine wave whose amplitude is +1 and a 4-Hz cosine wave having an amplitude of −1. These spectral components are illustrated in $Y_3(m)$ on the right side of Figure 1-8(d).

Notice that, when the sum of the two sinewaves is applied to the nonlinear system, the output contained sinusoids, Eq. (1-22), that were not present in either of the outputs when the individual sinewaves alone were applied. Those extra sinusoids were generated by an interaction of the

† The first term in Eq. (1-22) is $\cos(2\pi\cdot nt_s - 6\pi\cdot nt_s) = \cos(-4\pi\cdot nt_s) = \cos(-2\pi\cdot 2\cdot nt_s)$. However, because the cosine function is even, $\cos(-\alpha) = \cos(\alpha)$, we can express that first term as $\cos(2\pi\cdot 2\cdot nt_s)$.

two input sinusoids due to the squaring operation. That's nonlinearity; expression (1-13) was not satisfied. (Electrical engineers recognize this effect of internally generated sinusoids as *intermodulation distortion*.) Although nonlinear systems are usually difficult to analyze, they are occasionally used in practice. References [2], [3], and [4], for example, describe their application in nonlinear digital filters. Again, expressions (1-13) and (1-14) state that a linear system's output resulting from a sum of individual inputs, is the superposition (sum) of the individual outputs. They also stipulate that the output sequence $y_1(n)$ depends only on $x_1(n)$ combined with the system characteristics, and not on the other input $x_2(n)$, i.e., there's no interaction between inputs $x_1(n)$ and $x_2(n)$ at the output of a linear system.

1.6 Time-Invariant Systems

A time-invariant system is one where a time delay (or shift) in the input sequence causes a equivalent time delay in the system's output sequence. Keeping in mind that n is just an indexing variable we use to keep track of our input and output samples, let's say a system provides an output $y(n)$ given an input of $x(n)$, or

$$x(n) \xrightarrow{\text{results in}} y(n) \ . \tag{1-23}$$

For a system to be time invariant, with a shifted version of the original $x(n)$ input applied, $x'(n)$, the following applies:

$$x'(n) = x(n+k) \xrightarrow{\text{results in}} y'(n) = y(n+k) \ , \tag{1-24}$$

where k is some integer representing k sample period time delays. For a system to be time invariant, expression (1-24) must hold true for any integer value of k and any input sequence.

1.6.1 Example of a Time-Invariant System

Let's look at a simple example of time invariance illustrated in Figure 1-9. Assume that our initial $x(n)$ input is a unity-amplitude 1-Hz sinewave sequence with a $y(n)$ output, as shown in Figure 1-9(b). Consider a different input sequence $x'(n)$, where

$$x'(n) = x(n+4) \ . \tag{1-25}$$

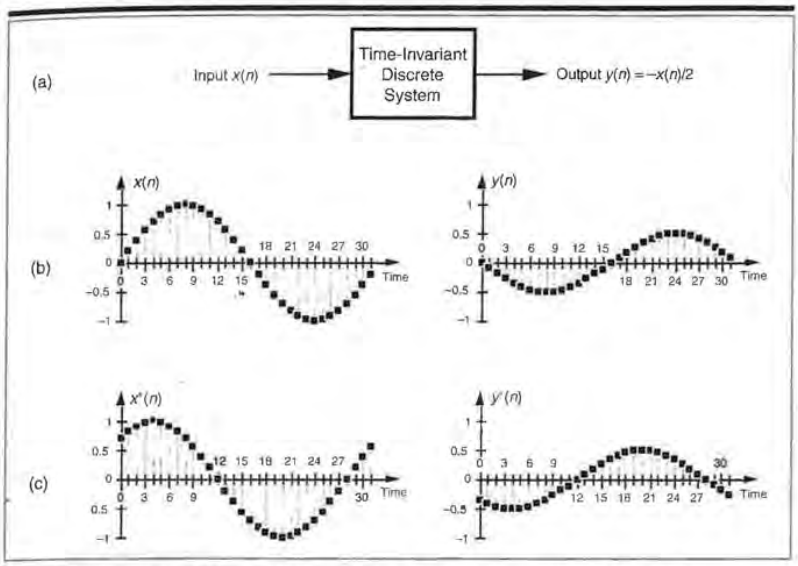


Figure 1-9 Time-invariant system input-to-output relationships: (a) system block diagram where $y(n) = -x(n)/2$; (b) system input and output with a 1-Hz sinewave applied; (c) system input and output when a 1-Hz sinewave, delayed by four samples, is applied. When $x'(n) = x(n+4)$, then, $y'(n) = y(n+4)$.

Equation (1-25) tells us that the input sequence $x'(n)$ is equal to sequence $x(n)$ shifted four samples to the left, that is, $x'(0) = x(4)$, $x'(1) = x(5)$, $x'(2) = x(6)$, and so on, as shown on the left of Figure 1-9(c). The discrete system is time invariant because the $y'(n)$ output sequence is equal to the $y(n)$ sequence shifted to the left by four samples, or $y'(n) = y(n+4)$. We can see that $y'(0) = y(4)$, $y'(1) = y(5)$, $y'(2) = y(6)$, and so on, as shown in Figure 1-9(c). For time-invariant systems, the y time shift is equal to the x time shift.

Some authors succumb to the urge to define a time-invariant system as one whose parameters do not change with time. That definition is incomplete and can get us in trouble if we're not careful. We'll just stick with the formal definition that a time-invariant system is one where a time shift in an input sequence results in an equal time shift in the output sequence. By the way, time-invariant systems in the literature are often called *shift-invariant* systems.†

† An example of a discrete process that's not time-invariant is the downsampling, or decimation, process described in Section 7.3.

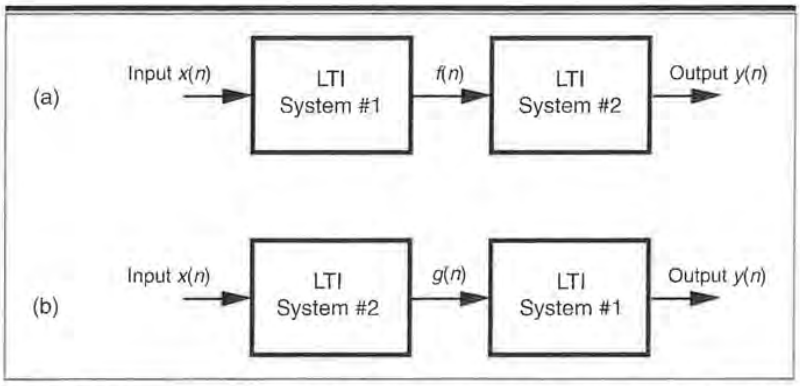


Figure 1-10 Linear time-invariant (LTI) systems in series: (a) block diagram of two LTI systems; (b) swapping the order of the two systems does not change the resultant output $y(n)$.

1.7 The Commutative Property of Linear Time-Invariant Systems

Although we don't substantiate this fact until we reach Section 6.8, it's not too early to realize that LTI systems have a useful commutative property by which their sequential order can be rearranged with no change in their final output. This situation is shown in Figure 1-10 where two different LTI systems are configured in series. Swapping the order of two cascaded systems does not alter the final output. Although the intermediate data sequences $f(n)$ and $g(n)$ will usually not be equal, the two pairs of LTI systems will have identical $y(n)$ output sequences. This commutative characteristic comes in handy for designers of digital filters, as we'll see in Chapters 5 and 6.

1.8 Analyzing Linear Time-Invariant Systems

As previously stated, LTI systems can be analyzed to predict their performance. Specifically, if we know the *unit impulse response* of an LTI system, we can calculate everything there is to know about the system; that is, the system's unit impulse response completely characterizes the system. By unit impulse response, we mean the system's time-domain output sequence when the input is a single unity-valued sample (unit impulse) preceded and followed by zero-valued samples as shown in Figure 1-11(b).

Knowing the (unit) impulse response of an LTI system, we can determine the system's output sequence for any input sequence because the output is equal to the *convolution* of the input sequence and the system's impulse response. Moreover, given an LTI system's time-domain impulse

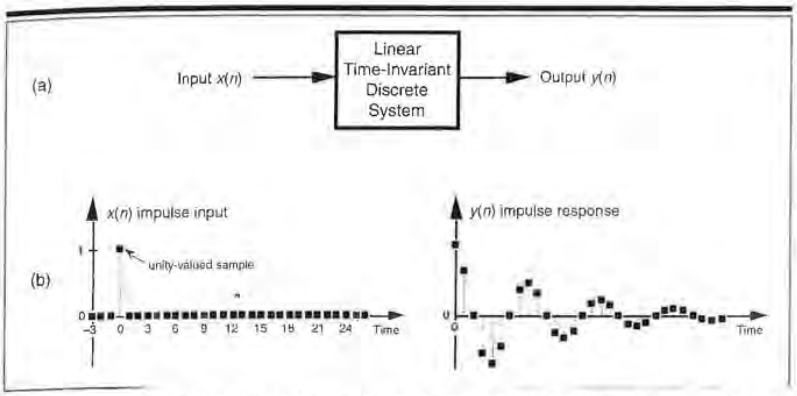


Figure 1-11 LTI system unit impulse response sequences: (a) system block diagram; (b) impulse input sequence $x(n)$ and impulse response output sequence $y(n)$.

response, we can find the system's *frequency response* by taking the Fourier transform in the form of a *discrete Fourier transform* of that impulse response[5].

Don't be alarmed if you're not exactly sure what is meant by convolution, frequency response, or the discrete Fourier transform. We'll introduce these subjects and define them slowly and carefully as we need them in later chapters. The point to keep in mind here is that LTI systems can be designed and analyzed using a number of straightforward and powerful analysis techniques. These techniques will become tools that we'll add to our signal processing toolboxes as we journey through the subject of digital signal processing.

References

[1] Karplus, W. J., and Soroka, W. W. *Analog Methods*, Second Edition, McGraw-Hill, New York, 1959, p. 117.

[2] Mikami, N., Kobayashi, M., and Yokoyama, Y. "A New DSP-Oriented Algorithm for Calculation of the Square Root Using a Nonlinear Digital Filter," *IEEE Trans. on Signal Processing*, Vol. 40, No. 7, July 1992.

[3] Heinen, P., and Neuvo, Y. "FIR-Median Hybrid Filters," *IEEE Trans. on Acoust. Speech, and Signal Processing*, Vol. ASSP-35, No. 6, June 1987.

[4] Oppenheim, A., Schafer, R., and Stockham, T. "Nonlinear Filtering of Multiplied and Convolved Signals," *Proc. IEEE*, Vol. 56, August 1968.

[5] Pickerd, John. "Impulse-Response Testing Lets a Single Test Do the Work of Thousands," *EDN*, April 27, 1995.

CHAPTER TWO

Periodic Sampling

Periodic sampling, the process of representing a continuous signal with a sequence of discrete data values, pervades the field of digital signal processing. In practice, sampling is performed by applying a continuous signal to an analog-to-digital (A/D) converter whose output is a series of digital values. Because sampling theory plays an important role in determining the accuracy and feasibility of any digital signal processing scheme, we need a solid appreciation for the often misunderstood effects of periodic sampling. With regard to sampling, the primary concern is just how fast must a given continuous signal be sampled in order to preserve its information content. We can sample a continuous signal at any sample rate we wish, and we'll get a series of discrete values—but the question is how well do these values represent the original signal? Let's learn the answer to that question and, in doing so, explore the various sampling techniques used in digital signal processing.

2.1 Aliasing: Signal Ambiguity in the Frequency Domain

There is a frequency-domain ambiguity associated with discrete-time signal samples that does not exist in the continuous signal world, and we can appreciate the effects of this uncertainty by understanding the sampled nature of discrete data. By way of example, suppose you were given the following sequence of values,

$$x(0) = 0$$
$$x(1) = 0.866$$
$$x(2) = 0.866$$
$$x(3) = 0$$
$$x(4) = -0.866$$
$$x(5) = -0.866$$
$$x(6) = 0\,,$$

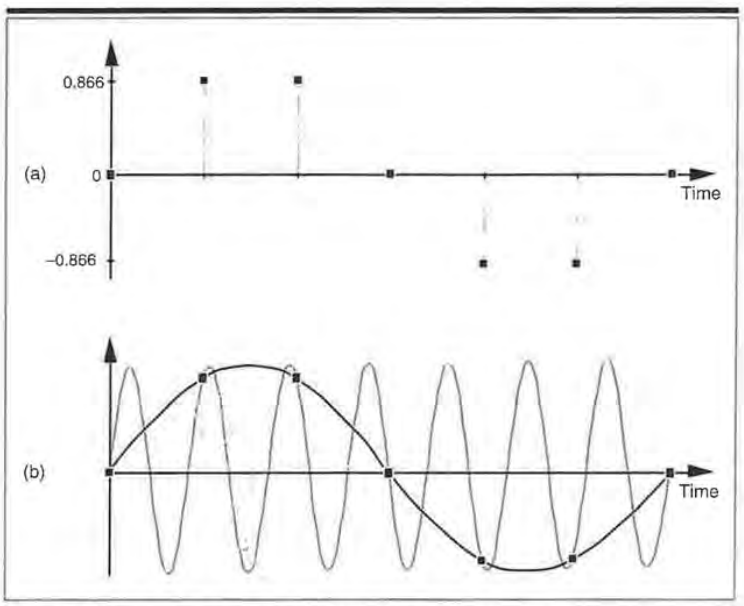


Figure 2-1 Frequency ambiguity: (a) discrete-time sequence of values; (b) two different sinewaves that pass through the points of the discrete sequence.

and were told that they represent instantaneous values of a time-domain sinewave taken at periodic intervals. Next, you were asked to draw that sinewave. You'd start by plotting the sequence of values shown by the dots in Figure 2-1(a). Next, you'd be likely to draw the sinewave, illustrated by the solid line in Figure 2-1(b), that passes through the points representing the original sequence.

Another person, however, might draw the sinewave shown by the shaded line in Figure 2-1(b). We see that the original sequence of values could, with equal validity, represent sampled values of both sinewaves. The key issue is that, if the data sequence represented periodic samples of a sinewave, we cannot unambiguously determine the frequency of the sinewave from those sample values alone.

Reviewing the mathematical origin of this frequency ambiguity enables us not only to deal with it, but to use it to our advantage. Let's derive an expression for this frequency-domain ambiguity and, then, look at a few specific examples. Consider the continuous time-domain sinusoidal signal defined as

$$x(t) = \sin(2\pi f_o t) \ . \tag{2-1}$$

This $x(t)$ signal is a garden variety sinewave whose frequency is f_o Hz. Now let's sample $x(t)$ at a rate of f_s samples/s, i.e., at regular periods of t_s seconds where $t_s = 1/f_s$. If we start sampling at time $t = 0$, we will obtain samples at times $0t_s$, $1t_s$, $2t_s$, and so on. So, from Eq. (2-1), the first n successive samples have the values

$$\begin{aligned}
&\text{0th sample:} && x(0) = \sin(2\pi f_o 0 t_s) \\
&\text{1st sample:} && x(1) = \sin(2\pi f_o 1 t_s) \\
&\text{2nd sample:} && x(2) = \sin(2\pi f_o 2 t_s) \\
&\quad \ldots && \quad \ldots \\
&\quad \ldots && \quad \ldots \\
&n\text{th sample:} && x(n) = \sin(2\pi f_o n t_s) \ .
\end{aligned} \tag{2-2}$$

Equation (2-2) defines the value of the nth sample of our $x(n)$ sequence to be equal to the original sinewave at the time instant nt_s. Because two values of a sinewave are identical if they're separated by an integer multiple of 2π radians, i.e., $\sin(\alpha) = \sin(\alpha + 2\pi m)$ where m is any integer, we can modify Eq. (2-2) as

$$x(n) = \sin(2\pi f_o n t_s) = \sin(2\pi f_o n t_s + 2\pi m) = \sin(2\pi(f_o + \frac{m}{nt_s})nt_s) \ . \tag{2-3}$$

If we let m be an integer multiple of n, $m = kn$, we can replace the m/n ratio in Eq. (2-3) with k so that

$$x(n) = \sin(2\pi(f_o + \frac{k}{t_s})nt_s) \ . \tag{2-4}$$

Because $f_s = 1/t_s$, we can equate the $x(n)$ sequences in Eqs. (2-2) and (2-4) as

$$x(n) = \sin(2\pi f_o n t_s) = \sin(2\pi(f_o + kf_s)nt_s) \ . \tag{2-5}$$

The f_o and $(f_o + kf_s)$ factors in Eq. (2-5) are therefore equal. The implication of Eq. (2-5) is critical. It means that an $x(n)$ sequence of digital sample values, representing a sinewave of f_o Hz, also exactly represents sinewaves at other frequencies, namely, $f_o + kf_s$. This is one of the most important relationships in the field of digital signal processing. It's the

thread with which all sampling schemes are woven. In words, Eq. (2-5) states that

> **When sampling at a rate of f_s samples/s, if k is any positive or negative integer, we cannot distinguish between the sampled values of a sinewave of f_o Hz and a sinewave of (f_o+kf_s) Hz.**

It's true. No sequence of values stored in a computer, for example, can unambiguously represent one and only one sinusoid without additional information. This fact applies equally to A/D-converter output samples as well as signal samples generated by computer software routines. The sampled nature of any sequence of discrete values makes that sequence also represent an infinite number of different sinusoids.

Equation (2-5) influences all digital signal processing schemes. It's the reason that, although we've only shown it for sinewaves, we'll see in Chapter 3 that the spectrum of any discrete series of sampled values contains periodic replications of the original continuous spectrum. The period between these replicated spectra in the frequency domain will always be f_s, and the spectral replications repeat all the way from *DC to daylight* in both directions of the frequency spectrum. That's because k in Eq. (2-5) can be any positive or negative integer. (In Chapters 5 and 6, we'll learn that Eq. (2-5) is the reason that all digital filter frequency responses are periodic in the frequency domain and is crucial to analyzing and designing a popular type of digital filter known as the infinite impulse response filter.)

To illustrate the effects of Eq. (2-5), let's build on Figure 2-1 and consider the sampling of a 7-kHz sinewave at a sample rate of 6 kHz. A new sample is determined every 1/6000 seconds, or once every 167 microseconds, and their values are shown as the dots in Figure 2-2(a).

Notice that the sample values would not change at all if, instead, we were sampling a 1-kHz sinewave. In this example f_o = 7 kHz, f_s = 6 kHz, and k = –1 in Eq. (2-5), such that f_o+kf_s = [7+(–1·6)] = 1 kHz. Our problem is that no processing scheme can determine if the sequence of sampled values, whose amplitudes are represented by the dots, came from a 7-kHz or a 1-kHz sinusoid. If these amplitude values are applied to a digital process that detects energy at 1 kHz, the detector output would indicate energy at 1 kHz. But we know that there is no 1-kHz tone there—our input is a spectrally pure 7-kHz tone. Equation (2-5) is causing a sinusoid, whose name is 7 kHz, to go by the *alias* of 1 kHz. Asking someone to determine which sinewave frequency accounts for the sample values in Figure 2-2(a) is like asking them "When I add two numbers I get a sum of four. What are the two numbers?" The answer is that there are an infinite number of number pairs that can add up to four.

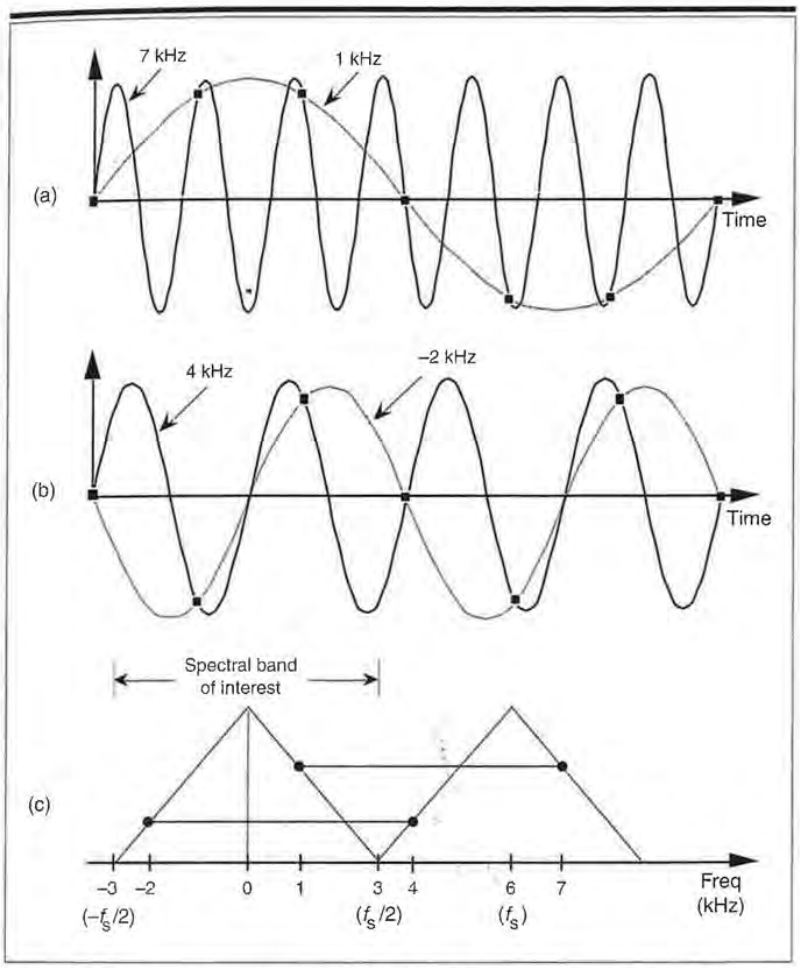


Figure 2-2 Frequency ambiguity effects of Eq. (2-5): (a) sampling a 7-kHz sinewave at a sample rate of 6 kHz; (b) sampling a 4-kHz sinewave at a sample rate of 6 kHz; (c) spectral relationships showing aliasing of the 7- and 4-kHz sinewaves.

Figure 2-2(b) shows another example of frequency ambiguity, that we'll call *aliasing*, where a 4-kHz sinewave could be mistaken for a –2-kHz sinewave. In Figure 2-2(b), f_0 = 4 kHz, f_s = 6 kHz, and k = –1 in Eq. (2-5), so that f_0+kf_s = [4+(–1 · 6)] = –2 kHz. Again, if we examine a sequence of numbers representing the dots in Figure 2-2(b), we could not determine if the sampled sinewave was a 4-kHz tone or a –2-kHz tone. (Although the

concept of negative frequencies might seem a bit strange, it provides a beautifully consistent methodology for predicting the spectral effects of sampling. Appendix C discusses negative frequencies and how they relate to real and complex signals.)

Now, if we restrict our spectral band of interest to the frequency range of $\pm f_s/2$ Hz, the previous two examples take on a special significance. The frequency $f_s/2$ is an important quantity in sampling theory and is referred to by different names in the literature, such as critical Nyquist, half Nyquist, and folding frequency. A graphical depiction of our two frequency aliasing examples is provided in Figure 2-2(c). We're interested in signal components that are aliased into the frequency band between $-f_s/2$ and $+f_s/2$. Notice in Figure 2-2(c) that, within the spectral band of interest (±3 kHz, because f_s = 6 kHz), there is energy at –2 kHz and +1 kHz, aliased from 4 kHz and 7 kHz, respectively. Note also that the vertical positions of the dots in Figure 2-2(c) have no amplitude significance but that their horizontal positions indicate which frequencies are related through aliasing.

A general illustration of aliasing is provided in the *shark's tooth* pattern in Figure 2-3(a). Note how the peaks of the pattern are located at integer multiples of f_s Hz. The pattern shows how signals residing at the intersection of a horizontal line and a sloped line will be aliased to all of the intersections of that horizontal line and all other lines with like slopes. For example, the pattern in Figure 2-3(b) shows that our sampling of a 7-kHz

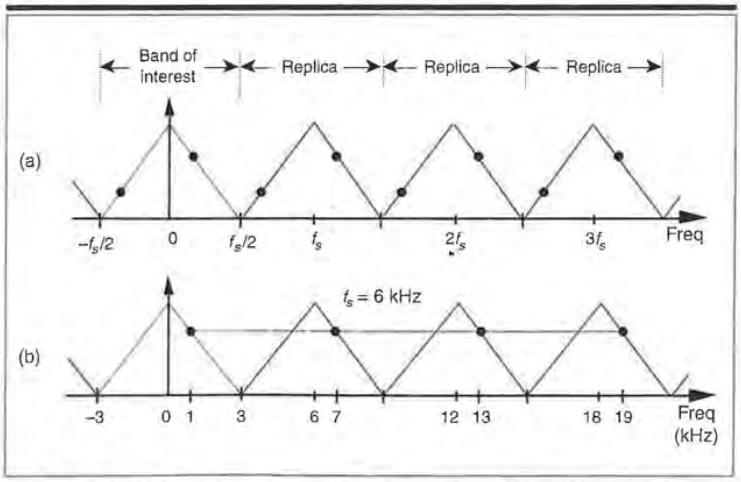


Figure 2-3 Shark's tooth pattern: (a) aliasing at multiples of the sampling frequency; (b) aliasing of the 7-kHz sinewave to 1 kHz, 13 kHz, and 19 kHz.

sinewave at a sample rate of 6 kHz will provide a discrete sequence of numbers whose spectrum ambiguously represents tones at 1 kHz, 7 kHz, 13 kHz, 19 kHz, etc. Let's pause for a moment and let these very important concepts soak in a bit. Again, discrete sequence representations of a continuous signal have unavoidable ambiguities in their frequency domains. These ambiguities must be taken into account in all practical digital signal processing algorithms.

OK, let's review the effects of sampling signals that are more interesting than just simple sinusoids.

2.2 Sampling Low-Pass Signals

Consider sampling a continuous real signal whose spectrum is shown in Figure 2-4(a). Notice that the spectrum is symmetrical about zero Hz, and the spectral amplitude is zero above $+B$ Hz and below $-B$ Hz, i.e., the signal is *band-limited*. (From a practical standpoint, the term *band-limited signal* merely implies that any signal energy outside the range of $\pm B$ Hz is below the sensitivity of our system.) Given that the signal is sampled at a rate of f_s samples/s, we can see the spectral replication effects of sampling in Figure 2-4(b), showing the original spectrum in addition to an infinite number of replications, whose period of replication is f_s Hz. (Although we stated in Section 1.1 that frequency-domain representations of discrete-time sequences are themselves discrete, the replicated spectra in Figure 2-4(b) are shown as continuous lines, instead of discrete dots, merely to keep the figure from looking too cluttered. We'll cover the full implications of discrete frequency spectra in Chapter 3.)

Let's step back a moment and understand Figure 2-4 for all it's worth. Figure 2-4(a) is the spectrum of a continuous signal, a signal that can only exist in one of two forms. Either it's a continuous signal that can be sampled, through A/D conversion, or it is merely an abstract concept such as a mathematical expression for a signal. It *cannot* be represented in a digital machine in its current band-limited form. Once the signal is represented by a sequence of discrete sample values, its spectrum takes the replicated form of Figure 2-4(b).

The replicated spectra are not just figments of the mathematics; they exist and have a profound effect on subsequent digital signal processing.[†] The replications may appear harmless, and it's natural to ask, "Why care about spectral replications? We're only interested in the frequency band

[†] Toward the end of Section 5.9, as an example of using the convolution theorem, another derivation of periodic sampling's replicated spectrums will be presented.

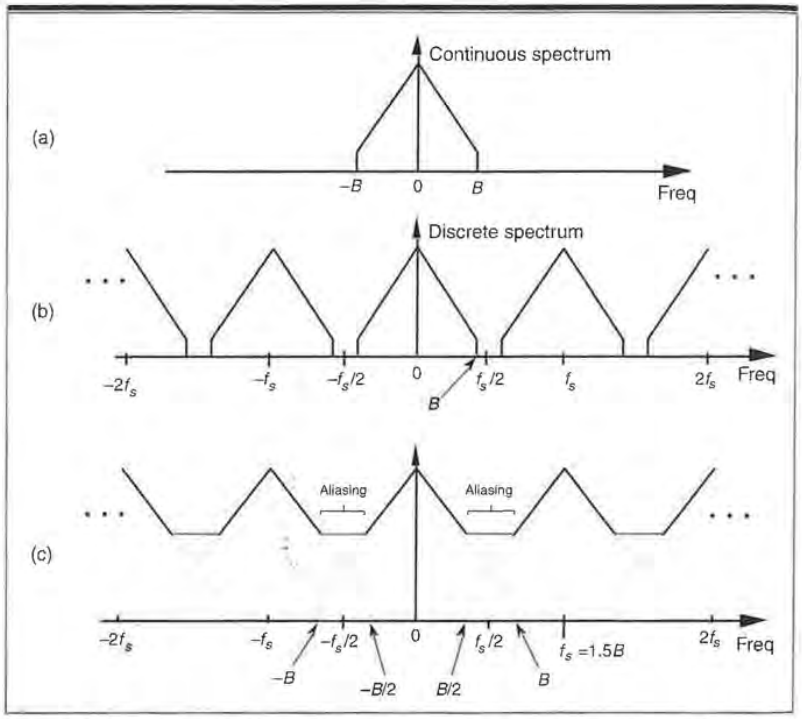


Figure 2-4 Spectral replications: (a) original continuous signal spectrum; (b) spectral replications of the sampled signal when $f_s/2 > B$; (c) frequency overlap and aliasing when the sampling rate is too low because $f_s/2 < B$.

within $\pm f_s/2$." Well, if we perform a frequency translation operation or induce a change in sampling rate through decimation or interpolation, the spectral replications will shift up or down right in the middle of the frequency range of interest $\pm f_s/2$ and could cause problems[1]. Let's see how we can control the locations of those spectral replications.

In practical A/D conversion schemes, f_s is always greater than $2B$ to separate spectral replications at the *folding frequencies* of $\pm f_s/2$. This very important relationship of $f_s \geq 2B$ is known as the Nyquist criterion. To illustrate why the term folding frequency is used, let's lower our sampling frequency to $f_s = 1.5B$ Hz. The spectral result of this *undersampling* is illustrated in Figure 2-4(c). The spectral replications are now overlapping the original baseband spectrum centered about zero Hz. Limiting our attention to the band $\pm f_s/2$ Hz, we see two very interesting effects. First, the lower edge and upper edge of the spectral replications centered

at $+f_s$ and $-f_s$ now lie in our band of interest. This situation is equivalent to the original spectrum folding to the left at $+f_s/2$ and folding to the right at $-f_s/2$. Portions of the spectral replications now combine with the original spectrum, and the result is aliasing errors. The discrete sampled values associated with the spectrum of Figure 2-4(c) no longer truly represent the original input signal. The spectral information in the bands of $-B$ to $-B/2$ and $B/2$ to B Hz has been corrupted. We show the amplitude of the aliased regions in Figure 2-4(c) as dashed lines because we don't really know what the amplitudes will be if aliasing occurs.

The second effect illustrated by Figure 2-4(c) is that the entire spectral content of the original continuous signal is now residing in the band of interest between $-f_s/2$ and $+f_s/2$. This key property was true in Figure 2-4(b) and will always be true, regardless of the original signal or the sample rate. This effect is particularly important when we're digitizing (A/D converting) continuous signals. It warns us that any signal energy located above $+B$ Hz and below $-B$ Hz in the original continuous spectrum of Figure 2-4(a) will always end up in the band of interest after sampling, regardless of the sample rate. For this reason, continuous (analog) *low-pass* filters are necessary in practice.

We illustrate this notion by showing a continuous signal of bandwidth B accompanied by noise energy in Figure 2-5(a). Sampling this composite continuous signal at a rate that's greater than $2B$ prevents replications of the signal of interest from overlapping each other, but all of the noise energy still ends up in the range between $-f_s/2$ and $+f_s/2$ of our discrete spectrum shown in Figure 2-5(b). This problem is solved in practice by

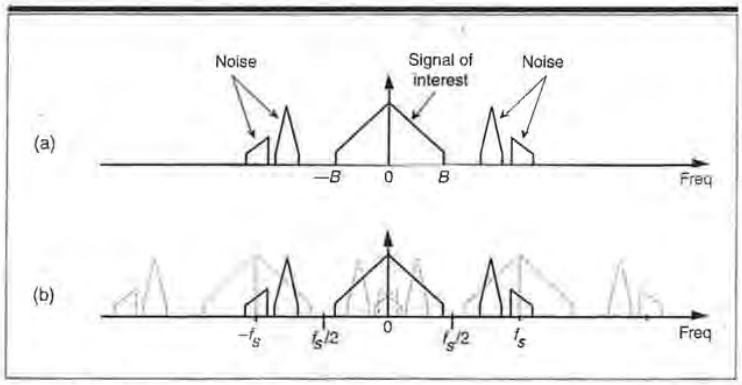


Figure 2-5 Spectral replications: (a) original continuous signal plus noise spectrum; (b) discrete spectrum with noise contaminating the signal of interest.

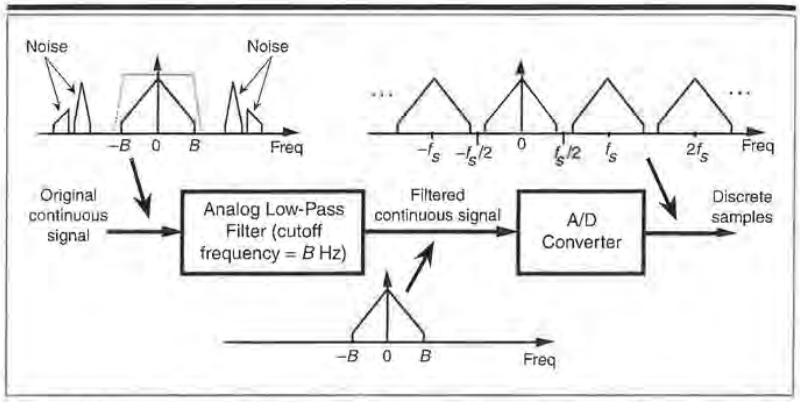


Figure 2-6 Low-pass analog filtering prior to sampling at a rate of f_s Hz.

using an analog low-pass *anti-aliasing* filter prior to A/D conversion to attenuate any unwanted signal energy above $+B$ and below $-B$ Hz as shown in Figure 2-6. An example low-pass filter response shape is shown as the dotted line superimposed on the original continuous signal spectrum in Figure 2-6. Notice how the output spectrum of the low-pass filter has been band-limited, and spectral aliasing is avoided at the output of the A/D converter.

This completes the discussion of simple low-pass sampling. Now let's go on to a more advanced sampling topic that's proven so useful in practice.

2.3 Sampling Bandpass Signals

Although satisfying the majority of sampling requirements, the sampling of low-pass signals, as in Figure 2-6, is not the only sampling scheme used in practice. We can use a technique known as *bandpass sampling* to sample a continuous bandpass signal that is centered about some frequency other than zero Hz. When a continuous input signal's bandwidth and center frequency permit us to do so, bandpass sampling not only reduces the speed requirement of A/D converters below that necessary with traditional low-pass sampling; it also reduces the amount of digital memory necessary to capture a given time interval of a continuous signal.

By way of example, consider sampling the band-limited signal shown in Figure 2-7(a) centered at $f_c = 20$ MHz, with a bandwidth $B = 5$ MHz. We use the term bandpass sampling for the process of sampling continuous signals whose center frequencies have been translated up from zero Hz. What we're calling bandpass sampling goes by various other names in the

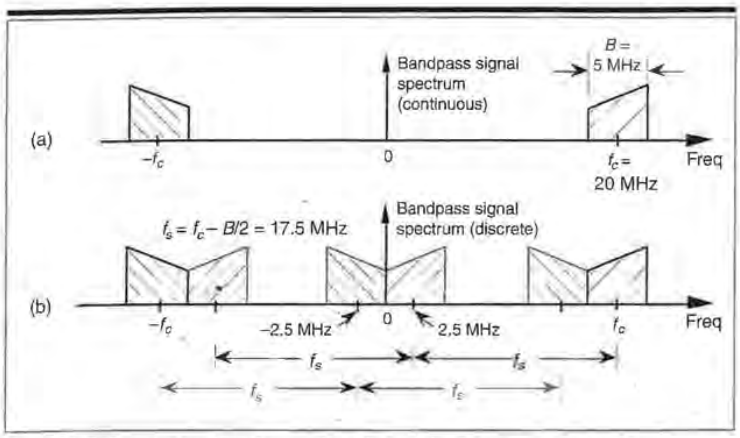


Figure 2-7 Bandpass signal sampling: (a) original continuous signal spectrum; (b) sampled signal spectrum replications when sample rate is 17.5 MHz.

literature, such as IF sampling, harmonic sampling[2], sub-Nyquist sampling, and undersampling[3]. In bandpass sampling, we're more concerned with a signal's bandwidth than its highest frequency component. Note that the negative frequency portion of the signal, centered at $-f_c$, is the mirror image of the positive frequency portion—as it must be for real signals. Our bandpass signal's highest frequency component is 22.5 MHz. Conforming to the Nyquist criterion (sampling at twice the highest frequency content of the signal) implies that the sampling frequency must be a minimum of 45 MHz. Consider the effect if the sample rate is 17.5 MHz shown in Figure 2-7(b). Note that the original spectral components remain located at $\pm f_c$, and spectral replications are located exactly at baseband, i.e., butting up against each other at zero Hz. Figure 2-7(b) shows that sampling at 45 MHz was unnecessary to avoid aliasing—instead we've used the spectral replicating effects of Eq. (2-5) to our advantage.

Bandpass sampling performs digitization and frequency translation in a single process, often called *sampling translation*. The processes of sampling and frequency translation are intimately bound together in the world of digital signal processing, and every sampling operation inherently results in spectral replications. The inquisitive reader may ask, "Can we sample at some still lower rate and avoid aliasing?" The answer is yes, but, to find out how, we have to grind through the derivation of an important bandpass sampling relationship. Our reward, however, will be worth the trouble because here's where bandpass sampling really gets interesting.

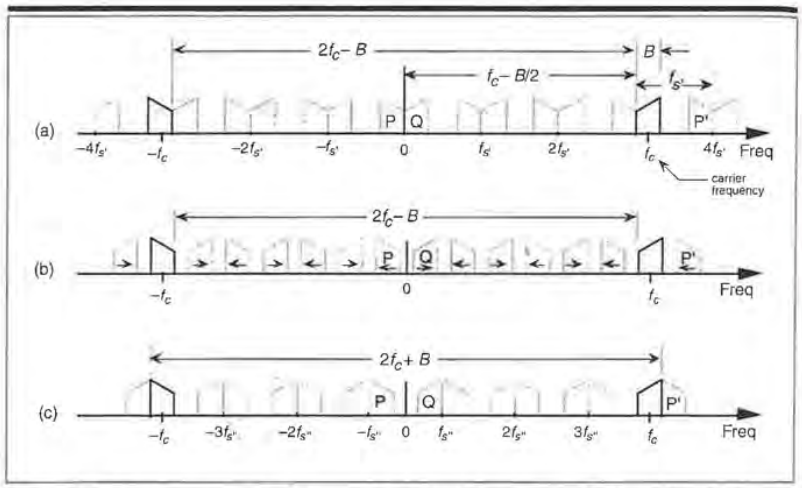


Figure 2-8 Bandpass sampling frequency limits: (a) sample rate $f_{s'} = (2f_c - B)/6$; (b) sample rate is less than $f_{s''}$; (c) minimum sample rate $f_{s''} < f_{s'}$.

Let's assume we have a continuous input bandpass signal of bandwidth B. Its *carrier frequency* is f_c Hz, i.e., the bandpass signal is centered at f_c Hz, and its sampled value spectrum is that shown in Figure 2-8(a). We can sample that continuous signal at a rate, say $f_{s'}$ Hz, so the spectral replications of the positive and negative bands, P and Q, just butt up against each other exactly at zero Hz. This situation, depicted in Figure 2-8(a), is reminiscent of Figure 2-7(b). With an arbitrary number of replications, say m, in the range of $2f_c - B$, we see that

$$mf_{s'} = 2f_c - B \quad \text{or} \quad f_{s'} = \frac{2f_c - B}{m} \,. \tag{2-6}$$

In Figure 2-8(a), $m = 6$ for illustrative purposes only. Of course m can be any positive integer so long as $f_{s'}$ is never less than $2B$. If the sample rate $f_{s'}$ is increased, the original spectra (bold) do not shift, but all the replications will shift. At zero Hz, the P band will shift to the right, and the Q band will shift to the left. These replications will overlap and aliasing occurs. Thus, from Eq. (2-6), for an arbitrary m, there is a frequency that the sample rate must not exceed, or

$$f_{s'} \le \frac{2f_c - B}{m} \quad \text{or} \quad \frac{2f_c - B}{m} \ge f_{s'} \,. \tag{2-7}$$

If we reduce the sample rate below the $f_{s'}$ value shown in Figure 2-8(a), the spacing between replications will decrease in the direction of the arrows in Figure 2-8(b). Again, the original spectra do not shift when the sample rate is changed. At some new sample rate $f_{s''}$, where $f_{s''} < f_{s'}$, the replication P' will just butt up against the positive original spectrum centered at f_c as shown in Figure 2-8(c). In this condition, we know that

$$(m+1)f_{s''} = 2f_c + B \quad \text{or} \quad f_{s''} = \frac{2f_c + B}{m+1} . \tag{2-8}$$

Should $f_{s''}$ be decreased in value, P' will shift further down in frequency and start to overlap with the positive original spectrum at f_c and aliasing occurs. Therefore, from Eq. (2-8) and for m+1, there is a frequency that the sample rate must always exceed, or

$$f_{s''} \geq \frac{2f_c + B}{m+1} . \tag{2-9}$$

We can now combine Eqs. (2-7) and (2-9) to say that f_s may be chosen anywhere in the range between $f_{s''}$ and $f_{s'}$ to avoid aliasing, or

$$\frac{2f_c - B}{m} \geq f_s \geq \frac{2f_c + B}{m+1} , \tag{2-10}$$

where m is an arbitrary, positive integer ensuring that $f_s \geq 2B$. (For this type of periodic sampling of real signals, known as real or first-order sampling, the Nyquist criterion $f_s \geq 2B$ must still be satisfied.)

To appreciate the important relationships in Eq. (2-10), let's return to our bandpass signal example, where Eq. (2-10) enables the generation of Table 2-1. This table tells us that our sample rate can be anywhere in the range of 22.5 to 35 MHz, anywhere in the range of 15 to 17.5 MHz, or anywhere in the range of 11.25 to 11.66 MHz. Any sample rate below 11.25 MHz is unacceptable because it will not satisfy Eq. (2-10) as well as $f_s \geq 2B$. The spectra resulting from several of the sampling rates from Table 2-1 are shown in Figure 2-9 for our bandpass signal example. Notice in Figure 2-9(f) that when f_s equals 7.5 MHz ($m = 5$), we have aliasing problems because neither the greater than relationships in Eq. (2-10) nor $f_s \geq 2B$ have been satisfied. The $m = 4$ condition is also unacceptable because $f_s \geq 2B$ is not satisfied. The last column in Table 2-1 gives the *optimum* sampling frequency for each acceptable m value. Optimum sampling frequency is defined here as that

Table 2-1 Equation (2-10) Applied to the Bandpass Signal Example

m	$(2f_c-B)/m$	$(2f_c+B)/(m+1)$	*Optimum Sampling Rate*
1	35.0 MHz	22.5 MHz	22.5 MHz
2	17.5 MHz	15.0 MHz	17.5 MHz
3	11.66 MHz	11.25 MHz	11.25 MHz
4	8.75 MHz	9.0 MHz	—
5	7.0 MHz	7.5 MHz	—

frequency where spectral replications do not butt up against each other except at zero Hz. For example, in the $m = 1$ range of permissible sampling frequencies, it is much easier to perform subsequent digital filtering or other processing on the signal samples whose spectrum is that of Figure 2-9(b), as opposed to the spectrum in Figure 2-9(a).

The reader may wonder, "Is the optimum sample rate always equal to the minimum permissible value for f_s using Eq. (2-10)?" The answer depends on the specific application—perhaps there are certain system constraints that must be considered. For example, in digital telephony, to simplify the follow-on processing, sample frequencies are chosen to be integer multiples of 8 kHz[4]. Another application-specific factor in choosing the optimum f_s is the shape of analog anti-aliasing filters[5]. Often, in practice, high-performance A/D converters have their hardware components *fine-tuned* during manufacture to ensure maximum linearity at high frequencies (>5 MHz). Their use at lower frequencies is not recommended.

An interesting way of illustrating the nature of Eq. (2-10) is to plot the minimum sampling rate, $(2f_c+B)/(m+1)$, for various values of m, as a function of R defined as

$$R = \frac{\text{highest signal frequency component}}{\text{bandwidth}} = \frac{f_c + B/2}{B} . \qquad (2\text{-}11)$$

If we normalize the minimum sample rate from Eq. (2-10) by dividing it by the bandwidth B, we get a curve whose axes are normalized to the bandwidth shown as the solid curve in Figure 2-10. This figure shows us the minimum normalized sample rate as a function of the normalized highest frequency component in the bandpass signal. Notice that, regardless of the value of R, the minimum sampling rate need never exceed $4B$ and approaches $2B$ as the carrier frequency increases. Surprisingly, the minimum acceptable sampling frequency actually decreases as the bandpass signal's carrier frequency increases. We can

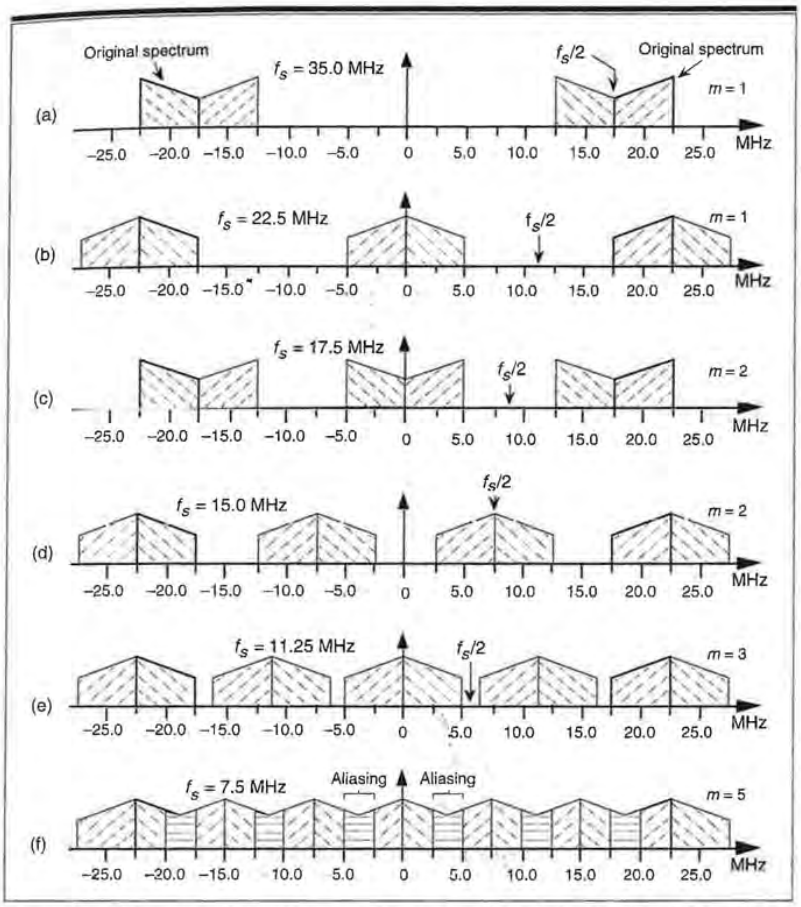


Figure 2-9 Various spectral replications from Table 2-1: (a) f_s = 35 MHz; (b) f_s = 22.5 MHz; (c) f_s = 17.5 MHz; (d) f_s = 15 MHz; (e) f_s = 11.25 MHz; (f) f_s = 7.5 MHz.

interpret Figure 2-10 by reconsidering our bandpass signal example from Figure 2-7 where $R = 22.5/5 = 4.5$. This R value is indicated by the dashed line in Figure 2-10 showing that $m = 3$ and f_s/B is 2.25. With $B = 5$ MHz, then, the minimum f_s = 11.25 MHz in agreement with Table 2-1. The leftmost line in Figure 2-10 shows the low-pass sampling case, where the sample rate f_s must be twice the signal's highest frequency component. So the normalized sample rate f_s/B is twice the highest frequency component over B or $2R$.

Figure 2-10 has been prominent in the literature, but its normal presentation enables the reader to jump to the false conclusion that any

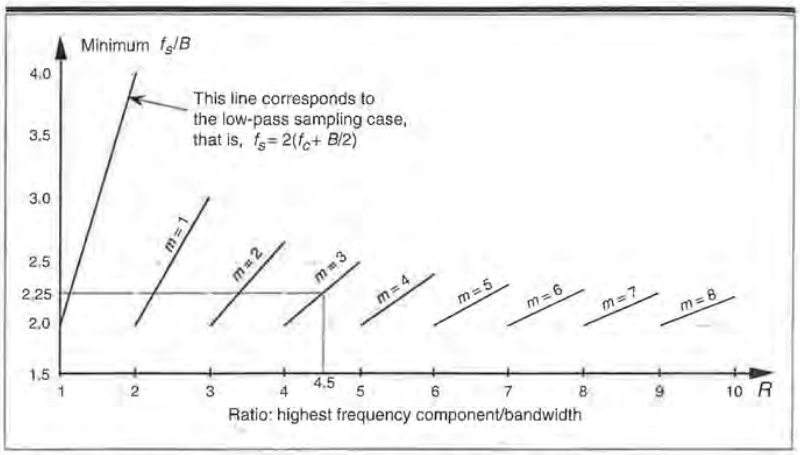


Figure 2-10 Minimum bandpass sampling rate from Eq. (2-10).

sample rate above the minimum shown in the figure will be an acceptable sample rate[6–12]. There's a clever way to avoid any misunderstanding[13]. If we plot the acceptable ranges of bandpass sample frequencies from Eq. (2-10) as a function of R we get the depiction shown in Figure 2-11. As we saw from Eq. (2-10), Table 2-1, and Figure 2-9, acceptable bandpass sample rates are a series of frequency ranges separated by unacceptable ranges of sample rate frequencies, that is, an acceptable bandpass sample frequency must be above the minimum shown in Figure 2-10, but cannot be just any frequency above that minimum. The shaded region in Figure 2-11 shows those normalized bandpass sample rates that will lead to spectral aliasing. Sample rates within the white regions of Figure 2-11 are acceptable. So, for bandpass sampling, we want our sample rate to be in the white wedged areas associated with some value of m from Eq. (2-10). Let's understand the significance of Figure 2-11 by again using our previous bandpass signal example from Figure 2-7.

Figure 2-12 shows our bandpass signal example R value (highest frequency component/bandwidth) of 4.5 as the dashed vertical line. Because that line intersects just three white wedged areas, we see that there are only three frequency regions of acceptable sample rates, and this agrees with our results from Table 2-1. The intersection of the $R = 4.5$ line and the borders of the white wedged areas are those sample rate frequencies listed in Table 2-1. So Figure 2-11 gives a depiction of bandpass sampling restrictions much more realistic than Figure 2-10.

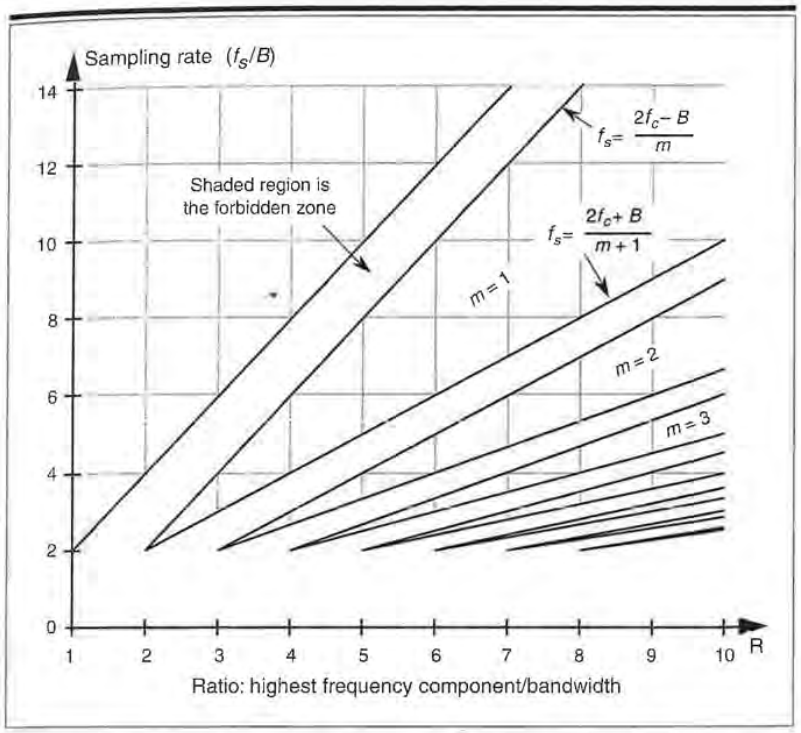


Figure 2-11 Regions of acceptable bandpass sampling rates from Eq. (2-10), normalized to the sample rate over the signal bandwidth (f_s/B).

Although Figures 2-11 and 2-12 indicate that we can use a sample rate that lies on the boundary between a white and shaded area, these sample rates should be avoided in practice. Nonideal analog bandpass filters, sample rate clock generator instabilities, and slight imperfections in available A/D converters make this *ideal* case impossible to achieve exactly. It's prudent to keep f_s somewhat separated from the boundaries. Consider the bandpass sampling scenario shown in Figure 2-13. With a typical (nonideal) analog bandpass filter, whose frequency response is indicated by the dashed line, it's prudent to consider the filter's bandwidth not as B, but as B_{gb} in our equations. That is, we create a guard band on either side of our filter so that there can be a small amount of aliasing in the discrete spectrum without distorting our desired signal, as shown at the bottom of Figure 2-13.

We can relate this idea of using guard bands to Figure 2-11 by looking more closely at one of the white wedges. As shown in Figure 2-14, we'd like

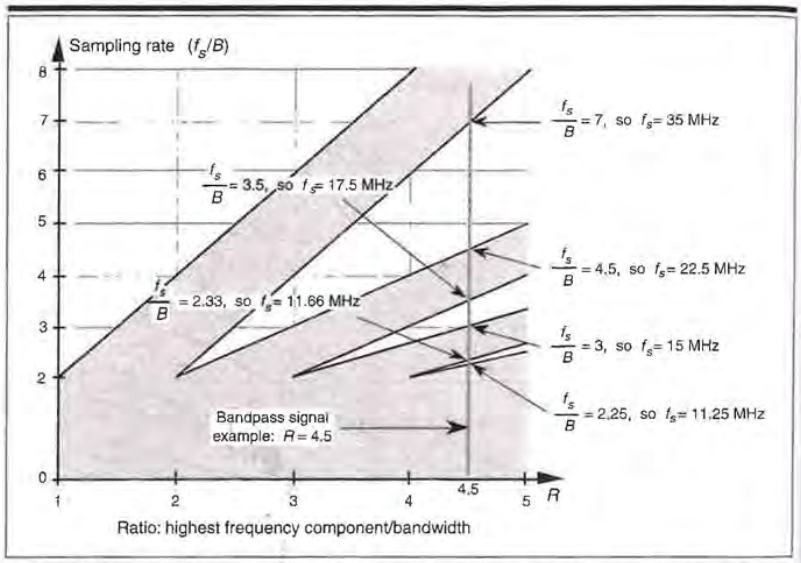


Figure 2-12 Acceptable sample rates for the bandpass signal example (B = 5 MHz) with a value of R = 4.5.

to set our sample rate as far down toward the vertex of the white area as we can—lower in the wedge means a lower sampling rate. However, the closer we operate to the boundary of a shaded area, the more narrow the guard band must be, requiring a sharper analog bandpass filter, as well as the tighter the tolerance we must impose on the stability and accuracy of our A/D clock generator. (Remember, operating on the boundary between a white and shaded area in Figure 2-11 causes spectral replications to butt up against each other.) So, to be safe, we operate at some intermediate point away from any shaded boundaries as shown in Figure 2-14. Further analysis of how guard band widths and A/D clock parameters relate to the geometry of Figure 2-14 is available in reference [13]. For this discussion, we'll just state that it's a good idea to ensure that our selected sample rate does not lie too close to the boundary between a white and shaded area in Figure 2-11.

There are a couple of ways to make sure we're not operating near a boundary. One way is to set the sample rate in the middle of a white wedge for a given value of R. We do this by taking the average between the maximum and minimum sample rate terms in Eq. (2-10) for a particular value of m, that is, to center the sample rate operating point within a wedge we use a sample rate of

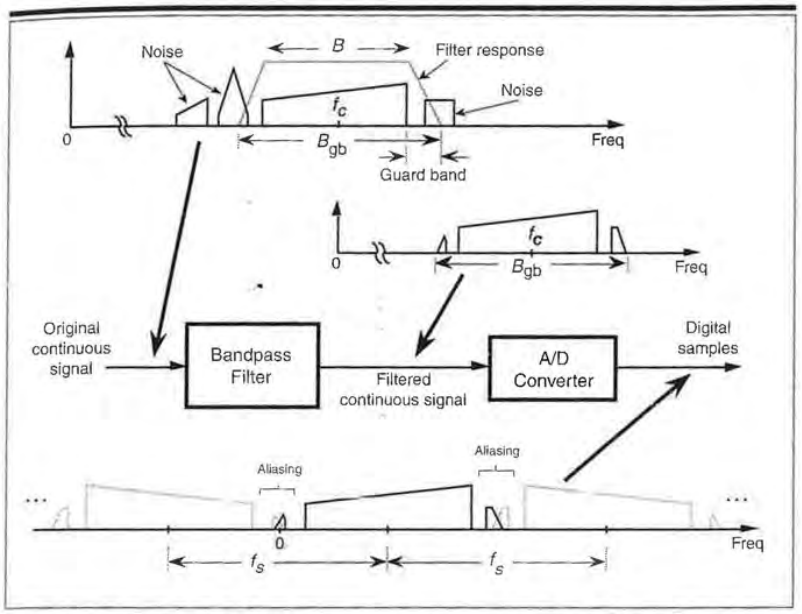


Figure 2-13 Bandpass sampling with aliasing occurring only in the filter guard bands.

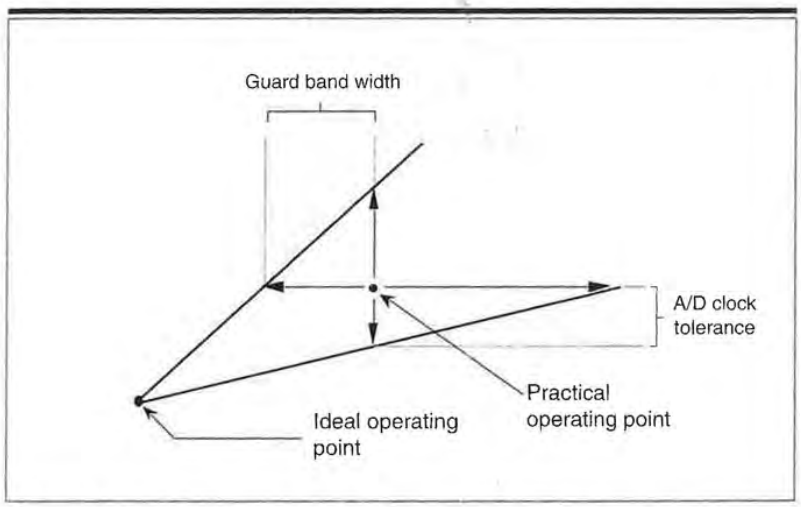


Figure 2-14 Typical operating point for f_s to compensate for nonideal hardware.

$$f_{s_{cntr}} = \frac{1}{2}\cdot\left[\frac{2f_c - B}{m} + \frac{2f_c + B}{m+1}\right] = \frac{f_c - B/2}{m} + \frac{f_c + B/2}{m+1} \quad . \tag{2-12}$$

Another way to avoid the boundaries of Figure 2-14 is to use the following expression to determine an intermediate f_{s_i} operating point:

$$f_{s_i} = \frac{4f_c}{m_{\text{odd}}} \quad , \tag{2-13}$$

where m_{odd} is an odd integer[14]. Of course the choice of m_{odd} must ensure that the Nyquist restriction of $f_{s_i} > 2B$ be satisfied. We show the results of Eqs. (2-12) and (2-13) for our bandpass signal example in Figure 2-15.

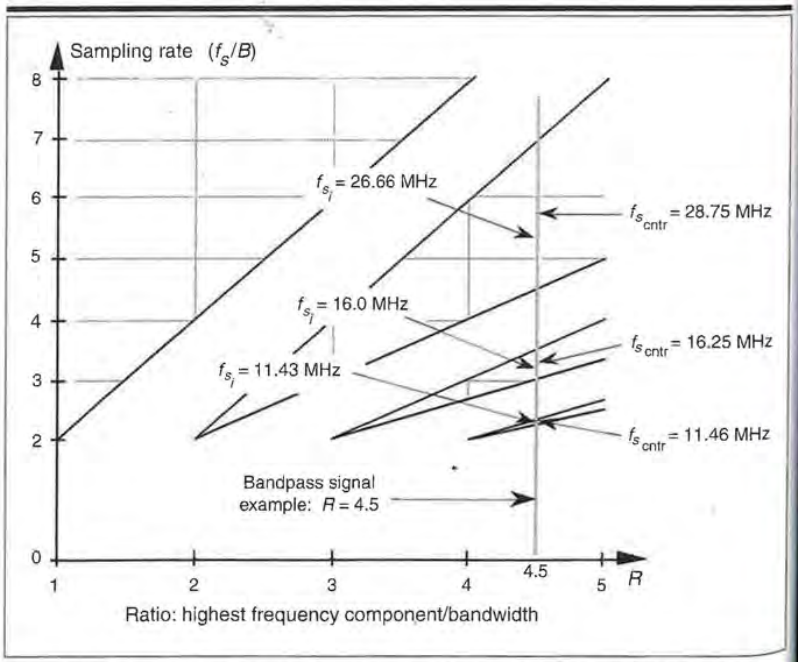


Figure 2-15 Intermediate f_{s_i} and $f_{s_{cntr}}$ operating points, from Eqs. (2-12) and (2-13), to avoid operating at the shaded boundaries for the bandpass signal example. B = 5 MHz and R = 4.5.

2.4 Spectral Inversion in Bandpass Sampling

Some of the permissible f_s values from Eq. (2-10) will, although avoiding aliasing problems, provide a sampled baseband spectrum (located near zero Hz) that is inverted from the original positive and negative spectral shapes, that is, the positive baseband will have the inverted shape of the negative half from the original spectrum. This spectral inversion happens whenever m, in Eq. (2-10), is an odd integer, as illustrated in Figures 2-9(b) and 2-9(e). When the original positive spectral bandpass components are symmetrical about the f_c frequency, spectral inversion presents no problem and any nonaliasing value for f_s from Eq. (2-10) may be chosen. However, if spectral inversion is something to be avoided, for example, when single sideband signals are being processed, the minimum applicable sample rate to avoid spectral inversion is defined by Eq. (2-10) with the restriction that m is the largest even integer such that $f_s \geq 2B$ is satisfied.† Using our definition of optimum sampling rate, the expression that provides the optimum noninverting sampling rates and avoids spectral replications butting up against each other, except at zero Hz, is

$$f_{s_o} = \frac{2f_c - B}{m_{\text{even}}}, \tag{2-14}$$

where $m_{\text{even}} = 2, 4, 6$, etc. For our bandpass signal example, Eq. (2-14) and $m = 2$ provide an optimum noninverting sample rate of $f_{s_o} = 17.5$ MHz, as shown in Figure 2-9(c). In this case, notice that the spectrum translated toward zero Hz has the same orientation as the original spectrum centered at 20 MHz.

Then again, if spectral inversion is unimportant for your application, we can determine the absolute minimum sampling rate without having to choose various values for m in Eq. (2-10) and creating a table like we did for Table 2-1. Considering Figure 2-16, the question is "How many replications of the positive and negative images of bandwidth B can we squeeze into the frequency range of $2f_c + B$ without overlap?" That number of replications is

$$R = \frac{\text{frequency span}}{\text{twice the bandwidth}} = \frac{2f_c + B}{2B} = \frac{f_c + B/2}{B} \tag{2-15}$$

† Single sideband signals are discussed in Section C.4 of Appendix C.

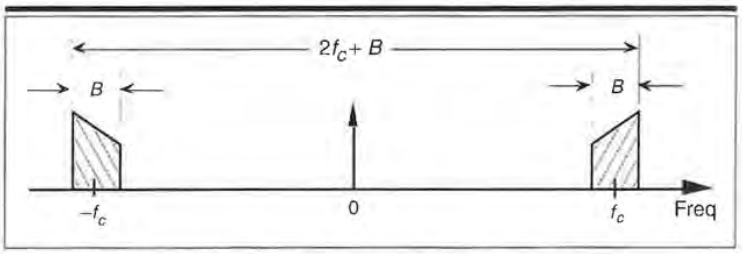


Figure 2-16 Frequency span of a continuous bandpass signal.

To avoid overlap, we have to make sure that the number of replications is an integer less than or equal to R in Eq. (2-15). So, we can define the integral number of replications to be R_{int} where

$$R_{int} \le R < R_{int} + 1 ,$$

or

$$R_{int} \le \frac{f_c + B/2}{B} < R_{int} + 1 . \tag{2-16}$$

With R_{int} replications in the frequency span of $2f_c + B$, then, the spectral repetition period, or minimum sample rate $f_{s_{min}}$, is

$$f_{s_{min}} = \frac{2f_c + B}{R_{int}} . \tag{2-17}$$

In our bandpass signal example, finding $f_{s_{min}}$ first requires the appropriate value for R_{int} in Eq. (2-13) as

$$R_{int} \le \frac{22.5}{5} < R_{int} + 1 ,$$

so $R_{int} = 4$. Then, from Eq. (2-17), $f_{s_{min}} = (40+5)/4 = 11.25$ MHz, which is the sample rate illustrated in Figures 2-9(e) and 2-12. So, we can use Eq. (2-17) and avoid using various values for m in Eq. (2-10) and having to create a table like Table 2-1. (Be careful though. Eq. (2-17) places our sampling rate at the boundary between a white and shaded area of Figure 2-12, and we have to consider the guard band strategy discussed above.) To recap the bandpass signal example, sampling at 11.25 MHz, from Eq. (2-17),

avoids aliasing and inverts the spectrum, while sampling at 17.5 MHz, from Eq. (2-14), avoids aliasing with no spectral inversion.

Now here's some good news. With a little additional digital processing, we can sample at 11.25 MHz, with its spectral inversion and easily reinvert the spectrum back to its original orientation. The discrete spectrum of any digital signal can be inverted by multiplying the signal's discrete-time samples by a sequence of alternating plus ones and minus ones (1, –1, 1, –1, etc.), indicated in the literature by the succinct expression $(-1)^n$. This scheme allows bandpass sampling at the lower rate of Eq. (2-17) while correcting for spectral inversion, thus avoiding the necessity of using the higher sample rates from Eq. (2-14). Although multiplying time samples by $(-1)^n$ is explored in detail in Section 10.1, all we need to remember at this point is the simple rule that multiplication of real signal samples by $(-1)^n$ is equivalent to multiplying by a cosine whose frequency is $f_s/2$. In the frequency domain, this multiplication flips the positive frequency band of interest, from zero to $+f_s/2$ Hz, about $f_s/4$ Hz, and flips the negative frequency band of interest, from $-f_s/2$ to zero Hz, about $-f_s/4$ Hz as shown in Figure 2-17. The $(-1)^n$ sequence is not only used for inverting the spectra of bandpass sampled sequences; it can be used to invert the spectra of low-pass sampled signals. Be aware, however, that, in the low-pass sampling case, any DC (zero Hz) component in the original continuous signal will be translated to both $+f_s/2$ and $-f_s/2$ after multiplication by $(-1)^n$.

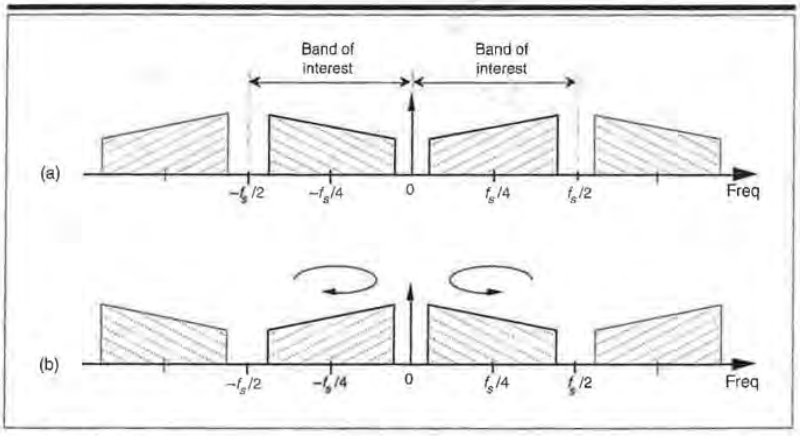


Figure 2-17 Spectral inversion through multiplication by $(-1)^n$: (a) original spectrum of a time-domain sequence; (b) new spectrum of the product of original time sequence and the $(-1)^n$ sequence.

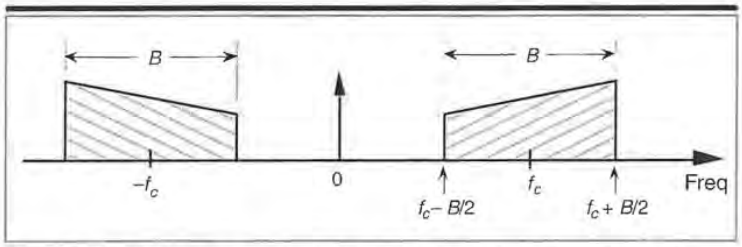


Figure 2-18 Continuous signal spectrum where bandpass sampling is not possible because $f_c - B/2 < B$.

Now that we have an understanding of bandpass sampling, we'd better remind ourselves of the situation where bandpass sampling is not possible. If a continuous bandpass signal's lowest frequency component is less than the bandwidth, we can't use bandpass sampling. This condition is shown in Figure 2-18 where $f_c - B/2 < B$. There's no way to squeeze any spectral replications between $f_c - B/2$ and zero Hz with bandpass sampling. In this case, we'd have to low-pass sample the signal in Figure 2-17 at a rate of at least twice the highest frequency component, or $f_s \geq 2(f_c + B/2)$.

We conclude this topic by consolidating in Table 2-2 what we need to know about bandpass sampling.

Table 2-2 Bandpass Sampling Relationships

Requirement	*Sample Rate Expression*	*Conditions*
Acceptable ranges of f_s for bandpass sampling: Eq. (2-10)	$\frac{2f_c - B}{m} \geq f_s \geq \frac{2f_c + B}{m+1}$	m = any positive integer so that $f_s \geq 2B$.
Sample rate in the middle of the acceptable sample rate bands: Eq. (2-12)	$f_{s_{cntr}} = \frac{f_c - B/2}{m} + \frac{f_c + B/2}{m+1}$	m = any positive integer so that $f_{s_{cntr}} \geq 2B$.
Sample rate at an intermediate point in the acceptable sample rate bands: Eq. (2-13)	$f_{s_i} = \frac{4f_c}{m_{odd}}$	m_{odd} = any positive odd integer so that $f_{s_i} \geq 2B$. (Spectral inversion occurs when m_{odd} = 3, 7, 11, etc.)
Optimum sample rate to avoid spectral inversion: Eq. (2-14)	$f_{s_o} = \frac{2f_c - B}{m_{even}}$	m_{even} = any even positive integer so that $f_{s_o} \geq 2B$.
Absolute minimum f_s to avoid aliasing: Eq. (2-17)	$f_{s_{min}} = \frac{2f_c + B}{R_{int}}$	where $R_{int} \leq \frac{f_c + B/2}{B} < R_{int} + 1$.

References

[1] Crochiere, R.E. and Rabiner, L.R. "Optimum FIR Digital Implementations for Decimation, Interpolation, and Narrow-band Filtering," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-23, No. 5, October 1975.

[2] Steyskal, H. "Digital Beamforming Antennas," *Microwave Journal*, January 1987.

[3] Hill, G. "The Benefits of Undersampling," *Electronic Design*, July 11, 1994.

[4] Yam, E., and Redman, M. "Development of a 60-channel FDM-TDM Transmultiplexer," *COMSAT Technical Review*, Vol. 13, No. 1, Spring 1983.

[5] Floyd, P., and Taylor, J. "Dual-Channel Space Quadrature-Interferometer System," *Microwave System Designer's Handbook*, Fifth Edition, Microwave Systems News, 1987.

[6] Lyons, R. G. "How Fast Must You Sample," *Test and Measurement World*, November 1988.

[7] Stremler, F. *Introduction to Communication Systems*, Chapter 3, Second Edition, Addison Wesley Publishing Co., Reading, Massachusetts, p. 125.

[8] Webb, R. C. "IF Signal Sampling Improves Receiver Detection Accuracy," *Microwaves & RF*, March 1989.

[9] Haykin, S. *Communications Systems*, Chapter 7, John Wiley and Sons, New York, 1983, p. 376.

[10] Feldman, C. B., and Bennett, W. R. "Bandwidth and Transmission Performance," *Bell System Tech. Journal*, Vol. 28, 1989, p. 490.

[11] Panter, P. F. *Modulation Noise, and Spectral Analysis*, McGraw-Hill, New York, 1965, p. 527.

[12] Shanmugam, K. S. *Digital and Analogue Communications Systems*, John Wiley and Sons, New York, 1979, p. 378.

[13] Vaughan, R., Scott, N. and White, D. "The Theory of Bandpass Sampling," *IEEE Trans. on Signal Processing*, Vol. 39, No. 9, September 1991, pp. 1973–1984.

[14] Xenakis B., and Evans, A. "Vehicle Locator Uses Spread Spectrum Technology," *RF Design*, October 1992.

CHAPTER THREE

The Discrete Fourier Transform

The discrete Fourier transform (DFT) is one of the two most common, and powerful, procedures encountered in the field of digital signal processing. (Digital filtering is the other.) The DFT enables us to analyze, manipulate, and synthesize signals in ways not possible with continuous (analog) signal processing. Even though it's now used in almost every field of engineering, we'll see applications for DFT continue to flourish as its utility becomes more widely understood. Because of this, a solid understanding of the DFT is mandatory for anyone working in the field of digital signal processing.

The DFT is a mathematical procedure used to determine the harmonic, or frequency, content of a discrete signal sequence. Although, for our purposes, a discrete signal sequence is a set of values obtained by periodic sampling of a continuous signal in the time domain, we'll find that the DFT is useful in analyzing any discrete sequence regardless of what that sequence actually represents. The DFT's origin, of course, is the continuous Fourier transform $X(f)$ defined as

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \ , \tag{3-1}$$

where $x(t)$ is some continuous time-domain signal.[†]

In the field of continuous signal processing, Eq. (3-1) is used to transform an expression of a continuous time-domain function $x(t)$ into a continuous frequency-domain function $X(f)$. Subsequent evaluation of the $X(f)$ expression enables us to determine the frequency content of any practical signal of interest and opens up a wide array of signal analysis and processing

[†] Fourier is pronounced 'for-yā. In engineering school, we called Eq. (3-1) the "four-year" transform because it took about four years to do one homework problem.

possibilities in the fields of engineering and physics. One could argue that the Fourier transform is the most dominant and widespread mathematical mechanism available for the analysis of physical systems. (A prominent quote from Lord Kelvin better states this sentiment: "Fourier's theorem is not only one of the most beautiful results of modern analysis, but it may be said to furnish an indispensable instrument in the treatment of nearly every recondite question in modern physics." By the way, the history of Fourier's original work in harmonic analysis, relating to the problem of heat conduction, is fascinating. References [1] and [2] are good places to start for those interested in the subject.)

With the advent of the digital computer, the efforts of early digital processing pioneers led to the development of the DFT defined as the discrete frequency-domain sequence $X(m)$, where

DFT equation (exponential form): $\rightarrow$

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N} \quad (3\text{-}2)$$

For our discussion of Eq. (3-2), $x(n)$ is a discrete sequence of time-domain sampled values of the continuous variable $x(t)$. The "e" in Eq. (3-2) is, of course, the base of natural logarithms and $j = \sqrt{-1}$.

3.1 Understanding the DFT Equation

Equation (3-2) has a tangled, almost unfriendly, look about it. Not to worry. After studying this chapter, Eq. (3-2) will become one of our most familiar and powerful tools in understanding digital signal processing. Let's get started by expressing Eq. (3-2) in a different way and examining it carefully. From *Euler's relationship* $e^{-j\phi} = \cos(\phi) - j\sin(\phi)$, Eq. (3-2) is equivalent to

DFT equation (rectangular form): $\rightarrow$

$$X(m) = \sum_{n=0}^{N-1} x(n)\left[\cos(2\pi nm/N) - j\sin(2\pi nm/N)\right] \quad (3\text{-}3)$$

We have separated the complex exponential of Eq. (3-2) into its real and imaginary components where

$X(m)$ = the mth DFT output component, i.e., $X(0)$, $X(1)$, $X(2)$, $X(3)$, etc.,

m = the index of the DFT output in the frequency domain, $m = 0, 1, 2, 3, \ldots, N-1$,

$x(n)$ = the sequence of input samples, $x(0)$, $x(1)$, $x(2)$, $x(3)$, etc.,

n = the time-domain index of the input samples, $n = 0, 1, 2, 3, \ldots, N-1$,

$j = \sqrt{-1}$, and

N = the number of samples of the input sequence and the number of frequency points in the DFT output.

Although it looks more complicated than Eq. (3-2), Eq. (3-3) turns out to be easier to understand. (If you're not too comfortable with it, don't let the $j = \sqrt{-1}$ concept bother you too much. It's merely a convenient abstraction that helps us compare the phase relationship between various sinusoidal components of a signal. Appendix C discusses the j operator in some detail.)† The indices for the input samples (n) and the DFT output samples (m) always go from 0 to $N-1$ in the standard DFT notation. This means that with N input time-domain sample values, the DFT determines the spectral content of the input at N equally spaced frequency points. The value N is an important parameter because it determines how many input samples are needed, the resolution of the frequency-domain results, and the amount of processing time necessary to calculate an N-point DFT.

It's useful to see the structure of Eq. (3-3) by eliminating the summation and writing out all the terms. For example, when $N = 4$, n and m both go from 0 to 3, and Eq. (3-3) becomes

$$X(m) = \sum_{n=0}^{3} x(n)[\cos(2\pi nm/4) - j\sin(2\pi nm/4)] \quad . \tag{3-4a}$$

Writing out all the terms for the first DFT output term corresponding to $m = 0$,

$$\begin{aligned} X(0) = {} & x(0)\cos(2\pi \cdot 0 \cdot 0/4) - jx(0)\sin(2\pi \cdot 0 \cdot 0/4) \\ & + x(1)\cos(2\pi \cdot 1 \cdot 0/4) - jx(1)\sin(2\pi \cdot 1 \cdot 0/4) \\ & + x(2)\cos(2\pi \cdot 2 \cdot 0/4) - jx(2)\sin(2\pi \cdot 2 \cdot 0/4) \\ & + x(3)\cos(2\pi \cdot 3 \cdot 0/4) - jx(3)\sin(2\pi \cdot 3 \cdot 0/4). \end{aligned} \tag{3-4b}$$

For the second DFT output term corresponding to $m = 1$, Eq. (3-4a) becomes

† Instead of the letter j, be aware that mathematicians often use the letter i to represent the $\sqrt{-1}$ operator.

$$\begin{aligned} X(1) = {} & x(0)\cos(2\pi \cdot 0 \cdot 1/4) - jx(0)\sin(2\pi \cdot 0 \cdot 1/4) \\ & + x(1)\cos(2\pi \cdot 1 \cdot 1/4) - jx(1)\sin(2\pi \cdot 1 \cdot 1/4) \\ & + x(2)\cos(2\pi \cdot 2 \cdot 1/4) - jx(2)\sin(2\pi \cdot 2 \cdot 1/4) \\ & + x(3)\cos(2\pi \cdot 3 \cdot 1/4) - jx(3)\sin(2\pi \cdot 3 \cdot 1/4). \end{aligned} \quad (3\text{-}4c)$$

For the third output term corresponding to $m = 2$, Eq. (3-4a) becomes

$$\begin{aligned} X(2) = {} & x(0)\cos(2\pi \cdot 0 \cdot 2/4) - jx(0)\sin(2\pi \cdot 0 \cdot 2/4) \\ & + x(1)\cos(2\pi \cdot 1 \cdot 2/4) - jx(1)\sin(2\pi \cdot 1 \cdot 2/4) \\ & + x(2)\cos(2\pi v2 \cdot 2/4) - jx(2)\sin(2\pi \cdot 2 \cdot 2/4) \\ & + x(3)\cos(2\pi \cdot 3 \cdot 2/4) - jx(3)\sin(2\pi \cdot 3 \cdot 2/4). \end{aligned} \quad (3\text{-}4d)$$

Finally, for the fourth and last output term corresponding to $m = 3$, Eq. (3-4a) becomes

$$\begin{aligned} X(3) = {} & x(0)\cos(2\pi \cdot 0 \cdot 3/4) - jx(0)\sin(2\pi \cdot 0 \cdot 3/4) \\ & + x(1)\cos(2\pi \cdot 1 \cdot 3/4) - jx(1)\sin(2\pi \cdot 1 \cdot 3/4) \\ & + x(2)\cos(2\pi \cdot 2 \cdot 3/4) - jx(2)\sin(2\pi \cdot 2 \cdot 3/4) \\ & + x(3)\cos(2\pi \cdot 3 \cdot 3/4) - jx(3)\sin(2\pi \cdot 3 \cdot 3/4). \end{aligned} \quad (3\text{-}4e)$$

The above multiplication symbol "·" in Eq. (3-4) is used merely to separate the factors in the sine and cosine terms. The pattern in Eq. (3-4b) through (3-4e) is apparent now, and we can certainly see why it's convenient to use the summation sign in Eq. (3-3). Each $X(m)$ DFT output term is the sum of the *point for point* product between an input sequence of signal values and a complex sinusoid of the form cos(ø) – *j*sin(ø). The exact frequencies of the different sinusoids depend on both the sampling rate f_s, at which the original signal was sampled, and the number of samples N. For example, if we are sampling a continuous signal at a rate of 500 samples/s and, then, perform a 16-point DFT on the sampled data, the fundamental frequency of the sinusoids is f_s/N = 500/16 or 31.25 Hz. The other $X(m)$ analysis frequencies are integral multiples of the fundamental frequency, i.e.,

$X(0)$ = 1st frequency term, with analysis frequency = 0 · 31.25 = 0 Hz,
$X(1)$ = 2nd frequency term, with analysis frequency = 1 · 31.25 = 31.25 Hz,
$X(2)$ = 3rd frequency term, with analysis frequency = 2 · 31.25 = 62.5 Hz,
$X(3)$ = 4th frequency term, with analysis frequency = 3 · 31.25 = 93.75 Hz,

. . .

. . .

$X(15)$ = 16th frequency term, with analysis frequency = 15 · 31.25 = 468.75 Hz.

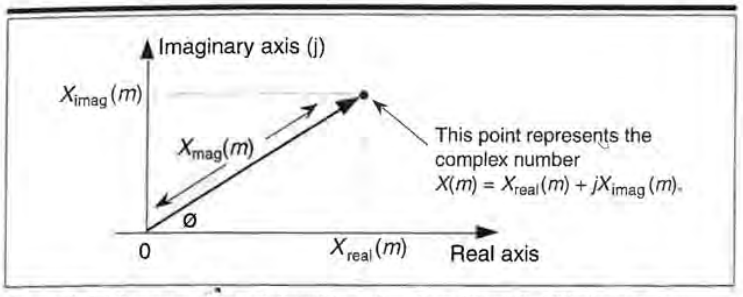


Figure 3-1 Trigonometric relationships of an individual DFT $X(m)$ complex output value.

The N separate DFT analysis frequencies are

$$f_{analysis}(m) = \frac{mf_s}{N} \quad . \tag{3-5}$$

So, in this example, the $X(0)$ DFT term tells us the magnitude of any 0-Hz ("DC") component contained in the input signal, the $X(1)$ term specifies the magnitude of any 31.25-Hz component in the input signal, and the $X(2)$ term indicates the magnitude of any 62.5-Hz component in the input signal, etc. Moreover, as we'll soon show by example, the DFT output terms also determine the phase relationship between the various analysis frequencies contained in an input signal.

Quite often we're interested in both the magnitude and the power (magnitude squared) contained in each $X(m)$ term, and the standard definitions for right triangles apply here as depicted in Figure 3-1.

If we represent an arbitrary DFT output value, $X(m)$, by its real and imaginary parts

$$X(m) = X_{real}(m) + jX_{imag}(m) = X_{mag}(m) \text{ at an angle of } X_{ø}(m) \ , \tag{3-6}$$

the magnitude of $X(m)$ is

$$X_{mag}(m) \ = \ |X(m)| \ = \sqrt{X_{real}(m)^2 + X_{imag}(m)^2} \quad . \tag{3-7}$$

By definition, the phase angle of $X(m)$, $X_{ø}(m)$, is

$$X_{ø}(m) = \tan^{-1}\left(\frac{X_{imag}(m)}{X_{real}(m)}\right) . \tag{3-8}$$

The power of $X(m)$, referred to as the power spectrum, is the magnitude squared where

$$X_{PS}(m) = X_{mag}(m)^2 = X_{real}(m)^2 + X_{imag}(m)^2 . \quad (3\text{-}9)$$

3.1.1 DFT Example 1

The above Eqs. (3-2) and (3-3) will become more meaningful by way of an example, so, let's go through a simple one step-by-step. Let's say we want to sample and perform an 8-point DFT on a continuous input signal containing components at 1 kHz and 2 kHz, expressed as

$$x_{in}(t) = \sin(2\pi \cdot 1000 \cdot t) + 0.5\sin(2\pi \cdot 2000 \cdot t + 3\pi/4) . \quad (3\text{-}10)$$

To make our example input signal $x_{in}(t)$ a little more interesting, we have the 2-kHz term shifted in phase by 135° ($3\pi/4$ radians) relative to the 1-kHz sinewave. With a sample rate of f_s, we sample the input every $1/f_s = t_s$ seconds. Because $N = 8$, we need 8 input sample values on which to perform the DFT. So the 8-element sequence $x(n)$ is equal to $x_{in}(t)$ sampled at the nt_s instants in time so that

$$x(n) = x_{in}(nt_s) = \sin(2\pi \cdot 1000 \cdot nt_s) + 0.5\sin(2\pi \cdot 2000 \cdot nt_s + 3\pi/4) . \quad (3\text{-}11)$$

If we choose to sample $x_{in}(t)$ at a rate of $f_s = 8000$ samples/s from Eq. (3-5), our DFT results will indicate what signal amplitude exists in $x(n)$ at the analysis frequencies of mf_s/N, or 0 kHz, 1 kHz, 2 kHz, . . ., 7 kHz. With $f_s = 8000$ samples/s, our eight $x(n)$ samples are

$$\begin{aligned} &x(0) = 0.3535, \quad x(1) = 0.3535, \\ &x(2) = 0.6464, \quad x(3) = 1.0607, \\ &x(4) = 0.3535, \quad x(5) = -1.0607, \\ &x(6) = -1.3535, \quad x(7) = -0.3535 . \end{aligned} \quad (3\text{-}11')$$

These $x(n)$ sample values are the dots plotted on the solid continuous $x_{in}(t)$ curve in Figure 3-2(a). (Note that the sum of the sinusoidal terms in Eq. (3-10), shown as the dashed curves in Figure 3-2(a), is equal to $x_{in}(t)$.)

Now we're ready to apply Eq. (3-3) to determine the DFT of our $x(n)$ input. We'll start with $m = 1$ because the $m = 0$ case leads to a special result that we'll discuss shortly. So, for $m = 1$, or the 1-kHz ($mf_s/N = 1 \cdot 8000/8$) DFT frequency term, Eq. (3-3) for this example becomes

$$X(1) = \sum_{n=0}^{7} x(n)\cos(2\pi n/8) - jx(n)\sin(2\pi n/8) \quad . \tag{3-12}$$

Next we multiply $x(n)$ by successive points on the cosine and sine curves of the first analysis frequency that have a single cycle over our 8 input samples. In our example, for $m = 1$, we'll sum the products of the $x(n)$ sequence with a 1-kHz cosine wave and a 1-kHz sinewave evaluated at the angular values of $2\pi n/8$. Those analysis sinusoids are shown as the dashed curves in Figure 3-2(b). Notice how the cosine and sinewaves have $m = 1$ complete cycles in our sample interval.

Substituting our $x(n)$ sample values into Eq. (3-12) and listing the cosine terms in the left column and the sine terms in the right column, we have

$$\begin{aligned}
X(1) &= 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) && \leftarrow \text{this is the } n = 0 \text{ term} \\
&+ 0.3535 \cdot 0.707 && - j(0.3535 \cdot 0.707) && \leftarrow \text{this is the } n = 1 \text{ term} \\
&+ 0.6464 \cdot 0.0 && - j(0.6464 \cdot 1.0) && \leftarrow \text{this is the } n = 2 \text{ term} \\
&+ 1.0607 \cdot -0.707 && - j(1.0607 \cdot 0.707) && \ldots \\
&+ 0.3535 \cdot -1.0 && - j(0.3535 \cdot 0.0) && \ldots \\
&- 1.0607 \cdot -0.707 && - j(-1.0607 \cdot -0.707) && \ldots \\
&- 1.3535 \cdot 0.0 && - j(-1.3535 \cdot -1.0) && \ldots \\
&- 0.3535 \cdot 0.707 && - j(-0.3535 \cdot -0.707) && \leftarrow \text{this is the } n = 7 \text{ term} \\
&= 0.3535 && + j0.0 \\
&\quad + 0.250 && - j0.250 \\
&\quad + 0.0 && - j0.6464 \\
&\quad - 0.750 && - j0.750 \\
&\quad + 0.3535 && - j0.0 \\
&\quad + 0.750 && - j0.750 \\
&\quad + 0.0 && - j1.3535 \\
&\quad - 0.250 && - j0.250 \\
&= 0.0 - j4.0 = 4 \angle -90°.
\end{aligned}$$

So we now see that the input $x(n)$ contains a signal component at a frequency of 1 kHz. Using Eq. (3-7), Eq. (3-8), and Eq. (3-9) for our $X(1)$ result, $X_{mag}(1) = 4$, $X_{PS}(1) = 16$, and $X(1)$'s phase angle relative to a 1-kHz cosine is $X_{\phi}(1) = -90°$.

For the $m = 2$ frequency term, we correlate $x(n)$ with a 2-kHz cosine wave and a 2-kHz sinewave. These waves are the dashed curves in Figure 3-2(c). Notice here that the cosine and sinewaves have $m = 2$ complete cycles in our

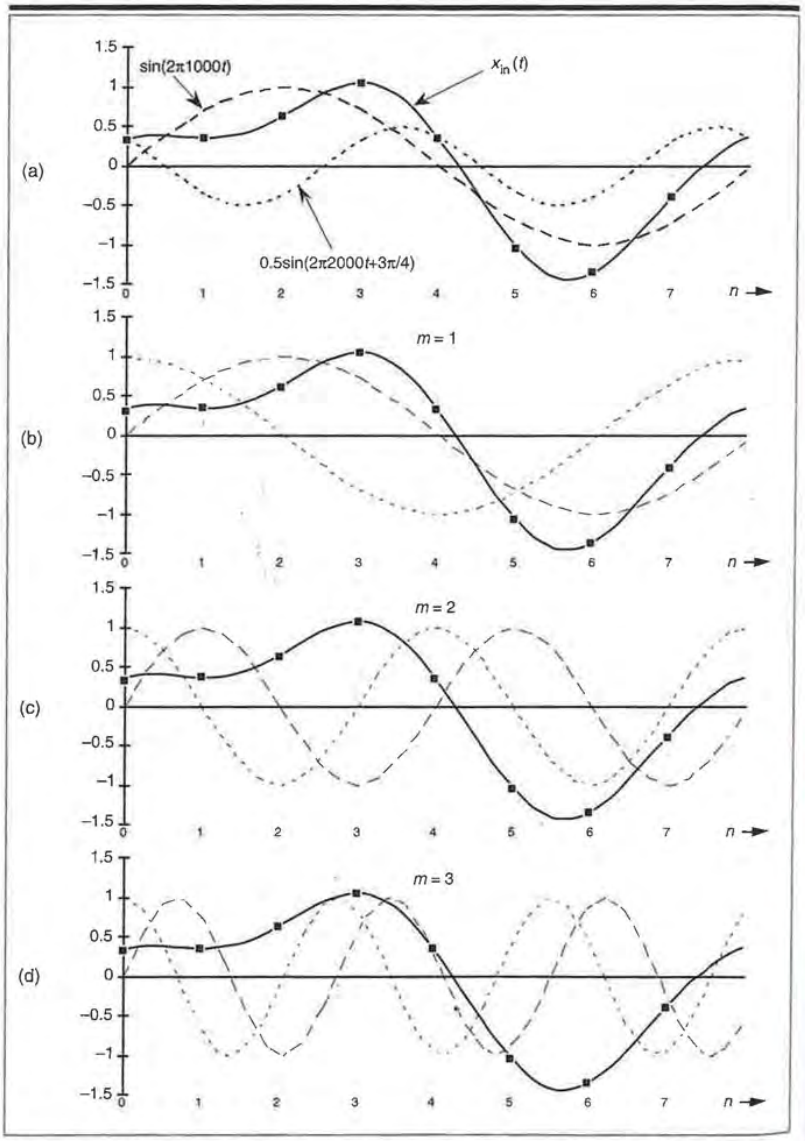


Figure 3-2 DFT Example 1: (a) the input signal; (b) the input signal and the $m = 1$ sinusoids; (c) the input signal and the $m = 2$ sinusoids; (d) the input signal and the $m = 3$ sinusoids.

sample interval in Figure 3-2(c). Substituting our $x(n)$ sample values in Eq. (3-3), for $m = 2$, gives

$$
\begin{aligned}
X(2) = {} & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.3535 \cdot 0.0 && - j(0.3535 \cdot 1.0) \\
& + 0.6464 \cdot -1.0 && - j(0.6464 \cdot 0.0) \\
& + 1.0607 \cdot 0.0 && - j(1.0607 \cdot -1.0) \\
& + 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& -1.0607 \cdot 0.0 && - j(-1.0607 \cdot 1.0) \\
& -1.3535 \cdot -1.0 && - j(-1.3535 \cdot 0.0) \\
& -0.3535 \cdot 0.0 && - j(-0.3535 \cdot -1.0)
\end{aligned}
$$

$$
\begin{aligned}
= {} & 0.3535 && + j0.0 \\
& + 0.0 && - j0.3535 \\
& + 0.6464 && - j0.0 \\
& - 0.0 && + j1.0607 \\
& + 0.3535 && - j0.0 \\
& + 0.0 && + j1.0607 \\
& + 1.3535 && - j0.0 \\
& - 0.0 && - j0.3535
\end{aligned}
$$

$$= 1.414 + j1.414 = 2 \angle 45°.$$

Here our input $x(n)$ contains a signal at a frequency of 2 kHz whose relative amplitude is 2, and whose phase angle relative to a 2-kHz cosine is 45°. For the $m = 3$ frequency term, we correlate $x(n)$ with a 3-kHz cosine wave and a 3-kHz sinewave. These waves are the dashed curves in Figure 3-2(d). Again, see how the cosine and sinewaves have $m = 3$ complete cycles in our sample interval in Figure 3-2(d). Substituting our $x(n)$ sample values in Eq. (3-3) for $m = 3$, gives

$$
\begin{aligned}
X(3) = {} & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.3535 \cdot -0.707 && - j(0.3535 \cdot 0.707) \\
& + 0.6464 \cdot 0.0 && - j(0.6464 \cdot -1.0) \\
& + 1.0607 \cdot 0.707 && - j(1.0607 \cdot 0.707) \\
& + 0.3535 \cdot -1.0 && - j(0.3535 \cdot 0.0) \\
& - 1.0607 \cdot 0.707 && - j(-1.0607 \cdot -0.707) \\
& - 1.3535 \cdot 0.0 && - j(-1.3535 \cdot 1.0) \\
& - 0.3535 \cdot -0.707 && - j(-0.3535 \cdot -0.707)
\end{aligned}
$$

$$
\begin{aligned}
= \; & 0.3535 && + j0.0 \\
& - 0.250 && - j0.250 \\
& + 0.0 && + j0.6464 \\
& + 0.750 && - j0.750 \\
& - 0.3535 && - j0.0 \\
& - 0.750 && - j0.750 \\
& + 0.0 && + j1.3535 \\
& + 0.250 && - j0.250
\end{aligned}
$$

$$= 0.0 - j0.0 = 0 \angle 0°.$$

Our DFT indicates that $x(n)$ contained no signal at a frequency of 3 kHz. Let's continue our DFT for the $m = 4$ frequency term using the sinusoids in Figure 3-3(a).

So Eq. (3-3) is

$$
\begin{aligned}
X(4) = \; & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.3535 \cdot -1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.6464 \cdot 1.0 && - j(0.6464 \cdot 0.0) \\
& + 1.0607 \cdot -1.0 && - j(1.0607 \cdot 0.0) \\
& + 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& - 1.0607 \cdot -1.0 && - j(-1.0607 \cdot 0.0) \\
& - 1.3535 \cdot 1.0 && - j(-1.3535 \cdot 0.0) \\
& - 0.3535 \cdot -1.0 && - j(-0.3535 \cdot 0.0)
\end{aligned}
$$

$$
\begin{aligned}
= \; & 0.3535 && - j0.0 \\
& - 0.3535 && - j0.0 \\
& + 0.6464 && - j0.0 \\
& - 1.0607 && - j0.0 \\
& + 0.3535 && - j0.0 \\
& + 1.0607 && - j0.0 \\
& - 1.3535 && - j0.0 \\
& + 0.3535 && - j0.0
\end{aligned}
$$

$$= 0.0 - j0.0 = 0 \angle 0°.$$

Our DFT for the $m = 5$ frequency term using the sinusoids in Figure 3-3(b) yields

$$
\begin{aligned}
X(5) = \; & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.3535 \cdot -0.707 && - j(0.3535 \cdot -0.707) \\
& + 0.6464 \cdot 0.0 && - j(0.6464 \cdot 1.0) \\
& + 1.0607 \cdot 0.707 && - j(1.0607 \cdot -0.707)
\end{aligned}
$$

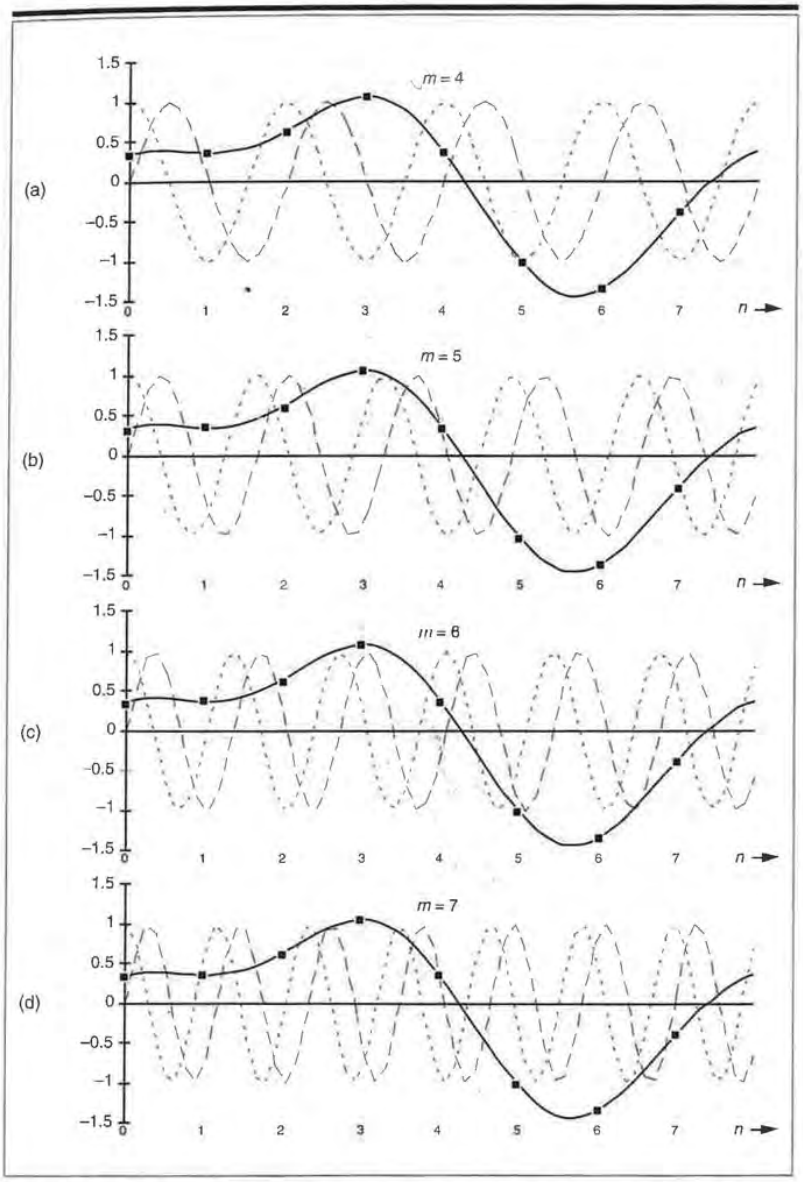


Figure 3-3 DFT Example 1: (a) the input signal and the $m = 4$ sinusoids; (b) the input and the $m = 5$ sinusoids; (c) the input and the $m = 6$ sinusoids; (d) the input and the $m = 7$ sinusoids.

$$
\begin{aligned}
&+ 0.3535 \cdot -1.0 && - j(0.3535 \cdot 0.0)\\
&- 1.0607 \cdot 0.707 && - j(-1.0607 \cdot 0.707)\\
&- 1.3535 \cdot 0.0 && - j(-1.3535 \cdot -1.0)\\
&- 0.3535 \cdot -0.707 && - j(-0.3535 \cdot 0.707)
\end{aligned}
$$

$$
\begin{aligned}
= \; & 0.3535 && - j0.0\\
& - 0.250 && + j0.250\\
& + 0.0 && - j0.6464\\
& + 0.750 && + j0.750\\
& - 0.3535 && - j0.0\\
& - 0.750 && + j0.750\\
& + 0.0 && - j1.3535\\
& + 0.250 && + j0.250
\end{aligned}
$$

$$= 0.0 - j.0 = 0 \angle 0°.$$

For the $m = 6$ frequency term using the sinusoids in Figure 3-3(c), Eq. (3-3) is

$$
\begin{aligned}
X(6) = & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0)\\
& + 0.3535 \cdot 0.0 && - j(0.3535 \cdot -1.0)\\
& + 0.6464 \cdot -1.0 && - j(0.6464 \cdot 0.0)\\
& + 1.0607 \cdot 0.0 && - j(1.0607 \cdot 1.0)\\
& + 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0)\\
& - 1.0607 \cdot 0.0 && - j(-1.0607 \cdot -1.0)\\
& - 1.3535 \cdot -1.0 && - j(-1.3535 \cdot 0.0)\\
& - 0.3535 \cdot 0.0 && - j(-0.3535 \cdot 1.0)
\end{aligned}
$$

$$
\begin{aligned}
= \; & 0.3535 && - j0.0\\
& + 0.0 && + j0.3535\\
& - 0.6464 && - j0.0\\
& + 0.0 && - j1.0607\\
& + 0.3535 && - j0.0\\
& + 0.0 && - j1.0607\\
& + 1.3535 && - j0.0\\
& + 0.0 && + j0.3535
\end{aligned}
$$

$$= 1.414 - j1.414 = 2 \angle -45°.$$

For the $m = 7$ frequency term using the sinusoids in Figure 3-3(d), Eq. (3-3) is

$$
\begin{aligned}
X(7) = & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0)\\
& + 0.3535 \cdot 0.707 && - j(0.3535 \cdot -0.707)\\
& + 0.6464 \cdot 0.0 && - j(0.6464 \cdot -1.0)\\
& + 1.0607 \cdot -0.707 && - j(1.0607 \cdot -0.707)
\end{aligned}
$$

$$
\begin{aligned}
&+ 0.3535 \cdot -1.0 && - j(0.3535 \cdot 0.0)\\
&- 1.0607 \cdot -0.707 && - j(-1.0607 \cdot 0.707)\\
&- 1.3535 \cdot 0.0 && - j(-1.3535 \cdot 1.0)\\
&- 0.3535 \cdot 0.707 && - j(-0.3535 \cdot 0.707)
\end{aligned}
$$

$$
\begin{aligned}
= \; & 0.3535 && + j0.0\\
& + 0.250 && + j0.250\\
& + 0.0 && + j0.6464\\
& - 0.750 && + j0.750\\
& - 0.3535 && - j0.0\\
& + 0.750 && + j0.750\\
& + 0.0 && + j1.3535\\
& - 0.250 && + j0.250
\end{aligned}
$$

$$= 0.0 + j4.0 = 4 \angle 90°.$$

If we plot the $X(m)$ output magnitudes as a function of frequency, we get the magnitude *spectrum* of the $x(n)$ input sequence, shown in Figure 3-4(a). The phase angles of the $X(m)$ output terms are depicted in Figure 3-4(b).

Hang in there, we're almost finished with our example. We've saved the calculation of the $m = 0$ frequency term to the end because it has a special significance. When $m = 0$, we correlate $x(n)$ with $\cos(0) - j\sin(0)$ so that Eq. (3-3) becomes

$$X(0) = \sum_{n=0}^{N-1} x(n)[\cos(0) - j\sin(0)]\,. \tag{3-13}$$

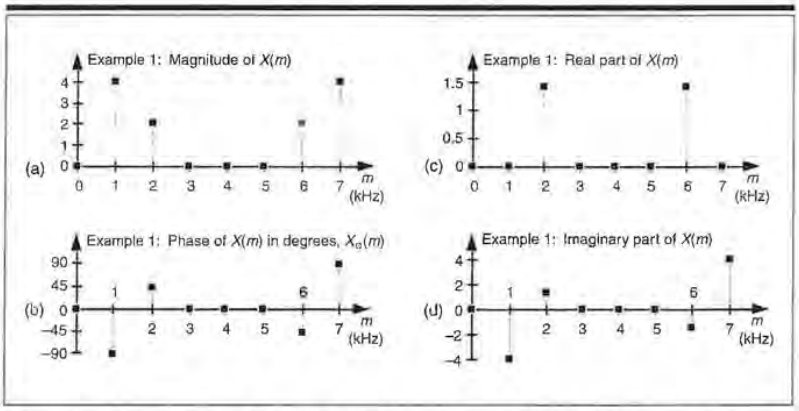


Figure 3-4 DFT results from Example 1: (a) magnitude of $X(m)$; (b) phase of $X(m)$; (c) real part of $X(m)$; (d) imaginary part of $X(m)$.

Because cos(0) = 1, and sin(0) = 0,

$$X(0) = \sum_{n=0}^{N-1} x(n) . \tag{3-13'}$$

We can see that Eq. (3-13') is the sum of the $x(n)$ samples. This sum is, of course, proportional to the average of $x(n)$. (Specifically, $X(0)$ is equal to N times $x(n)$'s average value). This makes sense because the $X(0)$ frequency term is the nontime-varying (DC) component of $x(n)$. If $X(0)$ were nonzero, this would tell us that the $x(n)$ sequence is riding on a DC bias and has some nonzero average value. For our specific example input from Eq. (3-10), the sum, however, is zero. The input sequence has no DC component, so we know that $X(0)$ will be zero. But let's not be lazy—we'll calculate $X(0)$ anyway just to be sure. Evaluating Eq. (3-3) or Eq. (3-13') for $m = 0$, we see that

$$\begin{aligned}
X(0) = {} & 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& + 0.6464 \cdot 1.0 && - j(0.6464 \cdot 0.0) \\
& + 1.0607 \cdot 1.0 && - j(1.0607 \cdot 0.0) \\
& + 0.3535 \cdot 1.0 && - j(0.3535 \cdot 0.0) \\
& - 1.0607 \cdot 1.0 && - j(-1.0607 \cdot 0.0) \\
& - 1.3535 \cdot 1.0 && - j(-1.3535 \cdot 0.0) \\
& - 0.3535 \cdot 1.0 && - j(-0.3535 \cdot 0.0)
\end{aligned}$$

$$\begin{aligned}
X(0) = {} & 0.3535 && - j0.0 \\
& + 0.3535 && - j0.0 \\
& + 0.6464 && - j0.0 \\
& + 1.0607 && - j0.0 \\
& + 0.3535 && - j0.0 \\
& - 1.0607 && - j0.0 \\
& - 1.3535 && - j0.0 \\
& - 0.3535 && - j0.0
\end{aligned}$$

$$= 0.0 - j0.0 = 0 \angle 0°.$$

So our $x(n)$ had no DC component, and, thus, its average value is zero. Notice that Figure 3-4 indicates that $x_{in}(t)$, from Eq. (3-10), has signal components at 1 kHz ($m = 1$) and 2 kHz ($m = 2$). Moreover, the 1-kHz tone has a magnitude twice that of the 2-kHz tone. The DFT results depicted in Figure 3-4 tell us exactly what's the spectral content of the signal defined by Eqs. (3-10) and (3-11).

The perceptive reader should be asking two questions at this point. First, what do those nonzero magnitude values at $m = 6$ and $m = 7$ in Figure 3-4(a) mean? Also, why do the magnitudes seem four times larger than we would expect? We'll answer those good questions shortly. The above 8-point DFT example, although admittedly simple, illustrates two very important characteristics of the DFT that we should never forget. First, any individual $X(m)$ output value is nothing more than the sum of the term-by-term products, a correlation, of an input signal sample sequence with a cosine and a sinewave whose frequencies are m complete cycles in the total sample interval of N samples. This is true no matter what the f_s sample rate is and no matter how large N is in an N-point DFT. The second important characteristic of the DFT of real input samples is the symmetry of the DFT output terms.

3.2 DFT Symmetry

Looking at Figure 3-4(a) again, there is an obvious symmetry in the DFT results. Although the standard DFT is designed to accept complex input sequences, most physical DFT inputs (such as digitized values of some continuous signal) are referred to as *real*, that is, real inputs have nonzero real sample values, and the imaginary sample values are assumed to be zero. When the input sequence $x(n)$ is real, as it will be for all of our examples, the complex DFT outputs for $m = 0$ to $m = (N/2) - 1$ are redundant with frequency output values for $m > (N/2)$. The mth DFT output will have the same magnitude as the $(N–m)$th DFT output. The phase angle of the DFT's mth output is the negative of the phase angle of the $(N–m)$th DFT output. So the mth and $(N–m)$th outputs are related by the following:

$$\begin{aligned} X(m) &= |X(m)| \text{ at } X_{\phi}(m) \text{ degrees} \\ &= |X(N-m)| \text{ at } -X_{\phi}(N-m) \text{ degrees}. \end{aligned} \tag{3-14}$$

We can state that when the DFT input sequence is real, $X(m)$ is the complex conjugate of $X(N–m)$, or

$$X(m) = X^*(N-m) \quad ,^{\dagger} \tag{3-14'}$$

where the superscript * symbol denotes conjugation.

† Using our notation, the complex conjugate of $x = a + jb$ is defined as $x^* = a - jb$; that is, we merely change the sign of the imaginary part of x. In an equivalent form, if $x = e^{j\phi}$, then $x^* = e^{-j\phi}$.

In our example above, notice in Figures 3-3(b) and 3-3(d), that $X(5)$, $X(6)$, and $X(7)$ are the complex conjugates of $X(3)$, $X(2)$, and $X(1)$, respectively. Like the DFT's magnitude symmetry, the real part of $X(m)$ has what is called *even symmetry*, as shown in Figure 3-4(c), while the DFT's imaginary part has *odd symmetry*, as shown in Figure 3-4(d). This relationship is what is meant when the DFT is called conjugate symmetric in the literature. It means that, if we perform an N-point DFT on a real input sequence, we'll get N separate complex DFT output terms, but only the first $N/2$ terms are independent. So to obtain the DFT of $x(n)$, we need only compute the first $N/2$ values of $X(m)$ where $0 \le m \le (N/2) - 1$; the $X(N/2)$ to $X(N-1)$ DFT output terms provide no additional information about the spectrum of the real sequence $x(n)$.

Although Eqs. (3-2) and (3-3) are equivalent, expressing the DFT in the exponential form of Eq. (3-2) has a terrific advantage over the form of Eq. (3-3). Not only does Eq. (3-2) save pen and paper, Eq. (3-2)'s exponentials are so much easier to manipulate when we're trying to analyze DFT relationships. Using Eq. (3-2), products of terms become the addition of exponents and, with due respect to Euler, we don't have all those trigonometric relationships to memorize. Let's demonstrate this by proving Eq. (3-14) to show the symmetry of the DFT of real input sequences. Substituting $N-m$ for m in Eq. (3-2), we get the expression for the $(N-m)$th component of the DFT:

$$X(N-m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi n(N-m)/N} = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nN/N} e^{-j2\pi n(-m)/N}$$

$$= \sum_{n=0}^{N-1} x(n) e^{-j2\pi n} e^{j2\pi nm/N}. \tag{3-15}$$

Because $e^{-j2\pi n} = \cos(2\pi n) - j\sin(2\pi n) = 1$ for all integer values of n,

$$X(N-m) = \sum_{n=0}^{N-1} x(n) e^{j2\pi nm/N}. \tag{3-15'}$$

We see that $X(N-m)$ in Eq. (3-15') is merely $X(m)$ in Eq. (3-2) with the sign reversed on $X(m)$'s exponent—and that's the definition of the complex conjugate. This is illustrated by the DFT output phase-angle plot in Figure 3-4(b) for our DFT Example 1. Try deriving Eq. (3-15') using the

cosines and sines of Eq. (3-3), and you'll see why the exponential form of the DFT is so convenient for analytical purposes.

There's an additional symmetry property of the DFT that deserves mention at this point. In practice, we're occasionally required to determine the DFT of real input functions where the input index n is defined over both positive and negative values. If that real input function is even, then $X(m)$ is always real and even; that is, if the real $x(n) = x(-n)$, then, $X_{real}(m)$ is in general nonzero and $X_{imag}(m)$ is zero. Conversely, if the real input function is odd, $x(n) = -x(-n)$, then $X_{real}(m)$ is always zero and $X_{imag}(m)$ is, in general, nonzero. This characteristic of input function symmetry is a property that the DFT shares with the continuous Fourier transform, and (don't worry) we'll cover specific examples of it later in Section 3.13 and in Chapter 5.

3.3 DFT Linearity

The DFT has a very important property known as *linearity*. This property states that the DFT of the sum of two signals is equal to the sum of the transforms of each signal; that is, if an input sequence $x_1(n)$ has a DFT $X_1(m)$ and another input sequence $x_2(n)$ has a DFT $X_2(m)$, then the DFT of the sum of these sequences $x_{sum}(n) = x_1(n) + x_2(n)$ is

$$X_{sum}(m) = X_1(m) + X_2(m) . \tag{3-16}$$

This is certainly easy enough to prove. If we plug $x_{sum}(n)$ into Eq. (3-2) to get $X_{sum}(m)$, then

$$X_{sum}(m) = \sum_{n=0}^{N-1} [x_1(n) + x_2(n)] e^{-j2\pi nm/N}$$

$$= \sum_{n=0}^{N-1} x_1(n) e^{-j2\pi nm/N} + \sum_{n=0}^{N-1} x_2(n) e^{-j2\pi nm/N} = X_1(m) + X_2(m) .$$

Without this property of linearity, the DFT would be useless as an analytical tool because we could transform only those input signals that contain a single sinewave. The real-world signals that we want to analyze are much more complicated than a single sinewave.

3.4 DFT Magnitudes

The DFT Example 1 results of $|X(1)| = 4$ and $|X(2)| = 2$ may puzzle the reader because our input $x(n)$ signal, from Eq. (3-11), had peak amplitudes of 1.0 and 0.5, respectively. There's an important point to keep in mind regarding DFTs defined by Eq. (3-2). When a real input signal contains a sinewave component of peak amplitude A_o with an integral number of cycles over N input samples, the output magnitude of the DFT for that particular sinewave is M_r where

$$M_r = A_o N/2 . \tag{3-17}$$

If the DFT input is a complex sinusoid of magnitude A_o (i.e., $A_o e^{j2\pi ft}$) with an integral number of cycles over N samples, the output magnitude of the DFT is M_c where

$$M_c = A_o N . \tag{3-17'}$$

As stated in relation to Eq. (3-13'), if the DFT input was riding on a DC value equal to D_o, the magnitude of the DFT's $X(0)$ output will be $D_o N$.

Looking at the real input case for the 1000 Hz component of Eq. (3-11), $A_o = 1$ and $N = 8$, so that $M_{real} = 1 \cdot 8/2 = 4$, as our example shows. Equation (3-17) may not be so important when we're using software or floating-point hardware to perform DFTs, but if we're implementing the DFT with fixed-point hardware, we have to be aware that the output can be as large as $N/2$ times the peak value of the input. This means that, for real inputs, hardware memory registers must be able to hold values as large as $N/2$ times the maximum amplitude of the input sample values. We discuss DFT output magnitudes in further detail later in this chapter. The DFT magnitude expressions in Eqs. (3-17) and (3-17') are why we occasionally see the DFT defined in the literature as

$$X'(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N} . \tag{3-18}$$

The $1/N$ scale factor in Eq. (3-18) makes the amplitudes of $X'(m)$ equal to half the time-domain input sinusoid's peak value at the expense of the additional division by N computation. Thus, hardware or software implementations of the DFT typically use Eq. (3-2) as opposed to Eq. (3-18). Of course, there are always exceptions. There are commercial software packages using

$$X''(m) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j2\pi nm/N},$$

and

$$x(n) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X''(m) e^{j2\pi nm/N} \qquad (3\text{-}18')$$

for the forward and inverse DFTs. (In Section 3.7, we discuss the meaning and significance of the inverse DFT.) The $1/\sqrt{N}$ scale factors in Eqs. (3-18') seem a little strange, but they're used so that there's no scale change when transforming in either direction. When analyzing signal spectra in practice, we're normally more interested in the relative magnitudes rather than absolute magnitudes of the individual DFT outputs, so scaling factors aren't usually that important to us.

3.5 DFT Frequency Axis

The frequency axis m of the DFT result in Figure 3-4 deserves our attention once again. Suppose we hadn't previously seen our DFT Example 1, were given the eight input sample values, from Eq. (3-11'), and asked to perform an 8-point DFT on them. We'd grind through Eq. (3-2) and get the $X(m)$ values shown in Figure 3-4. Next we ask, "What's the frequency of the highest magnitude component in $X(m)$ in Hz?" The answer is not "1." The answer depends on the original sample rate f_s. Without prior knowledge, we have no idea over what time interval the samples were taken, so we don't know the absolute scale of the $X(m)$ frequency axis. The correct answer to the question is to take f_s and plug it into Eq. (3-5) with $m = 1$. Thus, if $f_s = 8000$ samples/s, then, the frequency associated with the largest DFT magnitude term is

$$f_{\text{analysis}}(m) = \frac{mf_s}{N} = f_{\text{analysis}}(1) = \frac{1 \cdot 8000}{8} = 1000 \text{ Hz}.$$

If we said the sample rate f_s was 75 samples/s, we'd know, from Eq. (3-5), that the frequency associated with the largest magnitude term is now

$$f_{\text{analysis}}(1) - \frac{1 \cdot 75}{8} = 9.375 \text{ Hz}.$$

OK, enough of this—just remember that the DFT's frequency spacing (resolution) is f_s/N.

To recap what we've learned so far:

- each DFT output term is the sum of the term-by-term products of an input time-domain sequence with sequences representing a sine and a cosine wave,
- for real inputs, an N-point DFT's output provides only $N/2$ independent terms,
- the DFT is a linear operation,
- the magnitude of the DFT results are directly proportional to N, and
- the DFT's frequency resolution is f_s/N.

It's also important to realize, from Eq. (3-5), that $X(N/2)$, when $m = N/2$, corresponds to half the sample rate, i.e. the folding (Nyquist) frequency $f_s/2$.

3.6 DFT Shifting Theorem

There's an important property of the DFT known as the shifting theorem. It states that a shift in time of a periodic $x(n)$ input sequence manifests itself as a constant phase shift in the angles associated with the DFT results. (We won't derive the shifting theorem equation here because its derivation is included in just about every digital signal processing textbook in print.) If we decide to sample $x(n)$ starting at n equals some integer k, as opposed to $n = 0$, the DFT of those time-shifted sample values is $X_{\text{shifted}}(m)$ where

$$X_{\text{shifted}}(m) = e^{j2\pi km/N}\, X(m)\ . \tag{3-19}$$

Equation (3-19) tells us that, if the point where we start sampling $x(n)$ is shifted to the right by k samples, the DFT output spectrum of $X_{\text{shifted}}(m)$ is $X(m)$ with each of $X(m)$'s complex terms multiplied by the linear phase shift $e^{j2\pi km/N}$, which is merely a phase shift of $2\pi km/N$ radians or $360km/N$ degrees. Conversely, if the point where we start sampling $x(n)$ is shifted to the left by k samples, the spectrum of $X_{\text{shifted}}(m)$ is $X(m)$ multiplied by $e^{-j2\pi km/N}$. Let's illustrate Eq. (3-19) with an example.

3.6.1 DFT Example 2

Suppose we sampled our DFT Example 1 input sequence later in time by $k = 3$ samples. Figure 3-5 shows the original input time function,

$$x_{\text{in}}(t) = \sin(2\pi 1000t) + 0.5\sin(2\pi 2000t + 3\pi/4)\ .$$

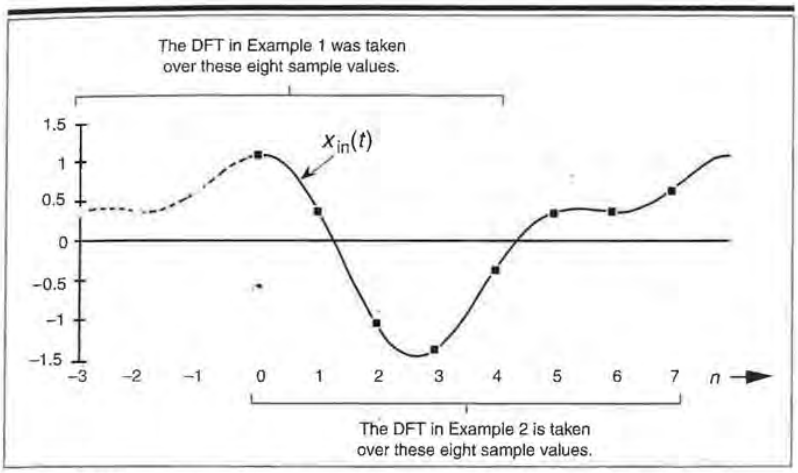


Figure 3-5 Comparison of sampling times between DFT Example 1 and DFT Example 2.

We can see that Figure 3-5 is a continuation of Figure 3-2(a). Our new $x(n)$ sequence becomes the values represented by the solid black dots in Figure 3-5 whose values are

$$\begin{aligned} x(0) &= 1.0607, & x(1) &= 0.3535, \\ x(2) &= -1.0607, & x(3) &= -1.3535, \\ x(4) &= -0.3535, & x(3) &= 0.3535, \\ x(6) &= 0.3535, & x(7) &= 0.6464\,. \end{aligned} \tag{3-20}$$

Performing the DFT on Eq. (3-20), $X_{\text{shifted}}(m)$ is

$X_{shifted}(m) =$

m	$X_{shifted}(m)$'s *magnitude*	$X_{shifted}(m)$'s *phase*	$X_{shifted}(m)$'s *real part*	$X_{shifted}(m)$'s *imaginary part*
0	0	0	0	0
1	4	+45	2.8284	2.8284
2	2	−45	1.4142	−1.414
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	2	+45	1.4142	1.4142
7	4	−45	2.8284	−2.828

(3-21)

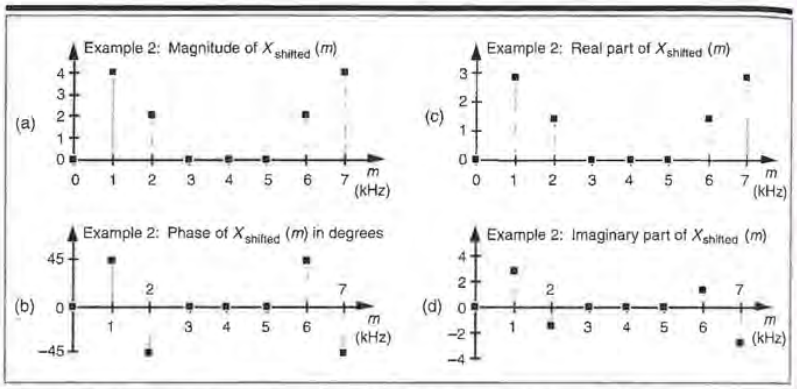


Figure 3-6 DFT results from Example 2: (a) magnitude of $X_{shifted}(m)$; (b) phase of $X_{shifted}(m)$; (c) real part of $X_{shifted}(m)$; (d) imaginary part of $X_{shifted}(m)$.

The values in Eq. (3-21) are illustrated as the dots in Figure 3-6. Notice that Figure 3-6(a) is identical to Figure 3-4(a). Equation (3-19) told us that the magnitude of $X_{shifted}(m)$ should be unchanged from that of $X(m)$. That's a comforting thought, isn't it? We wouldn't expect the DFT magnitude of our original periodic $x_{in}(t)$ to change just because we sampled it over a different time interval. The phase of the DFT result does, however, change depending on the instant at which we started to sample $x_{in}(t)$.

By looking at the $m = 1$ component of $X_{shifted}(m)$, for example, we can double-check to see that phase values in Figure 3-6(b) are correct. Using Eq. (3-19) and remembering that $X(1)$ from DFT Example 1 had a magnitude of 4 at a phase angle of –90 (or $-\pi/2$ radians), $k = 3$ and $N = 8$ so that

$$X_{shifted}(1) = e^{j2\pi km/N} \cdot X(1) = e^{j2\pi 3 \cdot 1/8} \cdot 4e^{-j\pi/2} = 4e^{j(6\pi/8 - 4\pi/8)} = 4e^{j\pi/4} . \quad (3\text{-}22)$$

So $X_{shifted}(1)$ has a magnitude of 4 and a phase angle of $\pi/4$ or +45°, which is what we set out to prove using Eq. (3-19).

3.7 Inverse DFT

Although the DFT is the major topic of this chapter, it's appropriate, now, to introduce the inverse discrete Fourier transform (IDFT). Typically we think of the DFT as transforming time-domain data into a frequency-domain representation. Well, we can reverse this process and obtain the original time-domain signal by performing the IDFT on the $X(m)$ frequency-domain values. The standard expressions for the IDFT are

$$x(n) = \frac{1}{N}\sum_{m=0}^{N-1} X(m)e^{j2\pi mn/N} \tag{3-23}$$

and equally,

$$x(n) = \frac{1}{N}\sum_{m=0}^{N-1} X(m)[\cos(2\pi mn/N) + j\sin(2\pi mn/N)] \ . \tag{3-23'}$$

Remember the statement we made in Section 3.1 that a discrete time-domain signal can be considered the sum of various sinusoidal analytical frequencies and that the $X(m)$ outputs of the DFT are a set of N complex values indicating the magnitude and phase of each analysis frequency comprising that sum. Equations (3-23) and (3-23') are the mathematical expressions of that statement. It's very important for the reader to understand this concept. If we perform the IDFT by plugging our results from DFT Example 1 into Eq. (3-23), we'll go from the frequency-domain back to the time-domain and get our original real Eq. (3-11') $x(n)$ sample values of

$$\begin{aligned} x(0) &= 0.3535 + j0.0 \quad x(1) = 0.3535 + j0.0 \\ x(2) &= 0.6464 + j0.0 \quad x(3) = 1.0607 + j0.0 \\ x(4) &= 0.3535 + j0.0 \quad x(3) = -1.0607 + j0.0 \\ x(6) &= -1.3535 + j0.0 \quad x(7) = -0.3535 + j0.0 \ . \end{aligned}$$

Notice that Eq. (3-23)'s IDFT expression differs from the DFT's Eq. (3-2) only by a $1/N$ scale factor and a change in the sign of the exponent. Other than the magnitude of the results, every characteristic that we've covered, thus far, regarding the DFT, also applies to the IDFT.

3.8 DFT Leakage

Hold on to your seat now. Here's where the DFT starts to get really interesting. The two previous DFT examples gave us correct results because the input $x(n)$ sequences were very carefully chosen sinusoids. As it turns out, the DFT of sampled real-world signals provides frequency-domain results that can be misleading. A characteristic, known as leakage, causes our DFT results to be only an approximation of the true spectra of the original input signals prior to digital sampling. Although there are ways to minimize leakage, we can't eliminate it entirely. Thus, we need to understand exactly what effect it has on our DFT results.

Let's start from the beginning. DFTs are constrained to operate on a finite set of N input values sampled at a sample rate of f_s, to produce an N-point transform whose discrete outputs are associated with the individual analytical frequencies $f_{analysis}(m)$, with

$$f_{analysis}(m) = \frac{mf_s}{N}, \text{where } m = 0, 1, 2, \ldots, N-1. \quad (3\text{-}24)$$

Equation (3-24), illustrated in DFT Example 1, may not seem like a problem, but it is. The DFT produces correct results only when the input data sequence contains energy precisely at the analysis frequencies given in Eq. (3-24), at integral multiples of our fundamental frequency f_s/N. If the input has a signal component at some intermediate frequency between our analytical frequencies of mf_s/N, say $1.5f_s/N$, this input signal will show up to some degree in *all* of the N output analysis frequencies of our DFT! (We typically say that input signal energy shows up in all of the DFT's output *bins*, and we'll see, in a moment, why the phrase "output bins" is appropriate.) Let's understand the significance of this problem with another DFT example.

Assume we're taking a 64-point DFT of the sequence indicated by the dots in Figure 3-7(a). The sequence is a sinewave with exactly three cycles contained in our $N = 64$ samples. Figure 3-7(b) shows the first half of the DFT of the input sequence and indicates that the sequence has an average value of zero ($X(0) = 0$) and no signal components at any frequency other than the $m = 3$ frequency. No surprises so far. Figure 3-7(a) also shows, for example, the $m = 4$ sinewave analysis frequency, superimposed over the input sequence, to remind us that the analytical frequencies always have an integral number of cycles over our total sample interval of 64 points. The sum of the products of the input sequence and the $m = 4$ analysis frequency is zero. (Or we can say, the correlation of the input sequence, and the $m = 4$ analysis frequency is zero.) The sum of the products of this particular three-cycle input sequence and any analysis frequency other than $m = 3$ is zero. Continuing with our leakage example, the dots in Figure 3-8(a) show an input sequence having 3.4 cycles over our $N = 64$ samples. Because the input sequence does not have an integral number of cycles over our 64-sample interval, input energy has leaked into all the other DFT output bins as shown in Figure 3-8(b). The $m = 4$ bin, for example, is not zero because the sum of the products of the input sequence and the $m = 4$ analysis frequency is no longer zero. This is leakage—it causes any input signal whose frequency is not exactly at a DFT bin center to leak into all of the other DFT output bins. Moreover,

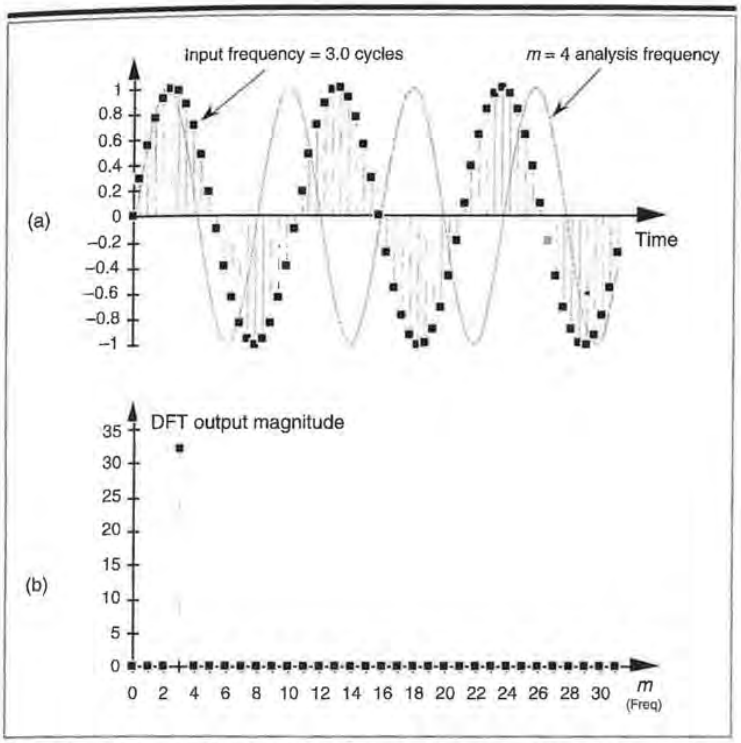


Figure 3-7 64-point DFT: (a) input sequence of three cycles and the $m = 4$ analysis frequency sinusoid; (b) DFT output magnitude.

leakage is an unavoidable fact of life when we perform the DFT on real-world finite-length time sequences.

Now, as the English philosopher Douglas Adams would say, "Don't panic." Let's take a quick look at the cause of leakage to learn how to predict and minimize its unpleasant effects. To understand the effects of leakage, we need to know the amplitude response of a DFT when the DFT's input is an arbitrary, real sinusoid. Although Sections 3.14 and 3.15 discuss this issue in detail, for our purposes, here, we'll just say that, for a real cosine input having k cycles in the N-point input time sequence, the amplitude response of an N-point DFT bin in terms of the bin index m is approximated by the sinc function

$$X(m) \approx \frac{N}{2} \cdot \frac{\sin[\pi(k-m)]}{\pi(k-m)} \quad . \tag{3-25}$$

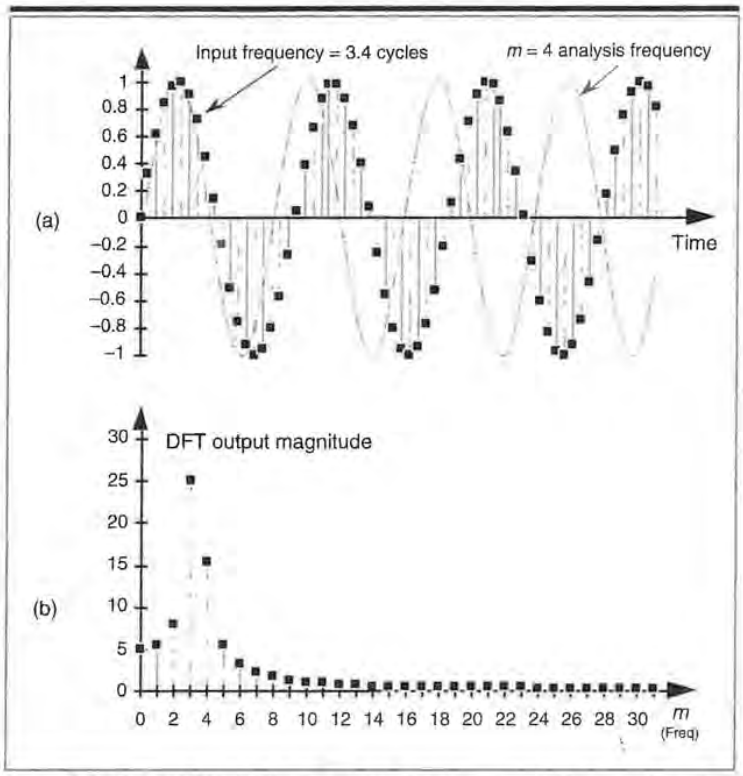


Figure 3-8 64-point DFT: (a) 3.4 cycles input sequence and the $m = 4$ analysis frequency sinusoid; (b) DFT output magnitude.

We'll use Eq. (3-25), illustrated in Figure 3-9(a), to help us determine how much leakage occurs in DFTs. We can think of the curve in Figure 3-9(a), comprising a main lobe and periodic peaks and valleys known as *sidelobes*, as the continuous positive spectrum of an N-point, real cosine time sequence having k complete cycles in the N-point input time interval. The DFT's outputs are discrete samples that reside on the curves in Figure 3-9; that is, our DFT output will be a sampled version of the continuous spectrum. (We show the DFT's magnitude response to a real input in terms of frequency (Hz) in Figure 3-9(b).) When the DFT's input sequence has exactly an integral k number of cycles (centered exactly in the $m = k$ bin), no leakage occurs, as in Figure 3-9, because when the angle in the numerator of Eq. (3-25) is a nonzero integral multiple of π, the sine of that angle is zero.

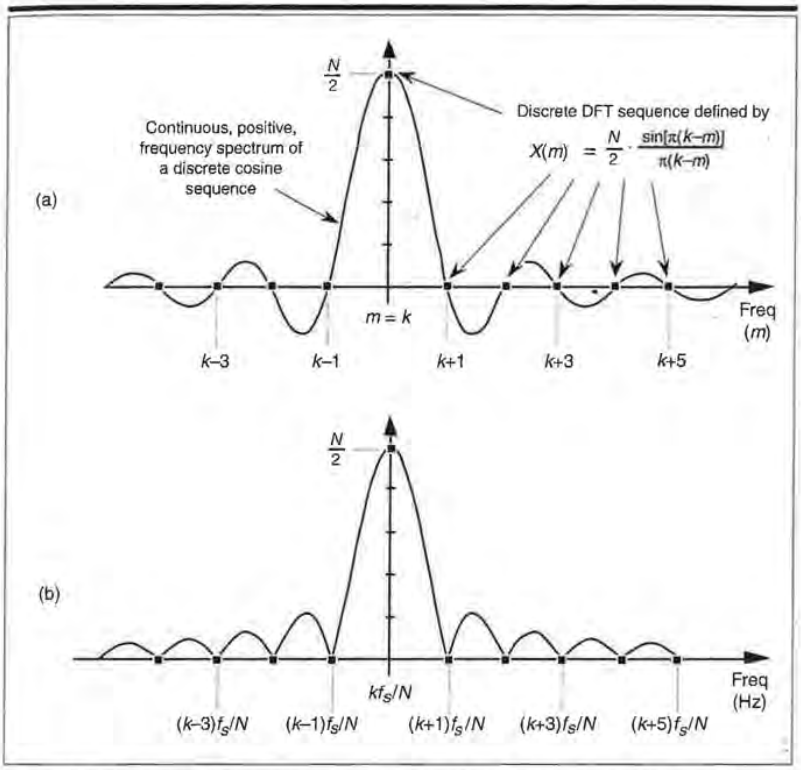


Figure 3-9 DFT positive frequency response due to an N-point input sequence containing k cycles of a real cosine: (a) amplitude response as a function of bin index m; (b) magnitude response as a function of frequency in Hz.

By way of example, we can illustrate again what happens when the input frequency k is not located at a bin center. Assume that a real 8-kHz sinusoid, having unity amplitude, has been sampled at a rate of $f_s = 32{,}000$ samples/s. If we take a 32-point DFT of the samples, the DFT's frequency resolution, or bin spacing, is $f_s/N = 32{,}000/32$ Hz = 1.0 kHz. We can predict the DFT's magnitude response by centering the input sinusoid's spectral curve at the positive frequency of 8 kHz, as shown in Figure 3-10(a). The dots show the DFT's output bin magnitudes.

Again, here's the important point to remember: the DFT output is a sampled version of the continuous spectral curve in Figure 3-10(a). Those sampled values in the frequency-domain, located at mf_s/N, are the dots in Figure 3-10(a). Because the input signal frequency is exactly at a DFT

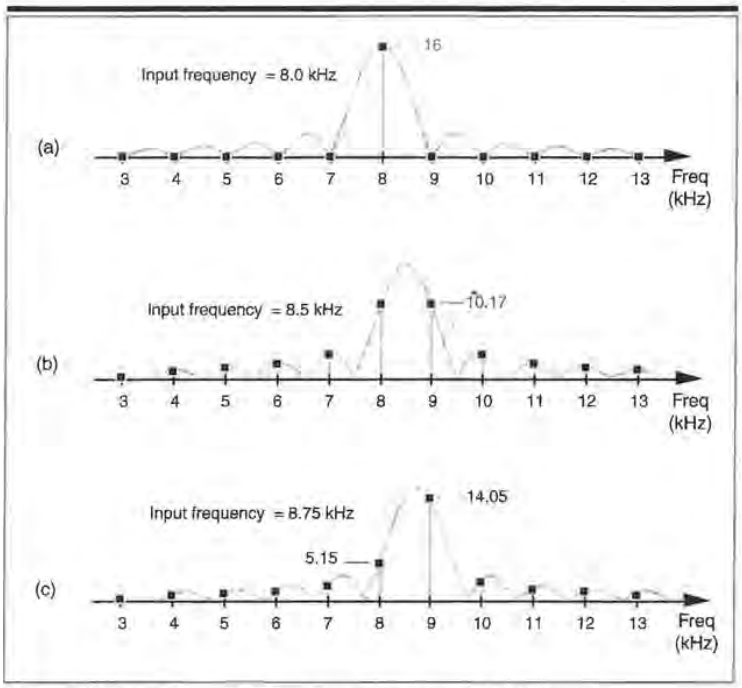


Figure 3-10 DFT bin positive frequency responses: (a) DFT input frequency = 8.0 kHz; (b) DFT input frequency = 8.5 kHz; (c) DFT input frequency = 8.75 kHz.

bin center, the DFT results have only one nonzero value. Stated in another way, when an input sinusoid has an integral number of cycles over N time-domain input sample values, the DFT outputs reside on the continuous spectrum at its peak and exactly at the curve's zero crossing points. From Eq. (3-25) we know the peak output magnitude is $32/2 = 16$. (If the real input sinusoid had an amplitude of 2, the peak of the response curve would be $2 \cdot 32/2$, or 32.) Figure 3-10(b) illustrates DFT leakage where the input frequency is 8.5 kHz, and we see that the frequency-domain sampling results in nonzero magnitudes for all DFT output bins. An 8.75-kHz input sinusoid would result in the leaky DFT output shown in Figure 3-10(c). If we're sitting at a computer studying leakage by plotting the magnitude of DFT output values, of course, we'll get the dots in Figure 3-10 and won't see the continuous spectral curves.

At this point, the attentive reader should be thinking: "If the continuous spectrums that we're sampling are symmetrical, why does the DFT output in

Figure 3-8(b) look so asymmetrical?" In Figure 3-8(b), the bins to the right of the third bin are decreasing in amplitude faster than the bins to the left of the third bin. "And another thing, evaluating the continuous spectrum's $X(mf_s)$ function at an abscissa value of 0.4 gives a magnitude scale factor of 0.75. Applying this factor to the DFT's maximum possible peak magnitude of 32, we should have a third bin magnitude of approximately $32 \cdot 0.75 = 24$—but Figure 3-8(b) shows that the third-bin magnitude is slightly greater than 25. What's going on here?" We answer this by remembering what Figure 3-8(b) really represents. When examining a DFT output, we're normally interested only in the $m = 0$ to $m = (N/2-1)$ bins. Thus, for our 3.4 cycles per sample interval example in Figure 3-8(b), only the first 32 bins are shown. Well, the DFT is periodic in the frequency domain as illustrated in Figure 3-11. Upon examining the DFT's output for higher and higher frequencies, we end up going in circles, and the spectrum repeats itself forever.

The more conventional way to view a DFT output is to *unwrap* the spectrum in Figure 3-11 to get the spectrum in Figure 3-12. Figure 3-12 shows some of the additional replications in the spectrum for the 3.4 cycles per sample interval example. Concerning our DFT output asymmetry problem, as some of the input 3.4-cycle signal amplitude leaks into the 2nd bin, the 1st bin, and the 0th bin, leakage continues into the –1st bin, the –2nd bin, the –3rd bin, etc. Remember, the 63rd bin is the –1st bin, the 62nd bin is the –2nd bin, and so on. These bin equivalencies allow

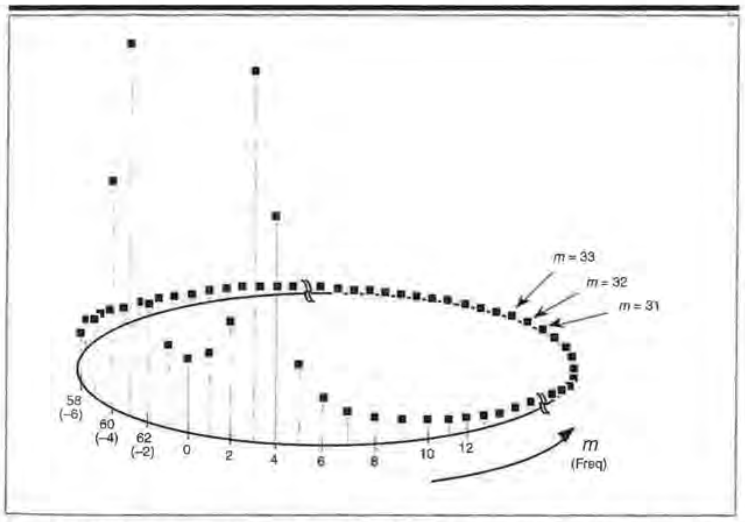


Figure 3-11 Cyclic representation of the DFT's spectral replication when the DFT input is 3.4 cycles per sample interval.

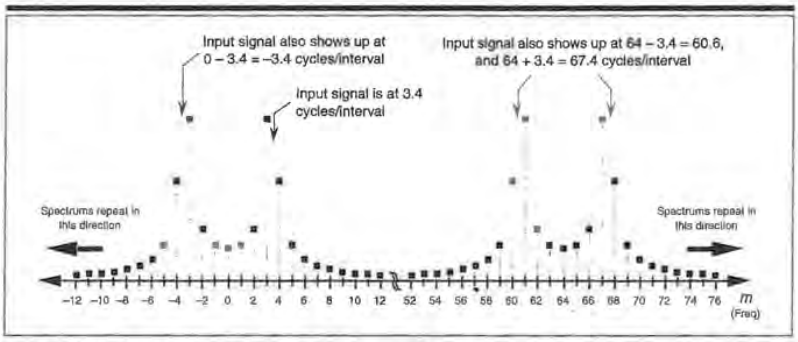


Figure 3-12 Spectral replication when the DFT input is 3.4 cycles per sample interval.

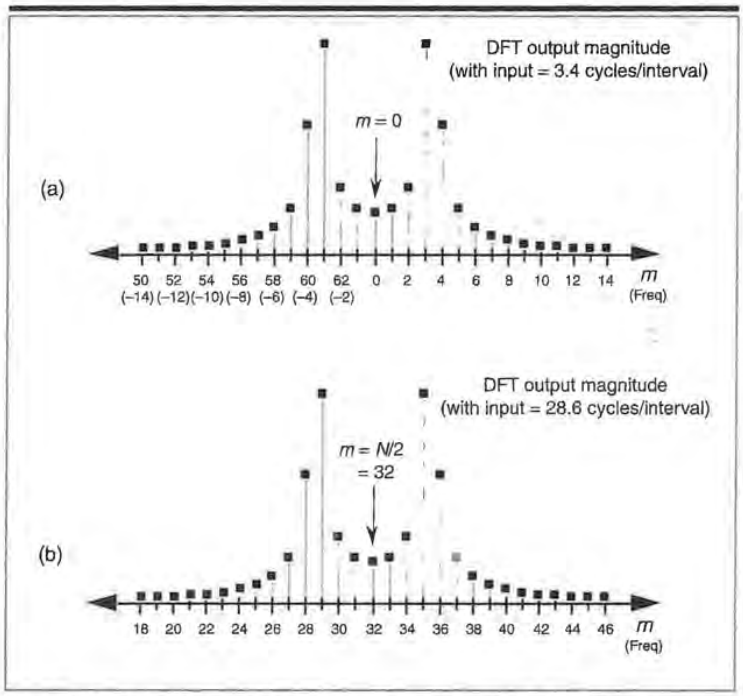


Figure 3-13 DFT output magnitude: (a) when the DFT input is 3.4 cycles per sample interval; (b) when the DFT input is 28.6 cycles per sample interval.

us to view the DFT output bins as if they extend into the negative frequency range, as shown in Figure 3-13(a). The result is that the leakage *wraps around* the $m = 0$ frequency bin, as well as around the $m = N$ frequency bin. This is not surprising because the $m = 0$ frequency *is* the $m = N$

frequency. The leakage wraparound at the $m = 0$ frequency accounts for the asymmetry about the DFT's $m = 3$ bin in Figure 3-8(b).

Recall from the DFT symmetry discussion, when a DFT input sequence $x(n)$ is real, the DFT outputs from $m = 0$ to $m = (N/2–1)$ are redundant with frequency bin values for $m > (N/2)$, where N is the DFT size. The mth DFT output will have the same magnitude as the $(N–m)$th DFT output. That is, $|X(m)| = |X(N–m)|$. What this means is that leakage wraparound also occurs about the $m = N/2$ bin. This can be illustrated using an input of 28.6 cycles per sample interval (32 – 3.4) whose spectrum is shown in Figure 3-13(b). Notice the similarity between Figure 3-13(a) and Figure 3-13(b). So the DFT exhibits leakage wraparound about the $m = 0$ and $m = N/2$ bins. Minimum leakage asymmetry will occur near the $N/4$th bin as shown in Figure 3-14(a) where the full spectrum of a 16.4 cycles per sample interval input is provided. Figure 3-14(b) shows a close-up view of the first 32 bins of the 16.4 cycles per sample interval spectrum.

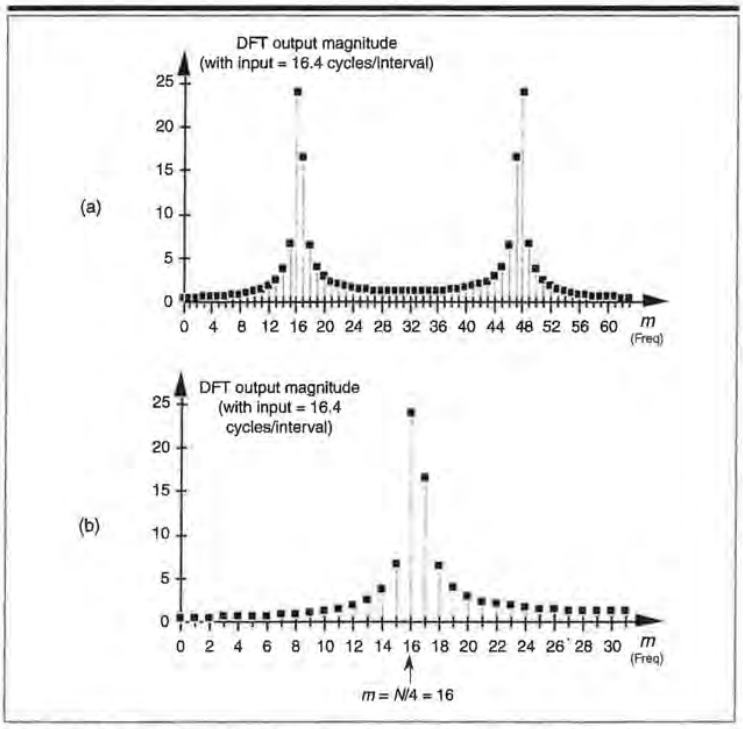


Figure 3-14 DFT output magnitude when the DFT input is 16.4 cycles per sample interval: (a) full output spectrum view; (b) close-up view showing minimized leakage asymmetry at frequency $m = N/4$.

You can read about leakage all day. However, the best way to appreciate its effects is to sit down at a computer and use a software program to take DFTs, in the form of fast Fourier transforms (FFTs), of your personally generated test signals like those in Figures 3-6 and 3-7. You can, then, experiment with different combinations of input frequencies and various DFT sizes. You'll be able to demonstrate that DFT leakage effect is troublesome because the bins containing low-level signals are corrupted by the sidelobe levels from neighboring bins containing high-amplitude signals.

Although there's no way to eliminate leakage completely, an important technique known as *windowing* is the most common remedy to reduce its unpleasant effects. Let's look at a few DFT window examples.

3.9 Windows

Windowing reduces DFT leakage by minimizing the magnitude of Eq. (3-25)'s sinc function's $\sin(x)/x$ sidelobes shown in Figure 3-9. We do this by forcing the amplitude of the input time sequence at both the beginning and the end of the sample interval to go smoothly toward a single common amplitude value. Figure 3-15 shows how this process works. If we consider the infinite-duration time signal shown in Figure 3-15(a), a DFT can only be performed over a finite-time sample interval like that shown in Figure 3-15(c). We can think of the DFT input signal in Figure 3-15(c) as the product of an input signal existing for all time, Figure 3-15(a), and the rectangular window whose magnitude is 1 over the sample interval shown in Figure 3-15(b). Anytime we take the DFT of a finite-extent input sequence we are, by default, multiplying that sequence by a window of all ones and effectively multiplying the input values outside that window by zeros. As it turns out, Eq. (3-25)'s sinc function's $\sin(x)/x$ shape, shown in Figure 3-9, is caused by this rectangular window because the continuous Fourier transform of the rectangular window in Figure 3-15(b) *is* the sinc function.

As we'll soon see, it's the rectangular window's abrupt changes between one and zero that are the cause of the sidelobes in the the $\sin(x)/x$ sinc function. To minimize the spectral leakage caused by those sidelobes, we have to reduce the sidelobe amplitudes by using window functions other than the rectangular window. Imagine if we multiplied our DFT input, Figure 3-15(c), by the triangular window function shown in Figure 3-15(d) to obtain the windowed input signal shown in Figure 3-15(e). Notice that the values of our final input signal appear to be the same at the beginning and end of the sample interval in Figure 3-15(e). The reduced discontinuity decreases the level of relatively high frequency components in our overall DFT output; that is, our DFT bin sidelobe levels are reduced in magnitude using a triangular window. There are other window func-

tions that reduce leakage even more than the triangular window, such as the Hanning window in Figure 3-15(f). The product of the window in Figure 3-15(f) and the input sequence provides the signal shown in Figure 3-15(g) as the input to the DFT. Another common window function is the Hamming window shown in Figure 3-15(h). It's much like the Hanning window, but it's raised on a *pedestal*.

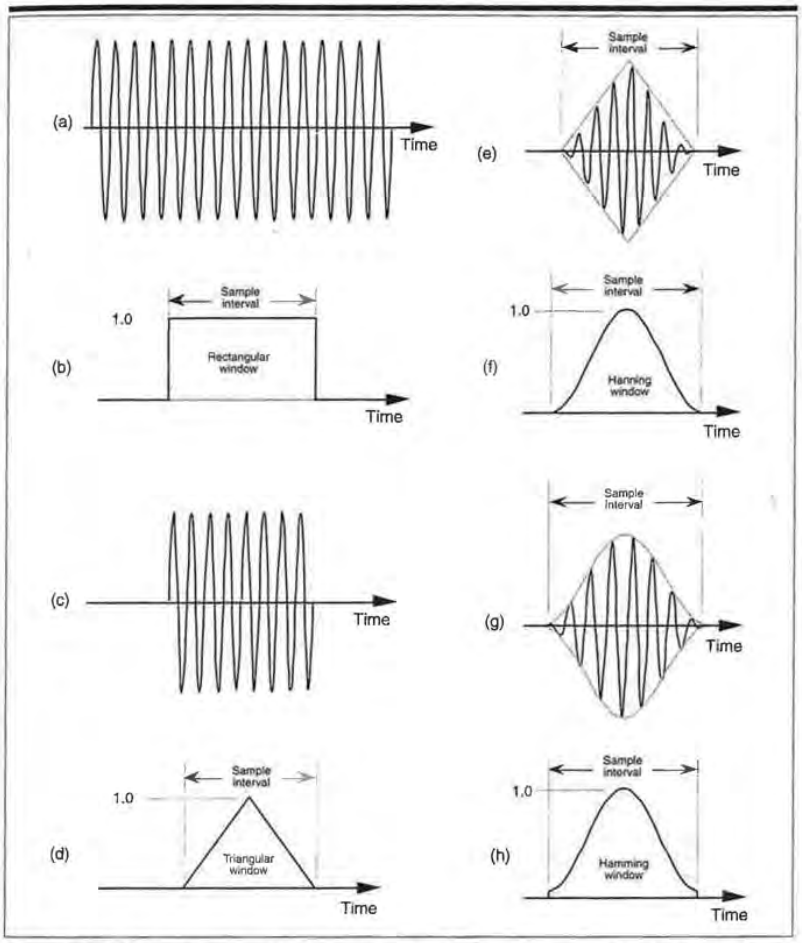


Figure 3-15 Minimizing sample interval endpoint discontinuities: (a) infinite duration input sinusoid; (b) rectangular window due to finite-time sample interval; (c) product of rectangular window and infinite-duration input sinusoid; (d) triangular window function; (e) product of triangular window and infinite-duration input sinusoid; (f) Hanning window function; (g) product of Hanning window and infinite-duration input sinusoid; (h) Hamming window function.

Before we see exactly how well these windows minimize DFT leakage, let's define them mathematically. Assuming that our original N input signal samples are indexed by n, where $0 \le n \le N-1$, we'll call the N time-domain window coefficients $w(n)$; that is, an input sequence $x(n)$ is multiplied by the corresponding window $w(n)$ coefficients before the DFT is performed. So the DFT of the windowed $x(n)$ input sequence, $X_w(m)$, takes the form of

$$X_\omega(m) = \sum_{n=0}^{N-1} \omega(n) \cdot x(n) e^{-j2\pi nm/N}. \tag{3-26}$$

To use window functions, we need mathematical expression of them in terms of n. The following expressions define our window function coefficients:

Rectangular window: (also called the uniform, or boxcar, window)

$$w(n) = 1, \text{ for } n = 0, 1, 2, \ldots, N-1. \tag{3-27}$$

Triangular window: (very similar to the Bartlett[3], and Parzen[4,5] windows)

$$\omega(n) = \frac{n}{N/2}, \text{ for } n = 0, 1, 2, \ldots, N/2, \text{ and}$$
$$= 2 - \frac{n}{N/2},$$
$$\text{for } n = N/2+1, N/2+2, \ldots, N-1. \tag{3-28}$$

Hanning window: (also called the raised cosine, Hann, or von Hann window)

$$w(n) = 0.5 - 0.5\cos(2\pi n/N), \text{ for } n = 0, 1, 2, \ldots, N-1. \tag{3-29}$$

Hamming window:

$$w(n) = 0.54 - 0.46\cos(2\pi n/N), \text{ for } n = 0, 1, 2, \ldots, N-1. \tag{3-30}$$

If we plot the $w(n)$ values from Eqs. (3-27) through (3-30), we'd get the corresponding window functions like those in Figures 3-15(b), 3-15(d), 3-15(f), and 3-15(h).†

† In the literature, the equations for window functions depend on the range of the sample index n. We define n to be in the range $0 < n < N-1$. Some authors define n to be in the range $-N/2 < n < N/2$, in which case, for example, the expression for the Hanning window would have a sign change and be $w(n) = 0.5 + 0.5\cos(2\pi n/N)$.

The rectangular window's amplitude response is the yardstick we normally use to evaluate another window function's amplitude response; that is, we typically get an appreciation for a window's response by comparing it to the rectangular window that exhibits the magnitude response shown in Figure 3-9(b). The rectangular window's $\sin(x)/x$ magnitude response, $|W(m)|$, is repeated in Figure 3-16(a). Also included in Figure 3-16(a) are the Hamming, Hanning, and triangular window magnitude responses. (The frequency axis in Figure 3-16 is such that the curves show the response of a single N-point DFT bin when the various window functions are used.) We can see that the last three windows give reduced sidelobe levels relative to the rectangular window. Because the Hamming, Hanning, and triangular windows reduce the time-domain signal levels applied to the DFT, their main lobe peak values are reduced relative to the rectangular window. (Because of the near-zero $w(n)$ coefficients at the beginning and end of the sample interval, this signal level loss is called the processing gain, or loss, of a window.) Be that as it may, we're primarily interested in the windows' sidelobe levels, which are difficult to see in Figure 3-16(a)'s linear scale. We will avoid this difficulty by plotting the windows' magnitude responses on a logarithmic decibel scale, and normalize each plot so its main lobe peak values are zero dB. (Appendix E provides a discussion of the origin and utility of measuring frequency-domain responses on a logarithmic scale using decibels.) Defining the log magnitude response to be $|W_{dB}(m)|$, we get $|W_{dB}(m)|$ by using the expression

$$|W_{dB}(m)| = 20 \cdot \log_{10}\left(\frac{|W(m)|}{|W(0)|}\right). \tag{3-31}$$

(The $|W(0)|$ term in the denominator of Eq. (3-31) is the value of $W(m)$ at the peak of the main lobe when $m = 0$.) The $|W_{dB}(m)|$ curves for the various window functions are shown in Figure 3-16(b). Now we can really see how the various window sidelobe responses compare to each other.

Looking at the rectangular window's magnitude response we see that its main lobe is the most narrow, $2f_s/N$. However, unfortunately, its first sidelobe level is only –13 dB below the main lobe peak, which is not so good. (Notice that we're only showing the positive frequency portion of the window responses in Figure 3-16.) The triangular window has reduced sidelobe levels, but the price we've paid is that the triangular window's main lobe width is twice as wide as that of the rectangular window's. The various nonrectangular windows' wide main lobes degrade the windowed DFT's frequency resolution by almost a factor of two. However, as we'll see, the important benefits of leakage reduction usually outweigh the loss in DFT frequency resolution.

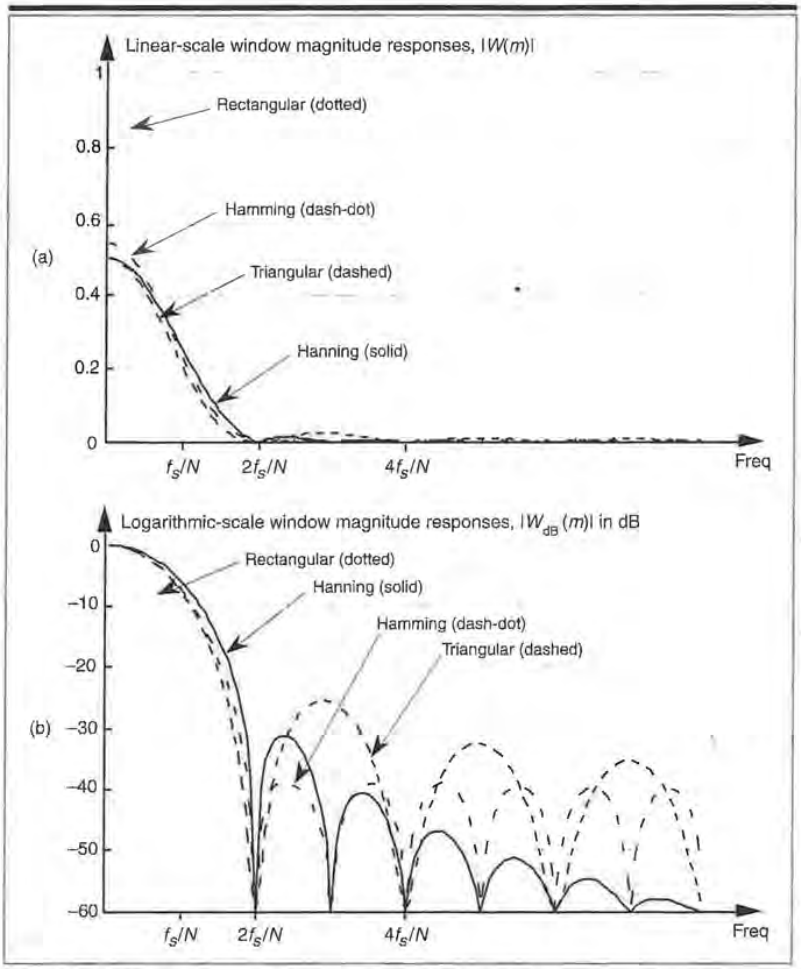


Figure 3-16 Window magnitude responses: (a) $|W(m)|$ on a linear scale; (b) $|W_{dB}(m)|$ on a normalized logarithmic scale.

Notice the further reduction of the first sidelobe level, and the rapid sidelobe roll-off of the Hanning window. The Hamming window has even lower first sidelobe levels, but this window's sidelobes roll off slowly relative to the Hanning window. This means that leakage three or four bins away from the center bin is lower for the Hamming window than for the Hanning window, and leakage a dozen or so bins away from the center bin is lower for the Hanning window than for the Hamming window.

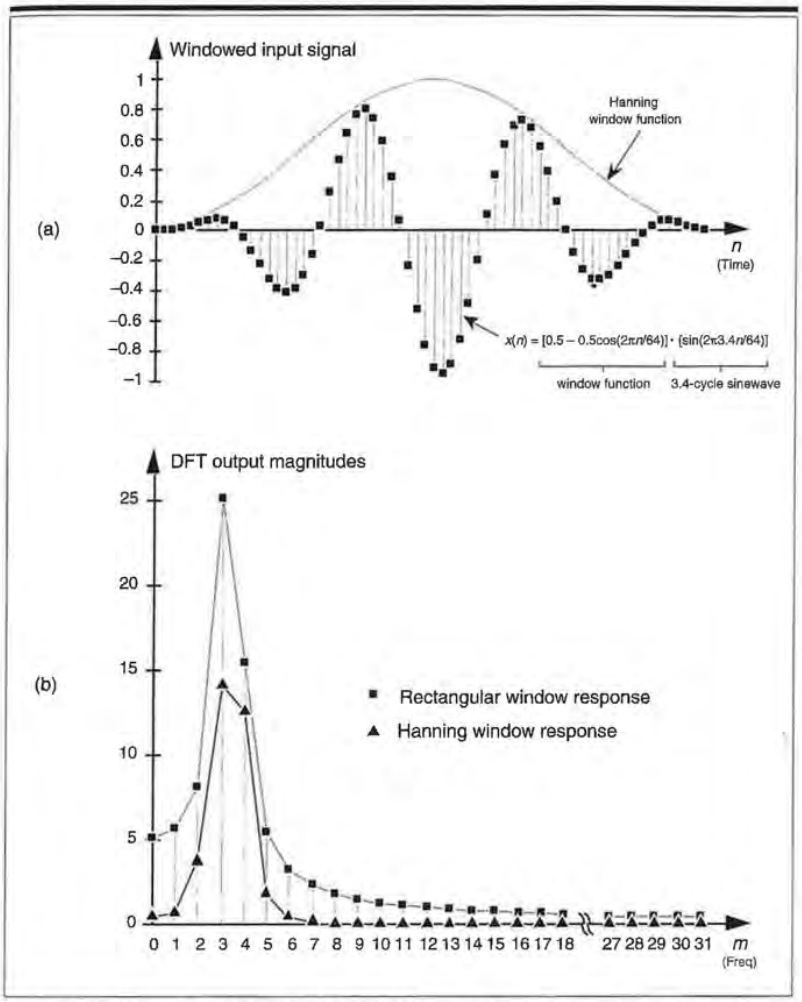


Figure 3-17 Hanning window: (a) 64-sample product of a Hanning window and a 3.4 cycles per sample interval input sinewave; (b) Hanning DFT output response vs. rectangular window DFT output response.

When we apply the Hanning window to Figure 3-8(a)'s 3.4 cycles per sample interval example, we end up with the DFT input shown in Figure 3-17(a) under the Hanning window envelope. The DFT outputs for the windowed waveform are shown in Figure 3-17(b) along with the DFT results with no windowing, i.e., the rectangular window. As we expected,

the shape of the Hanning window's response looks broader and has a lower peak amplitude, but its sidelobe leakage is noticeably reduced from that of the rectangular window.

We can demonstrate the benefit of using a window function to help us detect a low-level signal in the presence of a nearby high-level signal. Let's add 64 samples of a 7 cycles per sample interval sinewave, with a peak amplitude of only 0.1, to Figure 3-8(a)'s unity-amplitude 3.4 cycles per sample sinewave. When we apply a Hanning window to the sum of these sinewaves, we get the time-domain input shown in Figure 3-18(a). Had we not windowed the input data, our DFT output would be the squares in Figure 3-18(b) where DFT leakage causes the input signal component at $m = 7$ to be barely discernible. However, the DFT of the windowed data shown as the triangles in Figure 3-18(b), makes it easier for us to detect the presence of the $m = 7$ signal component. From a practical standpoint, people who use the DFT to perform real-world signal detection have learned that their overall frequency resolution and signal sensitivity are affected much more by the size and shape of their window function than the mere size of their DFTs.

As we become more experienced using window functions on our DFT input data, we'll see how different window functions have their own individual advantages and disadvantages. Furthermore, regardless of the window function used, we've decreased the leakage in our DFT output from that of the rectangular window. There are many different window functions described in the literature of digital signal processing—so many, in fact, that they've been named after just about everyone in the digital signal processing business. It's not that clear that there's a great deal of difference among many of these window functions. What we find is that window selection is a trade-off between main lobe widening, first sidelobe levels, and how fast the sidelobes decrease with increased frequency. The use of any particular window depends on the application[5], and there are many applications.

Windows are used to improve DFT spectrum analysis accuracy[6], to design digital filters[7,8], to simulate antenna radiation patterns, and even in the hardware world to improve the performance of certain mechanical force to voltage conversion devices[9]. So there's plenty of window information available for those readers seeking further knowledge. (The Mother of all technical papers on windows is that by Harris[10]. A useful paper by Nuttall corrected and extended some portions of Harris's paper[11].) Again, the best way to appreciate windowing effects is to have access to a computer software package that contains DFT, or FFT, routines and start analyzing windowed signals. (By the way,

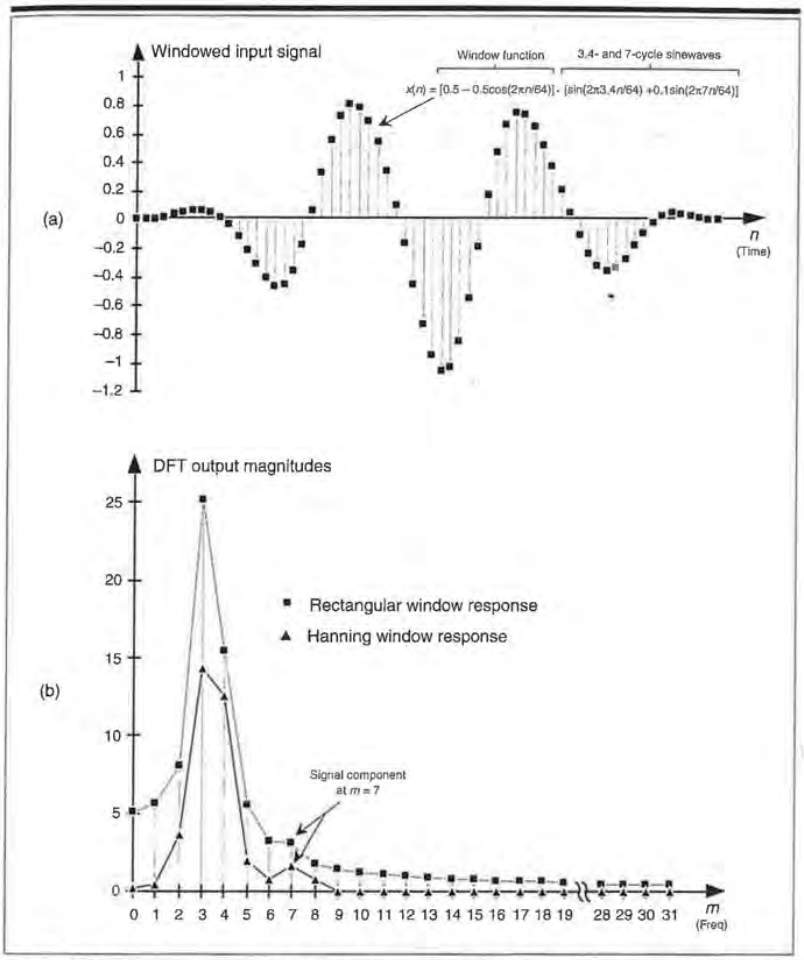


Figure 3-18 Increased signal detection sensitivity afforded using windowing: (a) 64-sample product of a Hanning window and the sum of a 3.4 cycles and a 7 cycles per sample interval sinewaves; (b) reduced leakage Hanning DFT output response vs. rectangular window DFT output response.

while we delayed their discussion until Section 5.3, there are two other commonly used window functions that can be used to reduce DFT leakage. They're the Chebyshev and Kaiser window functions, that have adjustable parameters, enabling us to strike a compromise between widening main lobe width and reducing sidelobe levels.)

3.10 DFT Scalloping Loss

Scalloping is the name used to describe fluctuations in the overall magnitude response of an N-point DFT. Although we derive this fact in Section 3.16, for now we'll just say that when no input windowing function is used, the $\sin(x)/x$ shape of the sinc function's magnitude response applies to each DFT output bin. Figure 3-19(a) shows a DFT's aggregate magnitude response by superimposing several $\sin(x)/x$ bin magnitude responses.† (Because the sinc function's sidelobes are not key to this discussion, we don't show them in Figure 3-19(a).) Notice from Figure 3-19(b) that the overall DFT frequency-domain response is indicated by the bold envelope curve. This rippled curve, also called the picket fence effect, illustrates the processing loss for input frequencies between the bin centers.

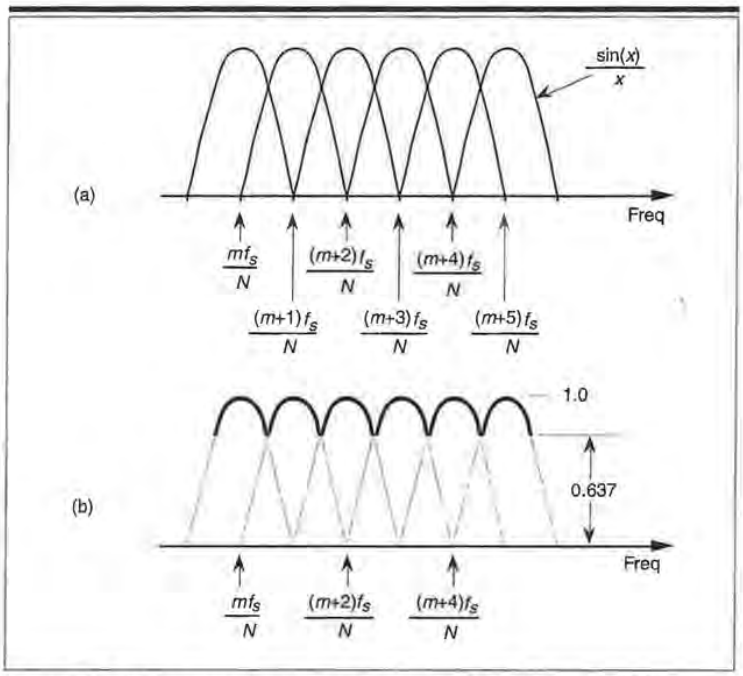


Figure 3-19 DFT bin magnitude response curves: (a) individual $\sin(x)/x$ responses for each DFT bin; (b) equivalent overall DFT magnitude response.

† Perhaps Figure 3-19(a) is why individual DFT outputs are called "bins." Any signal energy under a $\sin(x)/x$ curve will show up in the *enclosed storage compartment* of that DFT's output sample.

From Figure 3-19(b), we can determine that the magnitude of the DFT response fluctuates from 1.0, at bin center, to 0.637 halfway between bin centers. If we're interested in DFT output power levels, this envelope ripple exhibits a scalloping loss of almost −4 dB halfway between bin centers. Figure 3-19 illustrates a DFT output when no window (i.e., a rectangular window) is used. Because nonrectangular window functions broaden the DFT's main lobe, their use results in a scalloping loss that will not be as severe as the rectangular window[10,12]. That is, their wider main lobes overlap more and fill in the valleys of the envelope curve in Figure 3-19(b). For example, the scalloping loss of a Hanning window is approximately 0.82, or −1.45 dB, halfway between bin centers. Scalloping loss is not, however, a severe problem in practice. Real-world signals normally have bandwidths that span many frequency bins so that DFT magnitude response ripples can go almost unnoticed. Let's look at a scheme called *zero stuffing* that's used to both alleviate scalloping loss effects and to improve the DFT's frequency resolution.

3.11 DFT Resolution, Zero Stuffing, and Frequency-Domain Sampling

One popular method used to improve DFT frequency resolution is known as *zero stuffing*, or *zero padding*. This process involves the addition of zero-valued data samples to an original DFT input sequence to increase the total number of input data samples. Investigating this zero stuffing technique illustrates the DFT's important property of frequency-domain sampling alluded to in the discussion on leakage. When we sample a continuous time-domain function, having a continuous Fourier transform (CFT), and take the DFT of those samples, the DFT results in a frequency-domain sampled approximation of the CFT. The more points in our DFT, the better our DFT output approximates the CFT.

To illustrate this idea, suppose we want to approximate the CFT of the continuous $f(t)$ function in Figure 3-20(a). This $f(t)$ waveform extends to infinity in both directions but is nonzero only over the time interval of T seconds. If the nonzero portion of the time function is a sinewave of three cycles in T seconds, the magnitude of its CFT is shown in Figure 3-20(b). (Because the CFT is taken over an infinitely wide time interval, the CFT has infinitesimally small frequency resolution, resolution so fine-grained that it's continuous.) It's this CFT that we'll approximate with a DFT.

Suppose we want to use a 16-point DFT to approximate the CFT of $f(t)$ in Figure 3-20(a). The 16 discrete samples of $f(t)$, spanning the three periods of $f(t)$'s sinusoid, are those shown on the left side of Figure 3-21(a). Applying

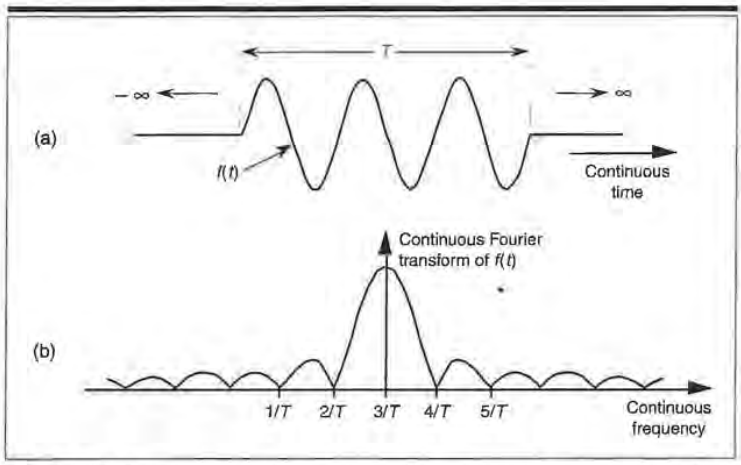


Figure 3-20 Continuous Fourier transform: (a) continuous time-domain $f(t)$ of a truncated sinusoid of frequency $3/T$; (b) continuous Fourier transform of $f(t)$.

those time samples to a 16-point DFT results in discrete frequency-domain samples, the positive frequency of which are represented by the dots on the right side of Figure 3-21(a). We can see that the DFT output samples Figure 3-20(b)'s CFT. If we append (or zero stuff) 16 zeros to the input sequence and take a 32-point DFT, we get the output shown on the right side of Figure 3-21(b), where we've increased our DFT frequency resolution by a factor of two. Our DFT is sampling the input function's CFT more often now. Adding 32 more zeros and taking a 64-point DFT, we get the output shown on the right side of Figure 3-21(c). The 64-point DFT output now begins to show the true shape of the CFT. Adding 64 more zeros and taking a 128-point DFT, we get the output shown on the right side of Figure 3-21(d). The DFT frequency-domain sampling characteristic is obvious now, but notice that the bin index for the center of the main lobe is different for each of the DFT outputs in Figure 3-21.

Does this mean we have to redefine the DFT's frequency axis when using the zero-stuffing technique? Not really. If we perform zero stuffing on L nonzero input samples to get a total of N time samples for an N-point DFT, the zero-stuffed DFT output bin center frequencies are related to the original f_s by our old friend Eq. (3-5), or

$$\text{center frequency of the } m\text{th bin} = \frac{mf_s}{N}. \tag{3-32}$$

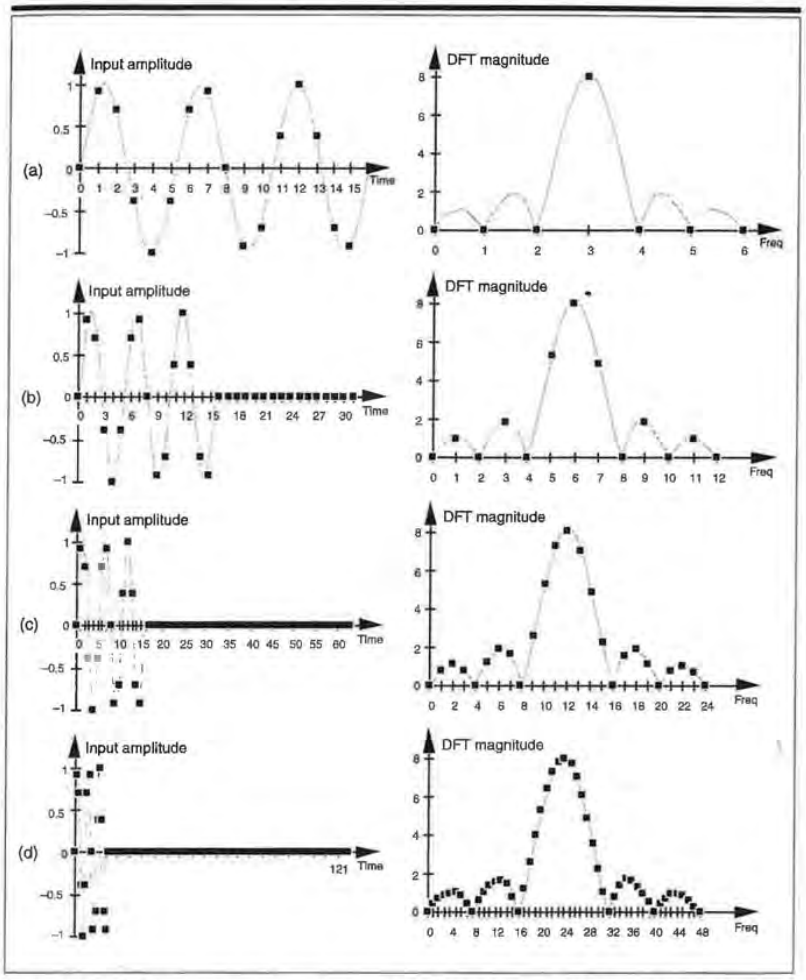


Figure 3-21 DFT frequency-domain sampling: (a) 16 input data samples and $N = 16$; (b) 16 input data samples, 16 stuffed zeros, and $N = 32$; (c) 16 input data samples, 48 stuffed zeros, and $N = 64$; (d) 16 input data samples, 112 stuffed zeros, and $N = 128$.

So in our Figure 3-21(a) example, we use Eq. (3-32) to show that, although the zero-stuffed DFT output bin index of the main lobe changes as N increases, the zero-stuffed DFT output frequency associated with the main lobe remains the same. The following list shows how this works:

Figure No.	*Main lobe peak located at* m =	L =	N =	*Frequency of main lobe peak relative to* f_s =
Figure 3-21(a)	3	16	16	$3f_s/16$
Figure 3-21(b)	6	16	32	$6 \cdot f_s/32 = 3f_s/16$
Figure 3-21(c)	12	16	64	$12 \cdot f_s/64 = 3f_s/16$
Figure 3-21(d)	24	16	128	$24 \cdot f_s/128 = 3f_s/16$

Do we gain anything by appending more zeros to the input sequence and taking larger DFTs? Not really, because our 128-point DFT is sampling the input's CFT sufficiently now in Figure 3-21(d). Sampling it more often with a larger DFT won't improve our understanding of the input's frequency content. The issue here is that adding zeros to an input sequence will improve our DFT's output resolution, but there's a practical limit on how much we gain by adding more zeros. For our example here, a 128-point DFT shows us the detailed content of the input spectrum. We've hit a *law of diminishing returns* here. Performing a 256-point or 512-point DFT, in our case, would serve little purpose.[†] There's no reason to *oversample* this particular input sequence's CFT. Of course, there's nothing sacred about stopping at a 128-point DFT. Depending on the number of samples in some arbitrary input sequence and the sample rate, we might, in practice, need to append any number of zeros to get some desired DFT frequency resolution.

There are two final points to be made concerning zero stuffing. First, the DFT magnitude expressions in Eqs. (3-17) and (3-17') don't apply if zero stuffing is being used. If we perform zero stuffing on L nonzero samples of a sinusoid whose frequency is located at a bin center to get a total of N input samples for an N-point DFT, we must replace the N with L in Eqs. (3-17) and (3-17') to predict the DFT's output magnitude for that particular sinewave. Second, in practical situations, if we want to perform both zero stuffing and windowing on a sequence of input data samples, we must be careful not to apply the window to the entire input including the appended zero-valued samples. The window function must be applied only to the original nonzero time samples, otherwise the stuffed zeros will *zero out* and distort part of the window function,

[†] Notice that the DFT sizes (N) we've discussed are powers of 2 (64, 128, 256, 512). That's because we actually perform DFTs using a special algorithm known as the fast Fourier transform (FFT). As we'll see in Chapter 4, the typical implementation of the FFT requires that N be a power of 2.

leading to erroneous results. (Section 4.5 gives additional practical pointers on performing the DFT using the FFT algorithm to analyze real-world signals.)

To digress slightly, now's a good time to define the term *discrete-time Fourier transform* (DTFT) that the reader may encounter in the literature. The DTFT is the continuous Fourier transform of an L-point discrete time-domain sequence; some authors use the DTFT to describe many of the digital signal processing concepts we've covered in this chapter. We can't perform the DTFT on a computer because it has an infinitely fine frequency resolution—but we could, if we wished, approximate the DTFT by performing an N-point DFT on an L-point discrete time sequence where $N > L$. That is, in fact, what we did in Figure 3-21 when we zero-stuffed the original 16-point time sequence. While we don't emphasize the DTFT in this book, keep in mind that when $N = L$ the DTFT approximation is identical to the DFT.

3.12 DFT Processing Gain

There are two types of processing gain associated with DFTs. People who use the DFT to detect signal energy embedded in noise often speak of the DFT's *processing gain* because the DFT can *pull* signals out of background noise. This is due to the inherent correlation gain that takes place in any N-point DFT. Beyond this natural processing gain, additional *integration gain* is possible when multiple DFT outputs are averaged. Let's look at the DFT's inherent processing gain first.

3.2.1 Processing Gain of a Single DFT

The concept of the DFT having processing gain is straightforward if we think of a particular DFT bin output as the output of a narrowband filter. Because a DFT output bin has the amplitude response of the $\sin(x)/x$ function, that bin's output is primarily due to input energy residing under, or very near, the bin's main lobe. It's valid to think of a DFT bin as a kind of *bandpass* filter whose band center is located at mf_s/N. We know from Eq. (3-17) that the maximum possible DFT output magnitude increases as the number of points (N) in a DFT increases. Also, as N increases, the DFT output bin main lobes become more narrow. So a DFT output bin can be treated as a bandpass filter whose gain can be increased and whose bandwidth can be reduced by increasing the value of N. Decreasing a bandpass filter's bandwidth is

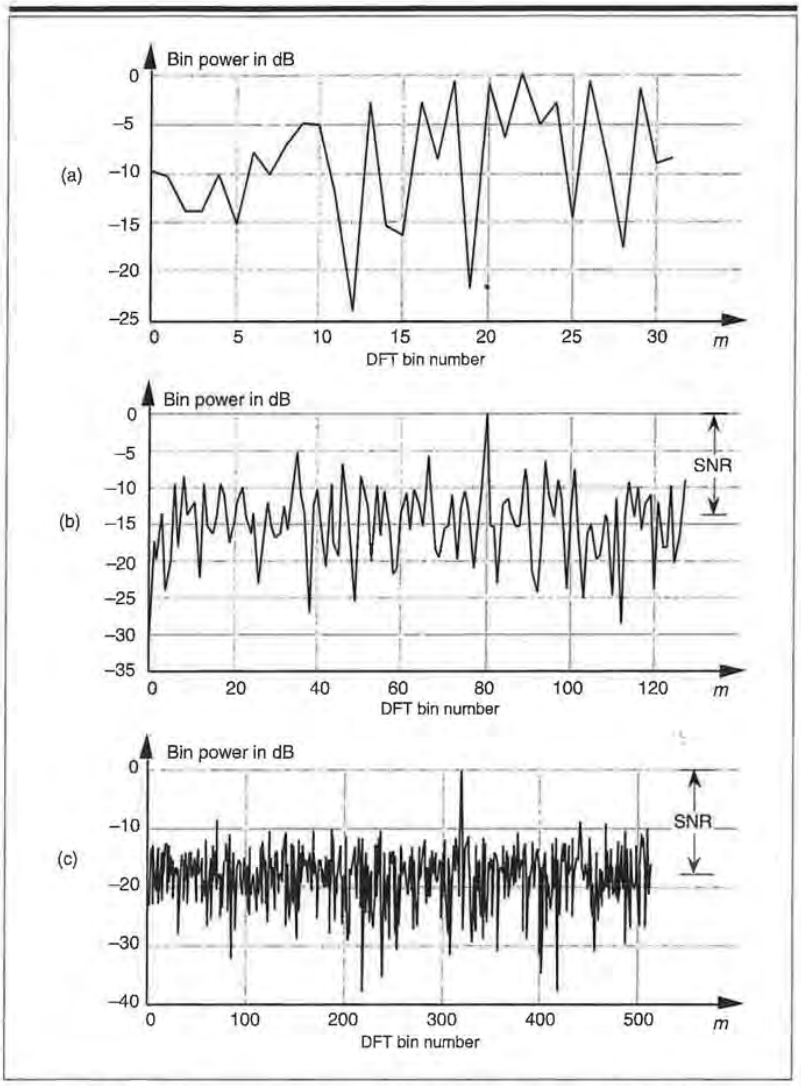


Figure 3-22 Single DFT processing gain: (a) $N = 64$; (b) $N = 256$; (c) $N = 1024$.

useful in energy detection because the frequency resolution improves in addition to the filter's ability to minimize the amount of background noise that resides within its passband. We can demonstrate this by looking at the DFT of a spectral tone (a constant frequency sinewave) added

to random noise. Figure 3-22(a) is a logarithmic plot showing the first 32 outputs of a 64-point DFT when the input tone is at the center of the DFT's $m = 20$th bin. The output power levels (DFT magnitude squared) in Figure 3-22(a) are normalized so that the highest bin output power is set to 0 dB. Because the tone's original signal power is below the average noise power level, the tone is a bit difficult to detect when $N = 64$. (The time-domain noise, used to generate Figure 3-22(a), has an average value of zero, i.e., no DC bias or amplitude offset.) If we quadruple the number of input samples and increase the DFT size to $N = 256$, we can now see the tone power raised above the average background noise power as shown for $m = 80$ in Figure 3-22(b). Increasing the DFT's size to $N = 1024$ provides additional processing gain to pull the tone further up out of the noise as shown in Figure 3-22(c).

To quantify the idea of DFT processing gain, we can define a signal-to-noise ratio (SNR) as the DFT's *output signal-power level* over the *average output noise-power level*. (In practice, of course, we like to have this ratio as large as possible.) For several reasons, it's hard to say what any given single DFT output SNR will be. That's because we can't exactly predict the energy in any given N samples of random noise. Also, if the input signal frequency is not at bin center, leakage will raise the effective background noise and reduce the DFT's output SNR. In addition, any window being used will have some effect on the leakage and, thus, on the output SNR. What we'll see is that the DFT's output SNR increases as N gets larger because a DFT bin's output noise standard deviation (*rms*) value is proportional to $\sqrt{N}$, and the DFT's output magnitude for the bin containing the signal tone is proportional to N. More generally for real inputs, if $N > N'$, an N-point DFT's output SNR_N increases over the N'-point DFT $SNR_{N'}$ by the following relationship:

$$SNR_N = SNR_{N'} + 20 \cdot \log_{10}\left(\frac{N/N'}{\sqrt{N/N'}}\right). \tag{3-33}$$

If we increase a DFT's size from N' to $N = 2N'$, from Eq. (3-33), the DFT's output SNR increases by 3 dB. So we say that a DFT's processing gain increases by 3 dB whenever N is doubled. Be aware that we may double a DFT's size and get a resultant processing gain of less than 3 dB in the presence of random noise; then again, we may gain slightly more than 3 dB. That's the nature of random noise. If we perform many DFTs, we'll see an average processing gain, shown in Figure 3-23(a), for various input signal SNRs. Because we're interested in the slope of the curves in Figure

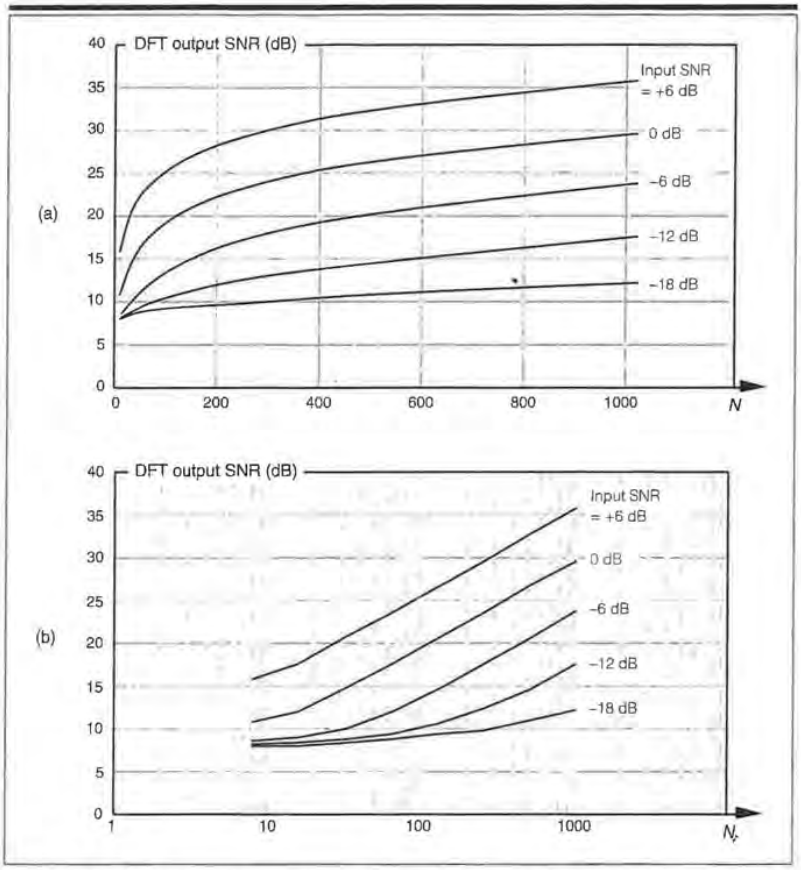


Figure 3-23 DFT processing gain vs. number of DFT points N for various input signal-to-noise ratios: (a) linear N axis; (b) logarithmic N axis.

3-23(a), we plot those curves on a logarithmic scale for N in Figure 3-23(b) where the curves straighten out and become linear. Looking at the slope of the curves in Figure 3-23(b), we can now see the 3 dB increase in processing gain as N doubles so long as N is greater than 20 or 30 and the signal is not overwhelmed by noise. There's nothing sacred about the absolute values of the curves in Figures 3-23(a) and 3-23(b). They were generated through a simulation of noise and a tone whose frequency was at a DFT bin center. Had the tone's frequency been between bin centers, the processing gain curves would have been shifted downward, but their

shapes would still be the same,[†] that is, Eq. (3-33) is still valid regardless of the input tone's frequency.

3.12.2 Integration Gain Due to Averaging Multiple DFTs

Theoretically, we could get very large DFT processing gains by increasing the DFT size arbitrarily. The problem is that the number of necessary DFT multiplications increases proportionally to N^2, and larger DFTs become very computationally intensive. Because addition is easier and faster to perform than multiplication, we can average the outputs of multiple DFTs to obtain further processing gain and signal detection sensitivity. The subject of averaging multiple DFT outputs is covered in Section 8.3.

3.13 The DFT of Rectangular Functions

We conclude this chapter by providing the mathematical details of two important aspects of the DFT. First, we obtain the expressions for the DFT of a rectangular function (rectangular window), and then we'll use these results to illustrate the magnitude response of the DFT. We're interested in the DFT's magnitude response because it provides an alternate viewpoint to understand the leakage that occurs when we use the DFT as a signal analysis tool.

One of the most prevalent and important computations encountered in digital signal processing is the DFT of a rectangular function. We see it in sampling theory, window functions, discussions of convolution, spectral analysis, and in the design of digital filters. As common as it is, however, the literature covering the DFT of rectangular functions can be confusing for several reasons for the digital signal processing beginner. The standard mathematical notation is a bit hard to follow at first, and sometimes the equations are presented with too little explanation. Compounding the problem, for the beginner, are the various expressions of this particular DFT. In the literature, we're likely to find any one of the following forms for the DFT of a rectangular function:

$$\mathrm{DFT}_{\mathrm{rect.function}} = \frac{\sin(x)}{\sin(x/N)}, \text{ or } \frac{\sin(x)}{x}, \text{ or } \frac{\sin(Nx/2)}{\sin(x/2)}. \tag{3-34}$$

[†] The curves would be shifted downward, indicating a lower SNR, because leakage would raise the average noise-power level, and scalloping loss would reduce the DFT bin's output power level.

In this section we'll show how the forms in Eq. (3-34) were obtained, see how they're related, and create a kind of *Rosetta stone* table allowing us to move back and forth between the various DFT expressions. Take a deep breath and let's begin our discussion with the definition of a rectangular function.

3.13.1 DFT of a General Rectangular Function

A general rectangular function $x(n)$ can be defined as N samples containing K unity-valued samples as shown in Figure 3-24. The full N-point sequence, $x(n)$, is the rectangular function that we want to transform. We call this the general form of a rectangular function because the K unity samples begin at a arbitrary index value of $-n_o$. Let's take the DFT of $x(n)$ in Figure 3-24 to get our desired $X(m)$. Using m as our frequency-domain sample index, the expression for an N-point DFT is

$$X(m) = \sum_{n=-(N/2)+1}^{N/2} x(n)e^{-j2\pi nm/N} \quad (3\text{-}35)$$

With $x(n)$ being nonzero only over the range of $-n_o \le n \le -n_o+(K-1)$, we can modify the summation limits of Eq. (3-35) to express $X(m)$ as

$$X(m) = \sum_{n=-n_o}^{-n_o+(K-1)} 1 \cdot e^{-j2\pi nm/N} \quad (3\text{-}36)$$

because only the K samples contribute to $X(m)$. That last step is important because it allows us to eliminate the $x(n)$ terms and make Eq. (3-36) easier to handle. To keep the following equations from being too messy, let's use the dummy variable $q = 2\pi m/N$.

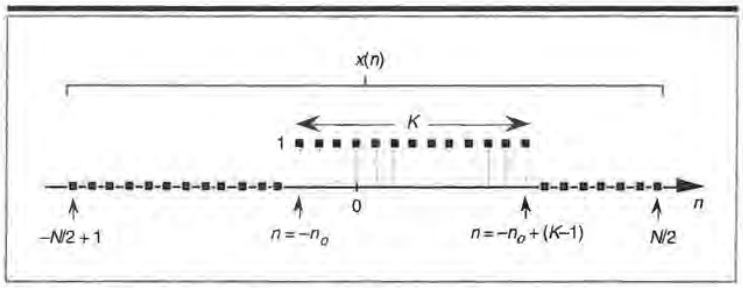


Figure 3-24 Rectangular function of width K samples defined over N samples where $K < N$.

OK, here's where the algebra comes in. Over our new limits of summation, we eliminate the factor of 1 and Eq. (3-36) becomes

$$X(q) = \sum_{n=-n_o}^{-n_o+(K-1)} e^{-jqn}$$

$$= e^{-jq(-n_o)} + e^{-jq(-n_o+1)} + e^{-jq(-n_o+2)} + \ldots + e^{-jq(-n_o+(K-1))}$$

$$= e^{-jq(-n_o)}e^{-j0q} + e^{-jq(-n_o)}e^{-j1q} + e^{-jq(-n_o)}e^{-j2q} + \ldots + e^{-jq(-n_o)}e^{-jq(K-1)}$$

$$= e^{jq(n_o)} \cdot \left[e^{-j0q} + e^{-j1q} + e^{-j2q} + \ldots + e^{jq(K-1)}\right] . \quad (3\text{-}37)$$

The series inside the brackets of Eq. (3-37), allows the use of a summation, such as

$$X(q) = e^{jq(n_o)} \sum_{p=0}^{K-1} e^{-jpq} . \quad (3\text{-}38)$$

Equation (3-38) certainly doesn't look any simpler than Eq. (3-36), but it is. Equation (3-38) is a *geometric series* and, from the discussion in Appendix B, it can be evaluated to the closed form of

$$\sum_{p=0}^{K-1} e^{-jpq} = \frac{1-e^{-jqK}}{1-e^{-jq}} . \quad (3\text{-}39)$$

We can now simplify Eq. (3-39)—here's the clever step. If we multiply and divide the numerator and denominator of Eq. (3-39)'s right-hand side by the appropriate half-angled exponentials, we break the exponentials into two parts and get

$$\sum_{p=0}^{K-1} e^{-jpq} = \frac{e^{-jqK/2}(e^{jqK/2} - e^{-jqK/2})}{e^{-jq/2}(e^{jq/2} - e^{-jq/2})}$$

$$= e^{-jq(K-1)/2} \cdot \frac{(e^{jqK/2} - e^{-jqK/2})}{(e^{jq/2} - e^{-jq/2})} . \quad (3\text{-}40)$$

Let's pause for a moment here to remind ourselves where we're going. We're trying to get Eq. (3-40) into a usable form because it's part of Eq. (3-38) that we're using to evaluate $X(m)$ in Eq. (3-36) in our quest for an understandable expression for the DFT of a rectangular function.

Equation (3-40) looks even more complicated than Eq. (3-39), but things can be simplified inside the parentheses. From Euler's equation: $\sin(\varnothing) = (e^{j\varnothing} - e^{-j\varnothing})/2j$, Eq. (3-40) becomes

$$\sum_{p=0}^{K-1} e^{-jpq} = e^{-jq(K-1)/2} \cdot \frac{2j\sin(qK/2)}{2j\sin(q/2)}$$

$$= e^{-jq(K-1)/2} \cdot \frac{\sin(qK/2)}{\sin(q/2)}. \tag{3-41}$$

Substituting Eq. (3-41) for the summation in Eq. (3-38), our expression for $X(q)$ becomes

$$X(q) = e^{jq(n_0)} \cdot e^{-jq(K-1)/2} \cdot \frac{\sin(qK/2)}{\sin(q/2)}$$

$$= e^{jq(n_0-(K-1)/2)} \cdot \frac{\sin(qK/2)}{\sin(q/2)}. \tag{3-42}$$

Returning our dummy variable q to its original value of $2\pi m/N$,

$$X(m) = e^{j(2\pi m/N)(n_0-(K-1)/2)} \cdot \frac{\sin(2\pi mK/2N)}{\sin(2\pi m/2N)}, \text{ or}$$

General form of the Dirichlet kernel: →

$$X(m) = e^{j(2\pi m/N)(n_0-(K-1)/2)} \cdot \frac{\sin(\pi mK/N)}{\sin(\pi m/N)}. \tag{3-43}$$

So there it is (Whew!). Equation (3-43) is the general expression for the DFT of the rectangular function as shown in Figure 3-24. Our $X(m)$ is a

complex expression (pun intended) where a ratio of sine terms is the amplitude of $X(m)$ and the exponential term is the phase angle of $X(m)$.[†] The ratio of sines factor in Eq. (3-43) lies on the periodic curve shown in Figure 3-25(a), and like all N-point DFT representations, the periodicity of $X(m)$ is N. This curve is known as the *Dirichlet kernel* (or the *aliased sinc function*) and has been thoroughly described in the literature[10,13,14]. (It's named after the nineteenth-century German mathematician Peter Dirichlet (pronounced dee-ree-'klay), who studied the convergence of trigonometric series used to represent arbitrary functions.)

We can zoom in on the curve at the $m = 0$ point and see more detail in Figure 3-25(b). The dots are shown in Figure 3-25(b) to remind us that the DFT of our rectangular function results in discrete amplitude values that lie on the curve. So when we perform DFTs, our discrete results are sampled values of the continuous sinc function's curve in Figure 3-25(a). As we'll show later, we're primarily interested in the absolute value, or magnitude, of the Dirichlet kernel in Eq. (3-43). That magnitude $|X(m)|$ is shown in Figure 3-25(c). Although we first saw the sinc function's curve in Figure 3-9 in Section 3.8, where we introduced the topic of DFT leakage, we'll encounter this curve often in our study of digital signal processing.

For now, there are just a few things we need to keep in mind concerning the Dirichlet kernel. First, the DFT of a rectangular function has a main lobe, centered about the $m = 0$ point. The peak amplitude of the main lobe is K. This peak value makes sense, right? The $m = 0$ sample of a DFT $X(0)$ is the sum of the original samples, and the sum of K unity-valued samples is K. We can show this in a more substantial way by evaluating Eq. (3-43) for $m = 0$. A difficulty arises when we plug $m = 0$ into Eq. (3-43) because we end up with sin(0)/sin(0), which is the indeterminate ratio 0/0. Well, hardcore mathematics to the rescue here. We can use L'Hospital's rule to take the derivative of the numerator and the denominator of Eq. (3-43), and *then* set $m = 0$ to determine the peak value of the magnitude of the Dirichlet kernel.[††] We proceed as

---

[†] N was an even number in Figure 3-24 depicting the $x(n)$. Had N been an odd number, the limits on the summation in Eq. (3-35) would have been $-(N-1)/2 \le n \le (N-1)/2$. Using these alternate limits would have led us to exactly the same $X(m)$ as in Eq. (3-43).

[††] L'Hospital is pronounced 'lō-pē-tōl, like baby doll.

$$|X(m)|_{m\to 0} = \frac{d}{dm}X(m) = \frac{d[\sin(\pi mK/N)]/dm}{d[\sin(\pi m/N)]/dm}$$

$$= \frac{\cos(\pi mK/N)}{\cos(\pi m/N)} \cdot \frac{d(\pi mK/N)/dm}{d(\pi m/N)/dm}$$

$$= \frac{\cos(0)}{\cos(0)} \cdot \frac{\pi K/N}{\pi/N} = 1 \cdot K = K \ , \qquad (3\text{-}44)$$

which is what we set out to show. (We could have been clever and evaluated Eq. (3-35) with $m = 0$ to get the result of Eq. (3-44). Try it, and keep in mind that $e^{j0} = 1$.) Had the amplitudes of the nonzero samples of $x(n)$ been

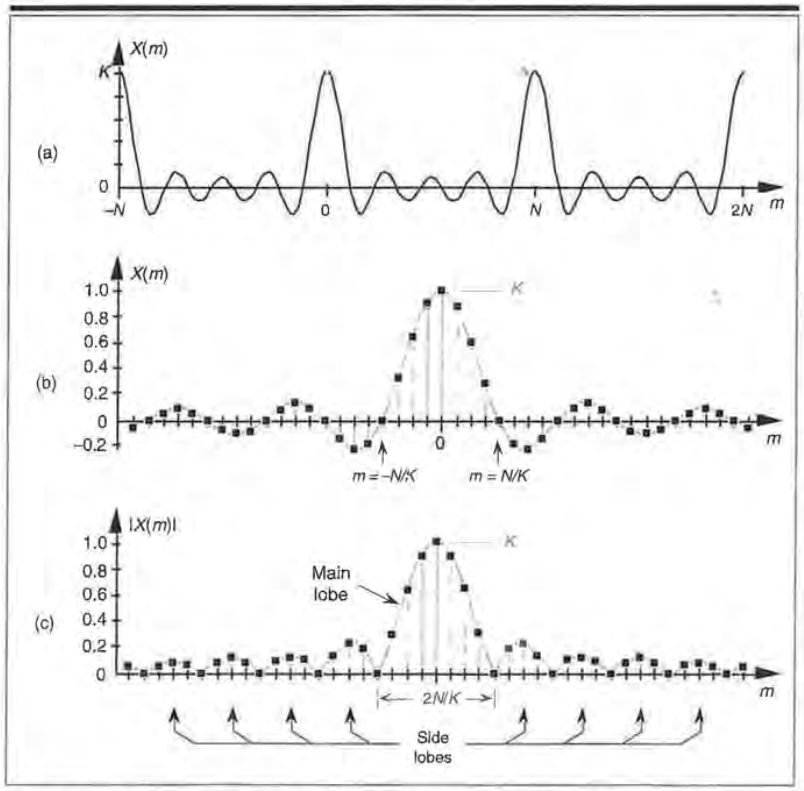


Figure 3-25 The Dirichlet kernel of $X(m)$: (a) periodic continuous curve on which the $X(m)$ samples lie; (b) $X(m)$ amplitudes about the $m = 0$ sample; (c) $|X(m)|$ magnitudes about the $m = 0$ sample.

different than unity, say some amplitude A_o, then, of course, the peak value of the Dirichlet kernel would be A_oK instead of just K. The next important thing to notice about the Dirichlet kernel is the main lobe's width. The first zero crossing of Eq. (3-43) occurs when the numerator's argument is equal to π. That is, when $\pi mK/N = \pi$. So the value of m at the first zero crossing is given by

$$m_{\text{first zero crossing}} = \frac{\pi N}{\pi K} = \frac{N}{K} \quad . \tag{3-45}$$

as shown in Figure 3-25(b). Thus the main lobe width $2N/K$, as shown in Figure 3-25(c), is inversely proportional to K.†

Notice that the main lobe in Figure 3-25(a) is surrounded by a series of oscillations, called *sidelobes*, as in Figure 3-25(c). These sidelobe magnitudes decrease the farther they're separated from the main lobe. However, no matter how far we look away from the main lobe, these sidelobes never reach zero magnitude—and they cause a great deal of heartache for practitioners in digital signal processing. These sidelobes cause high-amplitude signals to overwhelm and hide neighboring low-amplitude signals in spectral analysis, and they complicate the design of digital filters. As we'll see in Chapter 5, the unwanted ripple in the passband and the poor stopband attenuation in simple digital filters are caused by the rectangular function's DFT sidelobes. (The development, analysis, and application of window functions came about to minimize the ill effects of those sidelobes in Figure 3-25.)

Let's demonstrate the relationship in Eq. (3-45) by way of a simple but concrete example. Assume that we're taking a 64-point DFT of the 64-sample rectangular function, with eleven unity values, shown in Figure 3-26(a). In this example, $N = 64$ and $K = 11$. Taking the 64-point DFT of the sequence in Figure 3-26(a) results in an $X(m)$ whose real and imaginary parts, $X_{\text{real}}(m)$ and $X_{\text{imag}}(m)$, are shown in Figure 3-26(b) and Figure 3-26(c) respectively. Figure 3-26(b) is a good illustration of how the real part of the DFT of a real input sequence has even symmetry, and Figure 3-26(c) confirms that the imaginary part of the DFT of a real input sequence has odd symmetry. (These symmetry properties were introduced in Section 3.2 and are discussed further in Appendix C, Section C.3.)

† This is a fundamental characteristic of Fourier transforms. The narrower the function in one domain, the wider its transform will be in the other domain.

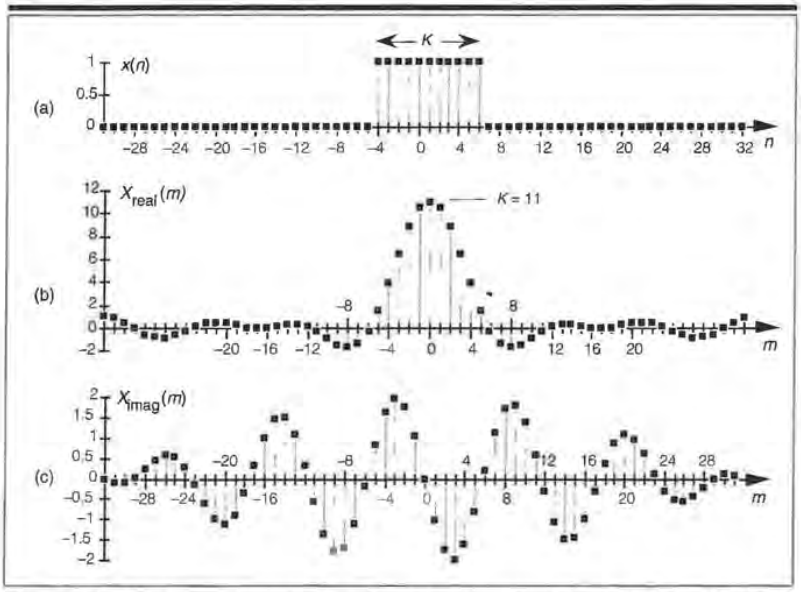


Figure 3-26 DFT of a rectangular function: (a) original function $x(n)$; (b) real part of the DFT of $x(n)$, $X_{real}(m)$; (c) imaginary part of the DFT of $x(n)$, $X_{imag}(m)$.

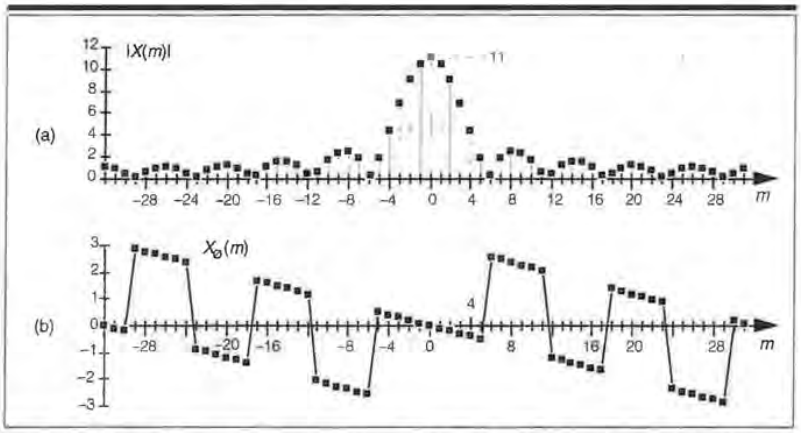


Figure 3-27 DFT of a generalized rectangular function: (a) magnitude $|X(m)|$; (b) phase angle in radians.

Although $X_{real}(m)$ and $X_{imag}(m)$ tell us everything there is to know about the DFT of $x(n)$, it's a bit easier to comprehend the true spectral nature of $X(m)$ by viewing its absolute magnitude. This magnitude, from Eq. (3-7), is provided in Figure 3-27(a) where the main and sidelobes are

clearly evident now. As we expected, the peak value of the main lobe is 11 because we had $K = 11$ samples in $x(n)$. The width of the main lobe from Eq. (3-45) is 64/11, or 5.82. Thus, the first positive frequency zero-crossing location lies just below the $m = 6$ sample of our discrete $|X(m)|$ represented by the squares in Figure 3-27(a). The phase angles associated with $|X(m)|$, first introduced in Equations (3-6) and (3-8), are shown in Figure 3-27(b).

To understand the nature of the DFT of rectangular functions more fully, let's discuss a few more examples using less general rectangular functions that are more common in digital signal processing than the $x(n)$ in Figure 3-24.

3.13.2 DFT of a Symmetrical Rectangular Function

Equation (3-43) is a bit complicated because our original function $x(n)$ was so general. In practice, special cases of rectangular functions lead to simpler versions of Eq. (3-43). Consider the symmetrical $x(n)$ rectangular function in Figure 3-28. As shown in Figure 3-28, we often need to determine the DFT of a rectangular function that's centered about the $n = 0$ index point. In this case, the K unity-valued samples begin at $n = -n_o = -(K-1)/2$. So substituting $(K-1)/2$ for n_o in Eq. (3-43), yields

$$X(m) = e^{j(2\pi m/N)((K-1)/2-(K-1)/2)} \cdot \frac{\sin(\pi mK/N)}{\sin(\pi m/N)}$$

$$= e^{j(2\pi m/N)(0)} \cdot \frac{\sin(\pi mK/N)}{\sin(\pi m/N)} \quad (3\text{-}46)$$

Because $e^{j0} = 1$, Eq. (3-46) becomes

Symmetrical form of the Dirichlet kernel: →

$$X(m) = \frac{\sin(\pi mK/N)}{\sin(\pi m/N)}. \quad (3\text{-}47)$$

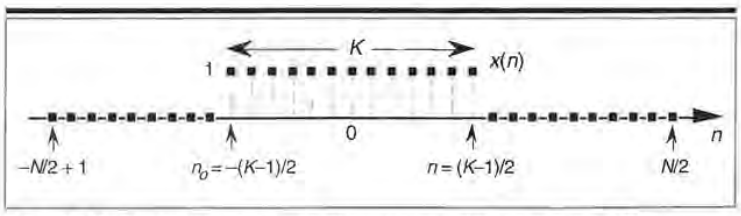


Figure 3-28 Rectangular $x(n)$ with K samples centered about $n = 0$.

Equation (3-47) indicates that the DFT of the symmetrical rectangular function in Figure 3-28 is itself a real function; that is, there's no complex exponential in Eq. (3-47), so this particular DFT contains no imaginary part or phase term. As we stated in Section 3.2, if $x(n)$ is real and even, $x(n) = x(-n)$, then $X_{real}(m)$ is nonzero and $X_{imag}(m)$ is always zero. We demonstrate this by taking the 64-point DFT of the sequence in Figure 3-29(a). Our $x(n)$ is 11 unity-valued samples centered about the $n = 0$ index. Here the DFT results in an $X(m)$ whose real and imaginary parts are shown in Figure 3-29(b) and Figure 3-29(c), respectively. As Eq. (3-47) predicted,

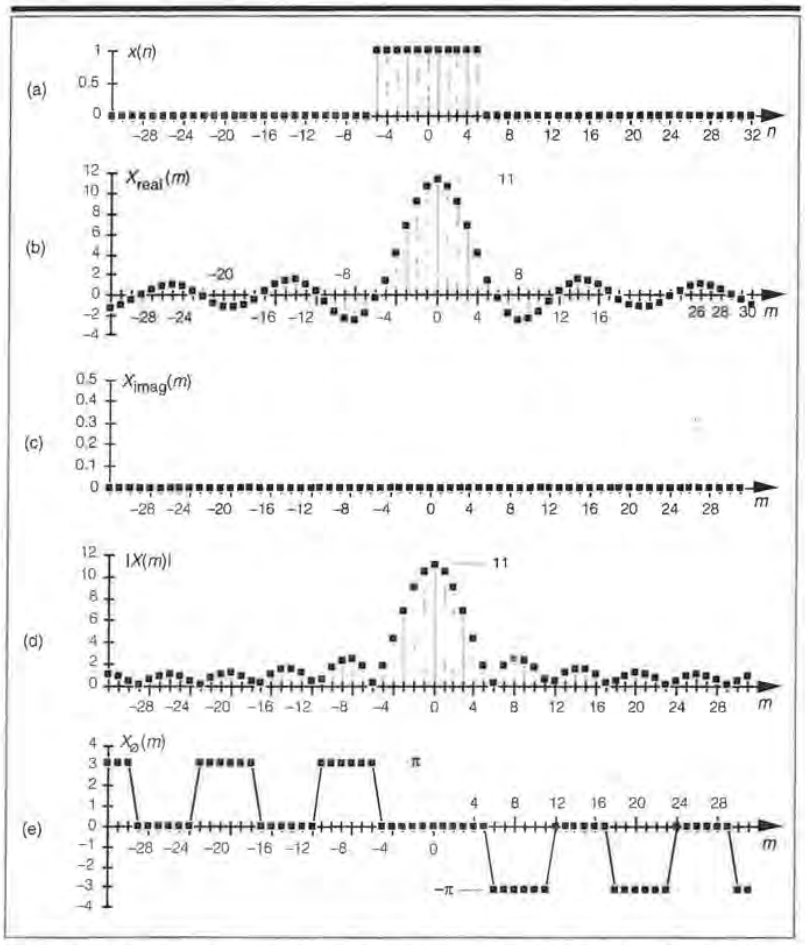


Figure 3-29 DFT of a rectangular function centered about $n = 0$: (a) original $x(n)$; (b) $X_{real}(m)$; (c) $X_{imag}(m)$; (d) magnitude of $X(m)$; (e) phase angle of $X(m)$ in radians.

$X_{real}(m)$ is nonzero and $X_{imag}(m)$ is zero. The magnitude and phase of $X(m)$ are depicted in Figure 3-29(d) and Figure 3-29(e).

Notice that the magnitudes in Figure 3-27(a) and Figure 3-29(d) are identical. This verifies the very important shifting theorem of the DFT; that is, the magnitude $|X(m)|$ depends only on the number of nonzero samples in $x(n)$, K, and *not* on their position relative to the $n = 0$ index value. Shifting the K unity-valued samples to center them about the $n = 0$ index merely affects the phase angle of $X(m)$, not its magnitude.

Speaking of phase angles, it's interesting to realize here that even though $X_{imag}(m)$ is zero in Figure 3-29(c), the phase angle of $X(m)$ is not always zero. In this case, $X(m)$'s individual phase angles in Figure 3-29(e) are either $+\pi$, zero, or $-\pi$ radians. With $+\pi$ and $-\pi$ radians both being equal to -1, we could easily reconstruct $X_{real}(m)$ from $|X(m)|$ and the phase angle $X_{\phi}(m)$ if we must. $X_{real}(m)$ is equal to $|X(m)|$ with the signs of $|X(m)|$'s alternate sidelobes reversed.† To gain some further appreciation of how the DFT of a rectangular function is a sampled version of the Dirichlet kernel, let's increase the number of our nonzero $x(n)$ samples. Figure 3-30(a) shows a 64-point $x(n)$ where 31 unity-valued samples are centered about the $n = 0$ index location. The magnitude of $X(m)$ is provided in Figure 3-30(b). By broadening the $x(n)$ function, i.e., increasing K, we've narrowed the Dirichlet kernel of $X(m)$. This follows from Eq. (3-45), right? The kernel's first zero crossing is inversely proportional to K, so, as we extend the width of K, we squeeze $|X(m)|$ in toward $m = 0$. In this

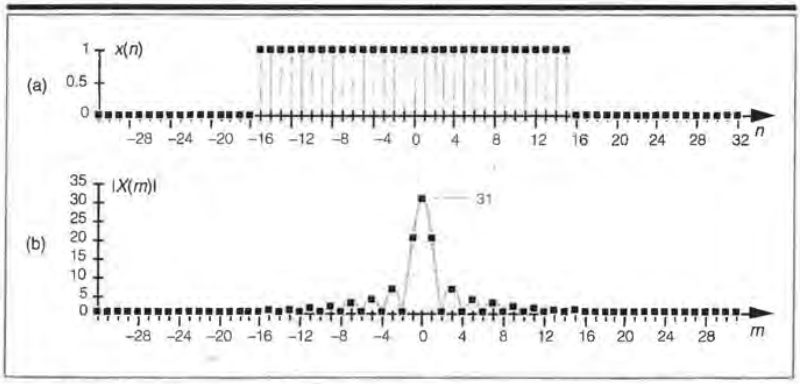


Figure 3-30 DFT of a symmetrical rectangular function with 31 unity values: (a) original $x(n)$; (b) magnitude of $X(m)$.

† The particular pattern of $+\pi$ and $-\pi$ values in Figure 3-29(e) is an artifact of the software used to generate that figure. A different software package may show a different pattern, but as long as the nonzero phase samples are either $+\pi$ or $-\pi$, the phase results will be correct.

example, $N = 64$ and $K = 31$. From Eq. (3-45) the first positive zero crossing of $X(m)$ occurs at $64/31$, or just slightly to the right of the $m = 2$ sample in Figure 3-30(b). Also notice that the peak value of $|X(m)| = K = 31$, as mandated by Eq. (3-44).

3.13.3 DFT of an All Ones Rectangular Function

The DFT of a special form of $x(n)$ is routinely called for, leading to yet another simplified form of Eq. (3-43). In the literature, we often encounter a rectangular function where $K = N$; that is, all N samples of $x(n)$ are nonzero, as shown in Figure 3-31. In this case, the N unity-valued samples begin at $n = -n_o = -(N-1)/2$. We obtain the expression for the DFT of the function in Figure 3-31 by substituting $K = N$ and $n_o = (N-1)/2$ in Eq. (3-43) to get

$$X(m) = e^{j(2\pi m/N)[(N-1)/2-(N-1)/2]} \cdot \frac{\sin(\pi mN/N)}{\sin(\pi m/N)}$$

$$= e^{j(2\pi m/N)(0)} \cdot \frac{\sin(\pi m)}{\sin(\pi m/N)}, \text{ or}$$

All Ones form of the Dirichlet kernel (Type 1): →

$$X(m) = \frac{\sin(\pi m)}{\sin(\pi m/N)}. \tag{3-48}$$

Equation (3-48) takes the first form of Eq. (3-34) that we alluded to at the beginning of this Section 3.13.† Figure 3-32 demonstrates the meaning of Eq. (3-48). The DFT magnitude of the all ones function, $x(n)$ in

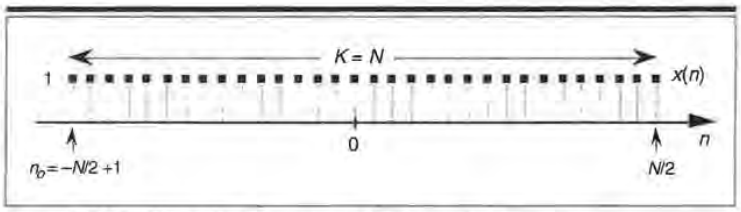


Figure 3-31 Rectangular function with N unity-valued samples.

† By the way, there's nothing *official* about calling Eq. (3-48) a Type 1 Dirichlet kernel. We're using the phrase *Type* 1 merely to distinguish Eq. (3-48) from other mathematical expressions for the Dirichlet kernel that we're about to encounter.

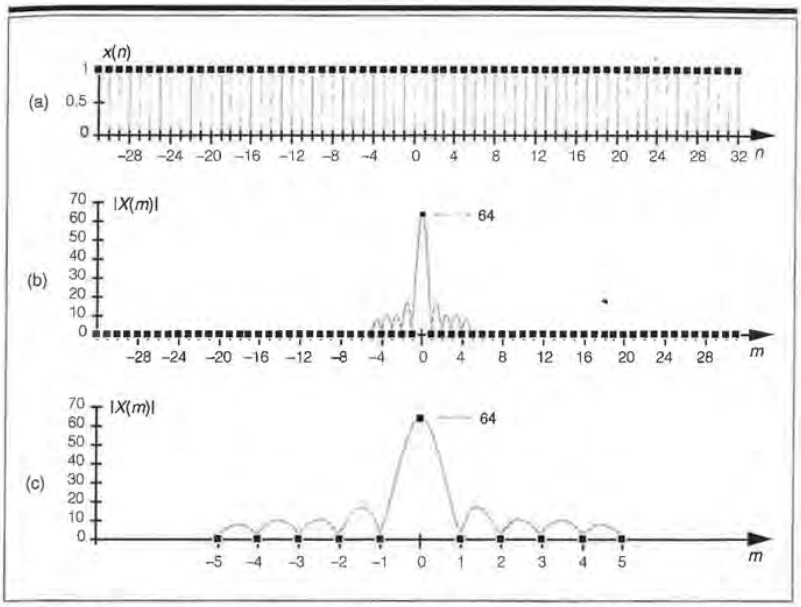


Figure 3-32 All ones function: (a) rectangular function with $N = 64$ unity-valued samples; (b) DFT magnitude of the all ones time function; (c) close-up view of the DFT magnitude of an all ones time function.

Figure 3-32(a), is shown in Figure 3-32(b) and Figure 3-32(c). Take note that if m was continuous Eq. (3-48) describes the shaded curves in Figure 3-32(b) and Figure 3-32(c). If m is restricted to be integers, then Eq. (3-48) represents the dots in those figures.

The Dirichlet kernel of $X(m)$ in Figure 3-32(b) is now as narrow as it can get. The main lobe's first positive zero crossing occurs at the $m = 64/64 = 1$ sample in Figure 3-32(b) and the peak value of $|X(m)| = N = 64$. With $x(n)$ being all ones, $|X(m)|$ is zero for all $m \neq 0$. The sinc function in Eq. (3-48) is of utmost importance—as we'll see at the end of this chapter, it defines the overall DFT frequency response to an input sinusoidal sequence, and it's also the amplitude response of a single DFT bin.

The form of Eq. (3-48) allows us to go one step further to identify the most common expression for the DFT of an all ones rectangular function found in the literature. To do this, we have to use an approximation principle found in the mathematics of trigonometry that you may have heard before. It states that when α is small, then $\sin(\alpha)$ is approximately equal to α, i.e., $\sin(\alpha) \approx \alpha$. This idea comes about when we consider a pie-shaped

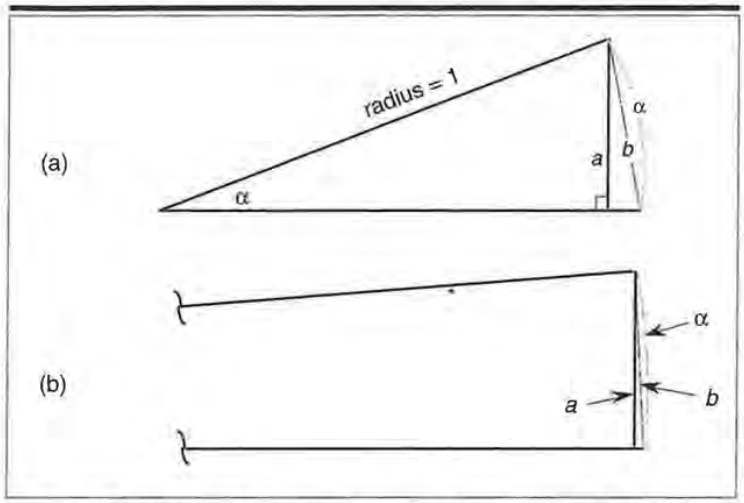


Figure 3-33 Relationships between an angle α, line $a = \sin(\alpha)$, and α's chord b: (a) large angle α; (b) small angle α.

section of a circle whose radius is 1 as shown in Figure 3-33(a). That section is defined by the length of the arc α measured in radians and α's chord b. If we draw a right triangle inside the section, we can say that $a = \sin(\alpha)$. As α gets smaller the long sides of our triangle become almost parallel, the length of chord b approaches the length of arc α, and the length of line a approaches the length of b. So, as depicted in Figure 3-33(b), when α is small, $\alpha \approx b \approx a = \sin(\alpha)$. We use this $\sin(\alpha) \approx \alpha$ approximation when we look at the denominator of Eq. (3-48). When $\pi m/N$ is small, then $\sin(\pi m/N)$ is approximately equal to $\pi m/N$. So we can, when N is large, state

All Ones form of the Dirichlet kernel (Type 2): →

$$X(m) \approx \frac{\sin(\pi m)}{\pi m / N} = N \cdot \frac{\sin(\pi m)}{\pi m}. \qquad (3\text{-}49)$$

It has been shown that when N is larger than, say, 10 in Eq. (3-48), Eq. (3-49) accurately describes the DFT's output.† Equation (3-49) is often normalized by dividing it by N, so we can express the normalized DFT of an all ones rectangular function as

† We can be comfortable with this result because, if we let $K = N$, we'll see that the peak value of $X(m)$ in Eq. (3-49), for $m = 0$, is equal to N, which agrees with Eq. (3-44).

All Ones form of the Dirichlet kernel (Type 3): →

$$X(m) \approx \frac{\sin(\pi m)}{\pi m}. \tag{3-50}$$

Equation (3-50), taking the second form of Eq. (3-34) that is so often seen in the literature, also has the DFT magnitude shown in Figure 3-32(b) and Figure 3-32(c).

3.13.4 Time and Frequency Axes Associated with Rectangular Functions

Let's establish the physical dimensions associated with the n and m index values. So far in our discussion, the n index was merely an integer enabling us to keep track of individual $x(n)$ sample values. If the n index represents instants in time, we can identify the time period separating adjacent $x(n)$ samples to establish the time scale for the $x(n)$ axis and the frequency scale for the $X(m)$ axis. Consider the time-domain rectangular function given in Figure 3-34(a). That function comprises N time samples obtained t_s seconds apart, and the full sample interval is Nt_s seconds. Each

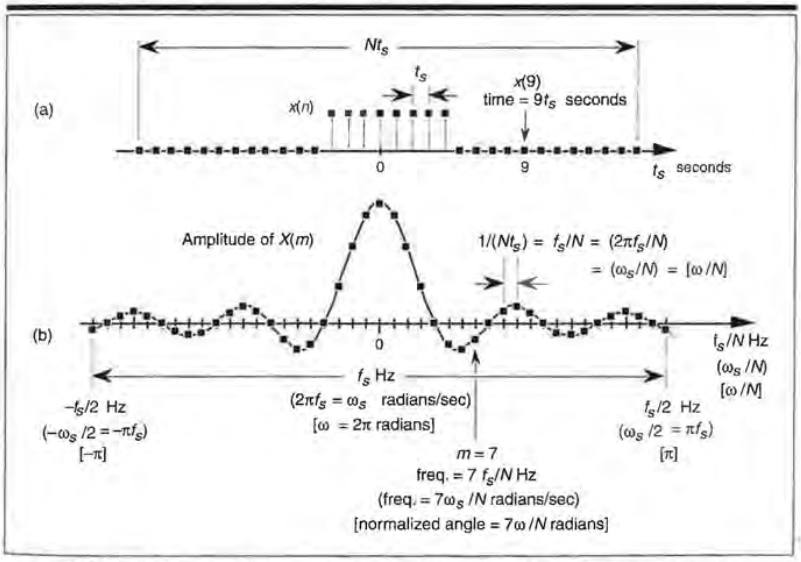


Figure 3-34 DFT time and frequency axis dimensions: (a) time-domain axis uses time index n; (b) various representations of the DFT's frequency axis.

Table 3-1 Characteristics of Various DFT Frequency Axis Representations

DFT Frequency Axis Representation	X(m) *Frequency Variable*	*Resolution of* X(m)	*Repetition Interval of* X(m)	*Frequency Axis Range*
Frequency in Hz	mf_s/N	f_s/N	f_s	$-f_s/2$ to $f_s/2$
Frequency in radians	$m\omega_s/N$ or $2\pi mf_s/N$	ω_s/N or $2\pi f_s/N$	ω_s or $2\pi f_s$	$-\omega_s/2$ to $\omega_s/2$ or $-\pi f_s$ to πf_s
Normalized angle in radians	$2\pi m/N$	$2\pi/N$	2π	$-\pi$ to π

$x(n)$ sample occurs at nt_s seconds for some value of n. For example, the $n = 9$ sample value, $x(9) = 0$, occurs at $9t_s$ seconds.

The frequency axis of $X(m)$ can be represented in a number of different ways. Three popular types of DFT frequency axis labeling are shown in Figure 3-34(b) and listed in Table 3-1. Let's consider each representation individually.

3.13.4.1 DFT Frequency Axis in Hz

If we decide to relate $X(m)$ to the time sample period t_s, or the sample rate $f_s = 1/t_s$, then the frequency axis variable is $m/Nt_s = mf_s/N$. So each $X(m)$ sample is associated with a cyclic frequency of mf_s/N Hz. In this case, the resolution of $X(m)$ is f_s/N. The DFT repetition period, or periodicity, is f_s Hz, as shown in Figure 3-34(b). If we substitute the cyclic frequency variable mf_s/N for the generic variable of m/N in Eq. (3-47), we obtain an expression for the DFT of a symmetrical rectangular function, where $K < N$, in terms of the sample rate f_s in Hz. That expression is

$$X(mf_s) = \frac{\sin(\pi mf_s K/N)}{\sin(\pi mf_s/N)}. \tag{3-51}$$

For an all ones rectangular function where $K = N$, the amplitude normalized $\sin(x)/x$ approximation in Eq. (3-50) can be rewritten in terms of sample rate f_s in Hz as

$$X(mf_s) \approx \frac{\sin(\pi mf_s)}{\pi mf_s}. \tag{3-52}$$

3.13.4.2 DFT Frequency Axis in Radians/Second

We can measure $X(m)$'s frequency in radians/s by defining the time-domain sample rate in radians/s as $\omega_s = 2\pi f_s$. So each $X(m)$ sample is associated with a radian frequency of $m\omega_s/N = 2\pi m f_s/N$ radians/s. In this case, $X(m)$'s resolution is $\omega_s/N = 2\pi f_s/N$ radians/s, and the DFT repetition period is $\omega_s = 2\pi f_s$ radians/s, as shown by the expressions in parenthesis in Figure 3-34(b). With $\omega_s = 2\pi f_s$, then $\pi f_s = \omega_s/2$. If we substitute $\omega_s/2$ for πf_s in Eq. (3-51), we obtain an expression for the DFT of a symmetrical rectangular function, where $K < N$, in terms of the sample rate ω_s in radians/s:

$$X(m\omega_s) = \frac{\sin(m\omega_s K/2N)}{\sin(m\omega_s/2N)}. \tag{3-53}$$

For an all ones rectangular function where $K = N$, the amplitude normalized $\sin(x)/x$ approximation in Eq. (3-50) can be stated in terms of a sample rate ω_s in radians/s as

$$X(m\omega_s) \approx \frac{2\sin(m\omega_s/2)}{m\omega_s}. \tag{3-54}$$

3.13.4.3 DFT Frequency Axis Using a Normalized Angle Variable

Many authors simplify the notation of their equations by using a normalized variable for the radian frequency $\omega_s = 2\pi f_s$. By normalized, we mean that the sample rate f_s is assumed to be equal to 1, and this establishes a normalized radian frequency ω_s equal to 2π. Thus, the frequency axis of $X(m)$ now becomes a normalized angle ω, and each $X(m)$ sample is associated with an angle of $m\omega/N$ radians. Using this convention, $X(m)$'s resolution is ω/N radians, and the DFT repetition period is $\omega = 2\pi$ radians, as shown by the expressions in brackets in Figure 3-34(b).

Unfortunately the usage of these three different representations of the DFT's frequency axis is sometimes confusing for the beginner. When reviewing the literature, the reader can learn to convert between these frequency axis notations by consulting Figure 3-34 and Table 3-1.

3.13.5 Alternate Form of the DFT of an All Ones Rectangular Function

Using the normalized radian angle notation for the DFT axis from the bottom row of Table 3-1 leads to another prevalent form of the DFT of the all

ones rectangular function in Figure 3-31. Letting our normalized discrete frequency axis variable be $\omega_m = 2\pi m/N$, then $\pi m = N\omega_m/2$. Substituting the term $N\omega_m/2$ for πm in Eq. (3-48), we obtain

All Ones form of the Dirichlet kernel (Type 4): →

$$X(\omega) = \frac{\sin(N\omega_m/2)}{\sin(\omega_m/2)}. \tag{3-55}$$

Table 3-2 DFT of Various Rectangular Functions

Description	*Expression*	
DFT of a general rectangular function, where $K < N$, in terms of the integral frequency m variable	$X(m) = \frac{\sin(\pi mK/N)}{\sin(\pi m/N)} \cdot e^{j(2\pi m/N)(n_o-(K-1)/2)}$	(3-46)
DFT of a symmetrical rectangular function, where $K < N$, in terms of the integral frequency variable m	$X(m) = \frac{\sin(\pi mK/N)}{\sin(\pi m/N)}$.	(3-47)
DFT of an all ones rectangular function in terms of the integer frequency variable m (Dirichlet kernel Type 1)	$X(m) = \frac{\sin(\pi m)}{\sin(\pi m/N)}$.	(3-48)
DFT of an all ones rectangular function in terms of the integer frequency variable m (Dirichlet kernel Type 2)	$X(m) = \frac{N\sin(\pi m)}{\pi m}$.	(3-49)
Amplitude normalized DFT of an all ones rectangular function in terms of the integral frequency variable m (Dirichlet kernel Type 3)	$X(m) = \frac{\sin(\pi m)}{\pi m}$.	(3-50)
DFT of a symmetrical rectangular function, where $K < N$, in terms of the sample rate f_s in Hz	$X(mf_s) = \frac{\sin(\pi mf_sK/N)}{\sin(\pi mf_s/N)}$.	(3-51)
Amplitude normalized DFT of an all ones rectangular function in terms of the sample rate f_s in Hz	$X(mf_s) = \frac{\sin(\pi mf_s)}{\pi mf_s}$.	(3-52)
DFT of a symmetrical rectangular function, where $K < N$, in terms of the sample rate ω_s in radians/s	$X(m\omega_s) = \frac{\sin(m\omega_sK/2N)}{\sin(m\omega_s/2N)}$.	(3-53)
Amplitude normalized DFT of an all ones rectangular function in terms of the sample rate ω_s in radians/s	$X(m\omega_s) = \frac{2\sin(m\omega_s/2)}{m\omega_s}$.	(3-54)
DFT of an all ones rectangular function in terms of the normalized discrete frequency variable ω_m (Dirichlet kernel Type 4).	$X(\omega) = \frac{\sin(N\omega_m/2)}{\sin(\omega_m/2)}$.	(3-55)

Equation (3-55), taking the third form of Eq. (3-34) sometimes seen in the literature, also has the DFT magnitude shown in Figure 3-32(b) and Figure 3-32(c).

We've covered so many different expressions for the DFT of various rectangular functions that it's reasonable to compile their various forms in Table 3-2.

3.13.6 Inverse DFT of a General Rectangular Function

Let's think now about computing the inverse DFT of a rectangular frequency-domain function; that is, given a rectangular $X(m)$ function, find a time-domain function $x(n)$. We can define the general form of a rectangular frequency-domain function, as we did for Figure 3-24, to be that shown in Figure 3-35. The inverse DFT of the rectangular function $X(m)$ in Figure 3-35 is

$$x(n)=\frac{1}{N}\sum_{m=-(N/2)+1}^{N/2}X(m)e^{j2\pi mn/N}. \tag{3-56}$$

The same algebraic acrobatics we used to arrive at Eq. (3-43) can be applied to Eq. (3-56), guiding us to

$$x(n)=e^{-j(2\pi n/N)(m_o-(K-1)/2)}\cdot\frac{1}{N}\cdot\frac{\sin(\pi nK/N)}{\sin(\pi n/N)}, \tag{3-57}$$

for the inverse DFT of the rectangular function in Figure 3-35. The descriptions we gave for Eq. (3-43) apply equally well to Eq. (3-57) with the exception of a $1/N$ scale factor term and a change in the sign of Eq. (3-43)'s phase angle. Let's use Eq. (3-57) to take a 64-point inverse DFT of the 64-sample rectangular function shown in Figure 3-36(a). The

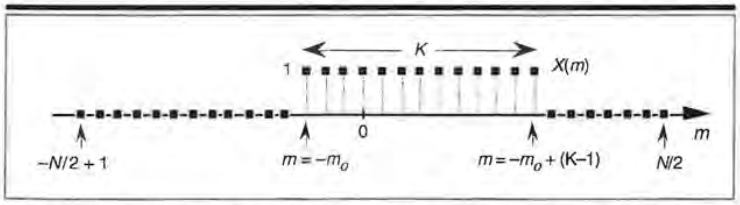


Figure 3-35 General rectangular frequency-domain function of width K samples defined over N samples where $K < N$.

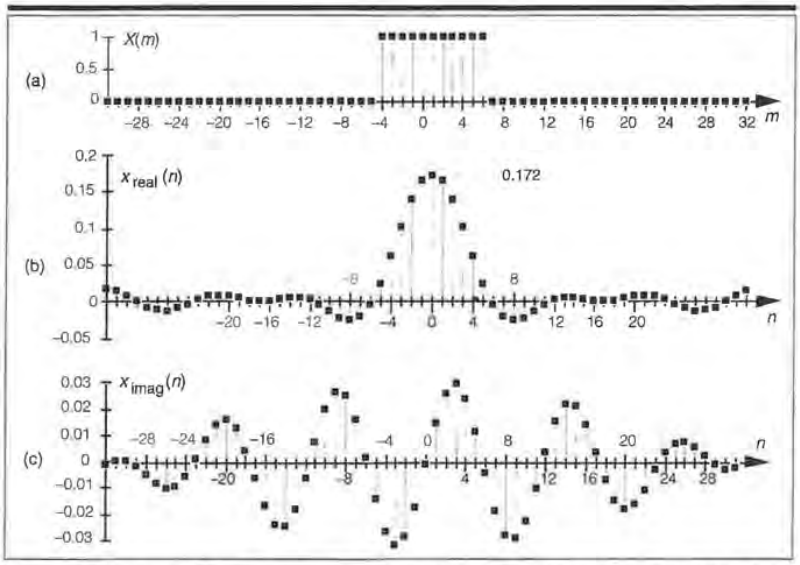


Figure 3-36 Inverse DFT of a general rectangular function: (a) original function $X(m)$; (b) real part, $x_{real}(n)$; (c) imaginary part, $x_{imag}(n)$.

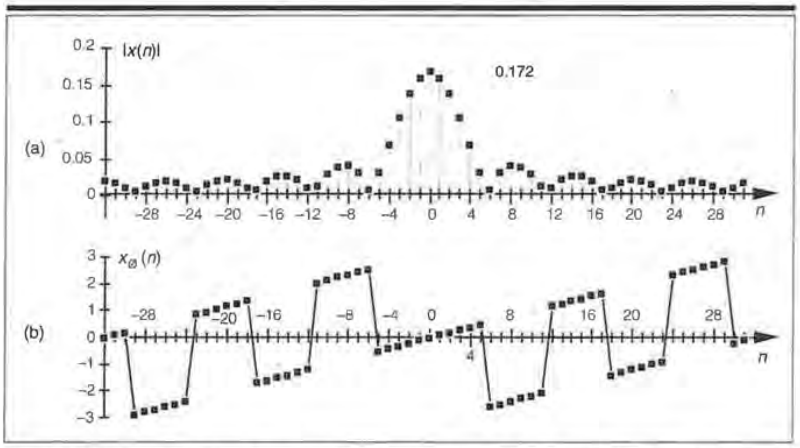


Figure 3-37 Inverse DFT of a generalized rectangular function: (a) magnitude $|x(n)|$; (b) phase angle of $x(n)$ in radians.

inverse DFT of the sequence in Figure 3-36(a) results in an $x(n)$ whose real and imaginary parts, $x_{real}(n)$ and $x_{imag}(n)$, are shown in Figure 3-36(b) and Figure 3-36(c), respectively. With $N = 64$ and $K = 11$ in this example, we've made the inverse DFT functions in Figure 3-36 easy to compare

with those *forward* DFT functions in Figure 3-26. See the similarity between the real parts, $X_{real}(m)$ and $x_{real}(n)$, in Figure 3-26(b) and Figure 3-36(b)? Also notice the sign change between the imaginary parts in Figure 3-26(c) and Figure 3-36(c).

The magnitude and phase angle of $x(n)$ are shown in Figure 3-37(a) and Figure 3-37(b). Note the differences in main lobe peak magnitude between Figure 3-27(a) and Figure 3-37(a). The peak magnitude value in Figure 3-37(a) is $K/N = 11/64$, or 0.172. Also notice the sign change between the phase angles in Figure 3-27(b) and Figure 3-37(b). The illustrations in Figures 3-26, 3-27, 3-36, and 3-37 are good examples of the fundamental duality relationships between the forward and inverse DFT.

3.13.7 Inverse DFT of a Symmetrical Rectangular Function

The inverse DFT of the general rectangular function in Figure 3-36 is not very common in digital signal processing. However, in discussions concerning digital filters, we will encounter the inverse DFT of symmetrical rectangular functions. This inverse DFT process is found in the window design method of what are called low-pass finite impulse response (FIR) digital filters. That method begins with the definition of a symmetrical frequency function, $H(m)$, such as that in Figure 3-38. The inverse DFT of the frequency samples in Figure 3-38 is then taken to determine the time-domain coefficients for use in a low-pass FIR filter. (Time-Domain FIR filter coefficients are typically denoted $h(n)$ instead of $x(n)$, so we'll use $h(n)$ throughout the remainder of this inverse DFT discussion.)

In the case of the frequency domain $H(m)$ in Figure 3-38, the K unity-valued samples begin at $m = -m_o = -(K-1)/2$. So plugging $(K-1)/2$ in for m_o in Eq. (3-57), gives

$$h(n) = e^{j(2\pi n/N)((K-1)/2-(K-1)/2)} \cdot \frac{1}{N} \cdot \frac{\sin(\pi nK/N)}{\sin(\pi n/N)}$$

$$= e^{j(2\pi n/N)(0)} \cdot \frac{1}{N} \cdot \frac{\sin(\pi nK/N)}{\sin(\pi n/N)}. \qquad (3\text{-}58)$$

Again, because $e^{j0} = 1$, Eq. (3-58) becomes

$$h(n) = \frac{1}{N} \cdot \frac{\sin(\pi nK/N)}{\sin(\pi n/N)}. \qquad (3\text{-}59)$$

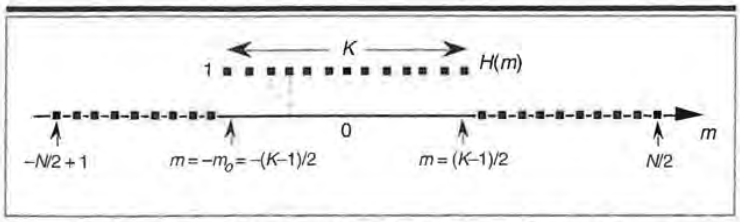


Figure 3-38 Frequency-domain rectangular function of width K samples defined over N samples.

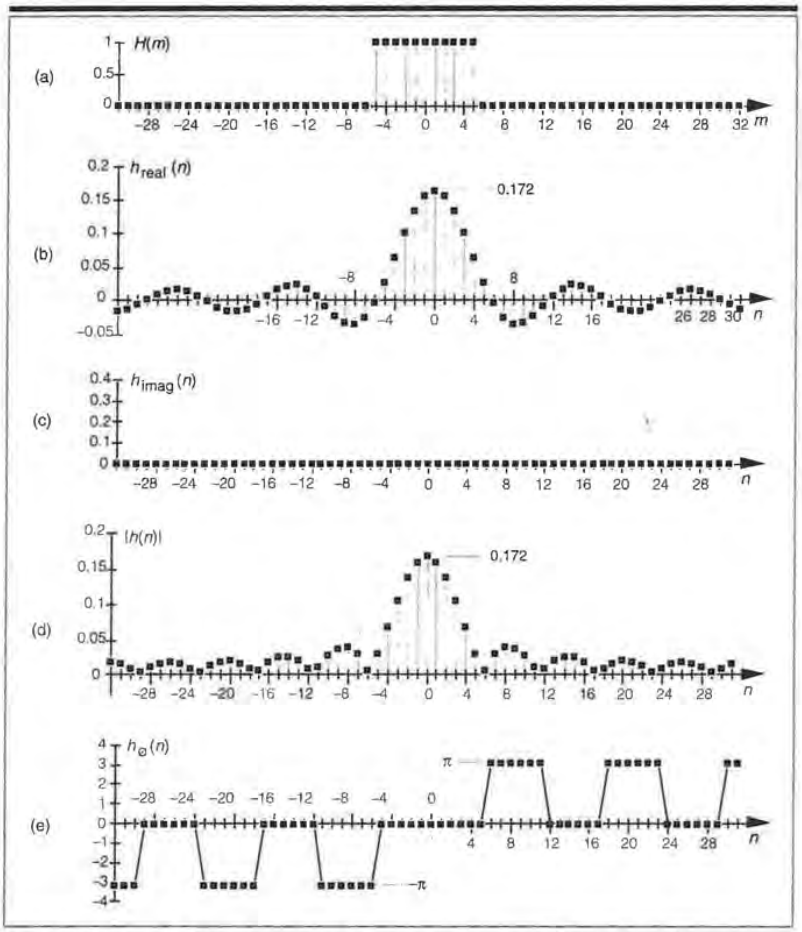


Figure 3-39 Inverse DFT of a rectangular function centered about $m = 0$: (a) original $H(m)$; (b) $h_{real}(n)$; (c) $h_{imag}(n)$; (d) magnitude of $h(n)$; (e) phase of $h(n)$ in radians.

Equation (3-59) tells us that the inverse DFT of the symmetrical rectangular function in Figure 3-38 is itself a real function that we can illustrate by example. We'll perform a 64-point inverse DFT of the sequence in Figure 3-39(a). Our $H(m)$ is 11 unity-valued samples centered about the $m = 0$ index. Here the inverse DFT results in an $h(n)$ whose real and imaginary parts are shown in Figure 3-39(b) and Figure 3-39(c), respectively. As Eq. (3-47) predicted, $h_{real}(n)$ is nonzero and $h_{imag}(n)$ is zero. The magnitude and phase of $h(n)$ are depicted in Figure 3-39(d) and Figure 3-39(e). (Again, we've made the functions in Figure 3-39 easy to compare with the forward DFT function in Figure 3-29.) It is $h(n)$'s real part that we're really after here. Those values of $h_{real}(n)$ are used for the time-domain filter coefficients in the design of FIR low-pass filters that we'll discuss in Section 5.3.

3.14 The DFT Frequency Response to a Complex Input

In this section, we'll determine the frequency response to an N-point DFT when its input is a discrete sequence representing a complex sinusoid expressed as $x_c(n)$. By frequency response we mean the DFT output samples when a complex sinusoidal sequence is applied to the DFT. We begin by depicting an $x_c(n)$ input sequence in Figure 3-40. This time sequence is of the form

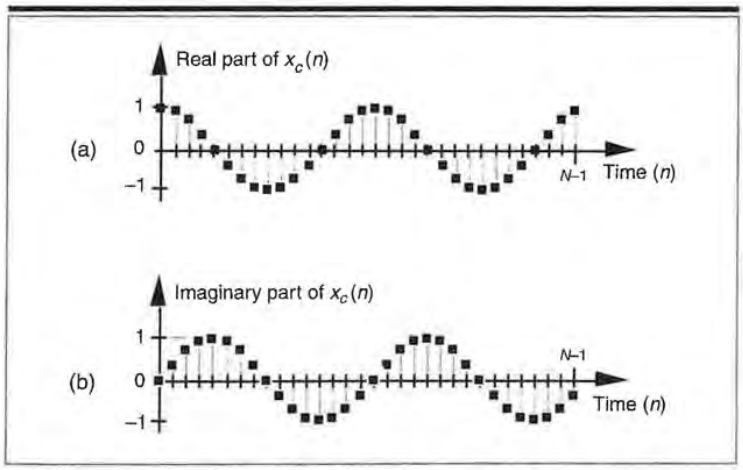


Figure 3-40 Complex time-domain sequence $x_c(n) = e^{j2\pi nk/N}$ having two complete cycles ($k = 2$) over N samples: (a) real part of $x_c(n)$; (b) imaginary part of $x_c(n)$.

$$x_c(n) = e^{j2\pi nk/N}, \tag{3-60}$$

where k is the number of complete cycles occurring in the N samples. Figure 3-40 shows $x_c(n)$ if we happen to let $k = 2$. If we denote our DFT output sequence as $X_c(m)$, and apply our $x_c(n)$ input to the DFT expression in Eq. (3-2) we have

$$X_c(m) = \sum_{n=0}^{N-1} x_c(n) e^{-j2\pi nm/N} = \sum_{n=0}^{N-1} e^{j2\pi nk/N} \cdot e^{-j2\pi nm/N}$$

$$= \sum_{n=0}^{N-1} e^{j2\pi n(k-m)/N}. \tag{3-61}$$

If we let $N = K$, $n = p$, and $q = -2\pi(k-m)/N$, Eq. (3-61) becomes

$$X_c(m) = \sum_{p=0}^{K-1} e^{-jpq}. \tag{3-62}$$

Why did we make the substitutions in Eq. (3-61) to get Eq. (3-62)? Because, happily, we've already solved Eq. (3-62) when it was Eq. (3-39). That closed form solution was Eq. (3-41) that we repeat here as

$$X_c(m) = \sum_{p=0}^{K-1} e^{-jpq} = e^{-jq(K-1)/2} \cdot \frac{\sin(qK/2)}{\sin(q/2)}. \tag{3-63}$$

Replacing our original variables from Eq. (3-61), we have our answer:

DFT of a complex sinusoid: →

$$X_c(m) = e^{j[\pi(k-m)-\pi(k-m)/N]} \cdot \frac{\sin[\pi(k-m)]}{\sin[\pi(k-m)/N]} \tag{3-64}$$

Like the Dirichlet kernel in Eq. (3-43), the $X_c(m)$ in Eq. (3-64) is a complex expression where a ratio of sine terms is the amplitude of $X_c(m)$ and the exponential term is the phase angle of $X_c(m)$. At this point, we're interested only in the ratio of sines factor in Eq. (3-64). Its magnitude is shown in Figure 3-41. Notice that, because $x_c(n)$ is complex, there are no negative frequency components in $X_c(m)$. Let's think about the shaded curve in

Figure 3-41 for a moment. That curve is the continuous Fourier transform of the complex $x_c(n)$ and can be thought of as the continuous spectrum of the $x_c(n)$ sequence.† By continuous spectrum we mean a spectrum that's defined at all frequencies, not just at the periodic f_s/N analysis frequencies of an N-point DFT. The shape of this spectrum with its main lobe and sidelobes is a direct and unavoidable effect of analyzing any finite-length time sequence, such as our $x_c(n)$ in Figure 3-40.

We can conceive of obtaining this continuous spectrum analytically by taking the continuous Fourier transform of our discrete $x_c(n)$ sequence, a process some authors call the discrete-time Fourier transform (DTFT), but we can't actually calculate the continuous spectrum on a computer. That's because the DTFT is defined only for infinitely long time sequences, and the DTFT's frequency variable is continuous with infinitely fine-grained resolution. What we can do, however, is use the DFT to calculate an approximation of $x_c(n)$'s continuous spectrum. The DFT outputs represented by the dots in Figure 3-41 are a discrete sampled version of $x_c(n)$'s continuous spectrum. We could have sampled that continuous spectrum more often, i.e., approximated it more closely, with a larger DFT by appending additional zeros to the original $x_c(n)$ sequence. We actually did that in Figure 3-21.

Figure 3-41 shows us why, when an input sequence's frequency is exactly at the $m = k$ bin center, the DFT output is zero for all bins except where $m = k$. If our input sequence frequency was k+0.25 cycles in the sample interval, the DFT will sample the continuous spectrum shown in Figure 3-42 where all of the DFT output bins would be nonzero. This effect is a demonstration of DFT leakage described in Section 3.8.

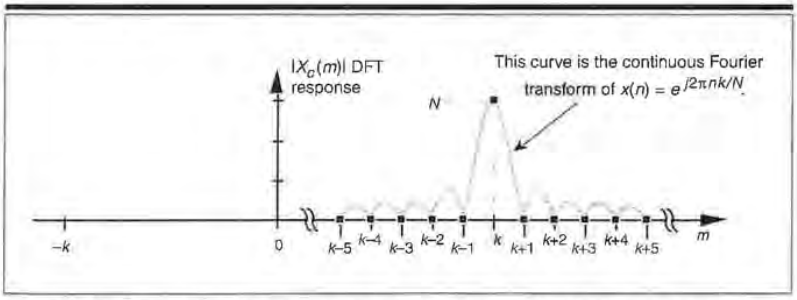


Figure 3-41 N-point DFT frequency magnitude response to a complex sinusoid having integral k cycles in the N-point time sequence $x_c(n) = e^{j2\pi nk/N}$.

† Just as we used L'Hospital's rule to find the peak value of the Dirichlet kernel in Eq. (3-44), we could also evaluate Eq. (3-64) to show that the peak of $X_c(m)$ is N when $m = k$.

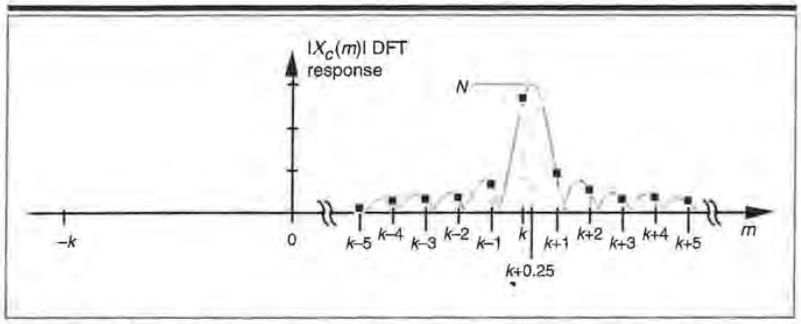


Figure 3-42 *N*-point DFT frequency magnitude response showing spectral leakage of a complex sinusoid having $k+0.25$ cycles in the *N*-point time sequence $x_c(n)$.

Again, just as there are several different expressions for the DFT of a rectangular function that we listed in Table 3-2, we can express the amplitude response of the DFT to a complex input sinusoid in different ways to arrive at Table 3-3.

Table 3-3 Various Forms of the Amplitude Response of the DFT to a Complex Input Sinusoid Having k Cycles in the Sample Interval

Description	*Expression*	
Complex input DFT amplitude response in terms of the integral frequency variable m [From Eq. (3-64)]	$X_c(m) = \dfrac{\sin[\pi(k-m)]}{\sin[\pi(k-m)/N]}$	(3-65)
Alternate form of the complex input DFT amplitude response in terms of the integral frequency variable m [based on Eq. (3-49)]	$X_c(m) = \dfrac{N\sin[\pi(k-m)]}{\pi(k-m)}$	(3-66)
Amplitude normalized complex input DFT response in terms of the integral frequency variable m	$X_c(m) = \dfrac{\sin[\pi(k-m)]}{\pi(k-m)}$	(3-67)
Complex input DFT response in terms of the sample rate f_s in Hz	$X_c(mf_s) = \dfrac{N\sin[\pi(k-m)f_s]}{\pi(k-m)f_s}$	(3-68)
Amplitude normalized complex input DFT response in terms of the sample rate f_s in Hz	$X_c(mf_s) = \dfrac{\sin[\pi(k-m)f_s]}{\pi(k-m)f_s}$	(3-69)
Amplitude normalized complex input DFT response in terms of the sample rate ω_s	$X_c(m\omega_s) = \dfrac{2\sin[(k-m)\omega_s/2]}{(k-m)\omega_s}$	(3-70)

At this point, the thoughtful reader may notice that the DFT's response to a complex input of k cycles per sample interval in Figure 3-41 looks suspiciously like the DFT's response to an all ones rectangular function in Figure 3-32(c). The reason the shapes of those two response curves look the same is because their shapes *are* the same. If our DFT input sequence was a complex sinusoid of $k = 0$ cycles, i.e., a sequence of identical constant values, the ratio of sines term in Eq. (3-64) becomes

$$\frac{\sin[\pi(0-m)]}{\sin[\pi(0-m)/N]} = \frac{-\sin(\pi m)}{-\sin(\pi m/N)} = \frac{\sin(\pi m)}{\sin(\pi m/N)},$$

which is identical to the all ones form of the Dirichlet kernel in Eq. (3-48). The shape of our $X_c(m)$ DFT response *is* the sinc function of the Dirichlet kernel.

3.15 The DFT Frequency Response to a Real Cosine Input

Now that we know what the DFT frequency response is to a complex sinusoidal input, it's easy to determine the DFT frequency response to a real input sequence. Say we want the DFT's response to a real cosine sequence, like that shown in Figure 3-40(a), expressed as

$$x_r(n) = \cos(2\pi nk/N), \tag{3-71}$$

where k is the integral number of complete cycles occurring in the N samples. Remembering Euler's relationship $\cos(\varnothing) = (e^{j\varnothing} + e^{-j\varnothing})/2$, we can show the desired DFT as $X_r(m)$ where

$$X_r(m) = \sum_{n=0}^{N-1} x_r(n)e^{-j2\pi nm/N} = \sum_{n=0}^{N-1} \cos(2\pi nk/N)\cdot e^{-j2\pi nm/N}$$

$$= \sum_{n=0}^{N-1} (e^{j2\pi nk/N} + e^{-j2\pi nk/N})/2 \cdot e^{-j2\pi nm/N}$$

$$= \frac{1}{2}\sum_{n=0}^{N-1} e^{j2\pi n(k-m)/N} + \frac{1}{2}\sum_{n=0}^{N-1} e^{-j2\pi n(k+m)/N}. \tag{3-72}$$

Fortunately, in the previous section we just finished determining the closed form of a summation expression like those in Eq. (3-72), so we can write the closed form for $X_r(m)$ as

DFT of a real cosine: →

$$X_r(m) = e^{j[\pi(k-m)-\pi(k-m)/N]} \cdot \frac{1}{2}\frac{\sin[\pi(k-m)]}{\sin[\pi(k-m)/N]} + e^{j[\pi(k+m)-\pi(k+m)/N]} \cdot \frac{1}{2}\frac{\sin[\pi(k+m)]}{\sin[\pi(k+m)/N]} . \quad (3\text{-}73)$$

We show the magnitude of those two ratio of sines terms as the sinc functions in Figure 3-43. Here again, the DFT is sampling the input cosine sequence's continuous spectrum and, because $k = m$, only one DFT bin is nonzero. Because the DFT's input sequence is real, $X_r(m)$ has both positive and negative frequency components. The positive frequency portion of Figure 3-43 corresponds to the first ratio of sines term in Eq.

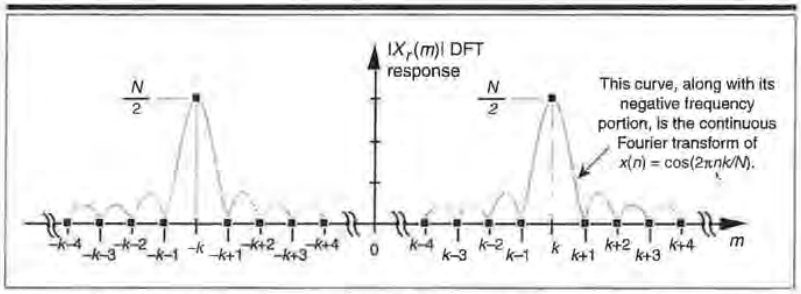


Figure 3-43 *N*-point DFT frequency magnitude response to a real cosine having integral k cycles in the N-point time sequence $x_r(n) = \cos(2\pi nk/N)$.

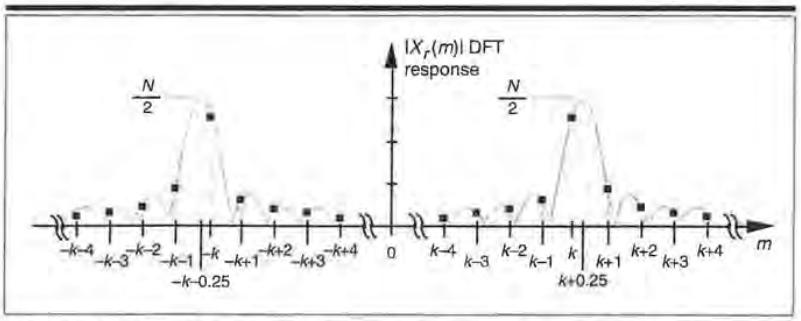


Figure 3-44 *N*-point DFT frequency magnitude response showing spectral leakage of a real cosine having k+0.25 cycles in the N-point time sequence $x_r(n)$.

Table 3-4 Various Forms of the *Positive Frequency* Amplitude Response of the DFT to a Real Cosine Input Having k Cycles in the Sample Interval

Description	*Expression*	
Real input DFT amplitude response in terms of the integral frequency variable m [From Eq. (3-73)]	$X_r(m) = \frac{\sin[\pi(k-m)]}{2\sin[\pi(k-m)/N]}$	(3-74)
Alternate form of the real input DFT amplitude response in terms of the integral frequency variable m [based on Eq. (3-49)]	$X_r(m) = \frac{N\sin[\pi(k-m)]}{2\pi(k-m)}$	(3-75)
Amplitude normalized real input DFT response in terms of the integral frequency variable m	$X_r(m) \approx \frac{\sin[\pi(k-m)]}{2\pi(k-m)}$	(3-76)
Real input DFT response in terms of the sample rate f_s in Hz	$X_r(mf_s) = \frac{N\sin[\pi(k-m)f_s]}{2\pi(k-m)f_s}$	(3-77)
Amplitude normalized real input DFT response in terms of the sample rate f_s in Hz	$X_r(mf_s) \approx \frac{\sin[\pi(k-m)f_s]}{2\pi(k-m)f_s}$	(3-78)
Amplitude normalized real input DFT response in terms of the sample rate ω_s in radians/s	$X_r(m\omega_s) \approx \frac{\sin[(k-m)\omega_s/2]}{(k-m)\omega_s}$	(3-79)

(3-73) and the second ratio of sines term in Eq. (3-73) produces the negative frequency components of $X_r(m)$.

DFT leakage is again demonstrated if our input sequence frequency were shifted from the center of the kth bin to k+0.25 as shown in Figure 3-44. (We used this concept of real input DFT amplitude response to introduce the effects of DFT leakage in Section 3.8.)

In Table 3-4, the various mathematical expressions for the (positive frequency) amplitude response of the DFT to a real cosine input sequence are simply those expressions in Table 3-3 reduced in amplitude by a factor of 2.

3.16 The DFT Single-Bin Frequency Response to a Real Cosine Input

Now that we understand the DFT's overall N-point (or N-bin) frequency response to a real cosine of k cycles per sample interval, we conclude this chapter by determining the frequency response of a *single* DFT bin. We can think of a single DFT bin as a kind of bandpass filter, and this useful notion is used, for example, to describe DFT scalloping loss (Section 3.10), employed in the design of frequency-domain filter banks, and applied in

a telephone frequency multiplexing technique known as transmultiplexing[15]. To determine a DFT single-bin's frequency response, consider applying a real $x_r(n)$ cosine sequence to a DFT and monitoring the output magnitude of just the $m = k$ bin. Let's say the initial frequency of the input cosine sequence starts at a frequency of $k<m$ cycles and increases up to a frequency of $k>m$ cycles in the sample interval. If we measure the DFT's

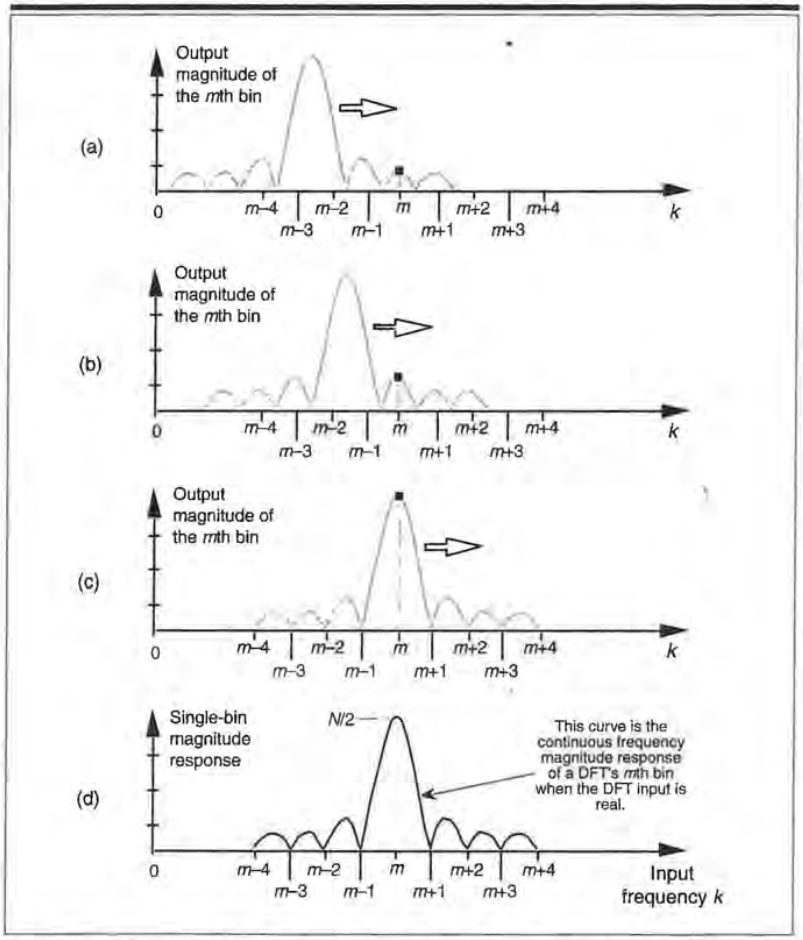


Figure 3-45 Determining the output magnitude of the mth bin of an N-point DFT: (a) when the real $x_r(n)$ has $k = m-2.5$ cycles in the time sequence; (b) when the real $x_r(n)$ has $k = m-1.5$ cycles in the time sequence; (c) when the real $x_r(n)$ has $k = m$ cycles in the time sequence; (d) the DFT single-bin frequency magnitude response of the $m = k$ bin.

$m = k$ bin during that frequency sweep, we'll see that its output magnitude must track the input cosine sequence's continuous spectrum, shown as the shaded curves in Figure 3-45.

Figure 3-45(a) shows the $m = k$ bin's output when the input $x_r(n)$'s frequency is $k = m–2.5$ cycles per sample interval. Increasing $x_r(n)$'s frequency to $k = m–1.5$ cycles per sample interval results in the $m = k$ bin's output, shown in Figure 3-45(b). Continuing to sweep $x_r(n)$'s frequency higher Figure 3-45(c) shows the $m = k$ bin's output when the input frequency is $k = m$. Throughout our input frequency sweeping exercise, we can see that the $m = k$ bin's output magnitude must trace out the cosine sequence's continuous spectrum, shown by the solid curve in Figure 3-45(d). This means that a DFT's single-bin frequency magnitude response, to a real input sinusoid, is that solid sinc function curve defined by Eqs. (3-74) through (3-79).

References

[1] Bracewell, R. "The Fourier Transform," *Scientific American*, June 1989.

[2] Struik, D. *A Concise History of Mathematics*, Dover Publications Inc., New York, 1967, p. 142.

[3] Williams, C. S. *Designing Digital Filters*. Section 8.6, Prentice-Hall, Englewood Cliffs, New Jersey, 1986, p. 122.

[4] Press, W., et al. *Numerical Recipes—The Art of Scientific Computing*. Cambridge University Press, 1989, p. 426.

[5] Geckinli, N. C., and Yavuz, D. "Some Novel Windows and a Concise Tutorial Comparison of Window Families," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-26, No. 6, December 1978. (By the way, on page 505 of this paper, the phrase "such that $W(f) \geq 0 \; \forall \; f$" indicates that W(f) is never negative. The symbol $\forall$ means "for all.")

[6] O'Donnell, J. "Looking Through the Right Window Improves Spectral Analysis," *EDN*, November 1984.

[7] Kaiser, J. F. "Digital Filters," in *System Analysis by Digital Computer*. Ed. by F. F. Kuo and J. F. Kaiser, John Wiley and Sons, New York, 1966, pp. 218–277.

[8] Rabiner, L. R., and Gold, B. *The Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975, p. 88.

[9] Schoenwald, J. "The Surface Acoustic Wave Filter: Window Functions," *RF Design*, March 1986.

[10] Harris, F. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, Vol. 66, No. 1, January 1978.

[11] Nuttall, A. H. "Some Windows with Very Good Sidelobe Behavior," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-29, No. 1, February 1981.

[12] Yanagimoto, Y. "Receiver Design for a Combined RF Network and Spectrum Analyzer," *Hewlett-Packard Journal*, October, 1993.

[13] Gullemin, E. A. *The Mathematics of Circuit Analysis*. John Wiley and Sons, New York, 1949, p. 511.

[14] Lanczos, C. *Discourse on Fourier Series*, Chapter 1, Hafner Publishing Co., New York, 1966, p. 7–47.

[15] Freeny, S. "TDM/FDM Translation As an Application of Digital Signal Processing," *IEEE Communications Magazine*, January 1980.

CHAPTER FOUR

The Fast Fourier Transform

Although the DFT is the most straightforward mathematical procedure for determining the frequency content of a time-domain sequence, it's terribly inefficient. As the number of points in the DFT is increased to hundreds, or thousands, the amount of necessary number crunching becomes excessive. In 1965 a paper was published by Cooley and Tukey describing a very efficient algorithm to implement the DFT[1]. That algorithm is now known as the fast Fourier transform (FFT).† Before the advent of the FFT, thousand-point DFTs took so long to perform that their use was restricted to the larger research and university computer centers. Thanks to Cooley, Tukey, and the semiconductor industry, 1024-point DFTs can now be performed in a few seconds on home computers.

Volumes have been written about the FFT, and, like no other innovation, the development of this algorithm transformed the discipline of digital signal processing by making the power of Fourier analysis affordable. In this chapter, we'll show why the most popular FFT algorithm (called the *radix*-2 FFT) is superior to the classical DFT algorithm, present a series of recommendations to enhance our use of the FFT in practice, and provide a list of sources for FFT routines in various software languages. We conclude this chapter, for those readers wanting to know the internal details, with a derivation of the radix-2 FFT and introduce several different ways in which this FFT is implemented.

† Actually, the FFT has an interesting history. While analyzing X-ray scattering data, a couple of physicists in the 1940s were taking advantage of the symmetries of sines and cosines using a mathematical method based on a technique published in the early 1900s. Remarkably, over 20 years passed before the FFT was (re)discovered. Reference [2] tells the full story.

4.1 Relationship of the FFT to the DFT

Although many different FFT algorithms have been developed, in this section we'll see why the radix-2 FFT algorithm is so popular and learn how it's related to the classical DFT algorithm. The radix-2 FFT algorithm is a very efficient process for performing DFTs under the constraint that the DFT size be an integral power of two. (That is, the number of points in the transform is $N = 2^k$, where k is some positive integer.) Let's see just why the radix-2 FFT is the favorite spectral analysis technique used by signal-processing practitioners.

Recall that our DFT Example 1 in Section 3.1 illustrated the number of redundant arithmetic operations necessary for a simple 8-point DFT. (For example, we ended up calculating the product of $1.0607 \cdot 0.707$ four separate times.) On the other hand, the radix-2 FFT eliminates these redundancies and greatly reduces the number of necessary arithmetic operations. To appreciate the FFT's efficiency, let's consider the number of complex multiplications necessary for our old friend, the expression for an N-point DFT,

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N}. \tag{4-1}$$

For an 8-point DFT, Eq. (4-1) tells us that we'd have to perform N^2 or 64 complex multiplications. (That's because, for each of the eight $X(m)$s, we have to sum eight complex products as n goes from 0 to 7.) As we'll verify in later sections of this chapter, the number of complex multiplications, for an N-point FFT, is approximately

$$\frac{N}{2} \cdot \log_2 N. \tag{4-2}$$

(We say approximately because some multiplications turn out to be multiplications by +1 or –1, which amount to mere sign changes.) Well, this $(N/2)\log_2 N$ value is a significant reduction from the N^2 complex multiplications required by Eq. (4-1), particularly for large N. To show just how significant, Figure 4-1 compares the number of complex multiplications required by DFTs and radix-2 FFTs as a function of the number of input data points N. When $N = 512$, for example, the DFT requires 200 times more complex multiplications than those needed by the FFT. When $N = 8192$, the DFT must calculate 1000 complex multiplications for *each* complex multiplication in the FFT!

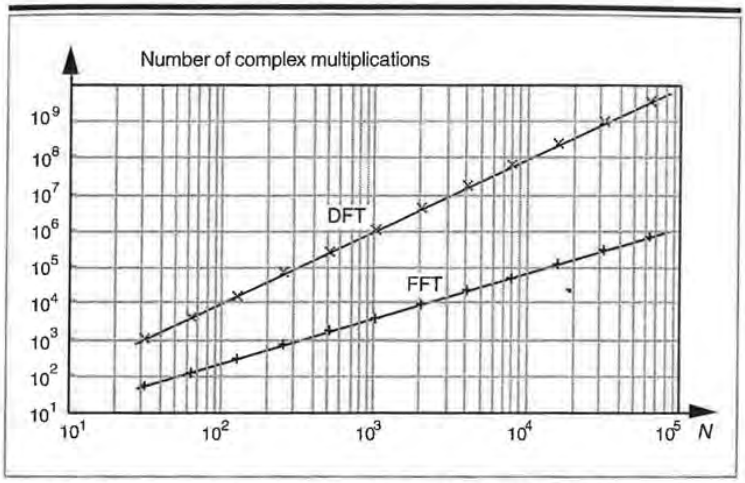


Figure 4-1 Number of complex multiplications in the DFT and the radix-2 FFT as a function of *N*.

It's appropriate now to make clear that the FFT is not an approximation to the DFT. It's exactly equal to the DFT; it *is* the DFT. Moreover, all of the performance characteristics of the DFT described in the previous chapter, output symmetry, linearity, output magnitudes, leakage, scalloping loss, etc., also describe the behavior of the FFT.

4.2 Hints on Using FFTs in Practice

Based on how useful FFTs are, here's a list of practical pointers, or tips, on acquiring input data samples and using the radix-2 FFT to analyze real-world signals or data.

4.2.1 Sample Fast Enough and Long Enough

When digitizing continuous signals with an A/D converter, for example, we know, from Chapter 2, that our sampling rate must be greater than twice the bandwidth of the continuous A/D input signal to prevent frequency-domain aliasing. Depending on the application, practitioners typically sample at 2.5 to four times the signal bandwidth. If we know that the bandwidth of the continuous signal is not too large relative to the maximum sample rate of our A/D converter, it's easy to avoid aliasing. If we don't know the continuous A/D input signal's bandwidth, how do we

tell if we're having aliasing problems? Well, we should mistrust any FFT results that have significant spectral components at frequencies near half the sample rate. Ideally, we'd like to work with signals whose spectral amplitudes decrease with increasing frequency. Be very suspicious of aliasing if there are any spectral components whose frequencies appear to depend on the sample rate. If we suspect that aliasing is occurring or that the continuous signal contains broadband noise, we'll have to use an analog low-pass filter prior to A/D conversion. The cutoff frequency of the low-pass filter must, of course, be greater than the frequency band of interest but less than half the sample rate.

Although we know that an N-point radix-2 FFT requires $N = 2^k$ input samples, just how many samples must we collect before we perform our FFT? The answer is that the data collection time interval must be long enough to satisfy our desired FFT frequency resolution for the given sample f_s rate. The data collection time interval is the reciprocal of the desired FFT frequency resolution, and the longer we sample at a fixed f_s sample rate, the finer our frequency resolution will be; that is, the total data collection time interval is N/f_s seconds, and our N-point FFT bin-to-bin frequency resolution is f_s/N Hz. So, for example, if we need a spectral resolution of 5 Hz, then, $f_s/N = 5$ Hz, and

$$N = \frac{f_s}{\text{desired resolution}} = \frac{f_s}{5} = 0.2f_s. \tag{4-3}$$

In this case, if f_s is, say, 10 kHz, then N must be at least 2,000, and we'd choose N equal to 2048 because this number is a power of 2.

4.2.2 Manipulating the Time Data Prior to Transformation

When using the radix-2 FFT, if we don't have control over the length of our time-domain data sequence, and that sequence length is not an integral power of two, we have two options. We could discard enough data samples so that the remaining FFT input sequence length is some integral power of two. This scheme is not recommended because ignoring data samples degrades our resultant frequency-domain resolution. (The larger N, the better our frequency resolution, right?) A better approach is to append enough zero-valued samples to the end of the time data sequence to match the number of points of the next largest radix-2 FFT. For example, if we have 1000 time samples to transform, rather than analyzing only 512 of them with a 512-point FFT, we should add 24 trailing zero-valued samples to the original sequence and use a

1024-point FFT. (This *zero-stuffing* technique is discussed in more detail in Section 3.11.)

FFTs suffer the same ill effects of spectral leakage that we discussed for the DFT in Section 3.8. We can multiply the time data by a window function to alleviate this leakage problem. Be prepared, though, for the frequency resolution degradation inherent when windows are used. By the way, if appending zeros is necessary to extend a time sequence, we have to make sure that we append the zeros *after* multiplying the original time data sequence by a window function. Applying a window function to the appended zeros will distort the resultant window and worsen our FFT leakage problems.

Although windowing will reduce leakage problems, it will not eliminate them altogether. Even when windowing is employed, high-level spectral components can obscure nearby low-level spectral components. This is especially evident when the original time data has a nonzero average, i.e., it's riding on a DC bias. When the FFT is performed in this case, a large-amplitude DC spectral component at 0 Hz will overshadow its spectral neighbors. We can eliminate this problem by calculating the average of the windowed time sequence and subtract that average value from each sample in the original windowed sequence. (Alternatively, the averaging and subtraction process could just as well be performed before windowing.) This technique makes the new time sequence's average (mean) value equal to zero and eliminates any high-level, 0-Hz component in the FFT results.

4.2.3 Enhancing FFT Results

If we're using the FFT to detect signal energy in the presence of noise and enough time-domain data is available, we can improve the sensitivity of our processing by averaging multiple FFTs. This technique, discussed in Section 10.7, can be implemented very efficiently to detect signal energy that's actually below the average noise level; that is, given enough time-domain data, we can detect signal components that have negative signal-to-noise ratios.

If our original time-domain data is real-valued only, we can take advantage of the 2*N*-Point Real FFT technique in Section 10.5 to speed up our processing; that is, a 2*N*-point real sequence can be transformed with a single *N*-point complex radix-2 FFT. Thus we can get the frequency resolution of a 2*N*-point FFT for just about the computational price of performing a standard *N*-point FFT. Another FFT speed enhancement is the possible use of the frequency-domain windowing technique discussed in

Section 10.3. If we need the FFT of unwindowed time-domain data and, at the same time, we also want the FFT of that same time data with a window function applied, we don't have to perform two separate FFTs. We can perform the FFT of the unwindowed data, and then we can perform frequency-domain windowing to reduce spectral leakage on any, or all, of the FFT bin outputs.

4.2.4 Interpreting FFT Results

The first step in interpreting FFT results is to compute the absolute frequency of the individual FFT bin centers. Like the DFT, the FFT bin spacing is the ratio of the sampling rate (f_s) over the number of points in the FFT, or f_s/N. With our FFT output designated by $X(m)$, where $m = 0, 1, 2, 3, \ldots, N-1$, the absolute frequency of the mth bin center is mf_s/N. If the FFT's input time samples are real, only the $X(m)$ outputs from $m = 0$ to $m = N/2$ are independent. So, in this case, we need determine only the absolute FFT bin frequencies for m over the range of $0 \le m \le N/2$. If the FFT input samples are complex, all N of the FFT outputs are independent, and we should compute the absolute FFT bin frequencies for m over the full range of $0 \le m \le N-1$.

If necessary, we can determine the true amplitude of time-domain signals from their FFT spectral results. To do so, we have to keep in mind that radix-2 FFT outputs are complex and of the form

$$X(m) = X_{\text{real}}(m) + jX_{\text{imag}}(m) . \tag{4-4}$$

Also, the FFT output magnitude samples,

$$X_{\text{mag}}(m) = |X(m)| = \sqrt{X_{\text{real}}(m)^2 + X_{\text{imag}}(m)^2} , \tag{4-5}$$

are all inherently multiplied by the factor $N/2$, as described in Section 3.4, when the input samples are real. If the FFT input samples are complex, the scaling factor is N. So to determine the correct amplitudes of the time-domain sinusoidal components, we'd have to divide the FFT magnitudes by the appropriate scale factor, $N/2$ for real inputs and N for complex inputs.

If a window function was used on the original time-domain data, some of the FFT input samples will be attenuated. This reduces the resultant FFT output magnitudes from their true unwindowed values. To calculate the correct amplitudes of various time-domain sinusoidal components, then, we'd have to further divide the FFT magnitudes by the appropriate

processing loss factor associated with the window function used. Processing loss factors for the most popular window functions are listed in Reference [3].

Should we want to determine the power spectrum $X_{PS}(m)$ of an FFT result, we'd calculate the magnitude-squared values using

$$X_{PS}(m) = |X(m)|^2 = X_{real}(m)^2 + X_{imag}(m)^2 . \quad (4\text{-}6)$$

Doing so would allow us to compute the power spectrum in decibels with

$$X_{dB}(m) = 10 \cdot \log_{10}(|X(m)|^2) \text{ dB} . \quad (4\text{-}7)$$

The normalized power spectrum in decibels can be calculated using

$$\text{normalized } X_{dB}(m) = 10 \cdot \log_{10}\left(\frac{|X(m)|^2}{(|X(m)|_{max})^2}\right), \quad (4\text{-}8)$$

or

$$\text{normalized } X_{dB}(m) = 20 \cdot \log_{10}\left(\frac{|X(m)|}{|X(m)|_{max}}\right) . \quad (4\text{-}9)$$

In Eqs. (4-8) and (4-9), the term $|X(m)|_{max}$ is the largest FFT output magnitude sample. In practice, we find that plotting $X_{dB}(m)$ is very informative because of the enhanced low-magnitude resolution afforded by the logarithmic decibel scale, as described in Appendix E. If either Eq. (4-8) or Eq. (4-9) is used, no compensation need be performed for the above-mentioned N or $N/2$ FFT scale or window processing loss factors. Normalization through division by $(|X(m)|_{max})^2$ or $|X(m)|_{max}$ eliminates the effect of any absolute FFT or window scale factors.

Knowing that the phase angles $X_{\phi}(m)$ of the individual FFT outputs are given by

$$X_{\phi}(m) = \tan^{-1}\left(\frac{X_{imag}(m)}{X_{real}(m)}\right), \quad (4\text{-}10)$$

it's important to watch out for $X_{real}(m)$ values that are equal to zero. That would invalidate our phase angle calculations in Eq. (4-10) due to division

by a zero condition. In practice, we want to make sure that our calculations (or software compiler) detect occurrences of $X_{real}(m) = 0$ and set the corresponding $X_\phi(m)$ to 90° if $X_{imag}(m)$ is positive, set $X_\phi(m)$ to 0° if $X_{imag}(m)$ is zero, and set $X_\phi(m)$ to –90° if $X_{imag}(m)$ is negative. While we're on the subject of FFT output phase angles, be aware that FFT outputs containing significant noise components can cause large fluctuations in the computed $X_\phi(m)$ phase angles. This means that the $X_\phi(m)$ samples are only meaningful when the corresponding $|X(m)|$ is well above the average FFT output noise level.

4.3 FFT Software Programs

For readers seeking actual FFT software routines without having to buy those high-priced signal processing software packages, public domain radix-2 FFT routines are readily available. References [4–7] provide standard FFT program listings using the FORTRAN language. Reference [8] presents an efficient FFT program written in FORTRAN for real-only input data sequences. Reference [9] provides a standard FFT program written in HP BASIC™, and reference [10] presents an FFT routine written in Applesoft BASIC™. Readers interested in the Ada language can find FFT-related subroutines in reference [11].

4.4 Derivation of the Radix-2 FFT Algorithm

This section and those that follow provide a detailed description of the internal data structures and operations of the radix-2 FFT for those readers interested in developing software FFT routines or designing FFT hardware. To see just exactly how the FFT evolved from the DFT, we return to the equation for an N-point DFT,

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N}. \tag{4-11}$$

A straightforward derivation of the FFT proceeds with the separation of the input data sequence $x(n)$ into two parts. When $x(n)$ is segmented into its even and odd indexed elements, we can, then, break Eq. (4-11) into two parts as

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n)e^{-j2\pi(2n)m/N} + \sum_{n=0}^{(N/2)-1} x(2n+1)e^{-j2\pi(2n+1)m/N}. \tag{4-12}$$

Pulling the constant phase angle outside the second summation,

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n)e^{-j2\pi(2n)m/N} + e^{-j2\pi m/N} \sum_{n=0}^{(N/2)-1} x(2n+1)e^{-j2\pi(2n)m/N}. \quad (4\text{-}13)$$

Well, here the equations get so long and drawn out that we'll use the standard notation to simplify things. We'll define $W_N = e^{-j2\pi/N}$ to represent the complex phase angle factor that is constant with N. So Eq. (4-13) becomes

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n)W_N^{2nm} + W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1)W_N^{2nm}. \quad (4\text{-}14)$$

Because $W_N^2 = e^{-j2\pi 2/(N)} = e^{-j2\pi/(N/2)}$, we can substitute $W_{N/2}$ for W_N^2 in Eq. (4-14), as

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n)W_{N/2}^{nm} + W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1)W_{N/2}^{nm}. \quad (4\text{-}15)$$

So we now have two $N/2$ summations whose results can be combined to give us the N-point DFT. We've reduced some of the necessary number crunching in Eq. (4-15) relative to Eq. (4-11) because the W terms in the two summations of Eq. (4-15) are identical. There's a further benefit in breaking the N-point DFT into two parts because the upper half of the DFT outputs are easy to calculate. Consider the $X(m+N/2)$ output. If we plug $m+N/2$ in for m in Eq. (4-15), then

$$X(m+N/2) = \sum_{n=0}^{(N/2)-1} x(2n)W_{N/2}^{n(m+N/2)} + W_N^{(m+N/2)} \sum_{n=0}^{(N/2)-1} x(2n+1)W_{N/2}^{n(m+N/2)}. \quad (4\text{-}16)$$

It looks like we're complicating things, right? Well, just hang in there for a moment. We can now simplify the phase angle terms inside the summations because

$$W_{N/2}^{n(m+N/2)} = W_{N/2}^{nm} W_{N/2}^{nN/2} = W_{N/2}^{nm} (e^{-j2\pi n 2N/2N})$$

$$= W_{N/2}^{nm}(1) = W_{N/2}^{nm}, \tag{4-17}$$

for any integer n. Looking at the so-called *twiddle factor* in front of the second summation in Eq. (4-16), we can simplify it as

$$W_N^{(m+N/2)} = W_N^m W_N^{N/2} = W_N^m (e^{-j2\pi N/2N}) = W_N^m(-1) = -W_N^m. \tag{4-18}$$

OK, using Eqs. (4-17) and (4-18), we represent Eq. (4-16)'s $X(m+N/2)$ as

$$X(m+N/2) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm} - W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm}. \tag{4-19}$$

Now, let's repeat Eqs. (4-15) and (4-19) to see the similarity;

$$X(m) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm} + W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm}, \tag{4-20}$$

and

$$X(m+N/2) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nm} - W_N^m \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nm}. \tag{4-20'}$$

So here we are. We need not perform any sine or cosine multiplications to get $X(m+N/2)$. We just change the sign of the twiddle factor W_N^m and use the results of the two summations from $X(m)$ to get $X(m+N/2)$. Of course, m goes from 0 to $(N/2)-1$ in Eq. (4-20) which means, for an N-point DFT, we perform an $N/2$-point DFT to get the first $N/2$ outputs and use those to get the last $N/2$ outputs. For $N = 8$, Eqs. (4-20) and (4-20') are implemented as shown in Figure 4-2.

If we simplify Eqs. (4-20) and (4-20') to the form

$$X(m) = A(m) + W_N^m B(m), \tag{4-21}$$

and

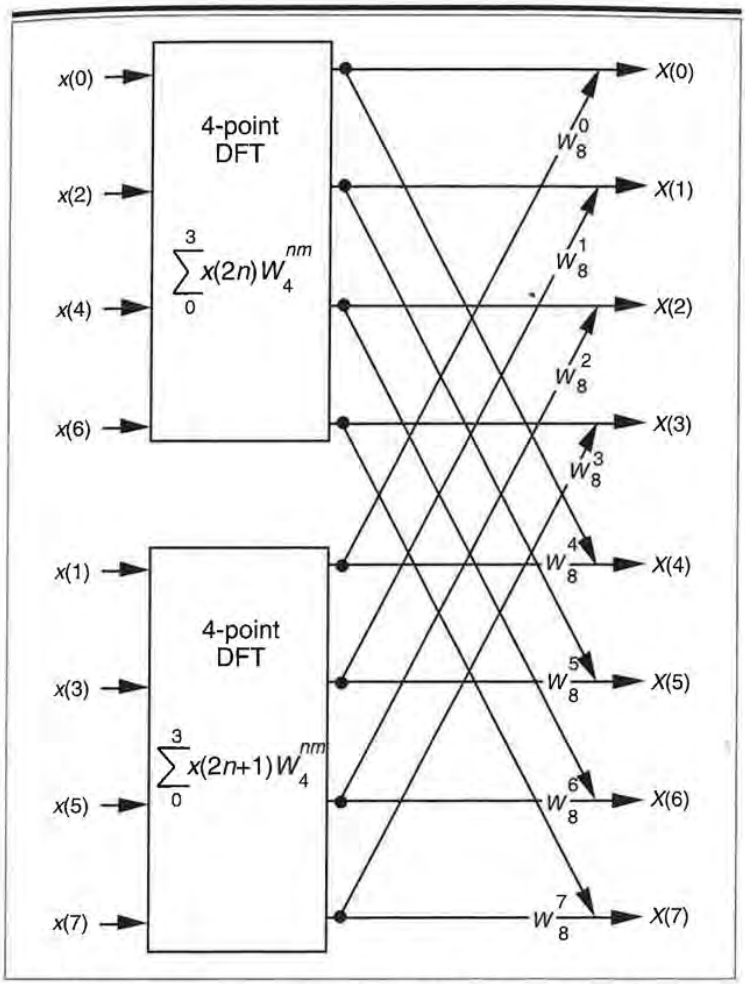


Figure 4-2 FFT implementation of an 8-point DFT using two 4-point DFTs.

$$X(m+N/2) = A(m) - W_N^m B(m)\,, \qquad (4\text{-}21')$$

we can go further and think about breaking the two 4-point DFTs into four 2-point DFTs. Let's see how we can subdivide the upper 4-point DFT in Figure 4-2 whose four outputs are $A(m)$ in Eqs. (4-21) and (4-21'). We segment the inputs to the upper 4-point DFT into their odd and even components

$$A(m) = \sum_{n=0}^{(N/2)-1} x(2n)W_{N/2}^{nm} = \sum_{2n=0}^{(N/4)-1} x(4n)W_{N/2}^{4nm} + \sum_{2n=0}^{(N/4)-1} x(4n+1)W_{N/2}^{(4n+1)m}$$

or

$$A(m) = \sum_{2n=0}^{(N/4)-1} x(4n)W_{N/4}^{2nm} + W_{N/2}^{m} \sum_{2n=0}^{(N/4)-1} x(4n+1)W_{N/4}^{2nm} . \tag{4-22}$$

If we let p = 2n, the index of the summations in Eq. (4-22) can be simplified to give $A(m)$ the more familiar form of

$$A(m) = \sum_{p=0}^{(N/4)-1} x(2p)W_{N/4}^{pm} + W_{N/2}^{m} \sum_{p=0}^{(N/4)-1} x(2p+1)W_{N/4}^{pm} . \tag{4-23}$$

Notice the similarity between Eq. (4-23) and Eq. (4-20). This capability to subdivide an $N/2$-point DFT into two $N/4$-point DFTs gives the FFT its capacity to greatly reduce the number of necessary multiplications to implement DFTs. (We're going to prove this shortly.) Following the same steps that we used to obtain $A(m)$, we can show that Eq. (4-21)'s $B(m)$ is given by

$$B(m) = \sum_{p=0}^{(N/4)-1} x(2p)W_{N/4}^{pm} - W_{N/2}^{m} \sum_{p=0}^{(N/4)-1} x(2p+1)W_{N/4}^{pm} . \tag{4-24}$$

For our $N = 8$ example, Eqs. (4-23) and (4-24) are implemented as shown in Figure 4-3. The FFT's well-known *butterfly* pattern of signal flows is certainly evident, and we see the further shuffling of the input data in Figure 4-3. The twiddle factor $W_{N/2}^{m}$ in Eqs. (4-23) and (4-24), for our $N = 8$ example, ranges from W_4^0 to W_4^3 because the m index, for $A(m)$ and $B(m)$, goes from 0 to 3. For any N-point DFT, we can break each of the $N/2$-point DFTs into two $N/4$-point DFTs to further reduce the number of sine and cosine multiplications. Eventually, we would arrive at an array of 2-point DFTs where no further computational savings could be realized. This is why the number of points in our FFTs are constrained to be some power of 2 and why this FFT algorithm is referred to as the radix-2 FFT.

Moving right along, let's go one step further, and then we'll be finished with our $N = 8$ point FFT derivation. The 2-point DFT functions in

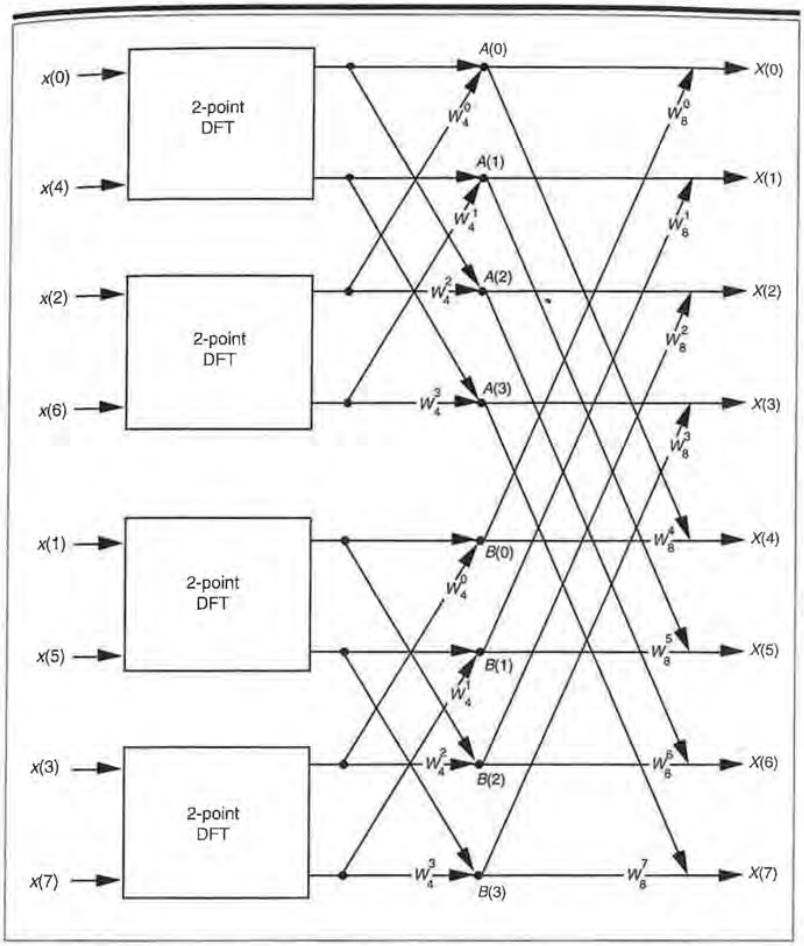


Figure 4-3 FFT implementation of an 8-point DFT as two 4-point DFTs and four 2-point DFTs.

Figure 4-3 cannot be partitioned into smaller parts—we've reached the end of our DFT reduction process arriving at the butterfly of a single 2-point DFT as shown in Figure 4-4. From the definition of W_N, $W_N^0 = e^{-j2\pi 0/N} = 1$ and $W_N^{N/2} = e^{-j2\pi N/2N} = e^{-j\pi} = -1$. So the 2-point DFT blocks in Figure 4-3 can be replaced by the butterfly in Figure 4-4 to give us a full 8-point FFT implementation of the DFT as shown in Figure 4-5.

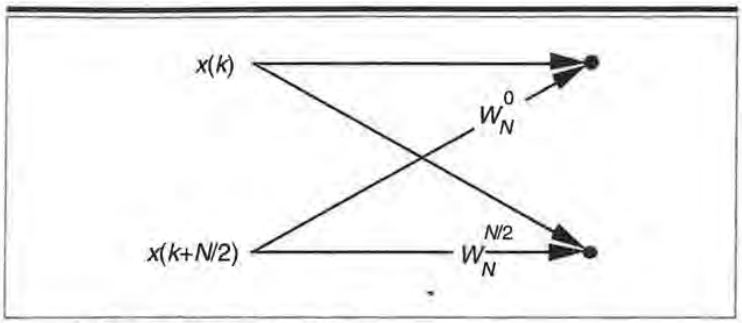


Figure 4-4 Single 2-point DFT butterfly.

OK, we've gone through a fair amount of algebraic foot shuffling here. To verify that the derivation of the FFT is valid, we can apply the 8-point data sequence of Chapter 3's DFT Example 1 to the 8-point FFT represented by Figure 4-5. The data sequence representing $x(n) = \sin(2\pi 1000nt_s) + 0.5\sin(2\pi 2000nt_s + 3\pi/4)$ is

$$\begin{aligned} x(0) &= 0.3535, & x(1) &= 0.3535, \\ x(2) &= 0.6464, & x(3) &= 1.0607, \\ x(4) &= 0.3535, & x(5) &= -1.0607, \\ x(6) &= -1.3535, & x(7) &= -0.3535. \end{aligned} \tag{4-25}$$

We begin grinding through this example by applying the input values from Eq. (4-25) to Figure 4-5, giving the data values shown on left side of Figure 4-6. The outputs of the second stage of the FFT are

$$A(0) = 0.707 + W_4^0(-0.707) = 0.707 + (1 + j0)(-0.707) = 0 + j0,$$
$$A(1) = 0.0 + W_4^1(1.999) = 0.0 + (0 - j1)(1.999) = 0 - j1.999,$$
$$A(2) = 0.707 + W_4^2(-0.707) = 0.707 + (-1 + j0)(-0.707) = 1.414 + j0,$$
$$A(3) = 0.0 + W_4^3(1.999) = 0.0 + (0 + j1)(1.999) = 0 + j1.999,$$
$$B(0) = -0.707 + W_4^0(0.707) = -0.707 + (1 + j0)(0.707) = 0 + j0,$$
$$B(1) = 1.414 + W_4^1(1.414) = 1.414 + (0 - j1)(1.414) = 1.414 - j1.414,$$
$$B(2) = -0.707 + W_4^2(0.707) = -0.707 + (-1 + j0)(0.707) = -1.414 + j0, \text{ and}$$
$$B(3) = 1.414 + W_4^3(1.414) = 1.414 + (0 + j1)(1.414) = 1.414 + j1.414\,.$$

Calculating the outputs of the third stage of the FFT to arrive at our final answer

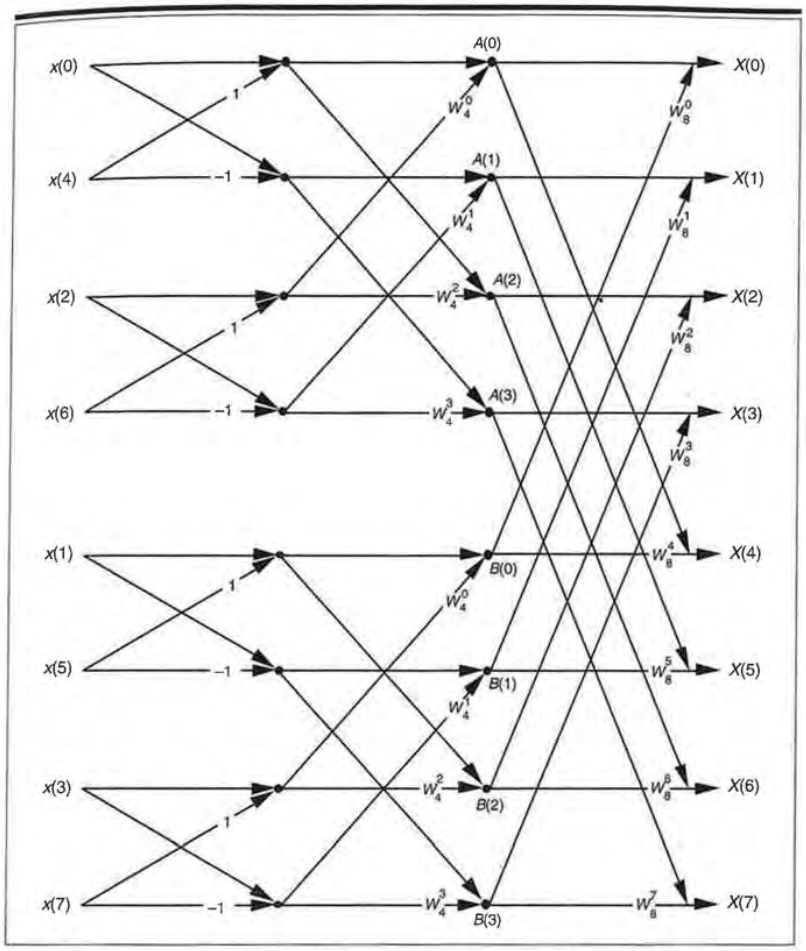


Figure 4-5 Full decimation-in-time FFT implementation of an 8-point DFT.

$$X(0) = A(0) + W_8^0 B(0) = 0 + j0 + (1 + j0)(0 + j0) = 0 + j0 + 0 + j0 = 0 \angle 0°,$$

$$X(1) = A(1) + W_8^1 B(1) = 0 - j1.999 + (0.707 - j0.707)(1.414 - j1.414)$$
$$= 0 - j1.999 + 0 - j1.999 = 0 - j4 = 4 \angle -90°$$

$$X(2) = A(2) + W_8^2 B(2) = 1.414 + j0 + (0 - j1)(-1.414 + j0)$$
$$= 1.414 + j0 + 0 + j1.4242 = 1.414 + j1.414 = 2 \angle 45°,$$

$$X(3) = A(3) + W_8^3 B(3) = 0 + j1.999 + (-0.707 - j0.707)(1.414 + j1.414)$$
$$= 0 + j1.999 + 0 - j1.999 = 0 \angle 0°,$$

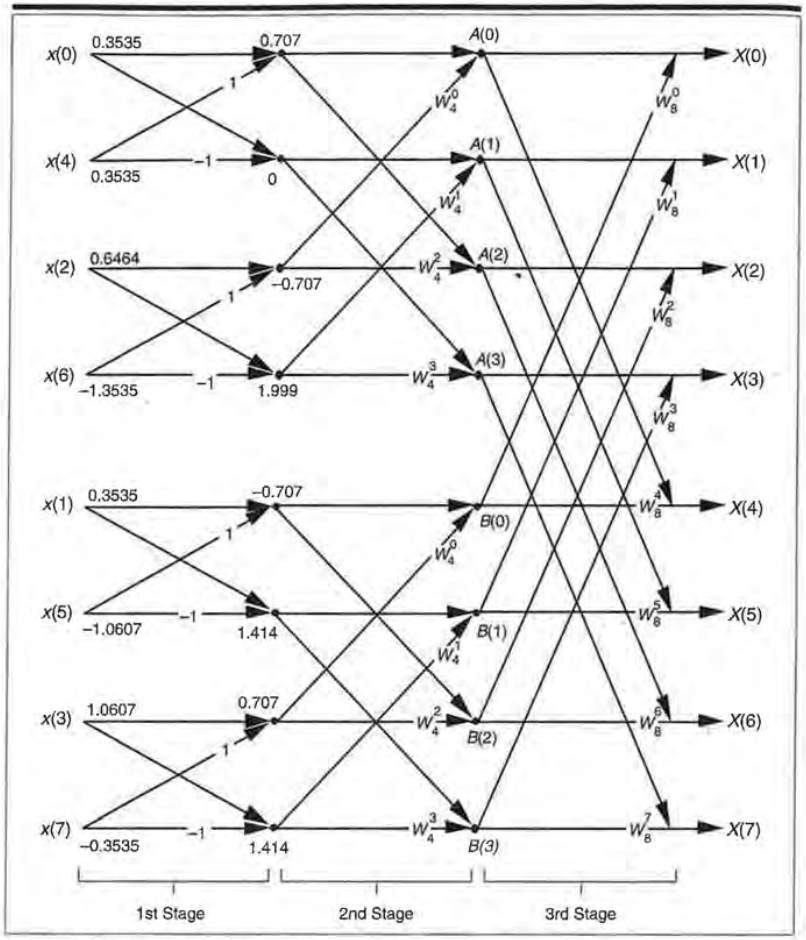


Figure 4-6 8-point FFT of Example 1 from Section 3.1.

$$X(4) = A(0) + W_8^4 B(0) = 0 + j0 + (-1 + j0)(0 + j0)$$
$$= 0 + j0 + 0 + j0 = 0 \angle 0°,$$

$$X(5) = A(1) + W_8^5 B(1) = 0 - j1.999 + (-0.707 + j0.707)(1.414 - j1.414)$$
$$= 0 - j1.999 + 0 + j1.999 = 0 \angle 0°,$$

$$X(6) = A(2) + W_8^6 B(2) = 1.414 + j0 + (0 + j1)(-1.414 + j0)$$
$$= 1.414 + j0 + 0 - j1.414 = 1.414 - j1.414 = 2 \angle -45°, \text{ and}$$

$$X(7) = A(3) + W_8^7 B(3) = 0 + j1.999 + (0.707 + j0.707)(1.414 + j1.414)$$
$$= 0 + j1.999 + 0 + j1.999 = 0 + j4 = 4 \angle 90°.$$

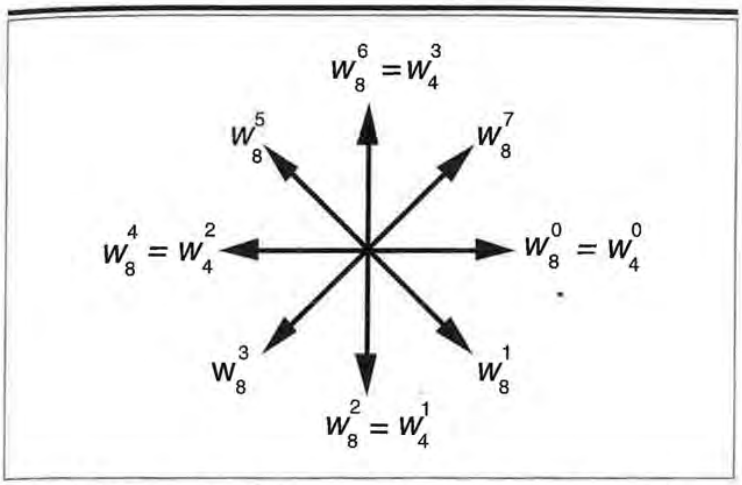


Figure 4-7 Cyclic redundancies in the twiddle factors of an 8-point FFT.

So, happily, the FFT gives us the correct results, and again we remind the reader that the FFT is not an approximation to a DFT; it is a DFT with a reduced number of necessary arithmetic operations. You've seen from the above example that the 8-point FFT example required less effort than the 8-point DFT Example 1 in Section 3.1. Some authors like to explain this arithmetic reduction by the redundancies inherent in the twiddle factors W_N^m. They illustrate this with the *starburst* pattern in Figure 4-7 showing the equivalencies of some of the twiddle factors in an 8-point DFT.

4.5 FFT Input/Output Data Index Bit Reversal

OK, let's look into some of the special properties of the FFT that are important to FFT software developers and FFT hardware designers. Notice that Figure 4-5 was titled "Full decimation-in-time FFT implementation of an 8-point DFT." The *decimation-in-time* phrase refers to how we broke the DFT input samples into odd and even parts in the derivation of Eqs. (4-20), (4-23), and (4-24). This time decimation leads to the scrambled order of the input data's index n in Figure 4-5. The pattern of this shuffled order can be understood with the help of Table 4-1. The shuffling of the input data is known as *bit reversal* because the scrambled order of the input data index can be obtained by reversing the bits of the binary representation of the normal input data index order. Sounds confusing, but it's really not—Table 4-1 illustrates the input index bit reversal for our 8-point FFT example. Notice the normal index order in the left

Table 4-1 Input Index Bit Reversal for an 8-point FFT

Normal order of index n	*Binary bits of index n*	*Reversed bits of index n*	*Bit-reversed order of index n*
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

column of Table 4-1 and the scrambled order in the right column that corresponds to the final decimated input index order in Figure 4-5. We've transposed the original binary bits representing the normal index order by reversing their positions. The most significant bit becomes the least significant bit and the least significant bit becomes the most significant bit, the next to the most significant bit becomes the next to the least significant bit, and the next to the least significant bit becomes the next to the most significant bit, and so on.[†]

4.6 Radix-2 FFT Butterfly Structures

Let's explore the butterfly signal flows of the decimation-in-time FFT a bit further. To simplify the signal flows, let's replace the twiddle factors in Figure 4-5 with their equivalent values referenced to W_N^m, where $N = 8$. We can show just the exponents m of W_N^m, to get the FFT structure shown in Figure 4-8. That is, W_4^1 from Figure 4-5 is equal to W_8^2 and is shown as a 2 in Figure 4-8, W_4^2 from Figure 4-5 is equal to W_8^4 and is shown as a 4 in Figure 4-8, etc. The 1s and –1s in the first stage of Figure 4-5 are replaced in Figure 4-8 by 0s and 4s, respectively. Other than the twiddle factor notation, Figure 4-8 is identical to Figure 4-5. We can shift around the signal nodes in Figure 4-5 and arrive at an 8-point decimation-in-time FFT as shown in Figure 4-9. Notice that the input data in Figure 4-9 is in its normal order and the output data indices are bit-reversed. In this case, a bit-reversal operation needs to be performed at the output of the FFT to unscramble the frequency-domain results.

[†] Many that are first shall be last; and the last first. [Mark 10:31]

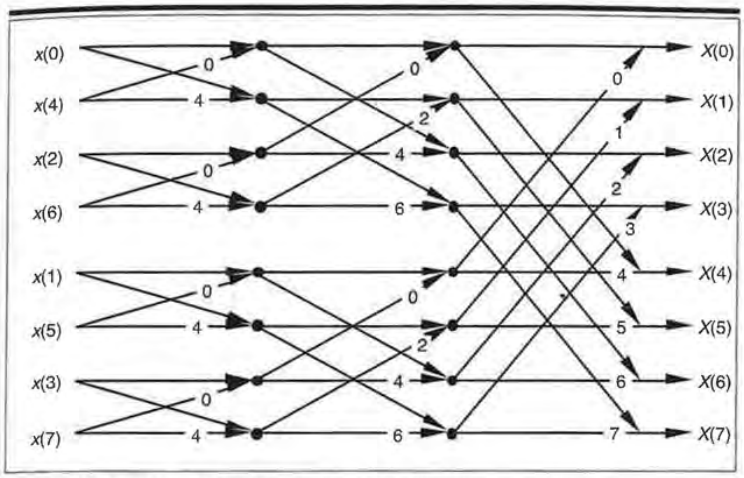


Figure 4-8 8-point decimation-in-time FFT with bit-reversed inputs.

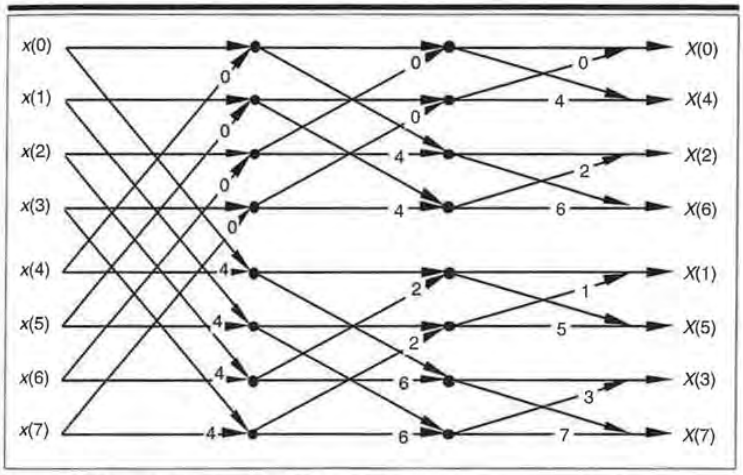


Figure 4-9 8-point decimation-in-time FFT with bit-reversed outputs.

Figure 4-10 shows an FFT signal-flow structure that avoids the bit-reversal problem altogether, and the graceful weave of the traditional FFT butterflies is replaced with a tangled, but effective, configuration.

Not too long ago, hardware implementations of the FFT spent most of their time (clock cycles) performing multiplications, and the bit-reversal process necessary to access data in memory wasn't a significant portion of

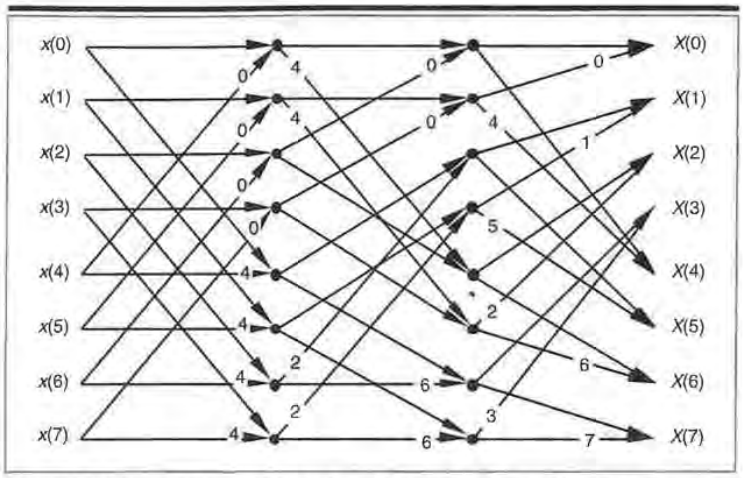


Figure 4-10 8-point decimation-in-time FFT with inputs and outputs in normal order.

the overall FFT computational problem. Now that high-speed multiplier/accumulator integrated circuits can multiply two numbers in a single clock cycle, FFT data multiplexing and memory addressing have become much more important. This has led to the development of efficient algorithms to perform bit reversal[12–15].

There's another derivation for the FFT that leads to butterfly structures looking like those we've already covered, but the twiddle factors in the butterflies are different. This alternate FFT technique is known as the decimation-in-frequency algorithm. Where the decimation-in-time FFT algorithm is based on subdividing the input data into its odd and even components, the decimation-in-frequency FFT algorithm is founded upon calculating the odd and even output frequency samples separately. The derivation of the decimation-in-frequency algorithm is straightforward and included in many tutorial papers and textbooks, so we won't go through the derivation here[4,5,15,16]. We will, however, illustrate decimation-in-frequency butterfly structures (analogous to the structures in Figures 4-8 through 4-10) in Figures 4-11 though 4-13.

So an equivalent decimation-in-frequency FFT structure exists for each decimation-in-time FFT structure. It's important to note that the number of necessary multiplications to implement the decimation-in-frequency FFT algorithms is the same as the number necessary for the decimation-in-time FFT algorithms. There are so many different FFT butterfly structures described in the literature, that it's easy to become confused about

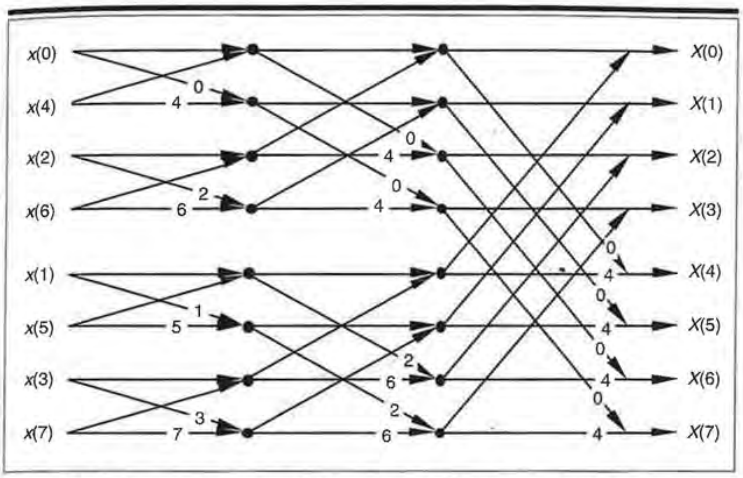


Figure 4-11 8-point decimation-in-frequency FFT with bit-reversed inputs

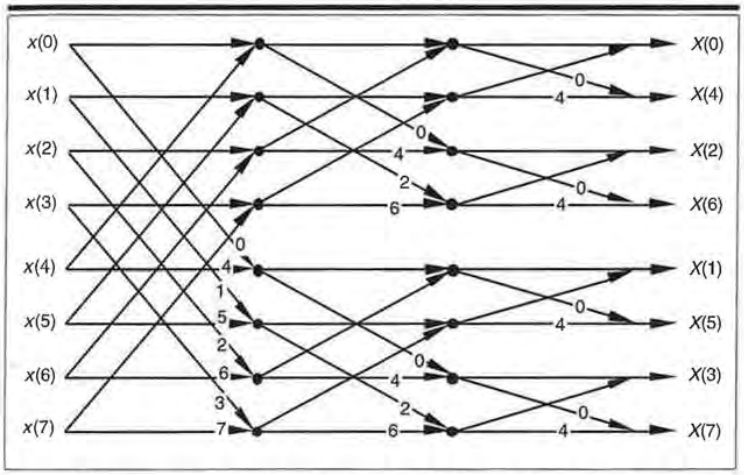


Figure 4-12 8-point decimation-in-frequency FFT with bit-reversed outputs.

which structures are decimation-in-time and which are decimation-in-frequency. Depending on how the material is presented, it's easy for a beginner to fall into the trap of believing that decimation-in-time FFTs always have their inputs bit-reversed and decimation-in-frequency FFTs always have their outputs bit-reversed. This is not true, as the above figures show. Decimation-in-time or -frequency is determined by whether

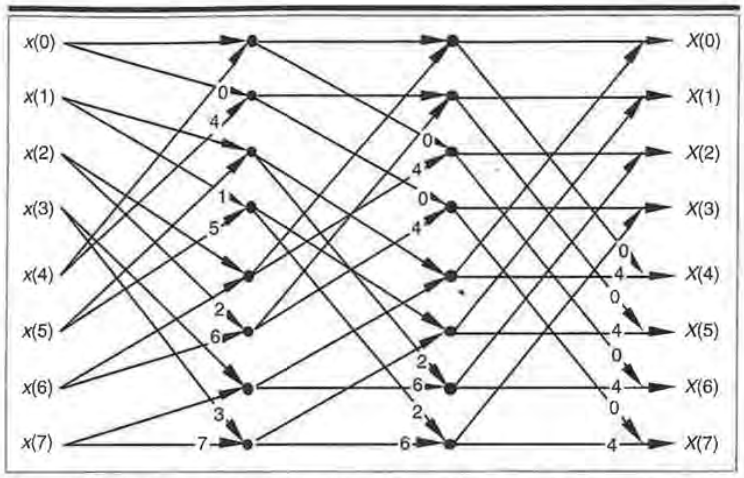


Figure 4-13 8-point decimation-in-frequency FFT with inputs and outputs in normal order.

the DFT inputs or outputs are partitioned when deriving a particular FFT butterfly structure from the DFT equations.

Let's take one more look at a single butterfly. The FFT butterfly structures in Figures 4-8, 4-9, 4-11, and 4-12 are the direct result of the derivations of the decimation-in-time and decimation-in-frequency algorithms. Although it's not very obvious at first, the twiddle factor exponents shown in these structures do have a consistent pattern. Notice how they always take the general forms shown in Figure 4-14(a).† To implement the decimation-in-time butterfly of Figure 4-14(a), we'd have to perform two complex multiplications and two complex additions. Well, there's a better way. Consider the decimation-in-time butterfly in Figure 4-14(a). If the top input is x and the bottom input is y, the top butterfly output would be

$$x' = x + W_N^k y\,, \tag{4-26}$$

and the bottom butterfly output would be

$$y' = x + W_N^{k+N/2} y\,. \tag{4-27}$$

† Remember, for simplicity the butterfly structures in Figures 4-8 through 4-13 show only the twiddle factor exponents, k and $k+N/2$, and not the entire complex twiddle factors.

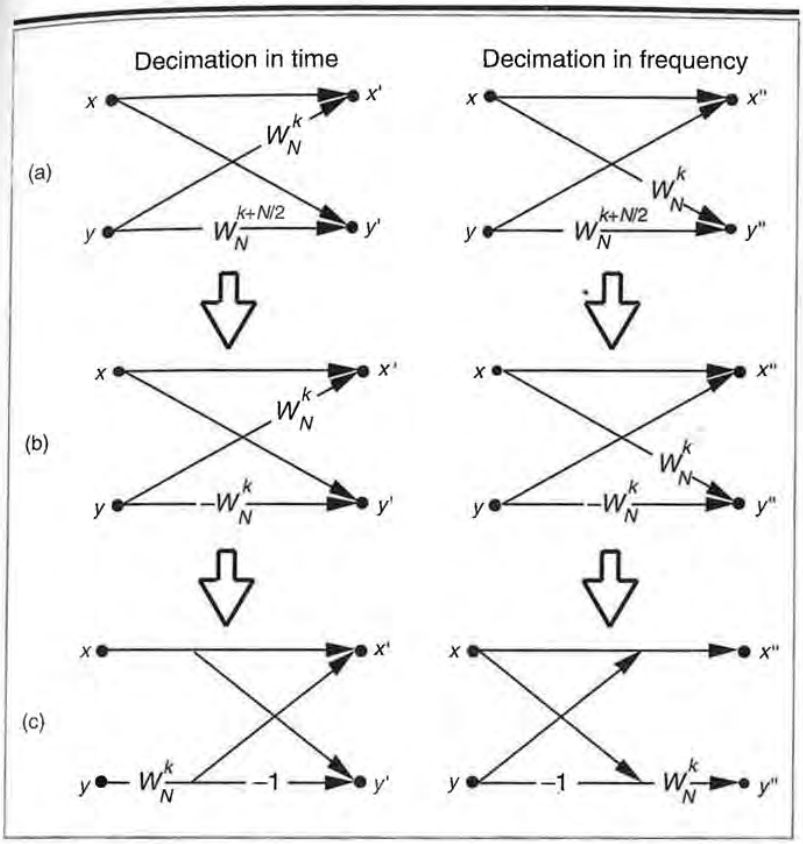


Figure 4-14 Decimation-in-time and decimation-in-frequency butterfly structures: (a) original form; (b) simplified form; (c) optimized form.

Fortunately, the operations in Eqs. (4-26) and (4-27) can be simplified because the two twiddle factors are related by

$$W_N^{k+N/2} = W_N^k W_N^{N/2} = W_N^k (e^{-j2\pi N/2N}) = W_N^k (-1) = -W_N^k . \qquad (4\text{-}28)$$

So we can replace the $W_N^{k+N/2}$ twiddle factors in Figure 4-14(a) with $-W_N^k$ to give us the simplified butterflies shown in Figure 4-14(b). Realizing that the twiddle factors in Figure 4-14(b) differ only by their signs, the optimized butterflies in Figure 4-14(c) can be used. Notice that these *optimized* butterflies require two complex additions but only

one complex multiplication, thus reducing our computational workload.†

We'll often see the optimized butterfly structures of Figure 4-14(c) in the literature instead of those in Figure 4-14(a). These optimized butterflies give us an easy way to recognize decimation-in-time and decimation-in-frequency algorithms. When we do come across the optimized butterflies from Figure 4-14(c), we'll know that the algorithm is decimation-in-time if the twiddle factor precedes the –1, or else the algorithm is decimation-in-frequency if the twiddle factor follows the –1.

Sometimes we'll encounter FFT structures in the literature that use the notation shown in Figure 4-15 [5, 17]. These wingless butterflies are equivalent to those shown in Figure 4-14(c). The signal-flow convention in Figure 4-15 is such that the plus output of a circle is the sum of the two samples that enter the circle from the left, and the minus output of a circle is the difference of the samples that enter the circle. So the outputs of the decimation-in-time butterflies in Figure 4-14(c) and Figure 4-15(a) are given by

$$x' = x + W_N^k y, \text{ and } y' = x - W_N^k y\,. \tag{4-29}$$

The outputs of the decimation-in-frequency butterflies in Figure 4-14(c) and Figure 4-15(b) are

$$x'' = x + y, \text{and } y'' = W_N^k(x - y) = W_N^k x - W_N^k y\,. \tag{4-30}$$

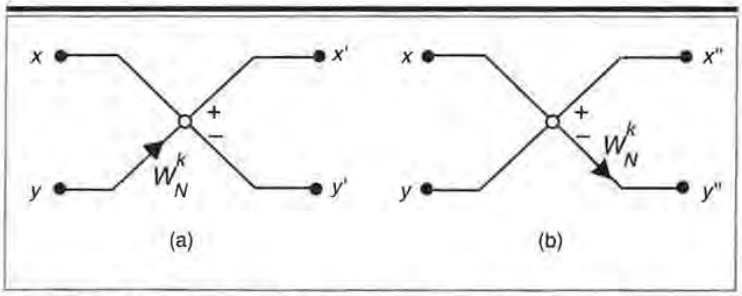


Figure 4-15 Alternate FFT butterfly notation: (a) decimation-in-time; (b) decimation-in-frequency.

† It's because there are $(N/2)\log_2 N$ butterflies in an N-point FFT that we said the number of complex multiplications performed by an FFT is $(N/2)\log_2 N$ in Eq. (4-2).

So which FFT structure is the best one to use? It depends on the application, the hardware implementation, and convenience. If we're using a software routine to perform FFTs on a general-purpose computer, we usually don't have a lot of choices. Most folks just use whatever existing FFT routines happen to be included in their commercial software package. Their code may be optimized for speed, but you never know. Examination of the software code may be necessary to see just how the FFT is implemented. If we *feel the need for speed*, we should check to see if the software calculates the sines and cosines each time it needs a twiddle factor. Trigonometric calculations normally take many machine cycles. It may be possible to speed up the algorithm by calculating the twiddle factors ahead of time and storing them in a table. That way, they can be *looked up*, instead of being calculated each time they're needed in a butterfly. If we're writing our own software routine, checking for butterfly output data overflow and careful magnitude scaling may allow our FFT to be performed using integer arithmetic that can be faster on some machines.[†] Care must be taken, however, when using integer arithmetic; some Reduced Instruction Set Computer (RISC) processors actually take longer to perform integer calculations because they're specifically designed to operate on floating-point numbers.

If we're using commercial array processor hardware for our calculations, the code in these processors is *always* optimized because their purpose in life is high speed. Array processor manufacturers typically publicize their products by specifying the speed at which their machines perform a 1024-point FFT. Let's look at some of our options in selecting a particular FFT structure in case we're designing special-purpose hardware to implement an FFT.

The FFT butterfly structures previously discussed typically fall into one of two categories: in-place FFT algorithms and double-memory FFT algorithms. An in-place algorithm is depicted in Figure 4-5. The output of a butterfly operation can be stored in the same hardware memory locations that previously held the butterfly's input data. No intermediate storage is necessary. This way, for an N-point FFT, only $2N$ memory locations are needed. (The 2 comes from the fact that each butterfly node represents a data value that has both a real and an imaginary part.) The rub with the in-place algorithms is that data routing and memory addressing is rather complicated. A double-memory FFT

[†] Overflow is what happens when the result of an arithmetic operation has too many bits, or digits, to be represented in the hardware registers designed to contain that result. FFT data overflow is described in Section 9.3.

structure is that depicted in Figure 4-10. With this structure, intermediate storage is necessary because we no longer have the standard butterflies, and $4N$ memory locations are needed. However, data routing and memory address control is much simpler in double-memory FFT structures than the in-place technique. The use of high-speed, floating-point integrated circuits to implement pipelined FFT architectures takes better advantage of their pipelined structure when the double-memory algorithm is used[18].

There's another class of FFT structures, known as constant-geometry algorithms, that make the addressing of memory both simple and constant for each stage of the FFT. These structures are of interest to those folks who build special-purpose FFT hardware devices[4,19]. From the standpoint of general hardware the decimation-in-time algorithms are optimum for real input data sequences, and decimation-in-frequency is appropriate when the input is complex[8]. When the FFT input data is symmetrical in time, special FFT structures exist to eliminate unnecessary calculations. These special butterfly structures based on input data symmetry are described in the literature[20].

For two-dimensional FFT applications, such as processing photographic images, the decimation-in-frequency algorithms appear to be the optimum choice[21]. Your application may be such that FFT input and output bit reversal is not an important factor. Some FFT applications allow manipulating a bit-reversed FFT output sequence in the frequency domain without having to unscramble the FFT's output data. Then an inverse transform that's expecting bit-reversed inputs will give a time-domain output whose data sequence is correct. This situation avoids the need to perform any bit reversals at all. Multiplying two FFT outputs to implement convolution or correlation are examples of this possibility.† As we can see, finding the optimum FFT algorithm and hardware architecture for an FFT is a fairly complex problem to solve, but the literature provides guidance[4,22,23].

References

[1] Cooley, J. and Tukey, J. "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comput.*, Vol. 19, No. 90, Apr. 1965, pp. 297–301.

[2] Cooley, J., Lewis, P., and Welch, P. "Historical Notes on the Fast Fourier Transform," *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-15, No. 2, June 1967.

† See Section 10.10 for an example of using the FFT to perform convolution.

[3] Harris, F. J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, Vol. 66, No. 1, pp. 54, January 1978.

[4] Oppenheim , A. V., and Schafer, R. W. *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989, pp. 608.

[5] Rabiner, L. R. and Gold, B. *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975, pp. 367.

[6] Stearns, S. *Digital Signal Analysis*, Hayden Book Co., Rochelle Park, New Jersey, 1975, pp. 265.

[7] *Programs for Digital Signal Processing*, Chapter 1, IEEE Press, New York, 1979.

[8] Sorenson, H. V., Jones, D. L., Heideman, M. T., and Burrus, C. S. "Real-Valued Fast Fourier Transform Algorithms," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-35, No. 6, June 1987.

[9] Bracewell, R. *The Fourier Transform and It's Applications*, 2nd Edition, Revised, McGraw-Hill, New York, 1986, pp. 405.

[10] Cobb, F. "Use Fast Fourier Transform Programs to Simplify, Enhance Filter Analysis," *EDN*, 8 March 1984.

[11] Carlin, F. "Ada and Generic FFT Generate Routines Tailored to Your Needs," *EDN*, 23 April 1992.

[12] Evans, D. "An Improved Digit-Reversal Permutation Algorithm for the Fast Fourier and Hartley Transforms," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-35, No. 8, August 1987.

[13] Burris, C. S. "Unscrambling for Fast DFT Algorithms," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. 36, No. 7, July 1988.

[14] Rodriguez, J. J. "An Improved FFT Digit-Reversal Algorithm," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-37, No. 8, August 1989.

[15] Land, A. "Bit Reverser Scrambles Data for FFT," *EDN*, March 2, 1995.

[16] JG-AE Subcommittee on Measurement Concepts, "What Is the Fast Fourier Transform?," *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-15, No. 2, June 1967.

[17] Cohen, R., and Perlman, R. "500 kHz Single-Board FFT System Incorporates DSP-Optimized Chips," *EDN*, 31 October 1984.

[18] Eldon, J., and Winter, G. E. "Floating-point Chips Carve Out FFT Systems," *Electronic Design*, 4 August 1983.

[19] Lamb, K. "CMOS Building Blocks Shrink and Speed Up FFT Systems," *Electronic Design*, 6 August 1987.

[20] Markel, J. D. "FFT Pruning," *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-19, No. 4, December 1971.

[21] Wu, H. R., and Paoloni, F. J. "The Structure of Vector Radix Fast Fourier Transforms," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-37, No. 8, August 1989.

[22] Ali, Z. M. "High Speed FFT Processor," *IEEE Trans. on Communications*, Vol. COM-26, No. 5, May 1978.

[23] Bergland, G. "Fast Fourier Transform Hardware Implementations—An Overview," *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-17, June 1969.

CHAPTER FIVE

Finite Impulse Response Filters

The filtering of digitized data, if not the most fundamental, is certainly the oldest discipline in the field of digital signal processing. Digital filtering's origins go back forty years. The growing availability of digital computers in the early 1950s led to efforts in the smoothing of discrete sampled data and the analysis of discrete data control systems. However it wasn't until the early to mid-1960s, around the time the Beatles came to America, that the analysis and development of digital equivalents of analog filters began in earnest. That's when digital signal processing experts realized that computers could go beyond the mere analysis of digitized signals into the domain of actually changing signal characteristics through filtering. Today, digital filtering is so widespread that the quantity of literature pertaining to it exceeds that of any other topic in digital signal processing. In this chapter, we introduce the fundamental attributes of digital filters, learn how to quantify their performance, and review the principles associated with the design of finite impulse response digital filters.

So let's get started by illustrating the concept of filtering a time-domain signal as shown in Figure 5-1.

In general, filtering is the processing of a time-domain signal resulting in some change in that signal's original spectral content. The change is usually the reduction, or filtering out, of some unwanted input spectral components; that is, filters allow certain frequencies to pass while attenuating other frequencies. Figure 5-1 shows both analog and digital versions of a filtering process. Where an analog filter operates on a continuous signal, a digital filter processes a sequence of discrete sample values. The digital filter in Figure 5-1(b), of course, can be a software program in a computer, a programmable hardware processor, or a dedicated integrated circuit. Traditional linear digital filters typically come in two flavors: finite

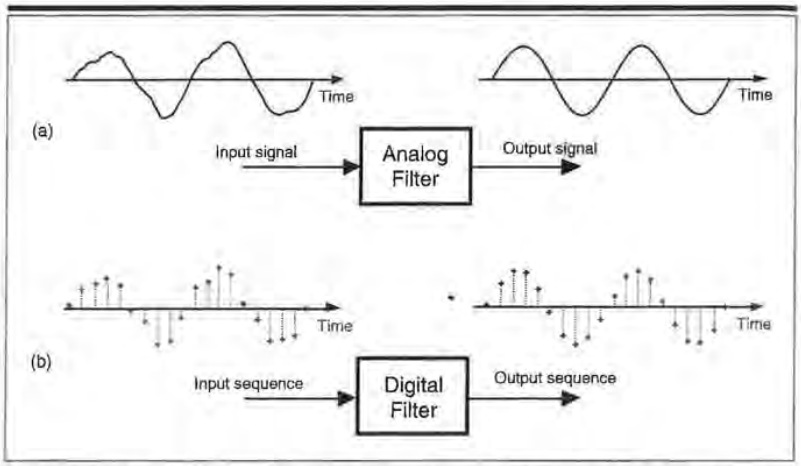


Figure 5-1 Filters: (a) an analog filter with a noisy tone input and a reduced-noise tone output; (b) the digital equivalent of the analog filter.

impulse response (FIR) filters and infinite impulse response (IIR) filters. Because FIR filters are the simplest type of digital filter to analyze, we'll examine them in this chapter and cover IIR filters in Chapter 6.

5.1 An Introduction to Finite Impulse Response FIR Filters

First of all, FIR digital filters use only current and past input samples, and none of the filter's previous output samples, to obtain a current output sample value. (That's also why FIR filters are sometimes called *nonrecursive* filters.) Given a finite duration of nonzero input values, the effect is that an FIR filter will always have a finite duration of nonzero output values and that's how FIR filters got their name. So, if the FIR filter's input suddenly becomes a sequence of all zeros, the filter's output will eventually be all zeros. While not sounding all that unusual, this characteristic is however, very important, and we'll soon find out why, as we learn more about digital filters.

FIR filters use addition to calculate their outputs in a manner much the same as the process of averaging uses addition. In fact, averaging is a kind of FIR filter that we can illustrate with an example. Let's say we're counting the number of cars that pass over a bridge every minute, and we need to know the average number of cars per minute over five-minute intervals; that is, every minute we'll calculate the average num-

Table 5-1 Values for the Averaging Example

Minute index	*Number of cars/minute over the last minute*	*Number of cars/minute averaged over the last five minutes*
1	10	–
2	22	–
3	24	–
4	42	–
5	37	27
6	77	40.4
7	89	53.8
8	22	53.4
9	63	57.6
10	9	52

ber of cars/minute over the last five minutes. If the results of our car counting for the first ten minutes are those values shown in the center column of Table 5-1, then the average number of cars/minute over the previous five one-minute intervals is listed in the right column of the table. We've added the number of cars for the first five one-minute intervals and divided by five to get our first five-minute average output value, (10+22+24+42+37)/5 = 27. Next we've averaged the number of cars/minute for the second to the sixth one-minute intervals to get our second five-minute average output of 40.4. Continuing, we average the number of cars/minute for the third to the seventh one-minute intervals to get our third average output of 53.8, and so on. With the number of cars/minute for the one-minute intervals represented by the dashed line in Figure 5-2, we show our five-minute average output as the solid line. (Figure 5-2 shows cars/minute input values beyond the first ten minutes listed in Table 5-1 to illustrate a couple of important ideas to be discussed shortly.)

There's much to learn from this simple averaging example. In Figure 5-2, notice that the sudden changes in our input sequence of cars/minute are flattened out by our averager. The averager output sequence is considerably smoother than the input sequence. Knowing that sudden transitions in a time sequence represent high frequency components, we can say that our averager is behaving like a low-pass filter and smoothing sudden changes in the input. Is our averager an FIR filter? It sure is—no previous averager output value is used to determine a current output

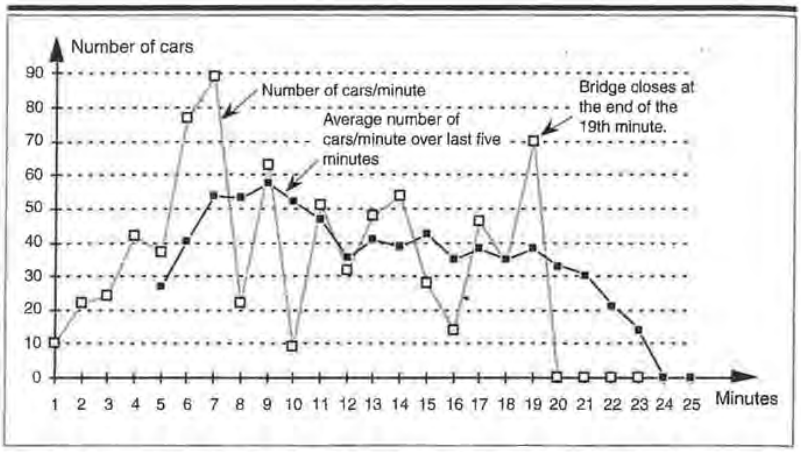


Figure 5-2 Averaging the number of cars/minute. The dashed line shows the individual cars/minute, and the solid line is the number of cars/minute averaged over the last five minutes.

value; only input values are used to calculate output values. In addition, we see that, if the bridge were suddenly closed at the end of the 19th minute, the dashed line immediately goes to zero cars/minute at the end of the 20th minute, and the averager's output in Figure 5-2 approaches and settles to a value of zero by the end of the 24th minute.

Figure 5-2 shows the first averager output sample occurring at the end of the 5th minute because that's when we first have five input samples to calculate a valid average. The 5th output of our averager can be denoted as $y_{ave}(5)$ where

$$y_{ave}(5) = \frac{1}{5}[x(1) + x(2) + x(3) + x(4) + x(5)] \quad . \tag{5-1}$$

In the general case, if the kth input sample is $x(k)$, then the nth output is

$$y_{ave}(n) = \frac{1}{5}[x(n-4) + x(n-3) + x(n-2) + x(n-1) + x(n)] = \frac{1}{5}\sum_{k=n-4}^{n} x(k) \quad . \tag{5-2}$$

Look at Eq. (5-2) carefully now. It states that the nth output is the average of the nth input sample and the four previous input samples.

We can formalize the digital filter nature of our averager by creating the block diagram in Figure 5-3 showing how the averager calculates its output samples.

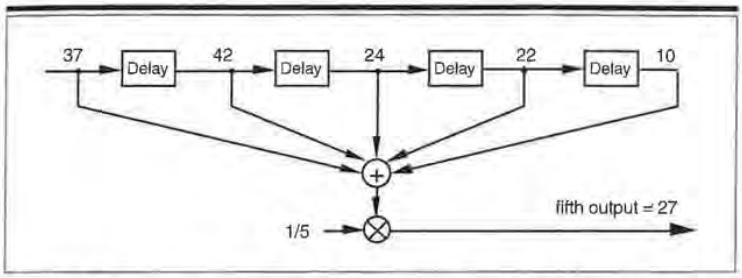


Figure 5-3 Averaging filter block diagram when the fifth input sample value, 37, is applied.

This block diagram, referred to as the filter *structure*, is a physical depiction of how we might calculate our averaging filter outputs with the input sequence of values shifted, in order, from left to right along the top of the filter as new output calculations are performed. This structure, implementing Eqs. (5-1) and (5-2), shows those values used when the first five input sample values are available. The delay elements in Figure 5-3, called *unit delays*, merely indicate a shift register arrangement where input sample values are temporarily stored during an output calculation.

In averaging, we add five numbers and divide the sum by five to get our answer. In a conventional FIR filter implementation, we can just as well multiply each of the five input samples by the coefficient 1/5 and then perform the summation as shown in Figure 5-4(a). Of course, the two methods in Figures 5-3 and 5-4(a) are equivalent because Eq. (5-2) describing the structure shown in Figure 5-3 is equivalent to

$$y_{ave}(n) = \frac{1}{5}x(n-4) + \frac{1}{5}x(n-3) + \frac{1}{5}x(n-2) + \frac{1}{5}x(n-1) + \frac{1}{5}x(n)$$

$$= \sum_{k=n-4}^{n} \frac{1}{5}x(k) \qquad (5\text{-}3)$$

that describes the structure in Figure 5-4(a).†

Let's make sure we understand what's happening in Figure 5-4(a). Each of the first five input values are multiplied by 1/5, and the five products are summed to give the 5th filter output value. The left to right

† We've used the venerable distributive law for multiplication and addition of scalars, $a(b+c+d) = ab+ac+ad$, in moving Eq. (5-2)'s factor of 1/5 inside the summation in Eq. (5-3).

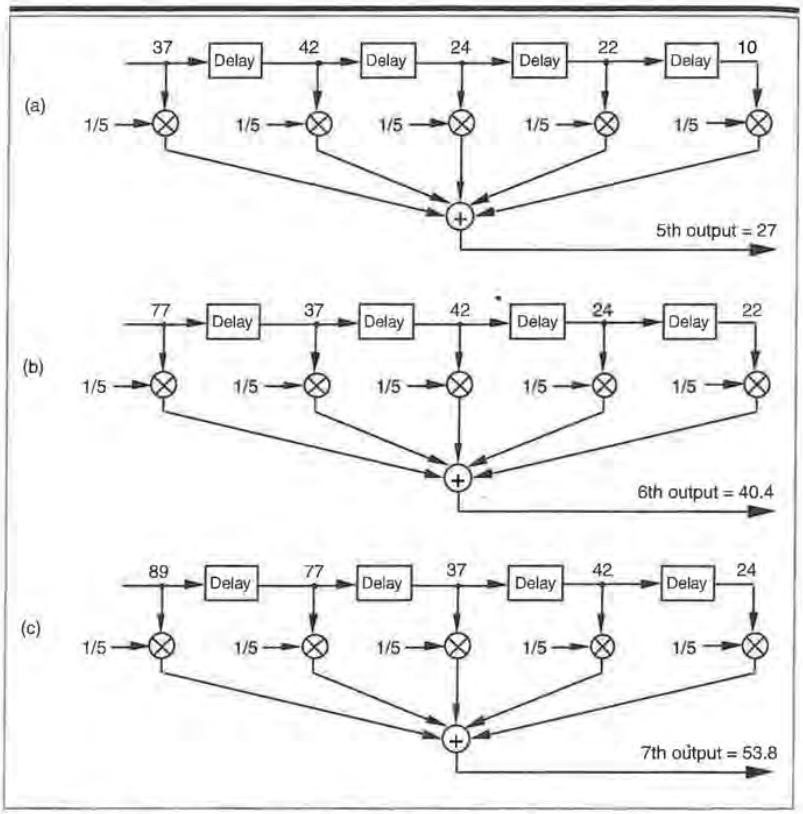


Figure 5-4 Alternate averaging filter structure: (a) input values used for the 5th output value; (b) input values used for the 6th output value; (c) input values used for the 7th output value.

sample shifting is illustrated in Figures 5-4(b) and 5-4(c). To calculate the filter's 6th output value, the input sequence is right shifted discarding the 1st input value of 10, and the 6th input value 77 is accepted on the left. Likewise, to calculate the filter's 7th output value, the input sequence is right shifted discarding the 2nd value of 22, and the 7th input value 89 arrives on the left. So, when a new input sample value is applied, the filter discards the oldest sample value, multiplies the samples by the coefficients of 1/5, and sums the products to get a single new output value. The filter's structure using this bucket brigade shifting process is often called a transversal filter due to the cross-directional flow of the input samples. Because we *tap off* five separate input sample

values to calculate an output value, the structure in Figure 5-4 is called a 5-tap FIR filter in digital filter vernacular.

One important and, perhaps, most interesting aspect of understanding FIR filters is learning how to predict their behavior when sinusoidal samples of various frequencies are applied to the input, i.e., how to estimate their frequency-domain response. Two factors affect an FIR filter's frequency response: the number of taps and the specific values used for the multiplication coefficients. We'll explore these two factors using our averaging example and, then, see how we can use them to design FIR filters. This brings us to the point where we have to introduce the C word: convolution. (Actually, we already slipped a convolution equation in on the reader without saying so. It was Eq. (5-3), and we'll examine it in more detail later.)

5.2 Convolution in FIR Filters

OK, here's where we get serious about understanding the mathematics behind FIR filters. We can graphically depict Eq. (5-3)'s and Figure 5-4's calculations as shown in Figure 5-5. Also, let's be formal and use the standard notation of digital filters for indexing the input samples and the filter coefficients by starting with an initial index value of zero; that is, we'll call the initial input value the 0th sample $x(0)$. The next input sample is represented by the term $x(1)$, the following input sample is called $x(2)$, and so on. Likewise, our five coefficient values will be indexed from zero to four, $h(0)$ through $h(4)$. (This indexing scheme makes the equations describing our example consistent with conventional filter notation found in the literature.)

In Eq. (5-3) we used the factor of 1/5 as the filter coefficients multiplied by our averaging filter's input samples. The left side of Figure 5-5 shows the alignment of those coefficients, black squares, with the filter input sample values represented by the white squares. Notice in Figure 5-5(a) through 5-5(e) that we're marching the input samples to the right, and, at each step, we calculate the filter output sample value using Eq. (5-3). The output samples on the right side of Figure 5-5 match the first five values represented by the black squares in Figure 5-2. The input samples in Figure 5-5 are those values represented by the white squares in Figure 5-2. Notice the time order of the inputs in Figure 5-5 has been reversed from the input sequence order in Figure 5-2! That is, the input sequence has been flipped in the time domain in Figure 5-5. This time order reversal is what happens to the input data using the filter structure in Figure 5-4.

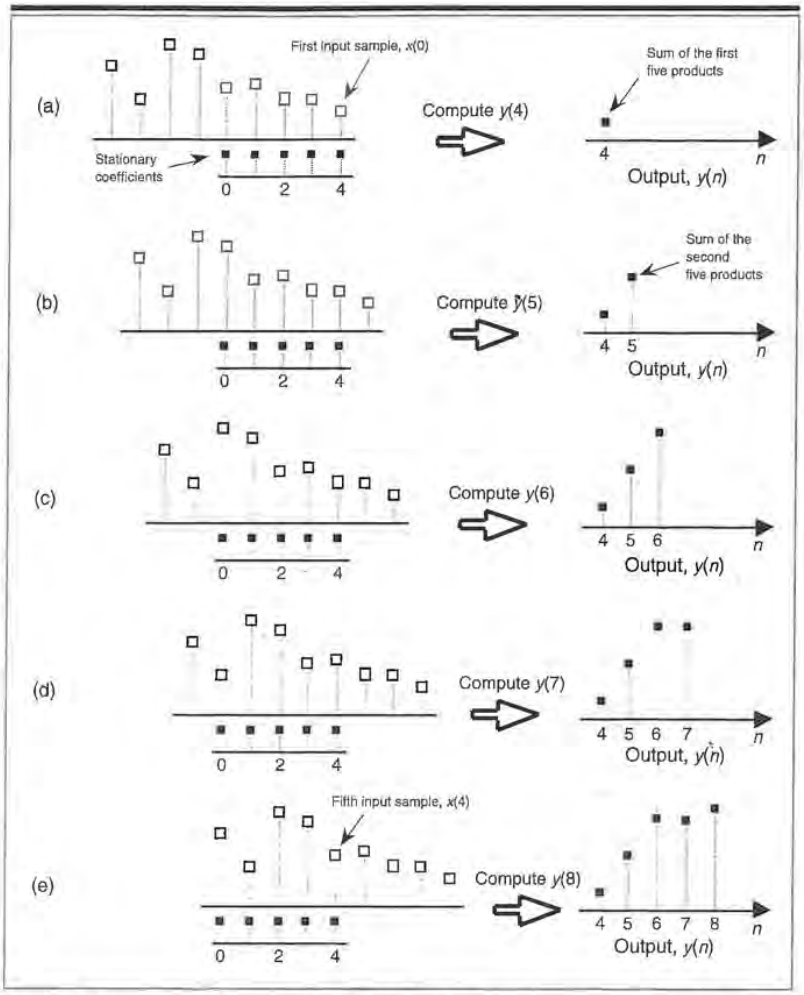


Figure 5-5 Averaging filter convolution: (a) first five input samples aligned with the stationary filter coefficients, index $n = 4$; (b) input samples shift to the right and index $n = 5$; (c) index $n = 6$; (d) index $n = 7$; (e) index $n = 8$.

Repeating the first part of Eq. (5-3) and omitting the subscript on the output term, our original FIR filter's $y(n)$th output is given by

$$y(n) = \frac{1}{5}x(n-4) + \frac{1}{5}x(n-3) + \frac{1}{5}x(n-2) + \frac{1}{5}x(n-1) + \frac{1}{5}x(n) \quad . \qquad (5\text{-}4)$$

Because we'll explore filters whose coefficients are not all the same value, we need to represent the individual filter coefficients by a variable, such as the term $h(k)$, for example. Thus we can rewrite the averaging filter's output from Eq. (5-4) in a more general way as

$$y(n) = h(4)x(n-4) + h(3)x(n-3) + h(2)x(n-2) + h(1)x(n-1) + h(0)x(n)$$

$$= \sum_{k=0}^{4} h(k)\,x(n-k) \ , \tag{5-5}$$

where $h(0)$ through $h(4)$ all equal 1/5. Equation (5-5) is a concise way of describing the filter structure in Figure 5-4 and the process illustrated in Figure 5-5.

Let's take Eq. (5-5) one step further and say, for a general M-tap FIR filter, the nth output is

$$y(n) = \sum_{k=0}^{M-1} h(k)\,x(n-k) \ . \tag{5-6}$$

Well, there it is. Eq. (5-6) is the infamous convolution equation as it applies to digital FIR filters. Beginners in the field of digital signal processing often have trouble understanding the concept of convolution. It need not be that way. Eq. (5-6) is merely a series of multiplications followed by the addition of the products. The process is actually rather simple. We just flip the time order of an input sample sequence and start stepping the flipped sequence across the filter's coefficients as shown in Figure 5-5. For each new filter input sample, we sum a series of products to compute a single filter output value.

Let's pause for a moment and introduce a new term that's important to keep in mind, the *impulse response*. The impulse response of a filter is exactly what its name implies—it's the filter's output time-domain sequence when the input is a single unity-valued sample (impulse) preceded and followed by zero-valued samples. Figure 5-6 illustrates this idea in the same way we determined the filter's output sequence in Figure 5-5. The left side of Figure 5-6 shows the alignment of the filter coefficients, black squares, with the filter input impulse sample values represented by the white squares. Again, in Figure 5-6(a) through 5-6(e) we're shifting the input samples to the right, and, at each step, we calculate the filter output sample value using Eq. (5-4). The output samples on the right side of

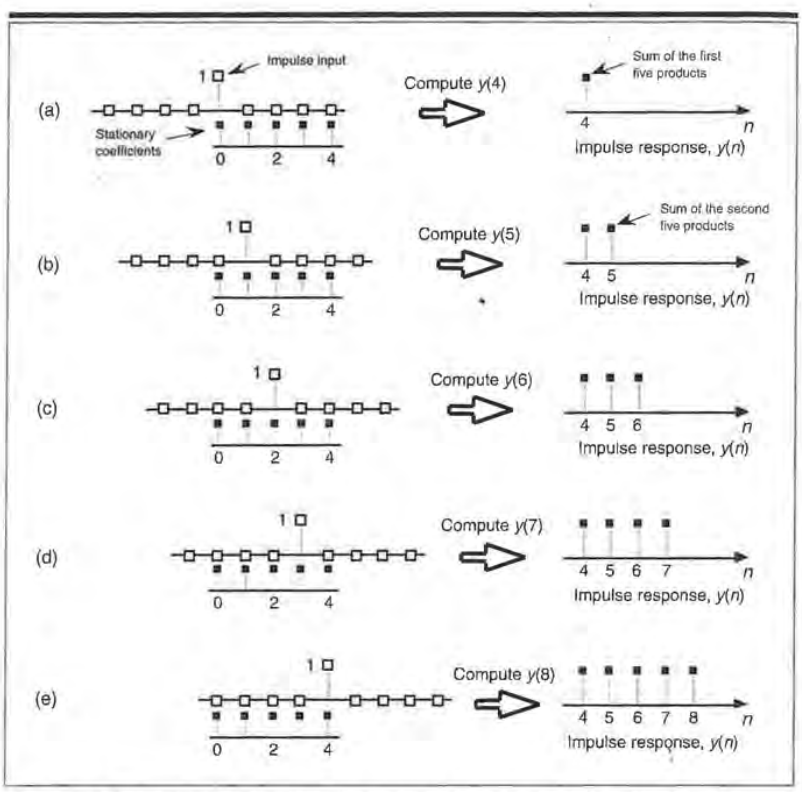


Figure 5-6 Convolution of filter coefficients and an input impulse to obtain the filter's output impulse response: (a) impulse sample aligned with the first filter coefficient, index $n = 4$; (b) impulse sample shifts to the right and index $n = 5$; (c) index $n = 6$; (d) index $n = 7$; (e) index $n = 8$.

Figure 5-6 are the filter's impulse response. Notice the key point here: the FIR filter's impulse response is identical to the five filter coefficient values. For this reason, the terms *FIR filter coefficients* and *impulse response* are synonymous. Thus, when someone refers to the impulse response of an FIR filter, they're also talking about the coefficients.

Returning to our averaging filter, recall that coefficients (or impulse response) $h(0)$ through $h(4)$ were all equal to 1/5. As it turns out, our filter's performance can be improved by using coefficients whose values are not all the same. By performance we mean how well the filter passes desired signals and attenuates unwanted signals. We judge that performance by determining the shape of the filter's frequency-domain response that we obtain by the convolution property of linear systems. To describe this concept, let's repeat Eq. (5-6) using the abbreviated notation of

$$y(n) = h(k) * x(n) \tag{5-7}$$

where the * symbol means convolution. (Equation 5-7 is read as "y of n equals the convolution of h of k and x of n.") The process of convolution, as it applies to FIR filters is as follows: The discrete Fourier transform (DFT) of the convolution of a filter's impulse response (coefficients), and an input sequence is equal to the product of the spectrum of the input sequence and the DFT of the impulse response. The idea we're trying to convey here is that if two time-domain sequences $h(k)$ and $x(n)$ have DFTs of $H(m)$ and $X(m)$, respectively, then the DFT of $y(n) = h(k) * x(n)$ is $H(m) \cdot X(m)$. Making this point in a more compact way, we state this relationship with the expression

$$y(n) = h(k) * x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) \cdot X(m). \tag{5-8}$$

With IDFT indicating the inverse DFT, Eq. (5-8) indicates that two sequences resulting from $h(k) * x(n)$ and $H(m) \cdot X(m)$ are Fourier transform pairs. So taking the DFT of $h(k) * x(n)$ gives us the product $H(m) \cdot X(m)$ that is the spectrum of our filter output $Y(m)$. Likewise, we can determine $h(k) * x(n)$ by taking the inverse DFT of $H(m) \cdot X(m)$. The very important conclusion to learn from Eq. (5-8) is that convolution in the time-domain is equivalent to multiplication in the frequency-domain. To help us appreciate this principle, Figure 5-7 sketches the relationship between convolution in the time domain and multiplication in the frequency domain. The process of convolution with regard to linear systems is discussed in more detail in Section 5.9. The beginner is encouraged to review that material to get a general idea of why and when the convolution process can be used to analyze digital filters.

Equation (5-8) and the relationships in Figure 5-7 tell us what we need to do to determine the frequency response of an FIR filter. The product $X(m) \cdot H(m)$ is the DFT of the filter output. Because $X(m)$ is the DFT of the filter's input sequence, the frequency response of the filter is then defined as $H(m)$, the DFT of filter's impulse response $h(k)$.[†] Getting back to our original problem, we can determine our averaging filter's frequency-domain response by taking the DFT of the individual filter coefficients (impulse response) in Eq. (5-4). If we take the five $h(k)$ coefficient values of 1/5 and append 59 zeros, we have the sequence depicted in Figure 5-8(a). Performing a 64-point DFT on that sequence, and normalizing the

[†] We use the *term impulse response* here, instead of *coefficients*, because this concept also applies to IIR filters. IIR filter frequency responses are also equal to the DFT of their impulse responses.

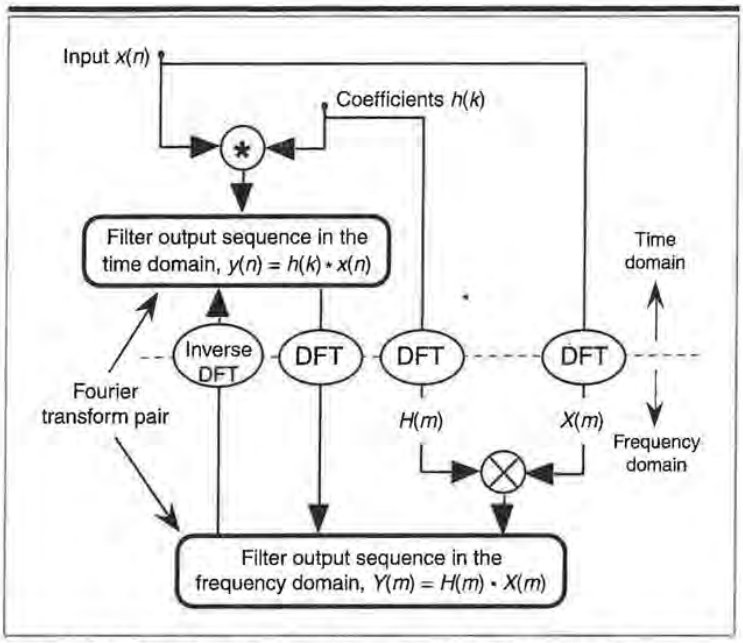


Figure 5-7 Relationships of convolution as applied to FIR digital filters.

DFT magnitudes, gives us the filter's frequency magnitude response $|H(m)|$ in Figure 5-8(b) and phase response shown in Figure 5-8(c).† $H(m)$ is our old friend, the $\sin(x)/x$ function from Section 3.13.

Let's relate the discrete frequency response samples in Figures 5-8(b) and 5-8(c) to the physical dimension of the sample frequency f_s. We know, from Section 3.5 and our experience with the DFT, that the $m = N/2$ discrete frequency sample, $m = 32$ in this case, is equal to the folding frequency, or half the sample rate, $f_s/2$. Keeping this in mind, we can convert the discrete frequency axis in Figure 5-8 to that shown in Figure 5-9. In Figure 5-9(a), notice that the filter's magnitude response is, of course, periodic in the frequency domain with a period of the equivalent sample rate f_s. Because we're primarily interested in the filter's response between 0 and half the sample rate, Figure 5-9(c) shows that frequency band in greater

† There's nothing sacred about using a 64-point DFT here. We could just as well have appended only enough zeros to take a 16- or 32-point FFT. We chose 64 points to get a frequency resolution that would make the shape of the response in Figure 5-8(b) reasonably smooth. Remember, the more points in the FFT, the finer the frequency resolution—right?

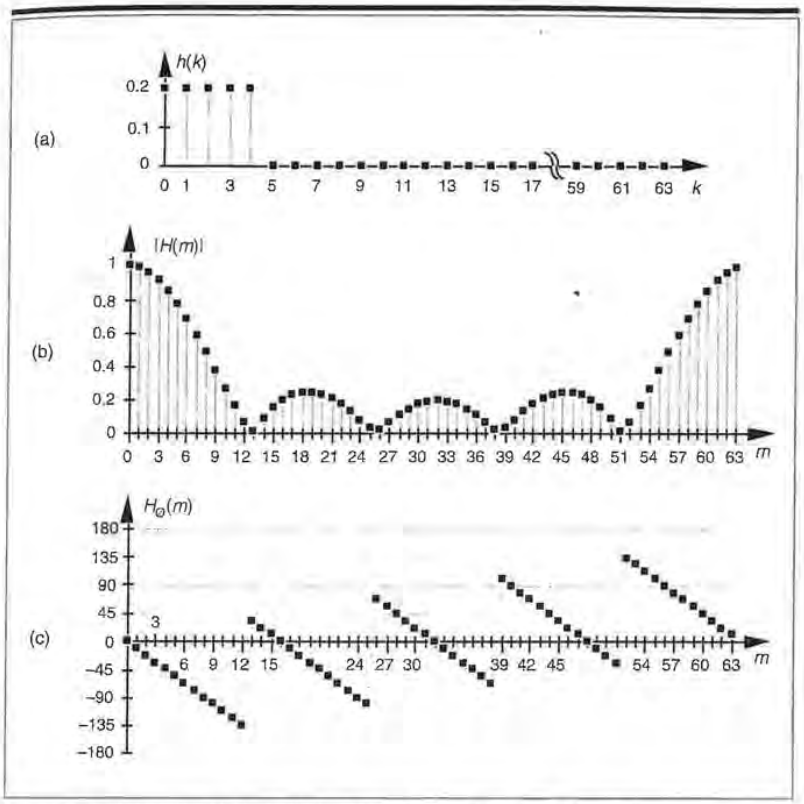


Figure 5-8 Averaging FIR filter: (a) filter coefficient sequence $h(k)$ with appended zeros; (b) normalized discrete frequency magnitude response $|H(m)|$ of the $h(k)$ filter coefficients; (c) phase-angle response of $H(m)$ in degrees.

detail affirming the notion that averaging behaves like a low-pass filter. It's a relatively poor low-pass filter compared to an arbitrary, *ideal* low-pass filter indicated by the dashed lines in Figure 5-9(c), but our averaging filter will attenuate higher frequency inputs relative to its response to low-frequency input signals.

We can demonstrate this by way of example. Suppose we applied a low-frequency sinewave to the averaging FIR filter, as shown in Figure 5-10(a), and that the input sinewave's frequency is $f_s/32$ and its peak amplitude is unity. The filter output in this case would be a sinewave of frequency $f_s/32$, but it's peak amplitude would be reduced to a value of 0.96, and the output sinewave's first sample value is delayed by a phase

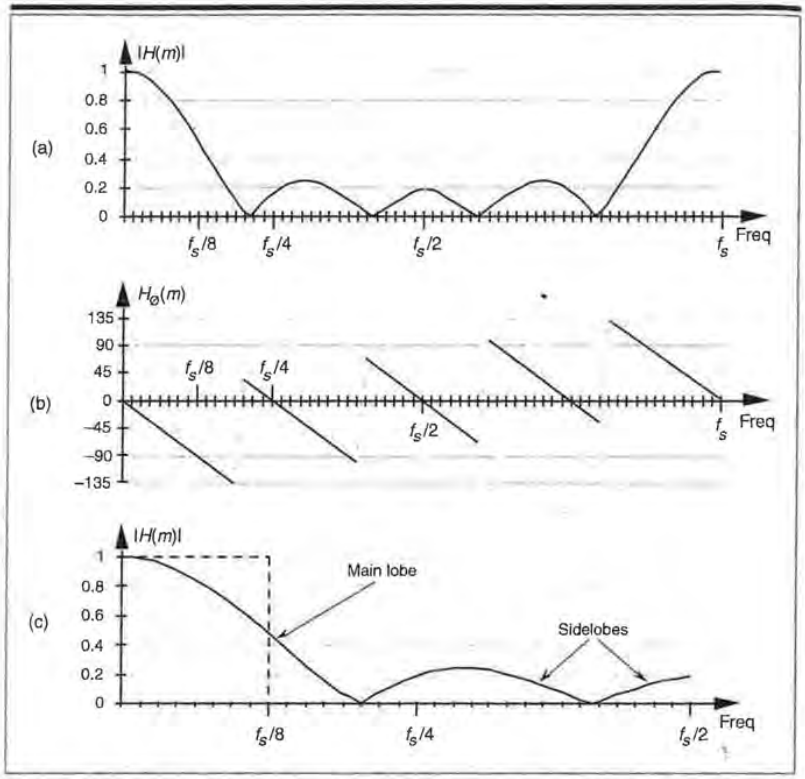


Figure 5-9 Averaging FIR filter frequency response shown as continuous curves: (a) normalized frequency magnitude response, $|H(m)|$; (b) phase angle response of $H(m)$ in degrees; (c) the filter's magnitude response between zero Hz and half the sample rate, $f_s/2$ Hz.

angle of 22.5°. Next, if we applied a higher frequency sinewave of $3f_s/32$ to the FIR filter as shown in Figure 5-10(b), the filter output would, then, be a sinewave of frequency $3f_s/32$, but it's peak amplitude is even further reduced to a value of 0.69. In addition, the Figure 5-10(b) output sinewave's first sample value has an even larger phase angle delay of 67.5°. Although the output amplitudes and phase delays in Figure 5-10 were measured values from actually performing a 5-tap FIR filter process on the input sinewave samples, we could have obtained those amplitude and phase delay values directly from Figures 5-8(a) and 5-8(b). The emphasis here is that we don't have to implement an FIR filter and apply various sinewave inputs to discover what its frequency

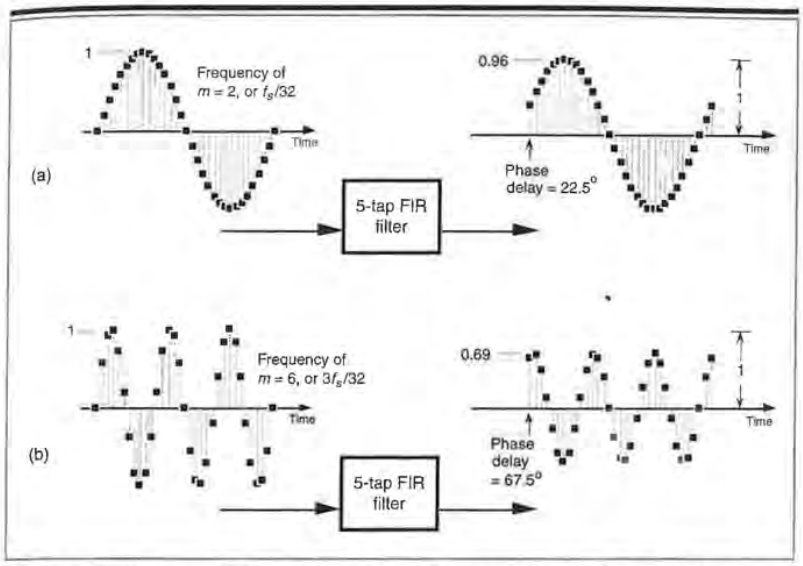


Figure 5-10 Averaging FIR filter input and output responses: (a) with an input sinewave of frequency $f_s/32$; (b) with an input sinewave of frequency $3f_s/32$.

response will be. We need merely take the DFT of the FIR filter's coefficients (impulse response) to determine the filter's frequency response as we did for Figure 5-8.

Figure 5-11 is another depiction of how well our 5-tap averaging FIR filter performs, where the dashed line is the filter's magnitude response $|H(m)|$, and the shaded line is the $|X(m)|$ magnitude spectrum of the filter's input values (the white squares in Figure 5-2). The solid line is the magnitude spectrum of the filter's output sequence which is shown by the black squares in Figure 5-2. So in Figure 5-11, the solid output spectrum is the product of the dashed filter response curve and the shaded input spectrum, or $|X(m) \cdot H(m)|$. Again, we see that our averager does indeed attenuate the higher frequency portion of the input spectrum.

Let's pause for a moment to let all of this soak in a little. So far we've gone through the averaging filter example to establish that

- FIR filters perform time-domain convolution by summing the products of the shifted input samples and a sequence of filter coefficients,
- an FIR filter's output sequence is equal to the convolution of the input sequence and a filter's impulse response (coefficients),

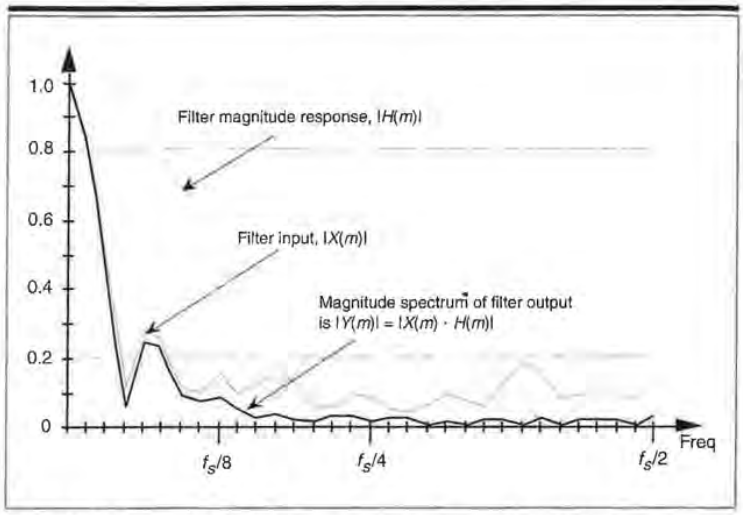


Figure 5-11 Averaging FIR filter input magnitude spectrum, frequency magnitude response, and output magnitude spectrum.

- an FIR filter's frequency response is the DFT of the filter's impulse response,[†]
- an FIR filter's output spectrum is the product of the input spectrum and the filter's frequency response, and
- convolution in the time domain and multiplication in the frequency domain are Fourier transform pairs.

OK, here's where FIR filters start to get really interesting. Let's change the values of the five filter coefficients to modify the frequency response of our 5-tap, low-pass filter. In fact, Figure 5-12(a) shows our original five filter coefficients and two other arbitrary sets of 5-tap coefficients. Figure 5-12(b) compares the frequency magnitude responses of those three sets of coefficients. Again, the frequency responses are obtained by taking the DFT of the three individual sets of coefficients and plotting the magnitude of the transforms, as we did for Figure 5-9(c). So we see three important characteristics in Figure 5-12. First, as we expected, different sets of coefficients give us different frequency magnitude responses. Second, a

[†] In Section 6.3, while treating an FIR filter as a special case of an IIR filter, we'll arrive at a mathematical expression for an FIR filter's frequency response in terms of its coefficient values.

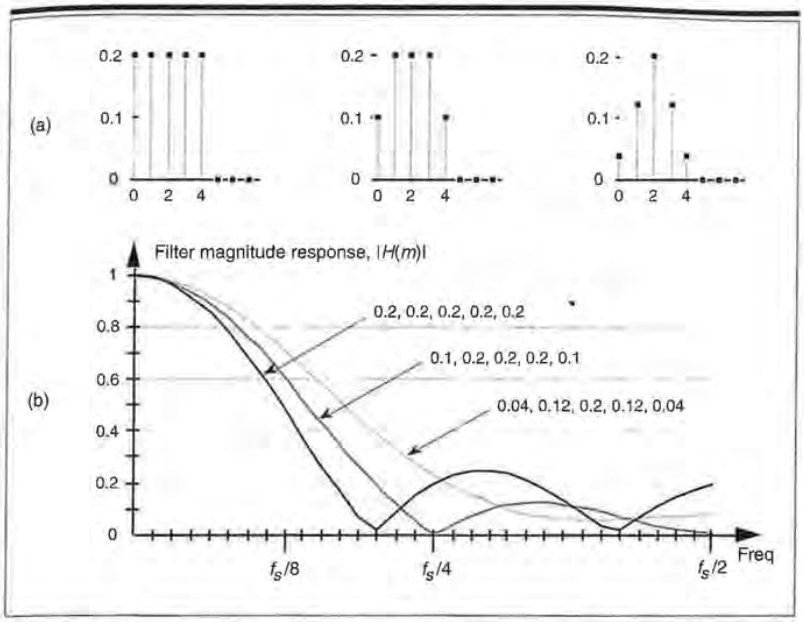


Figure 5-12 Three sets of 5-tap low-pass filter coefficients: (a) sets of coefficients; 0.2, 0.2, 0.2, 0.2, 0.2; 0.1, 0.2, 0.2, 0.2, 0.1; and 0.04, 0.12, 0.2, 0.12, 0.04; (b) frequency magnitude response of three low-pass FIR filters using those sets of coefficients.

sudden change in the values of the coefficient sequence, such as the 0.2 to 0 transition in the first coefficient set, causes ripples, or sidelobes, in the frequency response. Third, if we minimize the suddenness of the changes in the coefficient values, such as the third set of coefficients in Figure 5-12(a), we reduce the sidelobe ripples in the frequency response. However, reducing the sidelobes results in increasing the main lobe width of our low-pass filter. (As we'll see, this is exactly the same effect encountered in the discussion of window functions used with the DFT in Section 3.9.)

To reiterate the function of the filter coefficients, Figure 5-13 shows the 5-tap FIR filter structure using the third set of coefficients from Figure 5-12. The implementation of constant-coefficient transversal FIR filters does not get any more complicated than that shown in Figure 5-13. It's that simple. We can have a filter with more than five taps, but the input signal sample shifting, the multiplications by the constant coefficients, and the summation are all there is to it. (By constant coefficients, we don't mean coefficients whose values are all the same; we mean coefficients whose values remain unchanged, or time-invariant. There is a class of

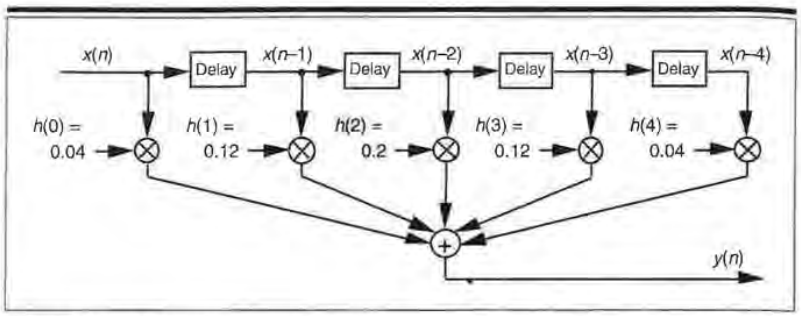


Figure 5-13 Five-tap, low-pass FIR filter implementation using the coefficients 0.04, 0.12, 0.2, 0.12, and 0.04.

digital filters, called adaptive filters, whose coefficient values are periodically changed to adapt to changing input signal parameters. While we won't discuss these adaptive filters in this introductory text, their descriptions are available in the literature[1–5].)

So far, our description of an FIR filter implementation has been presented from a hardware perspective. In Figure 5-13, to calculate a single filter output sample, five multiplications and five additions must take place before the arrival of the next input sample value. In a software implementation of a 5-tap FIR filter, however, all of the input data samples would be previously stored in memory. The software filter routine's job, then, is to access different five-sample segments of the $x(n)$ input data space, perform the calculations shown in Figure 5-13, and store the resulting filter $y(n)$ output sequence in an array of memory locations.†

Now that we have a basic understanding of what a digital FIR filter is, let's see what effect is had by using more than five filter taps by learning to design FIR filters.

5.3 Low-Pass FIR Filter Design

OK, instead of just accepting a given set of FIR filter coefficients and analyzing their frequency response, let's reverse the process and design our

† In reviewing the literature of FIR filters, the reader will often find the term z^{-1} replacing the delay function in Figure 5-13. This equivalence is explained in the next chapter when we study IIR filters.

own low-pass FIR filter. The design procedure starts with the determination of a *desired* frequency response followed by calculating the filter coefficients that will give us that response. There are two predominant techniques used to design FIR filters, the window method, and the so-called optimum method. Let's discuss them in that order.

5.3.1 Window Design Method

The window method of FIR filter design (also called the Fourier series method) begins with our deciding what frequency response we want for our low-pass filter. We can start by considering a continuous low-pass filter, and simulating that filter with a digital filter. We'll define the continuous frequency response $H(f)$ to be ideal, i.e., a low-pass filter with unity gain at low frequencies and zero gain (infinite attenuation) beyond some *cutoff frequency*, as shown in Figure 5-14(a). Representing this $H(f)$ response by a discrete frequency response is straightforward enough because the idea of a discrete frequency response is essentially the same as a continuous frequency response—with one important difference. As described in Sections 2.2 and 3.13, discrete frequency-domain representations are always periodic with the period being the sample rate f_s. The discrete representation of our ideal, continuous low-pass filter $H(f)$ is the periodic response $H(m)$ depicted by the frequency-domain samples in Figure 5-14(b).

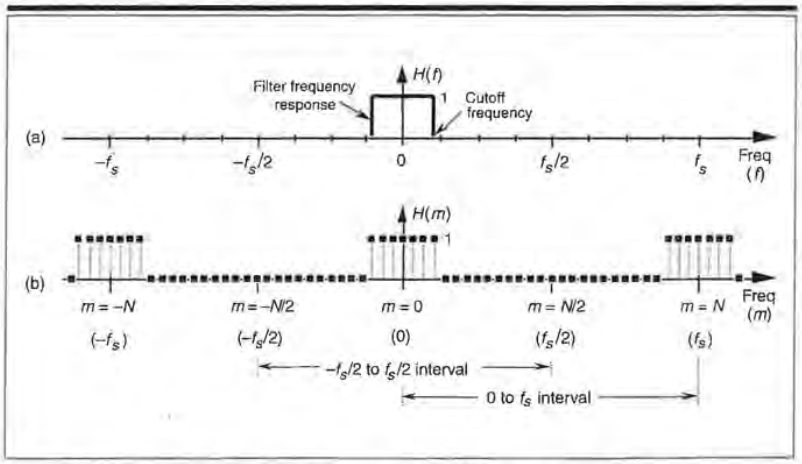


Figure 5-14 Low-pass filter frequency responses: (a) continuous frequency response $H(f)$; (b) periodic, discrete frequency response $H(m)$.

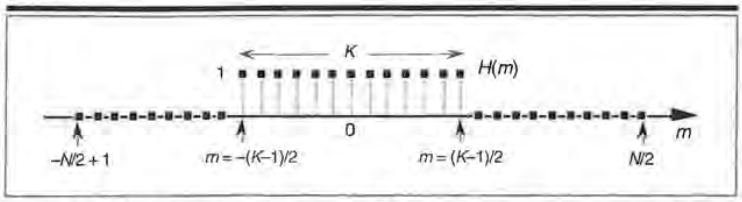


Figure 5-15 Arbitrary, discrete low-pass FIR filter frequency response defined over N frequency-domain samples covering the frequency range of f_s Hz.

We have two ways to determine our low-pass filter's time-domain coefficients. The first way is algebraic:

1. Develop an expression for the discrete frequency response $H(m)$.
2. Apply that expression to the inverse DFT equation to get the time domain $h(k)$.
3. Evaluate that $h(k)$ expression as a function of time index n.

The second method is to define the individual frequency-domain samples representing $H(m)$ and then have a software routine perform the inverse DFT of those samples, giving us the FIR filter coefficients. In either method, we need only define the periodic $H(m)$ over a single period of f_s Hz. As it turns out, defining $H(m)$ in Figure 5-14(b) over the frequency span $-f_s/2$ to $f_s/2$ is the easiest form to analyze algebraically, and defining $H(m)$ over the frequency span 0 to f_s is the best representation if we use the inverse DFT to obtain our filter's coefficients. Let's try both methods to determine the filter's time-domain coefficients.

In the algebraic method, we can define an arbitrary discrete frequency response $H(m)$ using N samples to cover the $-f_s/2$ to $f_s/2$ frequency range and establish K unity-valued samples for the passband of our low-pass filter as shown in Figure 5-15. To determine $h(k)$ algebraically we need to take the inverse DFT of $H(m)$ in the form of

$$h(k) = \frac{1}{N} \sum_{m=-(N/2)+1}^{N/2} H(m) e^{j2\pi mk/N} \qquad (5\text{-}9)$$

where our time-domain index is k. The solution to Eq. (5-9), derived in Section 3.13 as Eq. (3-59), is repeated here as

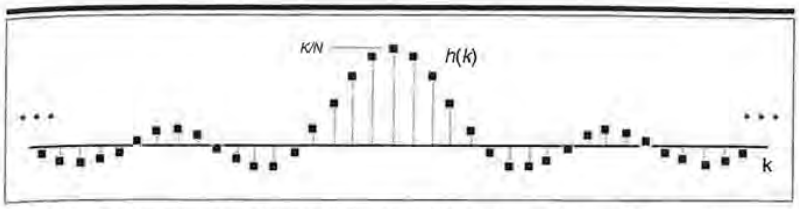


Figure 5-16 Time-domain $h(k)$ coefficients obtained by evaluating Eq. (5-10).

$$h(k) = \frac{1}{N} \cdot \frac{\sin(\pi kK / N)}{\sin(\pi k / N)}. \tag{5-10}$$

If we evaluate Eq. (5-10) as a function of k, we get the sequence shown in Figure 5-16 taking the form of the classic $\sin(x)/x$ function. By reviewing the material in Section 3.13, it's easy to see the great deal of algebraic manipulation required to arrive at Eq. (5-10) from Eq. (5-9). So much algebra, in fact, with its many opportunities for making errors, that digital filter designers like to avoid evaluating Eq. (5-9) algebraically. They prefer to use software routines to perform inverse DFTs (in the form of an inverse FFT) to determine $h(k)$, and so will we.

We can demonstrate the software inverse DFT method of FIR filter design with an example. Let's say we need to design a low-pass FIR filter simulating the continuous frequency response shown in Figure 5-17(a). The discrete representation of the filter's frequency response $H(m)$ is shown in Figure 5-17(b), where we've used $N = 32$ points to represent the frequency-domain variable $H(f)$. Because it's equivalent to Figure 5-17(b) but avoids the negative values of the frequency index m, we represent the discrete frequency samples over the range 0 to f_s in Figure 5-17(c), as opposed to the $-f_s/2$ to $+f_s/2$ range in Figure 5-17(b). OK, we're almost there. Using a 32-point inverse FFT to implement a 32-point inverse DFT of the $H(m)$ sequence in Figure 5-17(c), we get the 32 $h(k)$ values depicted by the dots from $k = -15$ to $k = 16$ in Figure 5-18(a).† We have one more step to perform. Because we want our final 31-tap $h(k)$ filter coefficients to be symmetrical with their peak value in the center of the coefficient sample set, we drop the $k = 16$ sample and shift the k index to the left from Figure 5-18(a) giving us the desired

† If you want to use this FIR design method but only have a forward FFT software routine available, Section 10.6 shows a slick way to perform an inverse FFT with the forward FFT algorithm.

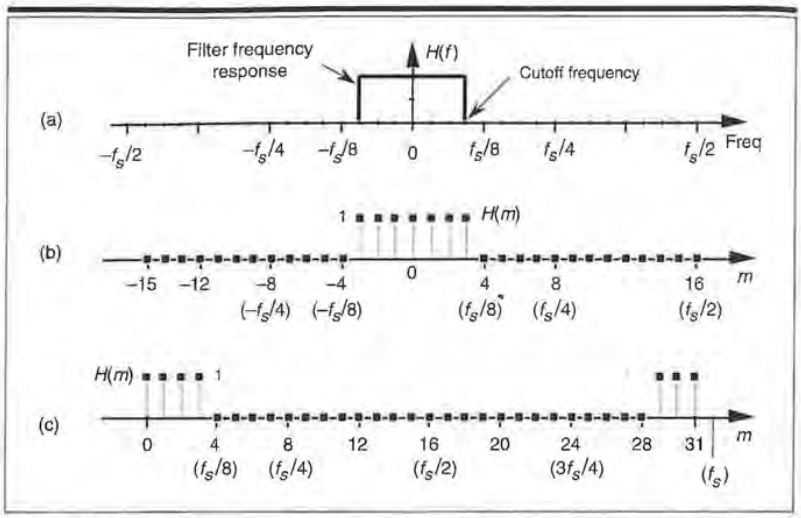


Figure 5-17 An ideal low-pass filter: (a) continuous frequency response $H(f)$; (b) discrete response $H(m)$ over the range of $-f_s/2$ to $f_s/2$ Hz; (c) discrete response $H(m)$ over the range 0 to f_s Hz.

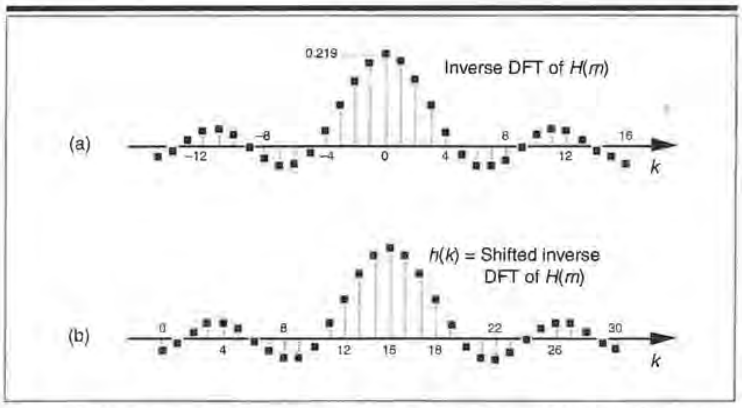


Figure 5-18 Inverse DFT of the discrete response in Figure 5-17(c): (a) normal inverse DFT indexing for k; (b) symmetrical coefficients used for a 31-tap low-pass FIR filter.

$\sin(x)/x$ form of $h(k)$ as shown in Figure 5-18(b). This shift of the index k will not change the frequency magnitude response of our FIR filter. (Remember from our discussion of the DFT Shifting Theorem in Section 3.6 that a shift in the time-domain manifests itself only as a linear phase shift in the frequency domain with no change in the frequency domain magnitude.) The sequence in Figure 5-18(b), then, is now the coeffi-

cients we use in the convolution process of Figure 5-5 to implement a low-pass FIR filter.

It's important to demonstrate that the more $h(k)$ terms we use as filter coefficients, the closer we'll approximate our ideal low-pass filter response. Let's be conservative, just use the center nine $h(k)$ coefficients, and see what our filter response looks like. Again, our filter's magnitude response in this case will be the DFT of those nine coefficients as shown on the right side of Figure 5-19(a). The ideal filter's frequency response is also shown for reference as the dashed curve. (To show the details of its shape, we've used a continuous curve for $|H(m)|$ in Figure 5-19(a), but we have to remember that $|H(m)|$ is really a sequence of discrete values.) Notice that using nine coefficients gives us a low-pass filter, but it's certainly far from ideal. Using more coefficients to improve our situation, Figure 5-19(b) shows 19 coefficients and their corresponding frequency magnitude response that is beginning to look more like our desired rectangular response. Notice that magnitude fluctuations, or ripples, are evident in the passband of our $H(m)$ filter response. Continuing, using all 31

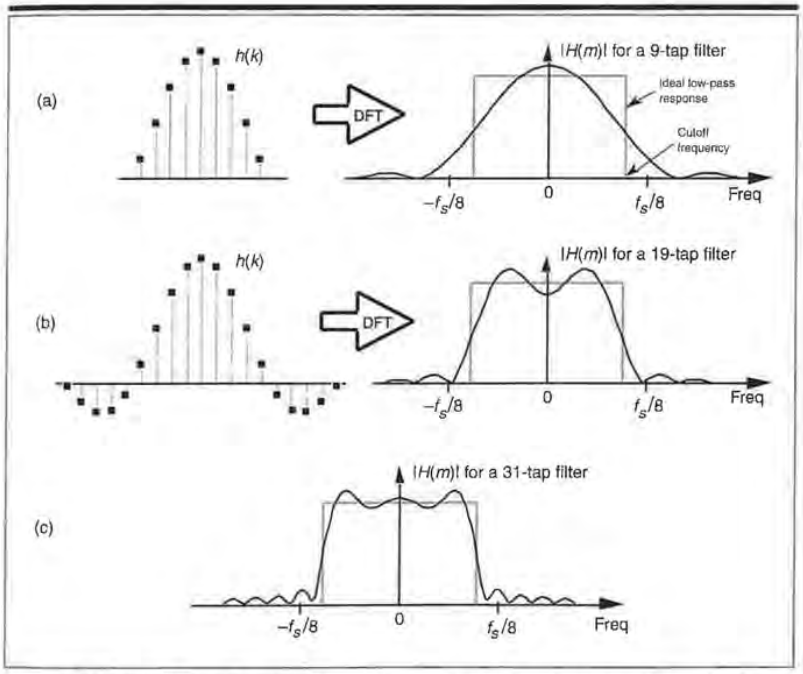


Figure 5-19 Coefficients and frequency responses of three low-pass filters: (a) 9-tap FIR filter; (b) a 19-tap FIR filter; (c) frequency response of the full 31-tap FIR filter.

of the $h(k)$ values for our filter coefficients results in the frequency response in Figure 5-19(c). Our filter's response is getting better (approaching the ideal), but those conspicuous passband magnitude ripples are still present.

It's important that we understand why those passband ripples are in the low-pass FIR filter response in Figure 5-19. Recall the above discussion of convolving the 5-tap averaging filter coefficients, or impulse response, with an input data sequence to obtain the averager's output. We established that convolution in the time domain is equivalent to multiplication in the frequency domain, that we symbolized with Eq. (5-8), and repeat it here as

$$h(k) * x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) \cdot X(m) . \tag{5-11}$$

This association between convolution in the time-domain and multiplication in the frequency domain, sketched in Figure 5-7, indicates that, if two time-domain sequences $h(k)$ and $x(n)$ have DFTs of $H(m)$ and $X(m)$, respectively, then the DFT of $h(k) * x(n)$ is $H(m) \cdot X(m)$. No restrictions whatsoever need be placed on what the time-domain sequences $h(k)$ and $x(n)$ in Eq. (5-11) actually represent. As detailed later in Section 5.9, convolution in one domain is equivalent to multiplication in the other domain allowing us to state that multiplication in the time domain is equivalent to convolution in the frequency domain, or

$$h(k) \cdot x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) * X(m) . \tag{5-12}$$

Now we're ready to understand why the magnitude ripples are present in Figure 5-19.

Rewriting Eq. (5-12) and replacing the $h(k)$ and $x(n)$ expressions with $h^{\infty}(k)$ and $w(k)$, respectively,

$$h^{\infty}(k) \cdot w(k) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H^{\infty}(m) * W(m) . \tag{5-13}$$

Let's say that $h^{\infty}(k)$ represents an infinitely long $\sin(x)/x$ sequence of ideal low-pass FIR filter coefficients and that $w(k)$ represents a window sequence that we use to truncate the $\sin(x)/x$ terms as shown in Figure 5-20. Thus, the $w(k)$ sequence is a finite-length set of unity values and its DFT is $W(m)$. The length of $w(k)$ is merely the number of coefficients, or taps, we intend

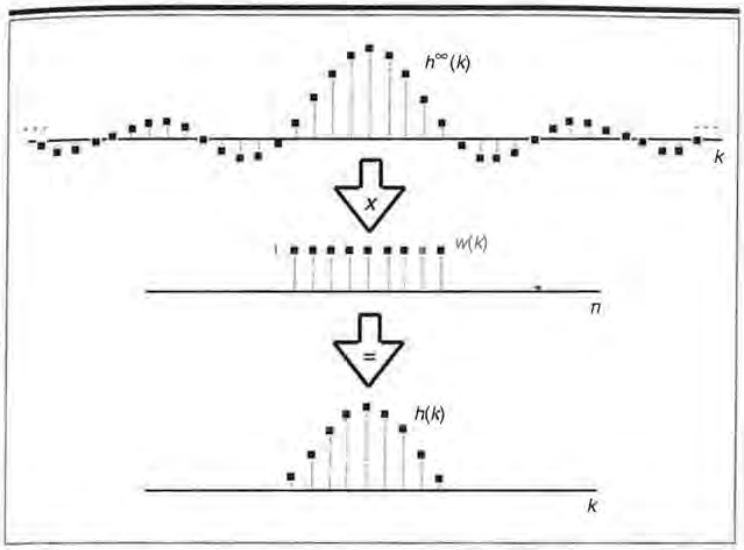


Figure 5-20 Infinite $h^{\infty}(k)$ sequence windowed by $w(k)$ to define the final filter coefficients $h(k)$.

to use to implement our low-pass FIR filter. With $h^{\infty}(k)$ defined as such, the product $h^{\infty}(k) \cdot w(k)$ represents the truncated set of filter coefficients $h(k)$ in Figures 5-19(a) and 5-19(b). So, from Eq. (5-13), the FIR filter's true frequency response $H(m)$ is the convolution

$$H(m) = H^{\infty}(m) * W(m) . \tag{5-14}$$

We depict this convolution in Figure 5-21 where, to keep the figure from being so busy, we show $H^{\infty}(m)$ (the DFT of the $h^{\infty}(k)$ coefficients) as the dashed rectangle. Keep in mind that it's really a sequence of constant-amplitude sample values.

Let's look at Figure 5-21(a) very carefully to see why all three $|H(m)|$s exhibit passband ripple in Figure 5-19. We can view a particular sample value of the $H(m) = H^{\infty}(m) * W(m)$ convolution as being the sum of the products of $H^{\infty}(m)$ and $W(m)$ for a particular frequency shift of $W(m)$. $H^{\infty}(m)$ and the unshifted $W(m)$ are shown in Figure 5-21(a.) With an assumed value of unity for all of $H^{\infty}(m)$, a particular $H(m)$ value is now merely the sum of the $W(m)$ samples that overlap the $H^{\infty}(m)$ rectangle. So, with a $W(m)$ frequency shift of 0 Hz, the sum of the $W(m)$ samples that overlap the $H^{\infty}(m)$ rectangle in Figure 5-21(a) is the value of $H(m)$ at 0 Hz.

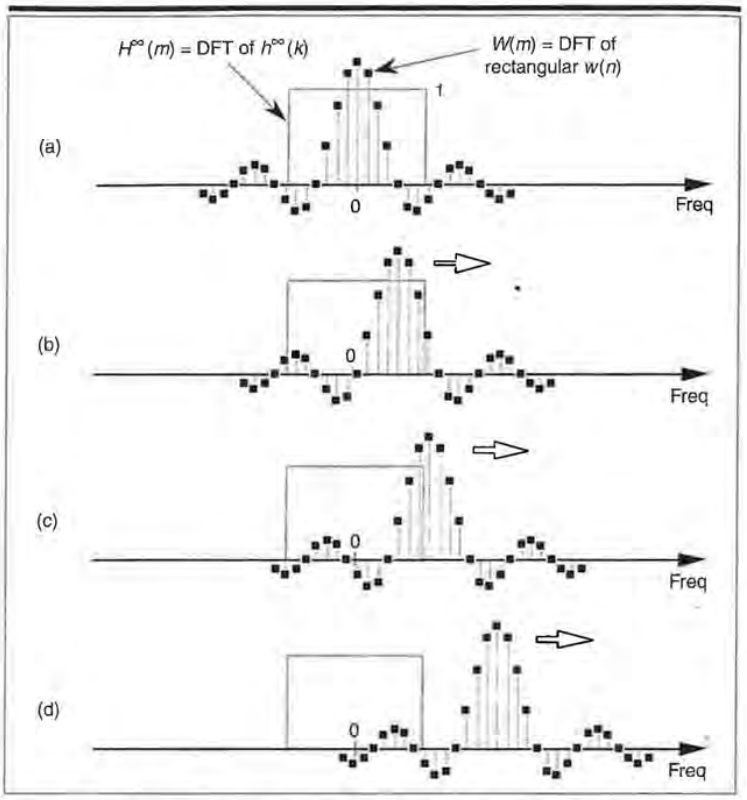


Figure 5-21 Convolution $W(m) * H^{\infty}(m)$: (a) unshifted $W(m)$ and $H^{\infty}(m)$; (b) shift of $W(m)$ leading to ripples within $H(m)$'s positive frequency passband; (c) shift of $W(m)$ causing response roll-off near $H(m)$'s positive cutoff frequency; (d) shift of $W(m)$ causing ripples beyond $H(m)$'s positive cutoff frequency.

As $W(m)$ is shifted to the right to give us additional positive frequency $H(m)$ values, we can see that the sum of the positive and negative values of $W(m)$ under the rectangle oscillate during the shifting of $W(m)$. As the convolution shift proceeds, Figure 5-21(b) shows why there are ripples in the passband of $H(m)$—again, the sum of the positive and negative $W(m)$ samples under the $H^{\infty}(m)$ rectangle continue to vary as the $W(m)$ function is shifted. The $W(m)$ frequency shift, indicated in Figure 5-21(c), where the peak of $W(m)$'s main lobe is now outside the $H^{\infty}(m)$ rectangle, corresponds to the frequency where $H(m)$'s passband begins to roll off. Figure 5-21(d) shows that, as the $W(m)$ shift continues, there will be ripples in $H(m)$

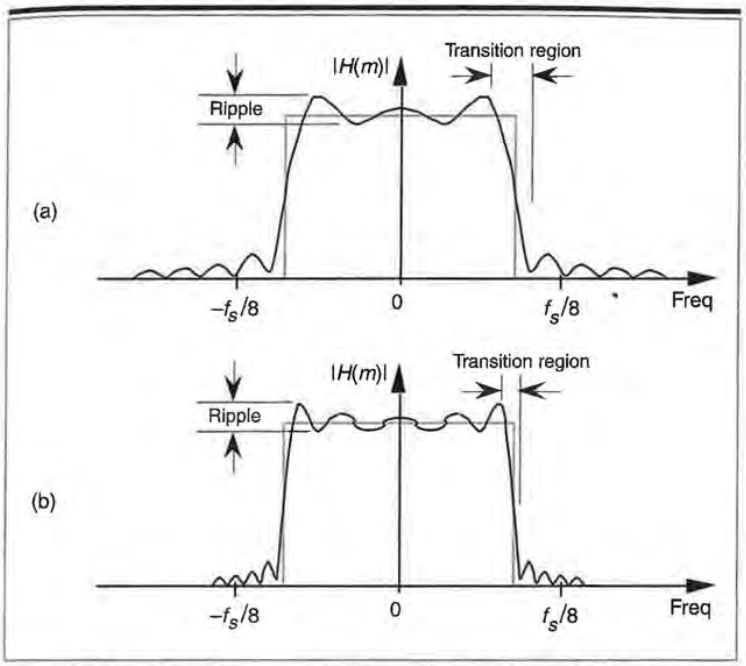


Figure 5-22 Passband ripple and transition regions: (a) for a 31-tap low-pass filter; (b) for a 63-tap low-pass filter.

beyond the positive cutoff frequency.† The point of all of this is that the ripples in $H(m)$ are caused by the sidelobes of $W(m)$.

Figure 5-22 helps us answer the question: How many $\sin(x)/x$ coefficients do we have to use (or how wide must $w(k)$ be) to get nice sharp falling edges and no ripples in our $H(m)$ passband? The answer is that we can't get there from here. It doesn't matter how many $\sin(x)/x$ coefficients (filter taps) we use, there will always be filter passband ripple. As long as $w(k)$ is a finite number of unity values (i.e., a rectangular window of finite width) there will be sidelobe ripples in $W(m)$, and this will induce passband ripples in the final $H(m)$ frequency response. To illustrate that increasing the number of $\sin(x)/x$ coefficients doesn't reduce passband ripple, we repeat the 31-tap, low-pass filter response in Figure 5-22(a). The frequency response, using 63 coefficients, is shown in

† In Figure 5-21(b), had we started to shift $W(m)$ to the left in order to determine the negative frequency portion of $H(m)$, we would have obtained the mirror image of the positive frequency portion of $H(m)$.

Figure 5-22(b), and the passband ripple remains. We can make the filter's transition region more narrow using additional $h(k)$ filter coefficients, but we cannot eliminate the passband ripple. That ripple, known as Gibbs' phenomenon, manifests itself anytime a function ($w(k)$ in this case) with a instantaneous discontinuity is represented by a Fourier series[6–8]. No finite set of sinusoids will be able to change fast enough to be exactly equal to an instantaneous discontinuity. Another way to state this Gibbs' dilemma is that, no matter how wide our $w(k)$ window is, its DFT of $W(m)$ will always have sidelobe ripples. As shown in Figure 5-22(b), we can use more coefficients by extending the width of the rectangular $w(k)$ to narrow the filter transition region, but a wider $w(k)$ does not eliminate the filter passband ripple nor does it even reduce their peak-to-peak ripple magnitudes, as long as $w(k)$ has sudden discontinuities.

5.3.2 Windows Used in FIR Filter Design

OK. The good news is that we can minimize FIR passband ripple with window functions the same way we minimized DFT leakage in Section 3.9. Here's how. Looking back at Figure 5-20, by truncating the infinitely long $h^{\infty}(k)$ sequence through multiplication by the rectangular $w(k)$, our final $h(k)$ exhibited ripples in the frequency-domain passband. Figure 5-21 shows us that the passband ripples were caused by $W(m)$'s sidelobes that, in turn, were caused by the sudden discontinuities from zero to one and one to zero in $w(k)$. If we think of $w(k)$ in Figure 5-20 as a rectangular window, then, it is $w(k)$'s abrupt amplitude changes that are the source of our filter passband ripple. The window FIR design method is the technique of reducing $w(k)$'s discontinuities by using window functions other than the rectangular window.

Consider Figure 5-23 to see how a nonrectangular window function can be used to design low-ripple FIR digital filters. Imagine if we replaced Figure 5-20's rectangular $w(k)$ with the Blackman window function whose discrete values are defined as[†]

$$\omega(k) = 0.42 - 0.5\cos\left(\frac{2\pi k}{N}\right) + 0.08\cos\left(\frac{4\pi k}{N}\right), \text{ for } k = 0, 1, 2, \ldots, N-1 \ . \quad (5\text{-}15)$$

[†] As we mentioned in Section 3.9, specific expressions for window functions depend on the range of the sample index k. Had we defined k to cover the range $-N/2, < k < N/2$, for example, the expression for the Blackman window would have a sign change and be $w(k) = 0.42 + 0.5\cos(2\pi k/N) + 0.08\cos(4\pi k/N)$.

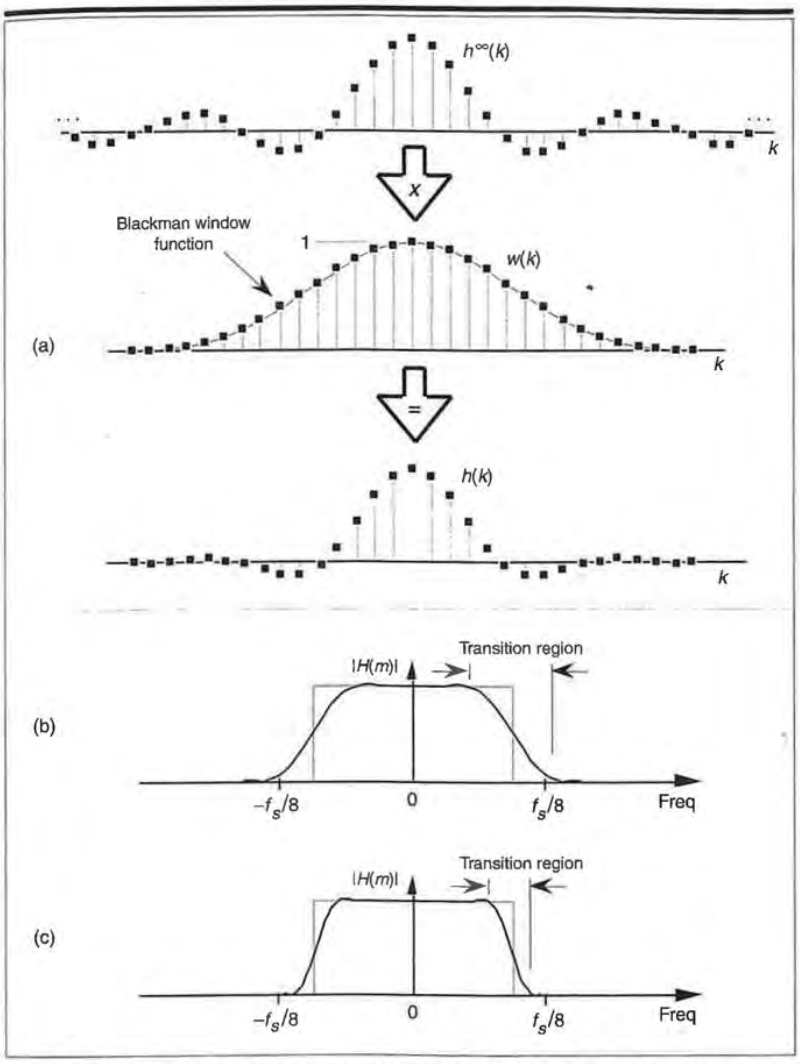


Figure 5-23 Coefficients and frequency response of a 31-tap Blackman-windowed FIR filter: (a) defining the windowed filter coefficients $h(k)$; (b) low-ripple 31-tap frequency response; (c) low-ripple 63-tap frequency response.

This situation is depicted for $N = 31$ in Figure 5-23(a) where Eq. (5-15)'s $w(k)$ looks very much like the Hanning window function in Figure 3-17(a). This Blackman window function results in the 31 smoothly tapered $h(k)$ coefficients at the bottom of Figure 5-23(a). Notice two things about the

resulting $H(m)$ in Figure 5-23(b). First, the good news. The passband ripples are greatly reduced from those evident in Figure 5-22(a)—so our Blackman window function did its job. Second, the price we paid for reduced passband ripple is a wider $H(m)$ transition region. We can get a steeper filter response roll-off by increasing the number of taps in our FIR filter. Figure 5-23(c) shows the improved frequency response had we used a 63-coefficient Blackman window function for a 63-tap FIR filter. So using a nonrectangular window function reduces passband ripple at the expense of slower passband to stopband roll-off.

A graphical comparison of the frequency responses for the rectangular and Blackman windows is provided in Figure 5-24. (The curves in Figure 5-24 were obtained for the window functions defined by 32 discrete samples, to which 480 zeros were appended, applied to a 512-point DFT.) The

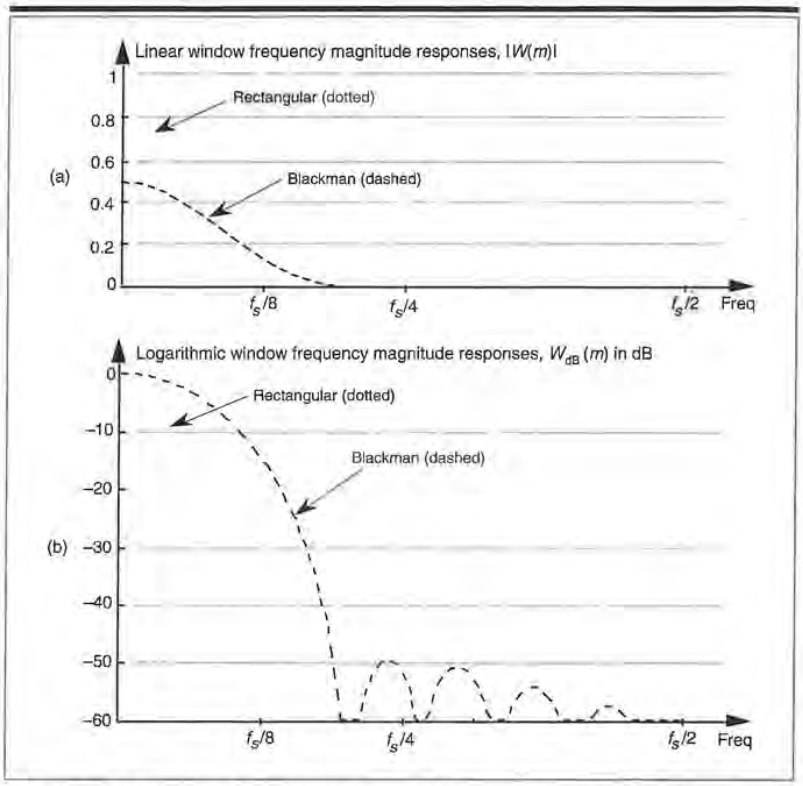


Figure 5-24 Rectangular vs. Blackman window frequency magnitude responses: (a) $|W(m)|$ on a linear scale; (b) normalized logarithmic scale of $W_{dB}(m)$.

sidelobe magnitudes of the Blackman window's $|W(m)|$ are too small to see on a linear scale. We can see those sidelobe details by plotting the two windows' frequency responses on a logarithmic scale and normalizing each plot so that their main lobe peak values are both zero dB. For a given window function, we can get the log magnitude response of $W_{dB}(m)$ by using the expression

$$W_{dB}(m) = 20 \cdot \log_{10}\left(\frac{|W(m)|}{|W(0)|}\right). \quad (5\text{-}16)$$

(The $|W(0)|$ term in Eq. (5-16) is the magnitude of $W(m)$ at the peak of the main lobe when $m = 0$.) Figure 5-24(b) shows us the greatly reduced sidelobe levels of the Blackman window and how that window's main lobe is almost three times as wide as the rectangular window's main lobe.

Of course, we could have used any of the other window functions, discussed in Section 3.9, for our low-pass FIR filter. That's why this FIR filter design technique is called the window design method. We pick a window function and multiply it by the $\sin(x)/x$ values from $H^{\infty}(m)$ in Figure 5-23(a) to get our final $h(k)$ filter coefficients. It's that simple. Before we leave the window method of FIR filter design, let's introduce two other interesting window functions.

Although the Blackman window and those windows discussed in Section 3.9 are useful in FIR filter design, we have little control over their frequency responses; that is, our only option is to select some window function and accept its corresponding frequency response. Wouldn't it be nice to have more flexibility in trading off, or striking a compromise between, a window's main lobe width and sidelobe levels? Fortunately, there are two popular window functions that give us this opportunity. Called the Chebyshev and the Kaiser window functions, they're defined by the following formidable expressions:

$$w(k) = \text{the } N\text{-point inverse DFT of}$$

Chebyshev window:→ (also called the Dolph–Chebyshev and the Tchebyschev window)

$$\frac{\cos\left[N \cdot \cos^{-1}\left[\alpha \cdot \cos\left(\pi \frac{m}{N}\right)\right]\right]}{\cosh\left[N \cdot \cosh^{-1}(\alpha)\right]},$$

$$\text{where } \alpha = \cosh\left(\frac{1}{N}\cosh^{-1}(10^{\gamma})\right) \text{ and } m = 0, 1, 2, \ldots, N-1, \quad (5\text{-}17)$$

Kaiser window:→
(also called the
Kaiser–Bessel window)

$$\omega(k) = \frac{I_o\left[\beta\sqrt{1-\left(\frac{k-p}{p}\right)^2}\right]}{I_o(\beta)},$$

for $k = 0, 1, 2, \ldots, N-1$, and $p = (N-1)/2$. (5-18)

Two typical Chebyshev and Kaiser window functions and their frequency magnitude responses are shown in Figure 5-25. For comparison, the rectangular and Blackman window functions are also shown in that figure. (Again, the curves in Figure 5-25(b) were obtained for window

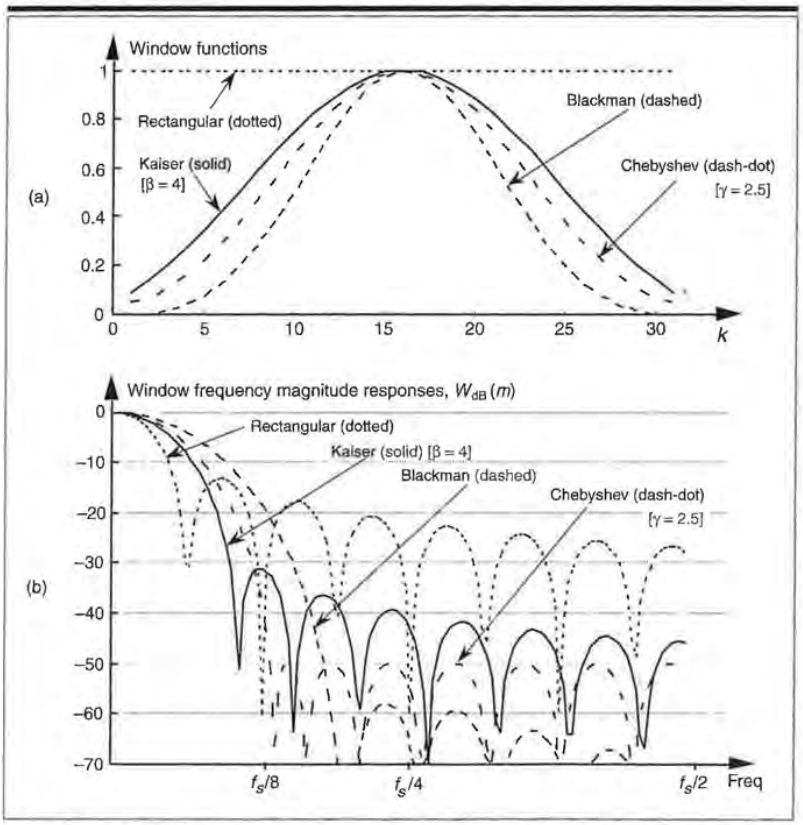


Figure 5-25 Typical window functions used with digital filters: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

functions defined by 32 discrete samples, with 480 zeros appended, applied to a 512-point DFT.)

Equation (5-17) was originally based on the analysis of antenna arrays using the mathematics of Chebyshev polynomials[9–11]. Equation (5-18) evolved from Kaiser's study of prolate spheroid functions using zeroth-order Bessel functions[12–13]. Don't be intimidated by the complexity of Eqs. (5-17) and (5-18)—at this point, we need not be concerned with the mathematical details of their development. We just need to realize that the *control* parameters γ and β, in Eqs. (5-17) and (5-18), give us control over the windows' main lobe widths and the sidelobe levels.

Let's see how this works for Chebyshev window functions, having four separate values of γ, and their frequency responses shown in Figure 5-26. FIR filter designers applying the window method typically use predefined software routines to obtain their Chebyshev window coefficients. Commercial digital signal processing software packages allow the user to specify three things: the window function (Chebyshev in this case), the desired number of coefficients (the number of taps in the FIR filter), and the value of γ. Selecting different values for γ enables us to adjust the sidelobe levels and see what effect those values have on main lobe width, a capability that we didn't have with the Blackman window or the window functions discussed in Section 3.9. The Chebyshev window function's stopband attenuation, in decibels, is equal to

$$\text{Atten}_{\text{Cheb}} = -20\gamma\,. \tag{5-19}$$

So, for example, if we needed our sidelobe levels to be no greater than –60 dB below the main lobe, we use Eq. (5-19) to establish a γ value of 3.0 and let the software generate the Chebyshev window coefficients.†

The same process applies to the Kaiser window, as shown in Figure 5-27. Commercial software packages allow us to specify β in Eq. (5-18) and provide us with the associated window coefficients. The curves in Figure 5-27(b), obtained for Kaiser window functions defined by 32 discrete samples, show that we can select the desired sidelobe levels and see what effect this has on the main lobe width.

Chebyshev or Kaiser, which is the best window to use? It depends on the application. Returning to Figure 5-25(b), notice that, unlike the constant sidelobe peak levels of the Chebyshev window, the Kaiser window's

† By the way, some digital signal processing software packages require that we specify $\text{Atten}_{\text{Cheb}}$ in decibels instead of γ. That way, we don't have to bother using Eq. (5-19) at all.

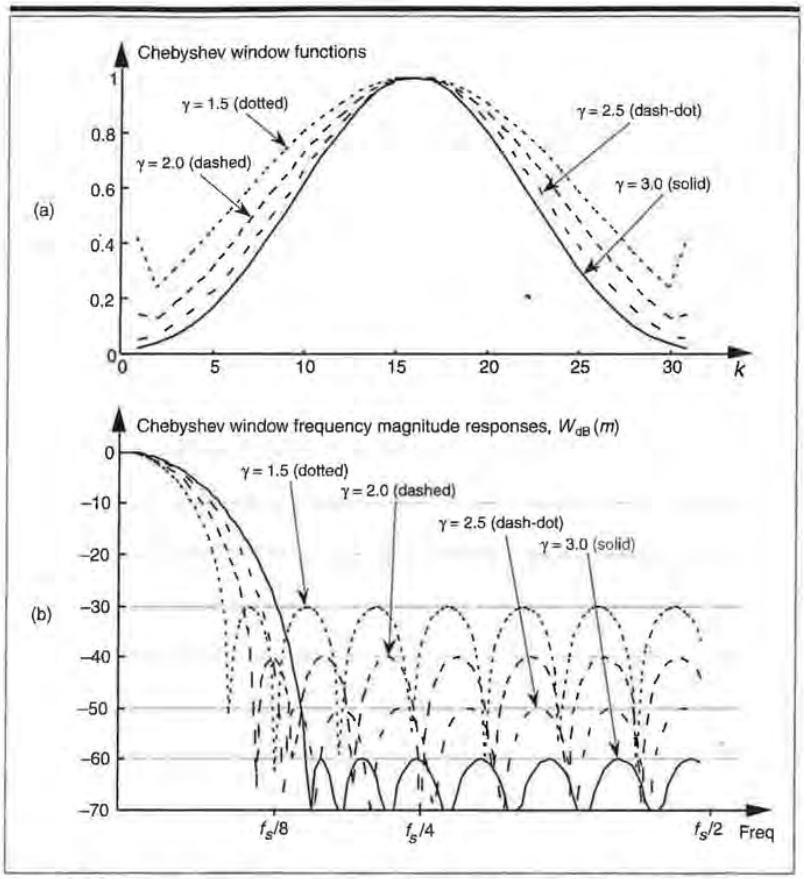


Figure 5-26 Chebyshev window functions for various γ values: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

sidelobes decrease with increased frequency. However, the Kaiser sidelobes are higher than the Chebyshev window's sidelobes near the main lobe. Our primary trade-off here is trying to reduce the sidelobe levels without broadening the main lobe too much. Digital filter designers typically experiment with various values of γ and β for the Chebyshev and Kaiser windows to get the optimum $W_{dB}(m)$ for a particular application. (For that matter, the Blackman window's very low sidelobe levels outweigh its wide main lobe in many applications.) Different window functions have their own individual advantages and disadvantages for FIR filter design. Regardless of the nonrectangular window function used, they

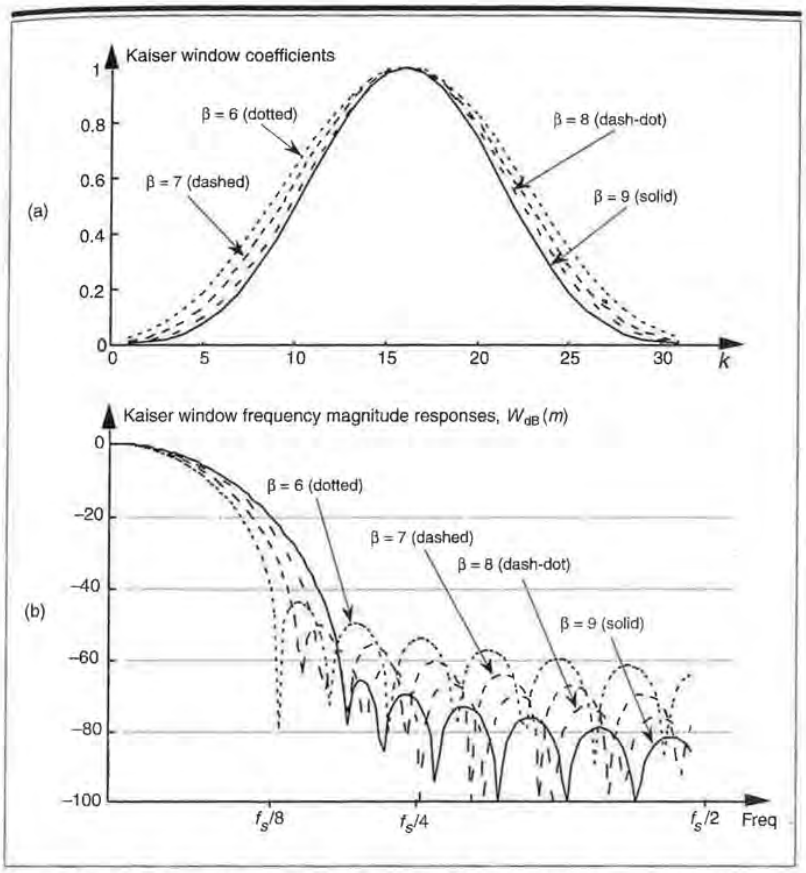


Figure 5-27 Kaiser window functions for various β values: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

always decrease the FIR filter passband ripple over that of the rectangular window. For the enthusiastic reader, a very thorough discussion of window functions can be found in reference [14].

5.4 Bandpass FIR Filter Design

The window method of low-pass FIR filter design can be used as the first step in designing a bandpass FIR filter. Let's say we want a 31-tap FIR filter with the frequency response shown in Figure 5-22(a), but instead of being centered about zero Hz, we want the filter's passband to be centered

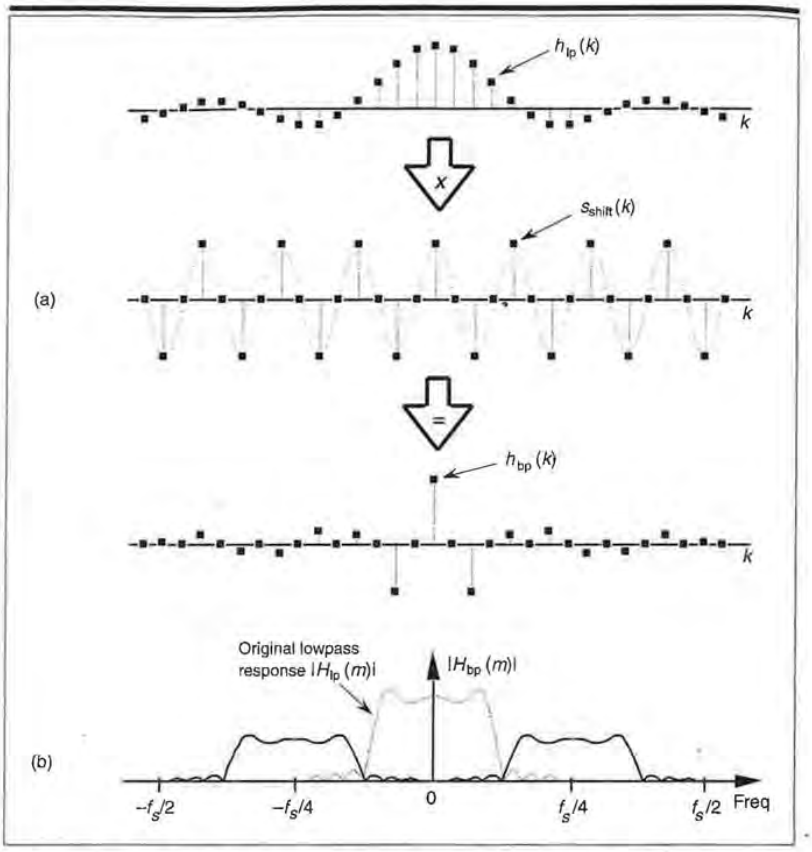


Figure 5-28 Bandpass filter with frequency response centered at $f_s/4$: (a) generating 31-tap filter coefficients $h_{bp}(k)$; (b) frequency magnitude response $|H_{bp}(m)|$.

about $f_s/4$ Hz. If we define a low-pass FIR filter's coefficients as $h_{lp}(k)$, our problem is to find the $h_{bp}(k)$ coefficients of a bandpass FIR filter. As shown in Figure 5-28, we can shift $H_{lp}(m)$'s frequency response by multiplying the filter's $h_{lp}(k)$ low-pass coefficients by a sinusoid of $f_s/4$ Hz. That sinusoid is represented by the $s_{shift}(k)$ sequence in Figure 5-28(a) whose values are a sinewave sampled at a rate of four samples per cycle. Our final 31-tap $h_{bp}(k)$ FIR bandpass filter coefficients are

$$h_{bp}(k) = h_{lp}(k) \cdot s_{shift}(k)\,, \tag{5-20}$$

whose frequency magnitude response $|H_{bp}(m)|$ is shown as the solid curves in Figure 5-28(b). The actual magnitude of $|H_{bp}(m)|$ is half that of the original $|H_{lp}(m)|$ because half the values in $h_{bp}(k)$ are zero when $s_{shift}(k)$ corresponds exactly to $f_s/4$. This effect has an important practical implication. It means that, when we design an N-tap bandpass FIR filter centered at a frequency of $f_s/4$ Hz, we only need to perform approximately $N/2$ multiplications for each filter output sample. (There's no reason to multiply an input sample value, $x(n-k)$, by zero before we sum all the products from Eq. (5-6) and Figure 5-13, right? We just don't bother to perform the unnecessary multiplications at all.) Of course, when the bandpass FIR filter's center frequency is other than $f_s/4$, we're forced to perform the full number of N multiplications for each FIR filter output sample.

Notice, here, that the $h_{lp}(k)$ low-pass coefficients in Figure 5-28(a) have not been multiplied by any window function. In practice, we'd use an $h_{lp}(k)$ that has been windowed prior to implementing Eq. (5-20) to reduce the passband ripple. If we wanted to center the bandpass filter's response at some frequency other than $f_s/4$, we merely need to modify $s_{shift}(k)$ to represent sampled values of a sinusoid whose frequency is equal to the desired bandpass center frequency. That new $s_{shift}(k)$ sequence would then be used in Eq. (5-20) to get the new $h_{bp}(k)$.

5.5 Highpass FIR Filter Design

Going one step further, we can use the bandpass FIR filter design technique to design a highpass FIR filter. To obtain the coefficients for a highpass filter, we need only modify the shifting sequence $s_{shift}(k)$ to make it represent a sampled sinusoid whose frequency is $f_s/2$. This process is shown in Figure 5-29. Our final 31-tap highpass FIR filter's $h_{hp}(k)$ coefficients are

$$\begin{aligned} h_{hp}(k) &= h_{lp}(k) \cdot s_{shift}(k) \\ &= h_{lp}(k) \cdot (1, -1, 1, -1, 1, -1, \text{etc.}) , \end{aligned} \qquad (5\text{-}21)$$

whose $|H_{hp}(m)|$ frequency response is the solid curve in Figure 5-29(b). Because $s_{shift}(k)$ in Figure 5-29(a) has alternating plus and minus ones, we can see that $h_{hp}(k)$ is merely $h_{lp}(k)$ with the sign changed for every other coefficient. Unlike $|H_{bp}(m)|$ in Figure 5-28(b), the $|H_{hp}(m)|$ response in Figure 5-29(b) has the same amplitude as the original $|H_{lp}(m)|$.

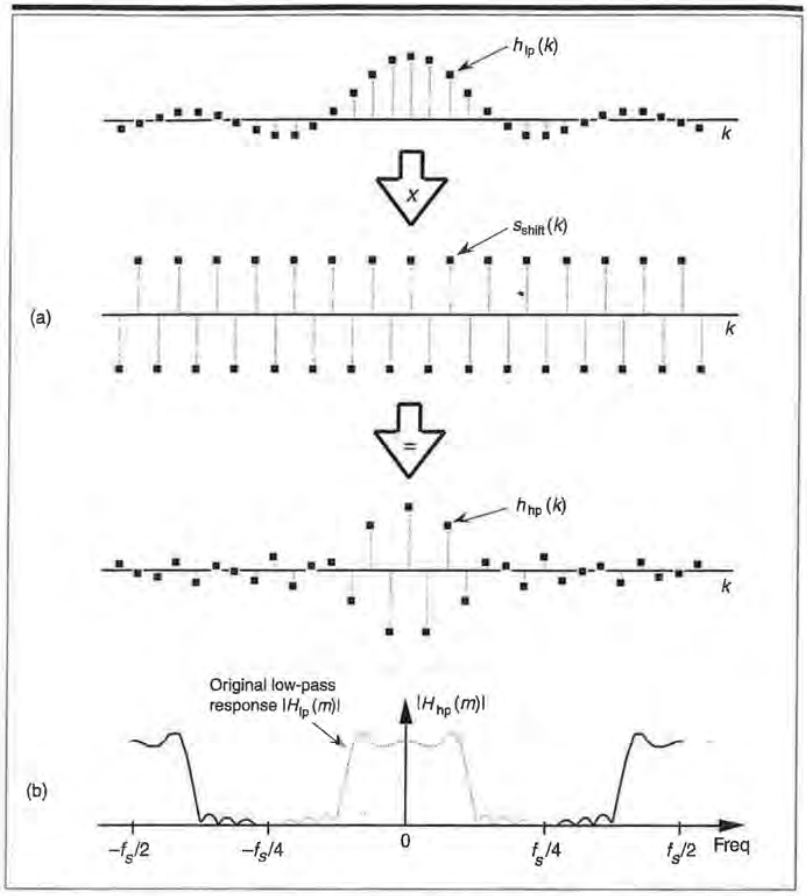


Figure 5-29 Highpass filter with frequency response centered at $f_s/2$: (a) generating 31-tap filter coefficients $h_{hp}(k)$; (b) frequency magnitude response $|H_{hp}(m)|$.

Again, notice that the $h_{lp}(k)$ low-pass coefficients in Figure 5-29(a) have not been modified by any window function. In practice, we'd use a windowed $h_{lp}(k)$ to reduce the passband ripple before implementing Eq. (5-21).

5.6 Remez Exchange FIR Filter Design Method

Let's introduce one last FIR filter design technique that has found wide acceptance in practice. The Remez Exchange FIR filter design method (also called the Parks-McClellan, or Optimal method) is a popular tech-

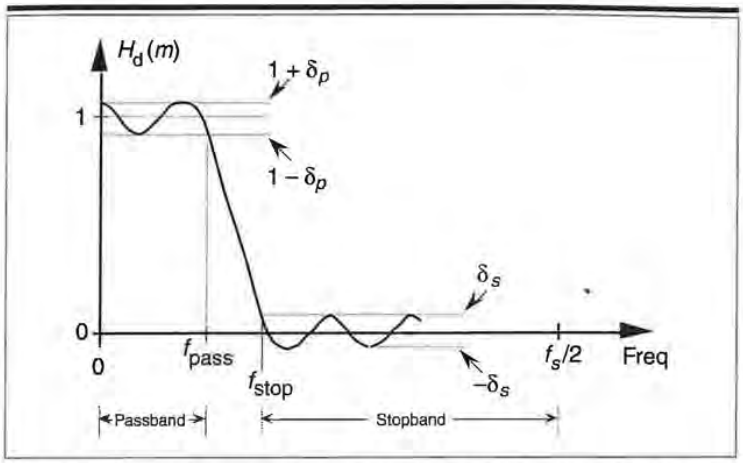


Figure 5-30 Desired frequency response definition of a low-pass FIR filter using the Remez Exchange design method.

nique used to design high-performance FIR filters.† To use this design method, we have to visualize a desired frequency response $H_d(m)$ like that shown in Figure 5-30.

We have to establish a desired passband cutoff frequency f_{pass} and the frequency where the attenuated stopband begins, f_{stop}. In addition, we must establish the variables δ_p and δ_s that define our desired passband and stopband ripple. Passband and stopband ripple, in decibels, are related to δ_p and δ_s by[15]

$$\text{Passband ripple} = 20 \cdot \log_{10}(1 + \delta_p)\,, \tag{5-22}$$

and

$$\text{Stopband ripple} = -20 \cdot \log_{10}(\delta_s)\,. \tag{5-22'}$$

(Some of the early journal papers describing the Remez design method used the equally valid expression $-20 \cdot \log_{10}(\delta_p)$ to define the passband ripple in decibels. However, Eq. (5-22) is the most common form used today.) Next, we apply these parameters to a computer software routine that generates the filter's N time-domain $h(k)$ coefficients where N is the minimum number of filter taps to achieve the desired filter response.

† *Remez* is pronounced re-'mā, like "away."

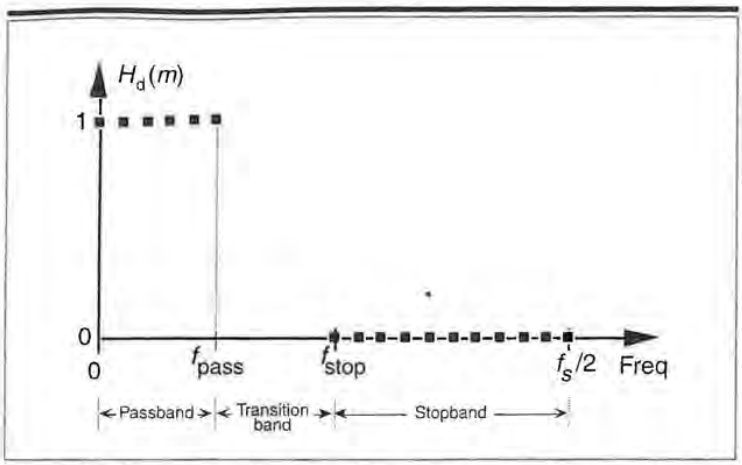


Figure 5-31 Alternate method for defining the desired frequency response of a low-pass FIR filter using the Remez Exchange technique.

On the other hand, some software Remez routines assume that we want δ_p and δ_s to be as small as possible and require us only to define the desired values of the $H_d(m)$ response as shown by the solid black dots in Figure 5-31. The software then adjusts the values of the undefined (shaded dots) values of $H_d(m)$ to minimize the error between our desired and actual frequency response while minimizing δ_p and δ_s. The filter designer has the option to define some of the $H_d(m)$ values in the transition band, and the software calculates the remaining undefined $H_d(m)$ transition band values. With this version of the Remez algorithm, the issue of most importance becomes how we define the transition region. We want to minimize its width while, at the same time, minimizing passband and stopband ripple. So exactly how we design an FIR filter using the Remez Exchange technique is specific to the available filter design software. Although the mathematics involved in the development of the Remez Exchange method is rather complicated, we don't have to worry about that here[16-20]. Just remember that the Remez Exchange design method gives us a Chebyshev-type filter whose actual frequency response is as close as possible to the desired $H_d(m)$ response for a given number of filter taps.

To illustrate the advantage of the Remez method, the solid curve in Figure 5-32 shows the frequency response of a 31-tap FIR designed using this technique. For comparison, Figure 5-32 also shows the frequency responses of two 31-tap FIR filters for the same passband width using the

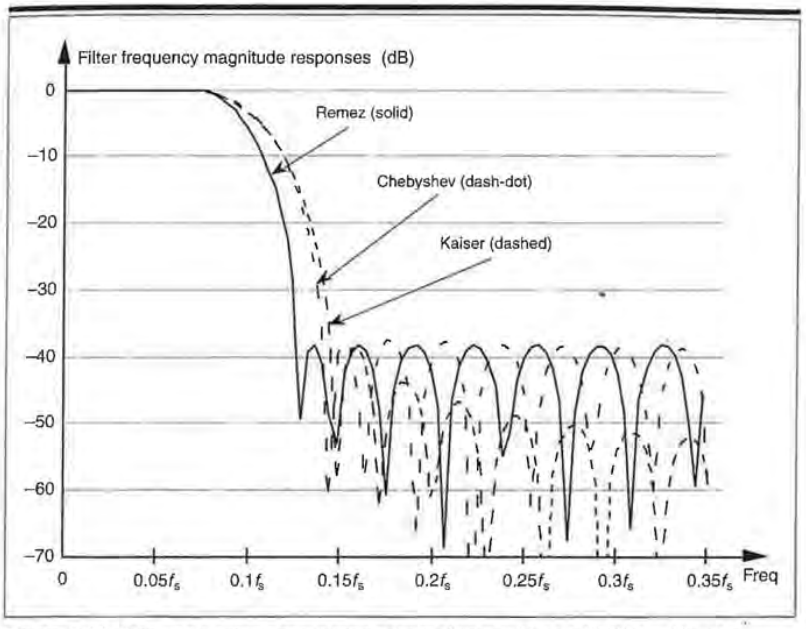


Figure 5-32 Frequency response comparison of three 31-tap FIR filters: Remez, Chebyshev windowed, and Kaiser windowed.

Chebyshev and Kaiser windowing techniques. Notice how the three filters have roughly the same stopband sidelobe levels, near the main lobe, but that the Remez filter has the more desirable (steeper) transition band roll-off.

5.7 Half-Band FIR Filters

There's a specialized FIR filter that's proved useful in decimation applications[21–25]. Called a *half-band FIR filter*, its frequency response is symmetrical about the $f_s/4$ point as shown in Figure 5-33(a). As such, the sum of f_{pass} and f_{stop} is $f_s/2$. This symmetry has the beautiful property that the time-domain FIR impulse response has every other filter coefficient being zero, except at the peak. This enables us to avoid approximately half the number of multiplications when implementing this kind of filter. By way of example, Figure 5-33(b) shows the coefficients for a 31-tap half-band filter where Δf was defined to be approximately $f_s/32$ using the Remez Exchange method. (To preserve further symmetry, the parameters δ_p and δ_s were specified to be equal to each other.)

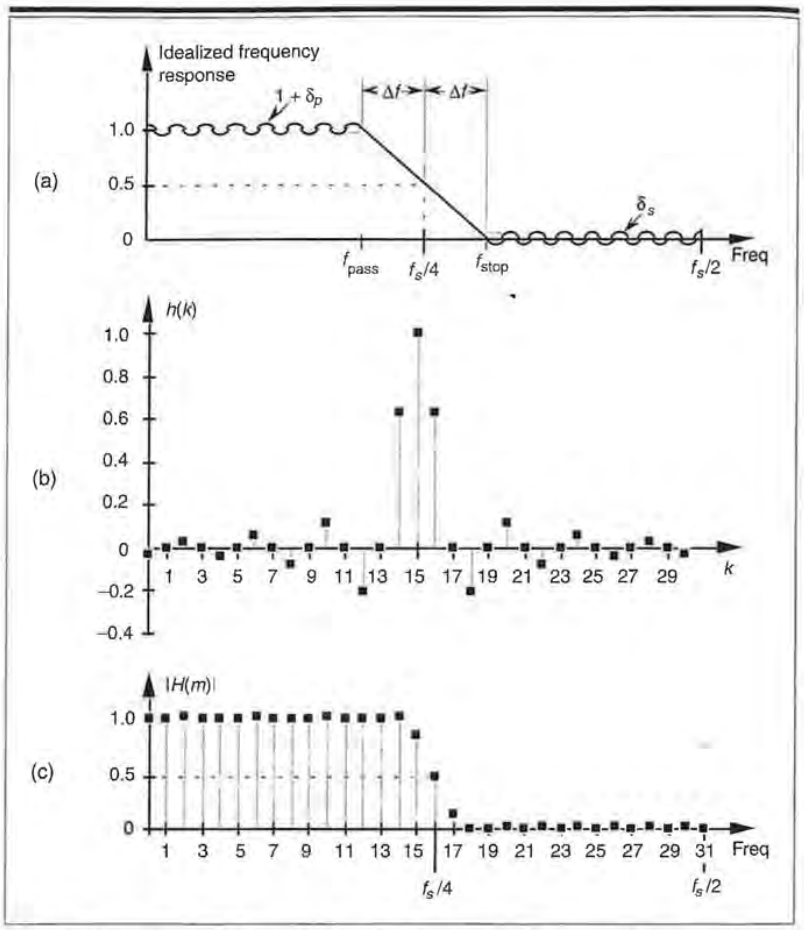


Figure 5-33 Half-Band FIR filter: (a) idealized continuous frequency response; (b) coefficients for a 31-tap half-band FIR filter; (c) discrete frequency magnitude response of a 31-tap half-band FIR filter.

Notice how the alternating $h(k)$ coefficients are zero, so we have to perform only 17 multiplications for each filter output sample instead of the expected 31 multiplications. For the general case of an S-tap half-band FIR filter, we'll need perform only $(S + 1)/2 + 1$ multiplications for each output sample.† Taking the DFT of those $h(k)$ coefficients gives us the half-band FIR frequency response shown in Figure 5-33(c).

† Section 10.8 shows a technique to further reduce the number of necessary multiplications for certain types of FIR filters including half-band filters.

5.8 Phase Response of FIR Filters

Although we illustrated a couple of output phase shift examples for our original averaging FIR filter in Figure 5-10, the subject of FIR phase response deserves additional attention. One of the dominant features of FIR filters is their linear phase response that we can demonstrate by way of example. Given the 25 $h(k)$ FIR filter coefficients in Figure 5-34(a), we can perform a DFT to determine the filter's $H(m)$ frequency response. The normalized real part, imaginary part, and magnitude of

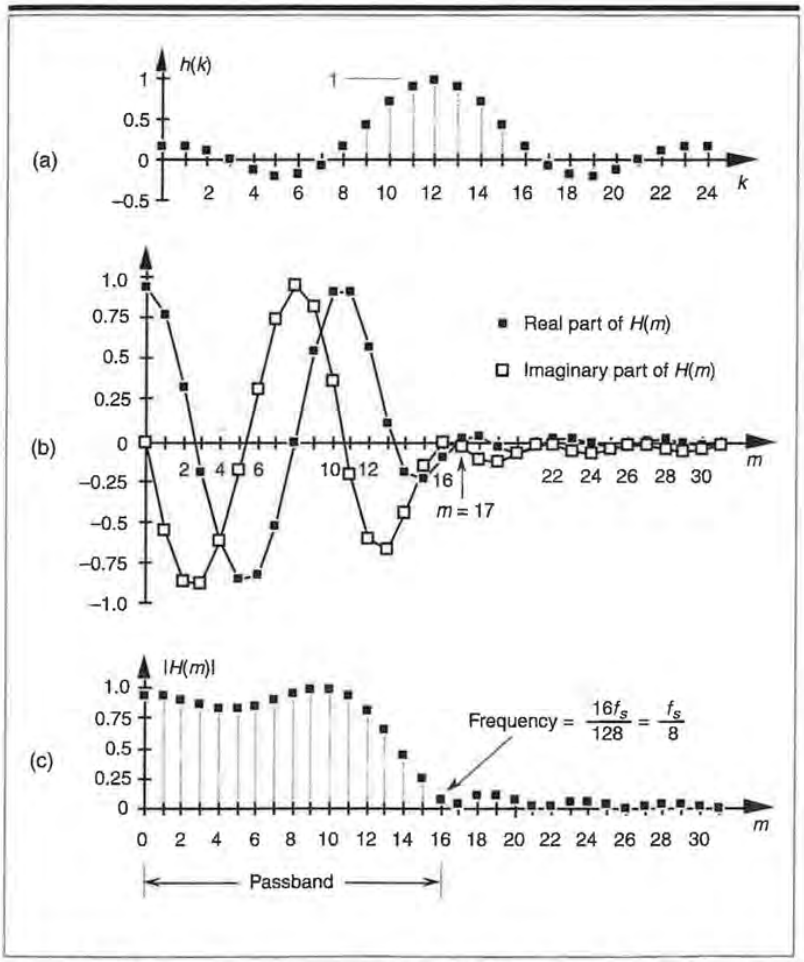


Figure 5-34 FIR filter frequency response $H(m)$: (a) $h(k)$ filter coefficients; (b) real and imaginary parts of $H(m)$; (c) magnitude of $H(m)$.

$H(m)$ are shown in Figures 5-34(b) and 5-34(c), respectively.† Being complex values, each $H(m)$ sample value can be described by its real and imaginary parts, or equivalently, by its magnitude $|H(m)|$ and its phase $H_\phi(m)$ shown in Figure 5-35(a).

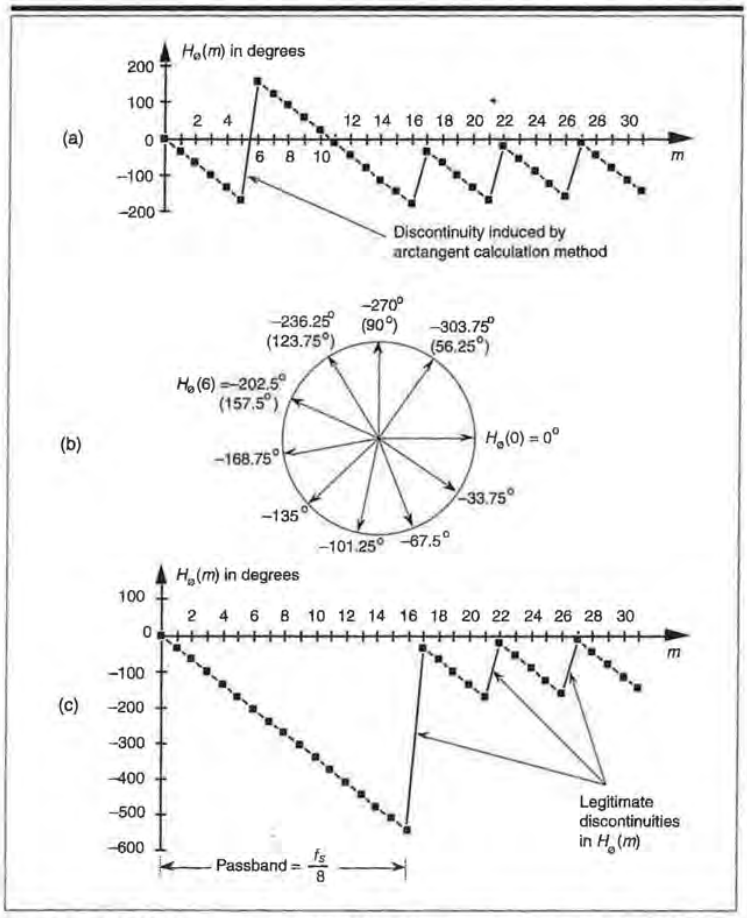


Figure 5-35 FIR filter phase response $H_\phi(m)$ in degrees: (a) calculated $H_\phi(m)$; (b) polar plot of $H_\phi(m)$'s first ten phase angles in degrees; (c) actual $H_\phi(m)$.

† Any DFT size greater than the $h(k)$ width of 25 is sufficient to obtain $H(m)$. The $h(k)$ sequence was padded with 103 zeros to take a 128-point DFT resulting in the $H(m)$ sample values in Figure 5-34.

The phase of a complex quantity is, of course, the arctangent of the imaginary part divided by the real part, or $\varnothing = \tan^{-1}(\text{imag}/\text{real})$. Thus the phase of $H_\varnothing(m)$ is determined from the samples in Figure 5-34(b).

The phase response in Figure 5-35(a) certainly looks linear over selected frequency ranges, but what do we make of those sudden jumps, or discontinuities, in this phase response? If we were to plot the angles of $H_\varnothing(m)$ starting with the $m = 0$ sample on a polar graph, using the nonzero real part of $H(0)$, and the zero-valued imaginary part of $H(0)$, we'd get the zero-angled $H_\varnothing(0)$ phasor shown on the right side of Figure 5-35(b). Continuing to use the real and imaginary parts of $H(m)$ to plot additional phase angles results in the phasors going clockwise around the circle in increments of –33.75°. It's at the $H_\varnothing(6)$ that we discover the cause of the first discontinuity in Figure 5-35(a). Taking the real and imaginary parts of $H(6)$, we'd plot our phasor oriented at an angle of –202.5°. But Figure 5-35(a) shows that $H_\varnothing(6)$ is equal to 157.5°. The problem lies in the software routine used to generate the arctangent values plotted in Figure 5-35(a). The software adds 360° to any negative angles in the range of $-180° > \varnothing \geq -360°$, i.e., angles in the upper half of the circle. This makes $\varnothing$ a positive angle in the range of $0° < \varnothing \leq 180°$ and that's what gets plotted. (This apparent discontinuity between $H_\varnothing(5)$ and $H_\varnothing(6)$ is called *phase wrapping*.) So the true $H_\varnothing(6)$ of –202.5° is converted to a +157.5° as shown in parentheses in Figure 5-35(b). If we continue our polar plot for additional $H_\varnothing(m)$ values, we'll see that their phase angles continue to decrease with an angle increment of –33.75°. If we compensate for the software's behavior and plot phase angles more negative than –180°, by *unwrapping* the phase, we get the true $H_\varnothing(m)$ shown in Figure 5-35(c).[†] Notice that $H_\varnothing(m)$ is, indeed, linear over the passband of $H(m)$. It's at $H_\varnothing(17)$ that our particular $H(m)$ experiences a polarity change of its real part while its imaginary part remains negative—this induces a true phase angle discontinuity that really is a constituent of $H(m)$ at $m = 17$. (Additional phase discontinuities occur each time the real part of $H(m)$ reverses polarity, as shown in Figure 5-35(c).) The reader may wonder why we care about the linear phase response of $H(m)$. The answer, an important one, requires us to introduce the notion of group delay.

Group delay is defined as the derivative of the phase with respect to frequency, or $G = \Delta\varnothing/\Delta f$. For FIR filters, then, group delay is the slope of the $H_\varnothing(m)$ response curve. When the group delay is constant, as it is over the

[†] When plotting filter phase responses, if we encounter a phase angle sample $\varnothing$ that looks like an unusual discontinuity, it's a good idea to add 360° to $\varnothing$ when $\varnothing$ is negative, or –360° when $\varnothing$ is positive, to see if that compensates for any software anomalies.

passband of all FIR filters having symmetrical coefficients, all frequency components of the filter input signal are delayed by an equal amount of time G before they reach the filter's output. This means that no phase distortion is induced in the filter's desired output signal, and this is crucial in communications signals. For amplitude modulation (AM) signals, constant group delay preserves the time waveform shape of the signal's modulation envelope. That's important because the modulation portion of an AM signal contains the signal's information. Conversely, a nonlinear phase will distort the audio of AM broadcast signals, blur the edges of television video images, blunt the sharp edges of received radar pulses, and increase data errors in digital communication signals. (Group delay is sometimes called *envelope delay* because group delay was originally the subject of analysis due to its affect on the envelope, or modulation signal, of amplitude modulation AM systems.) Of course we're not really concerned with the group delay outside the passband because signal energy outside the passband is what we're trying to eliminate through filtering.

Over the passband frequency range for an S-tap FIR digital filter, group delay has been shown to be given by

$$G_{odd} = \frac{(S-1)t_s}{2}, \tag{5-23}$$

when S is odd, and

$$G_{even} = \frac{S \cdot t_s}{2}, \tag{5-23'}$$

when S is even, where t_s is the sample period $(1/f_s)$.†

Although we used a 128-point DFT to obtain the frequency responses in Figures 5-34 and 5-35, we could just as well have used N = 32-point or N = 64-point DFTs. These smaller DFTs give us the phase response curves shown in Figure 5-36(a) and 5-36(b). Notice how different the phase response curves are when N = 32 in Figure 5-36(a) compared to when N = 128 in Figure 5-36(c). The phase angle resolution is much finer in Figure 5-36(c). The passband phase angle resolution, or increment $\Delta\phi$, is given by

$$\Delta\phi = \frac{-G \cdot 360°}{N}, \tag{5-24}$$

† As derived in Section 3.4 of reference [15], and Section 5.7.3 of reference [19].

where N is the number of points in the DFT. So, for our S = 25-tap filter in Figure 5-34(a), G = 12, and $\Delta\phi$ is equal to $-12 \cdot 360°/32 = -135°$ in Figure 5-36(a), and $\Delta\phi$ is -33.75° in Figure 5-36(c). If we look carefully at the sample values in Figure 5-36(a), we'll see that they're all included within the samples in Figures 5-36(b) and 5-36(c).

Let's conclude this FIR phase discussion by reiterating the meaning of phase response. The phase, or phase delay, at the output of an FIR filter is

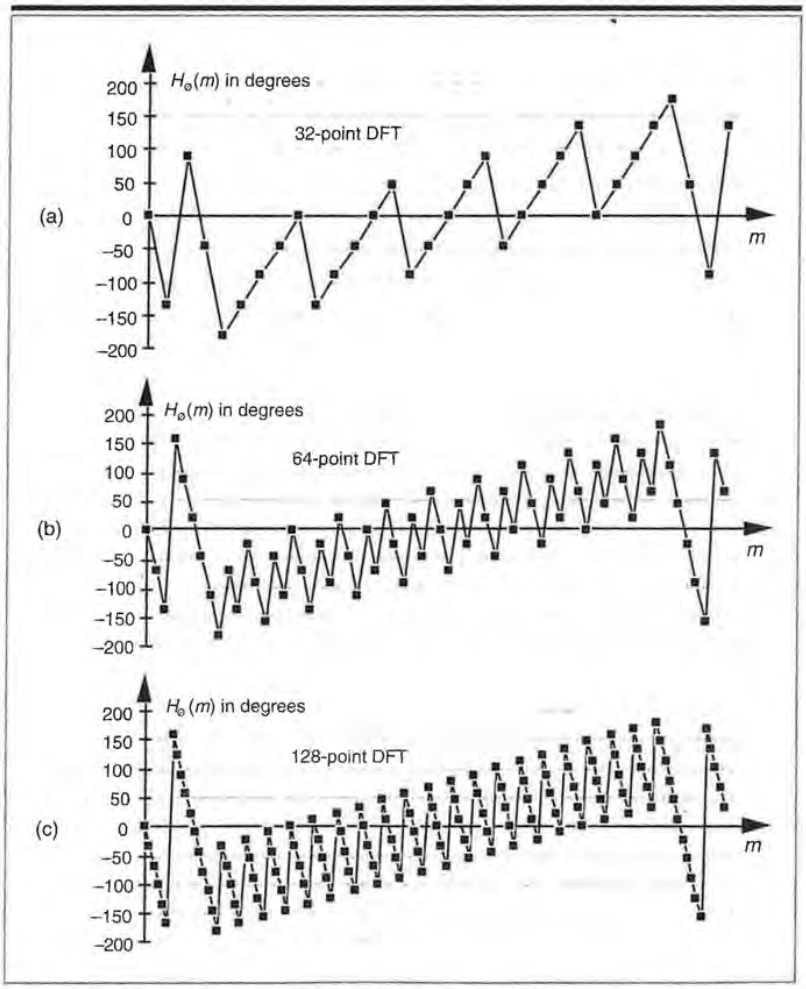


Figure 5-36 FIR filter phase response $H_\phi(m)$ in degrees: (a) calculated using a 32-point DFT; (b) using a 64-point DFT; (c) using a 128-point DFT.

Table 5-2 Values Used in Eq. (5-25) for the Frequency $f_s/32$

DFT size, N	*Index m*	$H_\phi(mf_s/N)$
32	1	-135°
64	2	-135°
128	4	-135°

the phase of the first output sample relative to the phase of the filter's first input sample. Over the passband, that phase shift, of course, is a linear function of frequency. This will be true only as long as the filter has symmetrical coefficients. Figure 5-10 is a good illustration of an FIR filter's output phase delay.

For FIR filters, the output phase shift measured in degrees, for the passband frequency $f = mf_s/N$, is expressed as

$$\text{phase delay} = H_\phi(mf_s/N) = m \cdot \Delta\phi = \frac{-m \cdot G \cdot 360^\circ}{N}. \qquad (5\text{-}25)$$

We can illustrate Eq. (5-25) and show the relationship between the phase responses in Figure 5-36, by considering the phase delay associated with the frequency of $f_s/32$ in Table 5-2. The subject of group delay is described further in Appendix F where an example of envelope delay distortion, due to a filter's nonlinear phase, is illustrated.

5.9 A Generic Description of Discrete Convolution

Although convolution was originally an analysis tool used to prove continuous signal processing theorems, we now know that convolution affects every aspect of digital signal processing. Convolution influences our results whenever we analyze or filter any finite set of data samples from a linear time-invariant system. Convolution not only constrains DFTs to be just approximations of the continuous Fourier transform; it is the reason that discrete spectra are periodic in the frequency domain. It's interesting to note that, although we use the process of convolution to implement FIR digital filters, convolution effects induce frequency response ripple preventing us from ever building a perfect digital filter. Its influence is so pervasive that, to repeal the law of convolution, quoting a

$$Y_j = \sum_{k=0}^{N-1} P_k \cdot Q_{j-k}, \text{ or}$$

$$\begin{bmatrix} Y \\ Y \\ Y \\ \cdot \\ \cdot \\ \cdot \\ Y_{N-1} \end{bmatrix} = \begin{bmatrix} Q_0 & Q_{N-1} & Q_{N-2} & \cdots & Q_1 \\ Q_1 & Q_0 & Q_{N-1} & \cdots & Q_2 \\ Q_2 & Q_1 & Q_0 & \cdots & Q_3 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ Q_{N-1} & Q_{N-2} & Q_{N-3} & \cdots & Q_0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ \cdot \\ \cdot \\ \cdot \\ P_{N-1} \end{bmatrix}$$

Theorem: if

$$P_j \leftarrow \text{DFT} \rightarrow A_n ,$$

$$Q_j \leftarrow \text{DFT} \rightarrow B_n ,$$

$$Y_j \leftarrow \text{DFT} \rightarrow C_n , \text{ and}$$

then,

$$C_n = N \cdot A_n \cdot B_n$$

Figure 5-37 One very efficient, but perplexing, way of defining convolution.

phrase from Dr. Who, would "unravel the entire causal nexus" of digital signal processing.

Convolution has always been a somewhat difficult concept for the beginner to grasp. That's not too surprising for several reasons. Convolution's effect on discrete signal processing is not intuitively obvious for those without experience working with discrete signals, and the mathematics of convolution does seem a little puzzling at first. Moreover, in their sometimes justified haste, many authors present the convolution equation and abruptly start using it as an analysis tool without explaining its origin and meaning. For example, this author once encountered what was called a *tutorial* article on the FFT in a professional journal that proceeded to define *convolution* merely by presenting something like that shown in Figure 5-37 with no further explanation!

Unfortunately, few beginners can gain an understanding of the convolution process from Figure 5-37 alone. Here, we avoid this dilemma by defining the process of convolution and gently proceed through a couple of simple convolution examples. We conclude this chapter with a discussion of the powerful convolution theorem and show why it's so useful as a qualitative tool in discrete system analysis.

5.9.1 Discrete Convolution in the Time-Domain

Discrete convolution is a process, whose input is two sequences, that provides a single output sequence. Convolution inputs can be two time-domain sequences giving a time-domain output, or two frequency-domain input sequences providing a frequency-domain result. (Although the two input sequences must both be in the same domain for the process of convolution to have any practical meaning, their sequence lengths need not be the same.) Let's say we have two input sequences $h(k)$ of length P and $x(k)$ of length Q in the time domain. The output sequence $y(n)$ of the convolution of the two inputs is defined mathematically as

$$y(n) = \sum_{k=0}^{P+Q-2} h(k)x(n-k). \tag{5-26}$$

Let's examine Eq. (5-26) by way of example using the $h(k)$ and $x(k)$ sequences shown in Figure 5-38. In this example, we can write the terms for each $y(n)$ in Eq. (5-26) as

$$\begin{aligned}
y(0) &= h(0)x(0-0) + h(1)x(0-1) + h(2)x(0-2) + h(3)x(0-3) + h(4)x(0-4) + h(5)x(0-5),\\
y(1) &= h(0)x(1-0) + h(1)x(1-1) + h(2)x(1-2) + h(3)x(1-3) + h(4)x(1-4) + h(5)x(1-5),\\
y(2) &= h(0)x(2-0) + h(1)x(2-1) + h(2)x(2-2) + h(3)x(2-3) + h(4)x(2-4) + h(5)x(2-5),\\
y(3) &= h(0)x(3-0) + h(1)x(3-1) + h(2)x(3-2) + h(3)x(3-3) + h(4)x(3-4) + h(5)x(3-5),\\
y(4) &= h(0)x(4-0) + h(1)x(4-1) + h(2)x(4-2) + h(3)x(4-3) + h(4)x(4-4) + h(5)x(4-5),
\end{aligned}$$

and

$$y(5) = h(0)x(5-0) + h(1)x(5-1) + h(2)x(5-2) + h(3)x(5-3) + h(4)x(5-4) + h(5)x(5-5). \tag{5-27}$$

With $P = 4$ and $Q = 3$, we need evaluate only $4 + 3 - 1 = 6$ individual $y(n)$ terms. Because $h(4)$ and $h(5)$ are zero, we can eliminate some of the terms in Eq. (5-27) and evaluate the remaining $x(n-k)$ indices giving the following expressions for $y(n)$ as

$$\begin{aligned}
y(0) &= h(0)x(0) + h(1)x(-1) + h(2)x(-2) + h(3)x(-3),\\
y(1) &= h(0)x(1) + h(1)x(0) + h(2)x(-1) + h(3)x(-2),\\
y(2) &= h(0)x(2) + h(1)x(1) + h(2)x(0) + h(3)x(-1),\\
y(3) &= h(0)x(3) + h(1)x(2) + h(2)x(1) + h(3)x(0),\\
y(4) &= h(0)x(4) + h(1)x(3) + h(2)x(2) + h(3)x(1),
\end{aligned}$$

and

$$y(5) = h(0)x(5) + h(1)x(4) + h(2)x(3) + h(3)x(2). \tag{5-28}$$

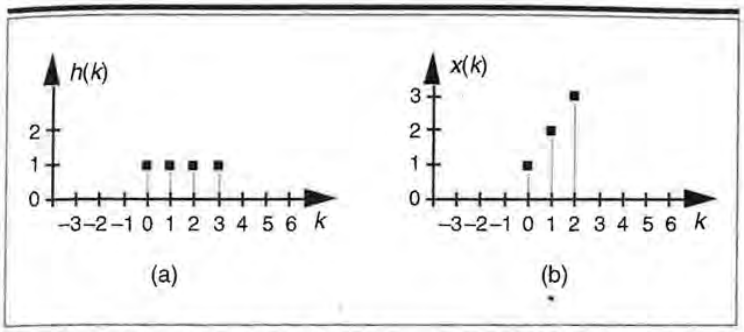


Figure 5-38 Convolution example input sequences: (a) first sequence $h(k)$ of length $P = 4$; (b) second sequence $x(k)$ of length $Q = 3$.

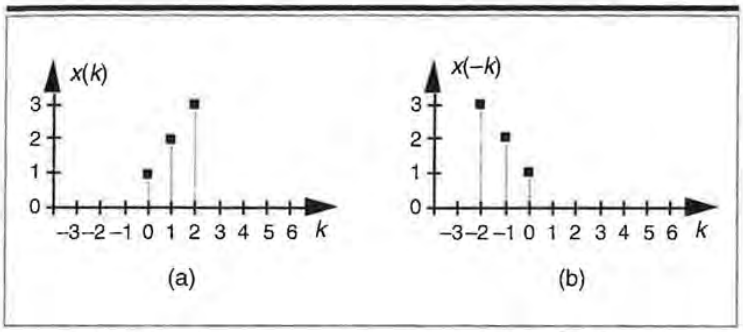


Figure 5-39 Convolution example input sequence: (a) second sequence $x(k)$ of length 3; (b) reflection of the second sequence about the $k = 0$ index.

Looking at the indices of the $h(k)$ and $x(k)$ terms in Eq. (5-28), we see two very important things occurring. First, convolution is merely the summation of a series of products—so the process itself is not very complicated. Second, notice that, for a given $y(n)$, $h(k)$'s index is increasing as $x(k)$'s index is decreasing. This fact has led many authors to introduce a new sequence $x(-k)$, and use that new sequence to graphically illustrate the convolution process. The $x(-k)$ sequence is simply our original $x(k)$ reflected about the 0 index of the k axis as shown in Figure 5-39. Defining $x(-k)$ as such enables us to depict the products and summations of Eq. (5-28)'s convolution as in Figure 5-40; that is, we can now align the $x(-k)$ samples with the samples of $h(k)$ for a given n index to calculate $y(n)$. As shown in Figure 5-40(a), the alignment of $h(k)$ and $x(n-k)$, for $n = 0$, yields $y(0) = 1$. This is the result of the first line in Eq. (5-28) repeated on the right side of Figure 5-40(a). The calculation of $y(1)$, for $n = 1$, is depicted

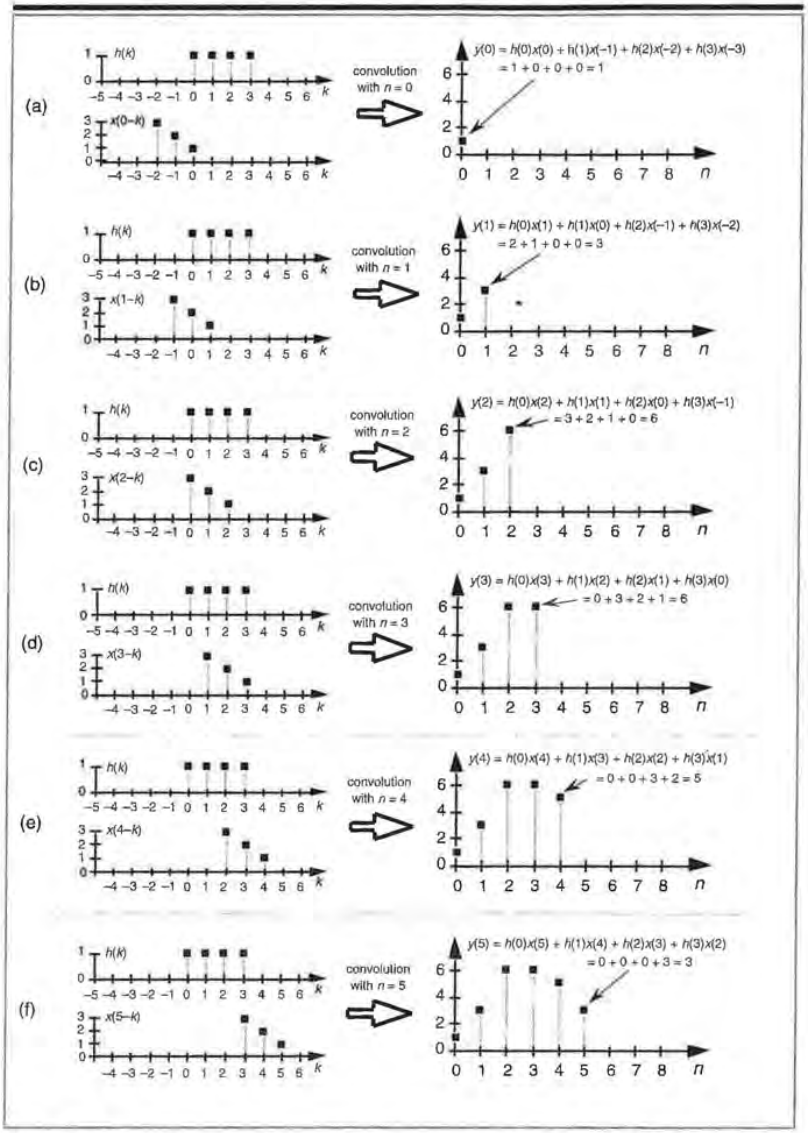


Figure 5-40 Graphical depiction of the convolution of $h(k)$ and $x(k)$ in Figure 5-38.

in Figure-34(b) where $x(n–k)$ is shifted one element to the right, resulting in $y(1) = 3$. We continue this $x(n–k)$ shifting and incrementing n until we arrive at the last nonzero convolution result of $y(5)$ shown in Figure 5-40(f). So, performing the convolution of $h(k)$ and $x(k)$ comprises

Step 1: plotting both $h(k)$ and $x(k)$,

Step 2: flipping the $x(k)$ sequence about the $k = 0$ value to get $x(-k)$,

Step 3: summing the products of $h(k)$ and $x(0-k)$ for all k to get $y(0)$,

Step 4: shifting $x(-k)$ one sample to the right,

Step 5: summing the products of $h(k)$ and $x(1-k)$ for all k to get $y(1)$, and

Step 6: shifting and summing products until there's no overlap of $h(k)$ and the shifted $x(n-k)$, in which case all further $y(n)$s are zero and we're done.

The full convolution of our $h(k)$ and $x(k)$ is the $y(n)$ sequence on the right side of Figure 5-40(f). We've scanned the $x(-k)$ sequence across the $h(k)$ sequence and summed the products where the sequences overlap. By the way, notice that the $y(n)$ sequence in Figure 5-40(f) has six elements where $h(k)$ had a length of four and $x(k)$ was of length three. In the general case, if $h(k)$ is of length P and $x(k)$ is of length Q, the length of $y(n)$ will have a sequence length of L, where

$$L = (P + Q - 1) . \tag{5-29}$$

At this point, it's fair for the beginner to ask, "OK, so what? What does this *strange* convolution process have to do with digital signal processing?" The answer to that question lies in understanding the effects of the convolution theorem.

5.9.2 The Convolution Theorem

The convolution theorem is a fundamental constituent of digital signal processing. It impacts our results anytime we filter or Fourier transform discrete data. To see why this is true, let's simplify the notation of Eq. (5-26) and use the abbreviated form

$$y(n) = h(k) * x(k) , \tag{5-30}$$

where, again, the "*" symbol means convolution. The convolution theorem may be stated as follows: If two time-domain sequences $h(k)$ and $x(k)$ have DFTs of $H(m)$ and $X(m)$, respectively, then the DFT of $h(k) * x(k)$ is the product $H(m) \cdot X(m)$. Likewise, the inverse DFT of $H(m) \cdot X(m)$ is $h(k) * x(k)$. We can represent this relationship with the expression

$$h(k) * x(k) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) \cdot X(m). \tag{5-31}$$

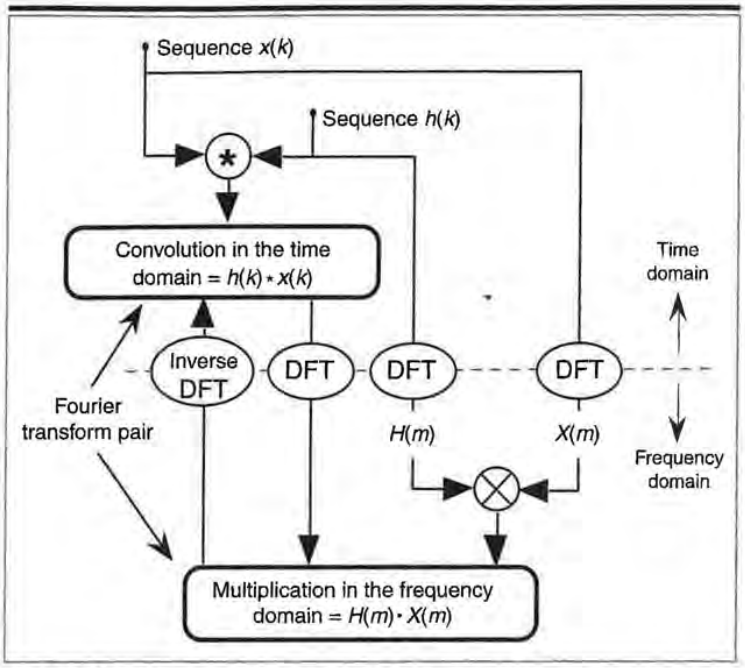


Figure 5-41 Relationships of the convolution theorem.

Equation (5-31) tells us that two sequences resulting from $h(k) * x(k)$ and $H(m) \cdot X(m)$ are Fourier transform pairs. So, taking the DFT of $h(k) * x(k)$ always gives us $H(m) \cdot X(m)$. Likewise, we can determine $h(k) * x(k)$ by taking the inverse DFT of $H(m) \cdot X(m)$. The important point to learn from Eq. (5-31) is that convolution in the time domain is equivalent to multiplication in the frequency domain. (We won't derive the convolution theorem here because it's derivation is readily available to the interested reader[26–29].) To help us appreciate this principle, Figure 5-41 sketches the relationship between convolution in the time domain and multiplication in the frequency domain.

We can easily illustrate the convolution theorem by taking 8-point DFTs of $h(k)$ and $x(k)$ to get $H(m)$ and $X(m)$, respectively, and listing these values as in Table 5-3. (Of course, we have to pad $h(k)$ and $x(k)$ with zeros, so they both have lengths of 8 to take 8-point DFTs.) Tabulating the inverse DFT of the product $H(m) \cdot X(m)$ allows us to verify Eq. (5-31), as listed in the last two columns of Table 5-3, where the acronym IDFT again means inverse DFT. The values from Table 5-3 are shown in Figure 5-42. (For simplicity, only the magnitudes of $H(m)$, $X(m)$, and $H(m) \cdot X(m)$ are shown in the figure.) We need to become comfortable with convolution in the time domain because,

Table 5-3 Convolution Values of $h(k)$ and $x(k)$ from Figure 5-38.

Index k *or* m	h(k)	x(k)	*DFT of* h(k) = H(m)	*DFT of* x(k) = X(m)	H(m) · X(m)	*IDFT of* H(m) · X(m)	h(k)*x(k)
0	1	1	4.0 + *j*0.0	6.0 + *j*0.0	24.0 + *j*0.0	1.0 + *j*0.0	1
1	1	2	1.00 – *j*2.41	2.41 – *j*4.41	–8.24 – *j*10.24	3.0 + *j*0.0	3
2	1	3	0	–2.0 – *j*2.0	0	6.0 + *j*0.0	6
3	1	0	1.00 – *j*0.41	–0.41 + *j*1.58	0.24 + *j*1.75	6.0 + *j*0.0	6
4	0	0	0	2.0 + *j*0.0	0	5.0 + *j*0.0	5
5	0	0	1.00 + *j*0.41	–0.41 – *j*1.58	0.24 – *j*1.75	3.0 + *j*0.0	3
6	0	0	0	–2.00 + *j*2.00	0	0.0 + *j*0.0	0
7	0	0	1.00 + *j*2.41	2.41 + *j*4.41	–8.24 + *j*10.24	0.0 + *j*0.0	0

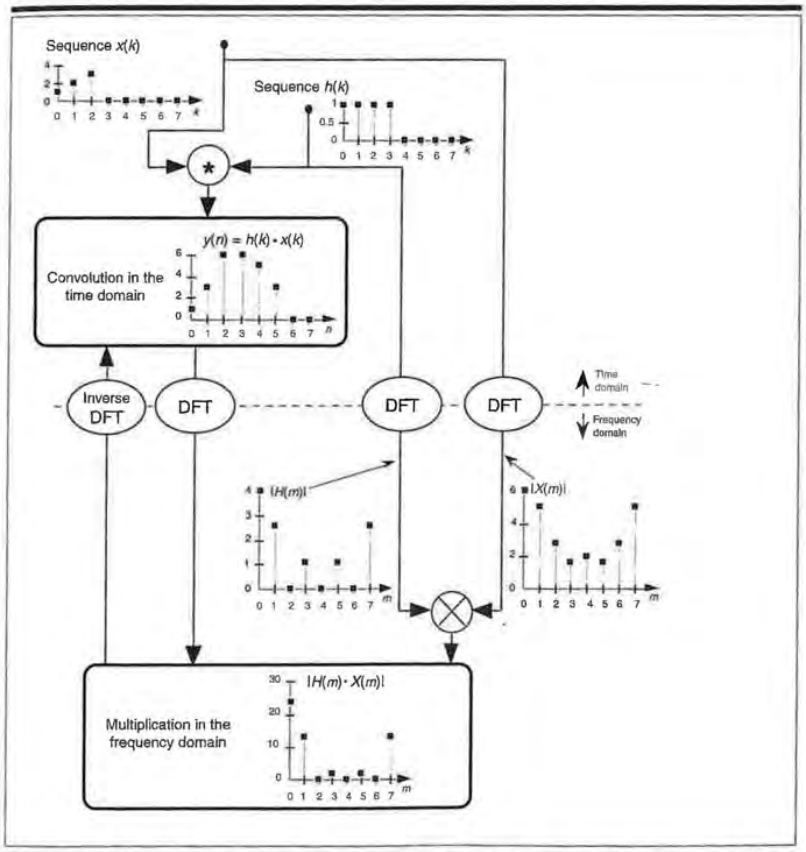


Figure 5-42 Convolution relationships of $h(k)$, $x(k)$, $H(m)$, and $X(m)$ from Figure 5-38.

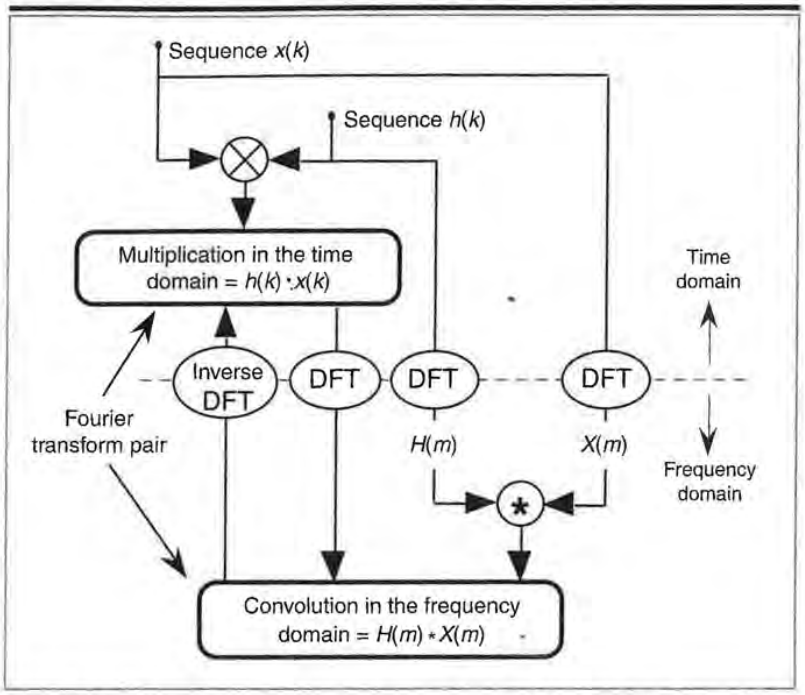


Figure 5-43 Relationships of the convolution theorem related to multiplication in the time domain.

as we've learned, it's the process used in FIR filters. As detailed in Section 5.2, we perform discrete time-domain FIR filtering by convolving an input sequence, $x(n)$ say, with the impulse response $x(k)$ of a filter, and, for FIR filters that impulse response happens to also be the filter's coefficients.[†] The result of that convolution is a filtered time-domain sequence whose spectrum is modified (multiplied) by the filter's frequency response $X(m)$. Section 10.10 describes a clever scheme to perform FIR filtering efficiently using the FFT algorithm to implement convolution.

Because of the duality of the convolution theorem, we could have swapped the time and frequency domains in our discussion of convolution and multiplication being a Fourier transform pair. This means that, similar to Eq. (5-31), we can also write

[†] As we'll see in Chapter 6, the coefficients used for an infinite impulse response (IIR) filter are not equal to that filter's impulse response.

$$h(k) \cdot x(k) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) * X(m) \ . \tag{5-32}$$

So the convolution theorem can be stated more generally as *Convolution in one domain is equivalent to multiplication in the other domain.* Figure 5-43 shows the relationship between multiplication in the time domain and convolution in the frequency domain. Equation (5-32) is the fundamental relationship used in the process of windowing time-domain data to reduce DFT leakage, as discussed in Section 3.9.

5.9.3 Applying the Convolution Theorem

The convolution theorem is useful as a qualitative tool in predicting the affects of different operations in discrete linear time-invariant systems. For example, many authors use the convolution theorem to show why periodic sampling of continuous signals results in discrete samples whose spectra are periodic in the frequency domain. Consider the real continuous time-domain waveform in Figure 5-44(a), with the one-sided spectrum of bandwidth B. Being a real signal, of course, its spectrum is symmetrical about 0 Hz. (In Figure 5-44, the large right-pointing shaded arrows represent Fourier transform operations.) Sampling this waveform is equivalent to multiplying it by a sequence of periodically spaced impulses, Figure 5-44(b), whose values are unity. If we say that the sampling rate is f_s samples/second, then the sample period $t_s = 1/f_s$ seconds. The result of this multiplication is the sequence of discrete time-domain impulses shown in Figure 5-44(c). We can use the convolution theorem to help us predict what the frequency-domain effect is of this multiplication in the time domain. From our theorem, we now realize that the spectrum of the time-domain product must be the convolution of the original spectra. Well, we know what the spectrum of the original continuous waveform is. What about the spectrum of the time-domain impulses? It has been shown that the spectrum of periodic impulses, whose period is t_s seconds, is also periodic impulses in the frequency domain with a spacing of f_s Hz as shown in Figure 5-44(b) [30].

Now, all we have to do is convolve the two spectra. In this case, convolution is straightforward because both of the frequency-domain functions are symmetrical about the zero-Hz point, and flipping one of them about zero Hz is superfluous. So we merely slide one of the functions across the other and plot the product of the two. The convolution of the original waveform spectrum and the spectral impulses results in replications of the waveform spectrum every f_s Hz, as shown in Figure 5-44(c).

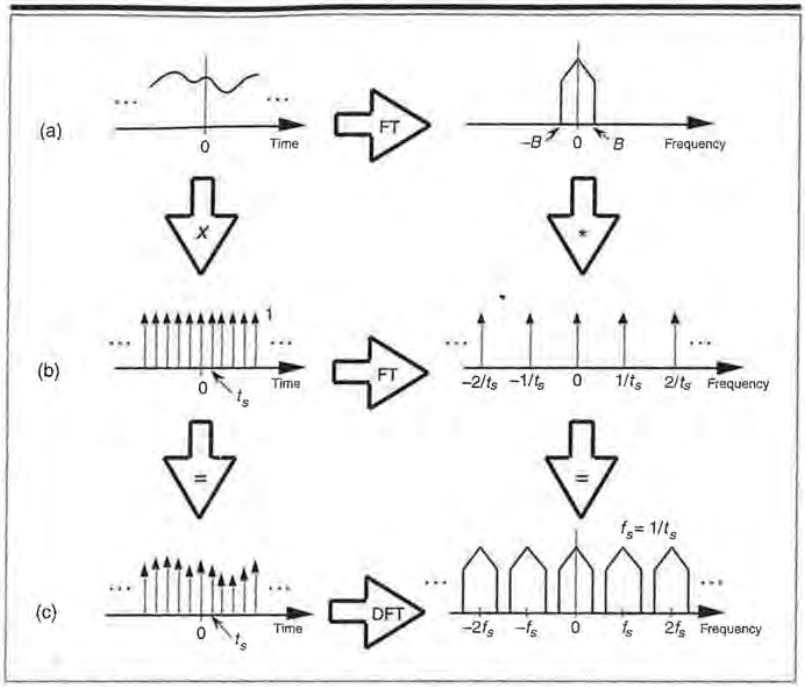


Figure 5-44 Using convolution to predict the spectral replication effects of periodic sampling.

This discussion reiterates the fact that the DFT is always periodic with a period of f_s Hz.

Here's another example of how the convolution theorem can come in handy when we try to understand digital signal processing operations. This author once used the theorem to resolve the puzzling result, at the time, of a triangular window function having its first frequency response null at twice the frequency of the first null of a rectangular window function. The question was "If a rectangular time-domain function of width T has its first spectral null at $1/T$ Hz, why does a triangular time-domain function of width T have its first spectral null at $2/T$ Hz?" We can answer this question by considering convolution in the time domain.

Look at the two rectangular time-domain functions shown in Figures 5-45(a) and 5-45(b). If their widths are each T seconds, their spectra are shown to have nulls at $1/T$ Hz as depicted in the frequency-domain functions in Figures 5-45(a) and 5-45(b). We know that the frequency magnitude

responses will be the absolute value of the classic $\sin(x)/x$ function.† If we convolve those two rectangular time-domain functions of width T, we'll get the triangular function shown in Figure 5-45(c). Again, in this case, flipping one rectangular function about the zero time axis is unnecessary. To convolve them, we need only scan one function across the other and determine

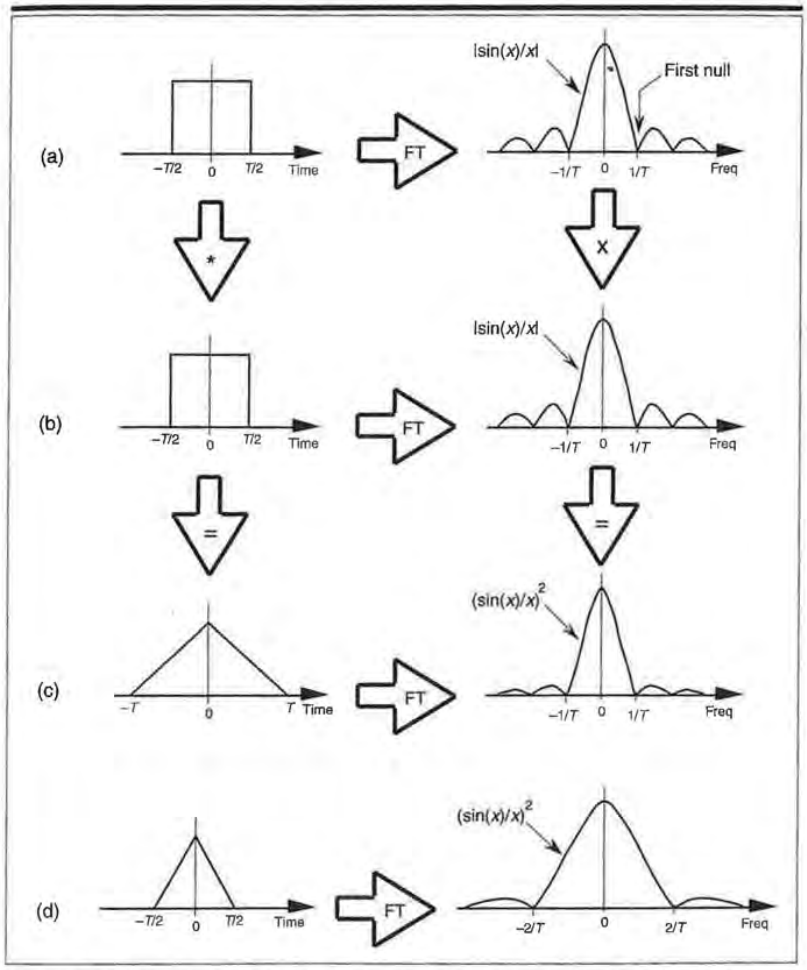


Figure 5-45 Using convolution to show that the Fourier transform of a triangular function has its first null at twice the frequency of the Fourier transform of a rectangular function.

† The $\sin(x)/x$ function was introduced in our discussion of window functions in Section 3.9 and is covered in greater detail in Section 3.13.

the area of their overlap. The time shift where they overlap the most happens to be a zero time shift. Thus, our resultant convolution has a peak at a time shift of zero seconds because there's 100 percent overlap. If we slide one of the rectangular functions in either direction, the convolution decreases linearly toward zero. When the time shift is $T/2$ seconds, the rectangular functions have a 50 percent overlap. The convolution is zero when the time shift is T seconds—that's when the two rectangular functions cease to overlap.

Notice that the triangular convolution result has a width of $2T$, and that's really the key to answering our question. Because convolution in the time domain is equivalent to multiplication in the frequency domain, the Fourier transform magnitude of our $2T$ width triangular function is the $|\sin(x)/x|$ in Figure 5-45(a) times the $|\sin(x)/x|$ in Figure 5-45(b), or the $(\sin(x)/x)^2$ function in Figure 5-45(c). If a triangular function of width 2T has its first frequency-domain null at $1/T$ Hz, then the same function of width T must have its first frequency null at $2/T$ Hz as shown in Figure 5-45(d), and that's what we set out to show. Comparison of Figure 5-45(c) and Figure 5-45(d) illustrates a fundamental Fourier transform property that compressing a function in the time domain results in an expansion of its corresponding frequency-domain representation.

References

[1] Shynk, J. J. "Adaptive IIR Filtering," *IEEE ASSP Magazine*, April 1989.

[2] Laundrie, A. "Adaptive Filters Enable Systems to Track Variations," *Microwaves & RF*, September 1989.

[3] Bullock, S. R. "High Frequency Adaptive Filter," *Microwave Journal*, September 1990.

[4] Haykin, S. S. *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[5] Goodwin, G. C., and Sin, K. S. *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[6] Gibbs, J. W. *Nature*, Vol. 59, 1899, pp. 606.

[7] Stockham, T. G. "High-Speed Convolution and Correlation with Applications to Digital Filtering," Chapter 7 in *Digital Processing of Signals*, Ed. by B. Gold et al., McGraw-Hill, New York, 1969, pp. 203–232.