

Understanding Digital Signal Processing

Third Edition

This page intentionally left blank

Understanding Digital Signal Processing

Third Edition

Richard G. Lyons

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • 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 the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com

Library of Congress Cataloging-in-Publication Data

Lyons, Richard G., 1948-

Understanding digital signal processing / Richard G. Lyons.—3rd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-13-702741-9 (hardcover : alk. paper)

1. Signal processing—Digital techniques. I. Title.

TK5102.9.L96 2011

621.382'2—dc22

2010035407

Copyright © 2011 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-702741-5

ISBN-10: 0-13-702741-9

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
Fourth printing, August 2012

I dedicate this book to my 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; and to the skilled folks on the USENET newsgroup comp.dsp. They ask the good questions and provide the best answers. Finally, to Sigi Pardula (Bat-girl): Without your keeping the place running, this book wouldn't exist.

This page intentionally left blank

Contents

PREFACE xv

ABOUT THE AUTHOR xxiii

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 12
- 1.6 Time-Invariant Systems 17
- 1.7 The Commutative Property of Linear Time-Invariant Systems 18
- 1.8 Analyzing Linear Time-Invariant Systems 19
 - References 21
 - Chapter 1 Problems 23

2 PERIODIC SAMPLING 33

- 2.1 Aliasing: Signal Ambiguity in the Frequency Domain 33
- 2.2 Sampling Lowpass Signals 38
- 2.3 Sampling Bandpass Signals 42
- 2.4 Practical Aspects of Bandpass Sampling 45
 - References 49
 - Chapter 2 Problems 50

3 THE DISCRETE FOURIER TRANSFORM 59

- 3.1 Understanding the DFT Equation 60
- 3.2 DFT Symmetry 73

- 3.3 DFT Linearity 75
- 3.4 DFT Magnitudes 75
- 3.5 DFT Frequency Axis 77
- 3.6 DFT Shifting Theorem 77
- 3.7 Inverse DFT 80
- 3.8 DFT Leakage 81
- 3.9 Windows 89
- 3.10 DFT Scalloping Loss 96
- 3.11 DFT Resolution, Zero Padding, and Frequency-Domain Sampling 98
- 3.12 DFT Processing Gain 102
- 3.13 The DFT of Rectangular Functions 105
- 3.14 Interpreting the DFT Using the Discrete-Time Fourier Transform 120
 - References 124
 - Chapter 3 Problems 125

4 THE FAST FOURIER TRANSFORM 135

- 4.1 Relationship of the FFT to the DFT 136
- 4.2 Hints on Using FFTs in Practice 137
- 4.3 Derivation of the Radix-2 FFT Algorithm 141
- 4.4 FFT Input/Output Data Index Bit Reversal 149
- 4.5 Radix-2 FFT Butterfly Structures 151
- 4.6 Alternate Single-Butterfly Structures 154
 - References 158
 - Chapter 4 Problems 160

5 FINITE IMPULSE RESPONSE FILTERS 169

- 5.1 An Introduction to Finite Impulse Response (FIR) Filters 170
- 5.2 Convolution in FIR Filters 175
- 5.3 Lowpass FIR Filter Design 186
- 5.4 Bandpass FIR Filter Design 201
- 5.5 Highpass FIR Filter Design 203
- 5.6 Parks-McClellan Exchange FIR Filter Design Method 204
- 5.7 Half-band FIR Filters 207
- 5.8 Phase Response of FIR Filters 209
- 5.9 A Generic Description of Discrete Convolution 214

- 5.10 Analyzing FIR Filters 226
 - References 235
 - Chapter 5 Problems 238

6 INFINITE IMPULSE RESPONSE FILTERS 253

- 6.1 An Introduction to Infinite Impulse Response Filters 254
- 6.2 The Laplace Transform 257
- 6.3 The z-Transform 270
- 6.4 Using the z-Transform to Analyze IIR Filters 274
- 6.5 Using Poles and Zeros to Analyze IIR Filters 282
- 6.6 Alternate IIR Filter Structures 289
- 6.7 Pitfalls in Building IIR Filters 292
- 6.8 Improving IIR Filters with Cascaded Structures 295
- 6.9 Scaling the Gain of IIR Filters 300
- 6.10 Impulse Invariance IIR Filter Design Method 303
- 6.11 Bilinear Transform IIR Filter Design Method 319
- 6.12 Optimized IIR Filter Design Method 330
- 6.13 A Brief Comparison of IIR and FIR Filters 332
 - References 333
 - Chapter 6 Problems 336

7 SPECIALIZED DIGITAL NETWORKS AND FILTERS 361

- 7.1 Differentiators 361
- 7.2 Integrators 370
- 7.3 Matched Filters 376
- 7.4 Interpolated Lowpass FIR Filters 381
- 7.5 Frequency Sampling Filters: The Lost Art 392
 - References 426
 - Chapter 7 Problems 429

8 QUADRATURE SIGNALS 439

- 8.1 Why Care about Quadrature Signals? 440
- 8.2 The Notation of Complex Numbers 440
- 8.3 Representing Real Signals Using Complex Phasors 446
- 8.4 A Few Thoughts on Negative Frequency 450

8.5	Quadrature Signals in the Frequency Domain	451
8.6	Bandpass Quadrature Signals in the Frequency Domain	454
8.7	Complex Down-Conversion	456
8.8	A Complex Down-Conversion Example	458
8.9	An Alternate Down-Conversion Method	462
	References	464
	Chapter 8 Problems	465

9 THE DISCRETE HILBERT TRANSFORM 479

9.1	Hilbert Transform Definition	480
9.2	Why Care about the Hilbert Transform?	482
9.3	Impulse Response of a Hilbert Transformer	487
9.4	Designing a Discrete Hilbert Transformer	489
9.5	Time-Domain Analytic Signal Generation	495
9.6	Comparing Analytical Signal Generation Methods	497
	References	498
	Chapter 9 Problems	499

10 SAMPLE RATE CONVERSION 507

10.1	Decimation	508
10.2	Two-Stage Decimation	510
10.3	Properties of Downsampling	514
10.4	Interpolation	516
10.5	Properties of Interpolation	518
10.6	Combining Decimation and Interpolation	521
10.7	Polyphase Filters	522
10.8	Two-Stage Interpolation	528
10.9	z -Transform Analysis of Multirate Systems	533
10.10	Polyphase Filter Implementations	535
10.11	Sample Rate Conversion by Rational Factors	540
10.12	Sample Rate Conversion with Half-band Filters	543
10.13	Sample Rate Conversion with IFIR Filters	548
10.14	Cascaded Integrator-Comb Filters	550
	References	566
	Chapter 10 Problems	568

11 SIGNAL AVERAGING 589

- 11.1 Coherent Averaging 590
- 11.2 Incoherent Averaging 597
- 11.3 Averaging Multiple Fast Fourier Transforms 600
- 11.4 Averaging Phase Angles 603
- 11.5 Filtering Aspects of Time-Domain Averaging 604
- 11.6 Exponential Averaging 608
 - References 615
 - Chapter 11 Problems 617

12 DIGITAL DATA FORMATS AND THEIR EFFECTS 623

- 12.1 Fixed-Point Binary Formats 623
- 12.2 Binary Number Precision and Dynamic Range 632
- 12.3 Effects of Finite Fixed-Point Binary Word Length 634
- 12.4 Floating-Point Binary Formats 652
- 12.5 Block Floating-Point Binary Format 658
 - References 658
 - Chapter 12 Problems 661

13 DIGITAL SIGNAL PROCESSING TRICKS 671

- 13.1 Frequency Translation without Multiplication 671
- 13.2 High-Speed Vector Magnitude Approximation 679
- 13.3 Frequency-Domain Windowing 683
- 13.4 Fast Multiplication of Complex Numbers 686
- 13.5 Efficiently Performing the FFT of Real Sequences 687
- 13.6 Computing the Inverse FFT Using the Forward FFT 699
- 13.7 Simplified FIR Filter Structure 702
- 13.8 Reducing A/D Converter Quantization Noise 704
- 13.9 A/D Converter Testing Techniques 709
- 13.10 Fast FIR Filtering Using the FFT 716
- 13.11 Generating Normally Distributed Random Data 722
- 13.12 Zero-Phase Filtering 725
- 13.13 Sharpened FIR Filters 726
- 13.14 Interpolating a Bandpass Signal 728
- 13.15 Spectral Peak Location Algorithm 730

- 13.16 Computing FFT Twiddle Factors 734
- 13.17 Single Tone Detection 737
- 13.18 The Sliding DFT 741
- 13.19 The Zoom FFT 749
- 13.20 A Practical Spectrum Analyzer 753
- 13.21 An Efficient Arctangent Approximation 756
- 13.22 Frequency Demodulation Algorithms 758
- 13.23 DC Removal 761
- 13.24 Improving Traditional CIC Filters 765
- 13.25 Smoothing Impulsive Noise 770
- 13.26 Efficient Polynomial Evaluation 772
- 13.27 Designing Very High-Order FIR Filters 775
- 13.28 Time-Domain Interpolation Using the FFT 778
- 13.29 Frequency Translation Using Decimation 781
- 13.30 Automatic Gain Control (AGC) 783
- 13.31 Approximate Envelope Detection 784
- 13.32 A Quadrature Oscillator 786
- 13.33 Specialized Exponential Averaging 789
- 13.34 Filtering Narrowband Noise Using Filter Nulls 792
- 13.35 Efficient Computation of Signal Variance 797
- 13.36 Real-time Computation of Signal Averages and Variances 799
- 13.37 Building Hilbert Transformers from Half-band Filters 802
- 13.38 Complex Vector Rotation with Arctangents 805
- 13.39 An Efficient Differentiating Network 810
- 13.40 Linear-Phase DC-Removal Filter 812
- 13.41 Avoiding Overflow in Magnitude Computations 815
- 13.42 Efficient Linear Interpolation 815
- 13.43 Alternate Complex Down-conversion Schemes 816
- 13.44 Signal Transition Detection 820
- 13.45 Spectral Flipping around Signal Center Frequency 821
- 13.46 Computing Missing Signal Samples 823
- 13.47 Computing Large DFTs Using Small FFTs 826
- 13.48 Computing Filter Group Delay without Arctangents 830
- 13.49 Computing a Forward and Inverse FFT Using a Single FFT 831
- 13.50 Improved Narrowband Lowpass IIR Filters 833
- 13.51 A Stable Goertzel Algorithm 838
- References 840

A THE ARITHMETIC OF COMPLEX NUMBERS 847

- A.1 Graphical Representation of Real and Complex Numbers 847
- A.2 Arithmetic Representation of Complex Numbers 848
- A.3 Arithmetic Operations of Complex Numbers 850
- A.4 Some Practical Implications of Using Complex Numbers 856

B CLOSED FORM OF A GEOMETRIC SERIES 859**C TIME REVERSAL AND THE DFT 863****D MEAN, VARIANCE, AND STANDARD DEVIATION 867**

- D.1 Statistical Measures 867
- D.2 Statistics of Short Sequences 870
- D.3 Statistics of Summed Sequences 872
- D.4 Standard Deviation (RMS) of a Continuous Sinewave 874
- D.5 Estimating Signal-to-Noise Ratios 875
- D.6 The Mean and Variance of Random Functions 879
- D.7 The Normal Probability Density Function 882

E DECIBELS (DB AND DBM) 885

- E.1 Using Logarithms to Determine Relative Signal Power 885
- E.2 Some Useful Decibel Numbers 889
- E.3 Absolute Power Using Decibels 891

F DIGITAL FILTER TERMINOLOGY 893**G FREQUENCY SAMPLING FILTER DERIVATIONS 903**

- G.1 Frequency Response of a Comb Filter 903
- G.2 Single Complex FSF Frequency Response 904
- G.3 Multisection Complex FSF Phase 905
- G.4 Multisection Complex FSF Frequency Response 906

- G.5 Real FSF Transfer Function 908
- G.6 Type-IV FSF Frequency Response 910

H FREQUENCY SAMPLING FILTER DESIGN TABLES 913

I COMPUTING CHEBYSHEV WINDOW SEQUENCES 927

- I.1 Chebyshev Windows for FIR Filter Design 927
- I.2 Chebyshev Windows for Spectrum Analysis 929

INDEX 931

Preface

This book is an expansion of previous editions of *Understanding Digital Signal Processing*. Like those earlier editions, its goals are (1) to help beginning students understand the theory of digital signal processing (DSP) and (2) to provide practical DSP information, not found in other books, to help working engineers/scientists design and test their signal processing systems. Each chapter of this book contains new information beyond that provided in earlier editions.

It's traditional at this point in the preface of a DSP textbook for the author to tell readers why they should learn DSP. I don't need to tell you how important DSP is in our modern engineering world. You already know that. I'll just say that the future of electronics is DSP, and with this book you will not be left behind.

FOR INSTRUCTORS

This third edition is appropriate as the text for a one- or two-semester undergraduate course in DSP. It follows the DSP material I cover in my corporate training activities and a signal processing course I taught at the University of California Santa Cruz Extension. To aid students in their efforts to learn DSP, this third edition provides additional explanations and examples to increase its tutorial value. To test a student's understanding of the material, homework problems have been included at the end of each chapter. (For qualified instructors, a Solutions Manual is available from Prentice Hall.)

FOR PRACTICING ENGINEERS

To help working DSP engineers, the changes in this third edition include, but are not limited to, the following:

- Practical guidance in building discrete differentiators, integrators, and matched filters
- Descriptions of statistical measures of signals, variance reduction by way of averaging, and techniques for computing real-world signal-to-noise ratios (SNRs)
- A significantly expanded chapter on sample rate conversion (multirate systems) and its associated filtering
- Implementing fast convolution (FIR filtering in the frequency domain)
- IIR filter scaling
- Enhanced material covering techniques for analyzing the behavior and performance of digital filters
- Expanded descriptions of industry-standard binary number formats used in modern processing systems
- Numerous additions to the popular “Digital Signal Processing Tricks” chapter

FOR STUDENTS

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 look rather complicated. The content here, you say, looks suspiciously like material in technical journals and textbooks whose meaning has eluded you in the past. Well, this is not just another book on digital signal processing.

In this book I provide a gentle, but thorough, explanation of the theory and practice of DSP. The text is not written so that you *may* understand the material, but so that you *must* understand the material. I’ve attempted to avoid the traditional instructor–student relationship and have tried to make reading this book seem like talking to a friend while walking in the park. I’ve used just enough mathematics to help you develop a fundamental understanding of DSP theory and have illustrated that theory with practical examples.

I have designed the homework problems to be more than mere exercises that assign values to variables for the student to plug into some equation in order to compute a result. Instead, the homework problems are designed to

be as educational as possible in the sense of expanding on and enabling further investigation of specific aspects of DSP topics covered in the text. Stated differently, the homework problems are not designed to induce “death by algebra,” but rather to improve your understanding of DSP. Solving the problems helps you become proactive in your own DSP education instead of merely being an inactive recipient of DSP information.

THE JOURNEY

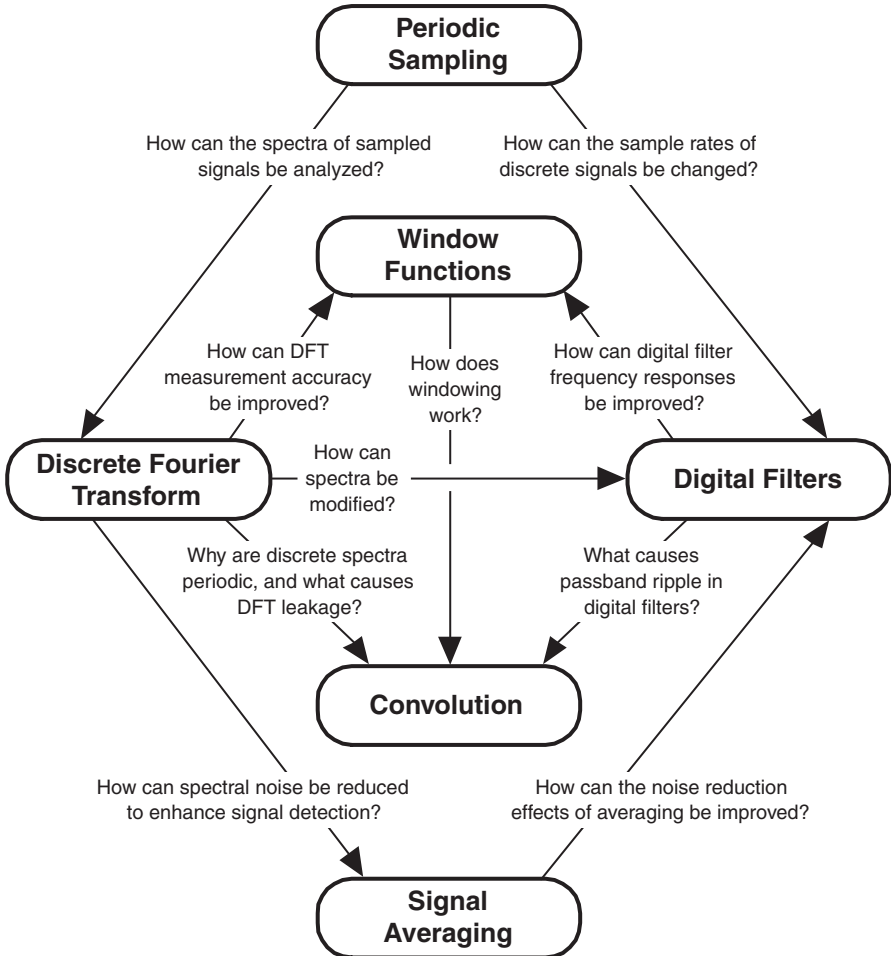
Learning digital signal processing is not something you accomplish; it’s a journey you take. When you gain an understanding of one topic, questions arise that cause you to investigate some other facet of digital signal processing.[†] Armed with more knowledge, you’re likely to begin exploring further aspects of digital signal processing much like those shown in the diagram on page xviii. 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 certainly help. DSP simulation software allows the beginner to verify signal processing theory through the time-tested trial and error process.[‡] In particular, software routines that plot signal data, perform the fast Fourier transforms, 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 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 the only way to learn mathematics is through careful study. The same applies to digital signal processing. Also, don’t worry if you need 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 can become a little involved. They deserve your thoughtful attention. So, go slowly and read the material twice if necessary; you’ll be glad you did. If you show persistence, to quote Susan B. Anthony, “Failure is impossible.”

[†]“You see I went on with this research just the way it led me. This is the only way I ever heard of research going. I asked a question, devised some method of getting an answer, and got—a fresh question. Was this possible, or that possible? You cannot imagine what this means to an investigator, what an intellectual passion grows upon him. You cannot imagine the strange colourless delight of these intellectual desires” (Dr. Moreau—infamous physician and vivisectionist from H.G. Wells’ *Island of Dr. Moreau*, 1896).

[‡]“One must learn by doing the thing; for though you think you know it, you have no certainty until you try it” (Sophocles, 496–406 B.C.).



COMING ATTRACTIONS

Chapter 1 begins by establishing the notation used throughout the remainder of the book. 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 the concept of sampling a

continuous signal is not complicated, there are mathematical subtleties in the process that require thoughtful attention. Beginning gradually with simple examples of lowpass sampling, we then proceed to the interesting subject of bandpass sampling. Chapter 2 explains and quantifies the frequency-domain ambiguity (aliasing) associated with these important topics.

Chapter 3 is devoted to one of the foremost topics in digital signal processing, the discrete Fourier transform (DFT) used for spectrum analysis. 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 reader is likely to encounter in the literature.

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 the reader use the FFT in practice.

Chapter 5 ushers in the subject of digital filtering. Beginning with a simple lowpass 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 lowpass FIR filter designs to bandpass and highpass digital filters are presented, and the popular Parks-McClellan (Remez) Exchange 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 is devoted to 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 that of 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 introduces specialized networks known as *digital differentiators*, *integrators*, and *matched filters*. In addition, this chapter covers two specialized digital filter types that have not received their deserved exposure in traditional DSP textbooks. Called *interpolated FIR* and *frequency sampling* filters, providing improved lowpass filtering computational efficiency, they belong in our arsenal of filter design techniques. Although these are FIR filters, their introduction is delayed to this chapter because familiarity with the z-transform (in Chapter 6) makes the properties of these filters easier to understand.

Chapter 8 presents a detailed description of quadrature signals (also called *complex* signals). Because quadrature signal theory has become so important in recent years, in both signal analysis and digital communications implementations, it deserves its own chapter. Using three-dimensional illustrations, this chapter gives solid physical meaning to the mathematical notation, processing advantages, and use of quadrature signals. Special emphasis is given to quadrature sampling (also called *complex down-conversion*).

Chapter 9 provides a mathematically gentle, but technically thorough, description of the Hilbert transform—a process used to generate a quadrature (complex) signal from a real signal. In this chapter we describe the properties, behavior, and design of practical Hilbert transformers.

Chapter 10 presents an introduction to the fascinating and useful process of sample rate conversion (changing the effective sample rate of discrete data sequences through decimation or interpolation). Sample rate conversion—so useful in improving the performance and reducing the computational complexity of many signal processing operations—is essentially an exercise in lowpass filter design. As such, polyphase and cascaded integrator-comb filters are described in detail in this chapter.

Chapter 11 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 that chapter the popular scheme known as *exponential averaging* is covered in some detail.

Chapter 12 presents an introduction to the various binary number formats 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 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 that 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 13 provides the literature's most comprehensive collection of *tricks of the trade* used by DSP professionals to make their processing algorithms more efficient. These 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 clever schemes require an understanding of the material from the previous chapters, making the last chapter an appropriate place to keep our arsenal 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 nature and mathematics of digital signal processing. A comprehensive description of the arithmetic of complex numbers is covered in Appendix A, and Appendix B derives the often used, but seldom explained, closed form of a geometric series. The subtle aspects and two forms of time reversal in discrete systems (of which zero-phase digital filtering is an application) are explained in Appendix C. 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. Appendix F, in a slightly different vein, provides a glossary of the terminology used in the field of digital filters. Appendices G and H provide supplementary information for designing and analyzing specialized digital filters. Appendix I explains the computation of Chebyshev window sequences.

ACKNOWLEDGMENTS

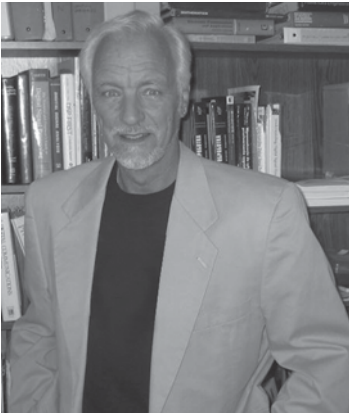
Much of the new material in this edition is a result of what I've learned from those clever folk on the USENET newsgroup comp.dsp. (I could list a dozen names, but in doing so I'd make 12 friends and 500 enemies.) So, I say thanks to my DSP pals on comp.dsp for teaching me so much signal processing theory.

In addition to the reviewers of previous editions of this book, I thank Randy Yates, Clay Turner, and Ryan Groulx for their time and efforts to help me improve the content of this book. I am especially indebted to my eagle-eyed mathematician friend Antoine Trux for his relentless hard work to both enhance this DSP material and create a homework Solutions Manual.

As before, I thank my acquisitions editor, Bernard Goodwin, for his patience and guidance, and his skilled team of production people, project editor Elizabeth Ryan in particular, at Prentice Hall.

If you're still with me this far in this Preface, I end by saying I had a ball writing this book and sincerely hope you benefit from reading it. If you have any comments or suggestions regarding this material, or detect any errors no matter how trivial, please send them to me at R.Lyons@ieee.org. I promise I will reply to your e-mail.

About the Author

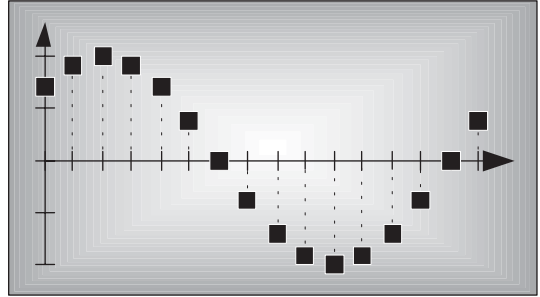


Richard Lyons is a consulting systems engineer and lecturer with Besser Associates in Mountain View, California. He has been the lead hardware engineer for numerous signal processing systems for both the National Security Agency (NSA) and Northrop Grumman Corp. Lyons has taught DSP at the University of California Santa Cruz Extension and authored numerous articles on DSP. As associate editor for the *IEEE Signal Processing Magazine* he created, edits, and contributes to the magazine's "DSP Tips & Tricks" column.

This page intentionally left blank

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 perform spectral analysis, design digital filters, develop voice recognition, data communication, and image compression processes using software that's interactive both in 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 process-

[†] 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.

ing, 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_0 described by the equation

$$x(t) = \sin(2\pi f_0 t). \quad (1-1)$$

The frequency f_0 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

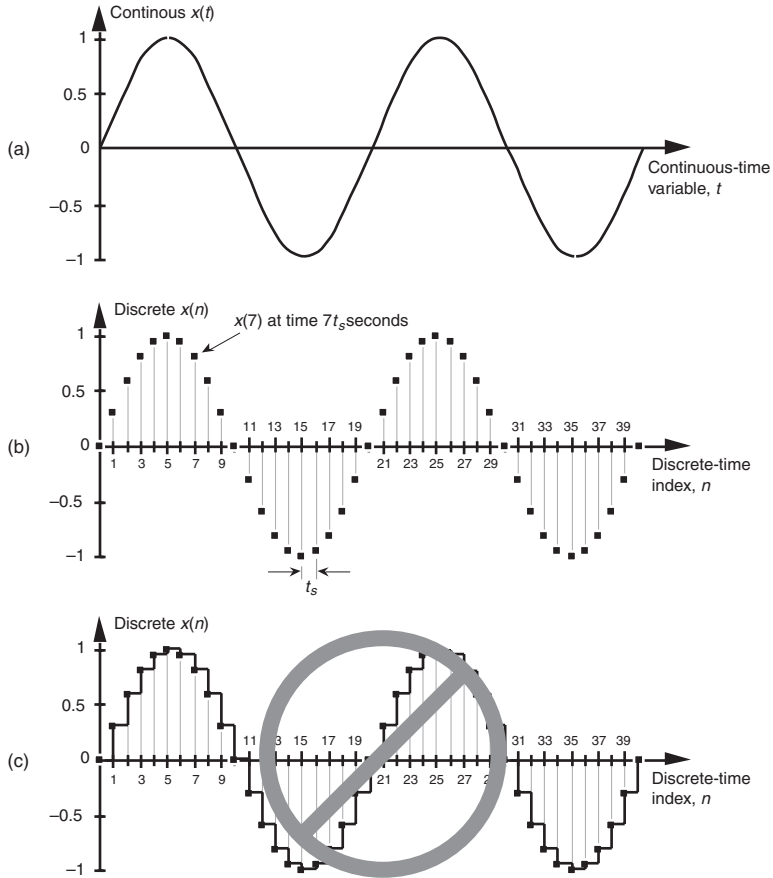


Figure 1-1 A time-domain sinewave: (a) continuous waveform representation; (b) discrete sample representation; (c) discrete samples with connecting lines.

one million Hz.[†]) With t in Eq. 1-1 representing time in seconds, the $f_0 t$ factor has dimensions of cycles, and the complete $2\pi f_0 t$ 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 volt-

[†] 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.

age, 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 stair-step 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:

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

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

$$x(n) = \sin(2\pi f_0 n t_s). \tag{1-3}$$

(Here again, the $2\pi f_0 n t_s$ 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 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 $n t_s$ 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

$$y(n) = 2x(n) - 1. \quad (1-4)$$

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 $y(4) = 17$.

Equation (1–4) is formally called a *difference equation*. (In this book we won't be working with differential equations or partial differential equations. However, we will, now and then, work with partially difficult equations.)

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.

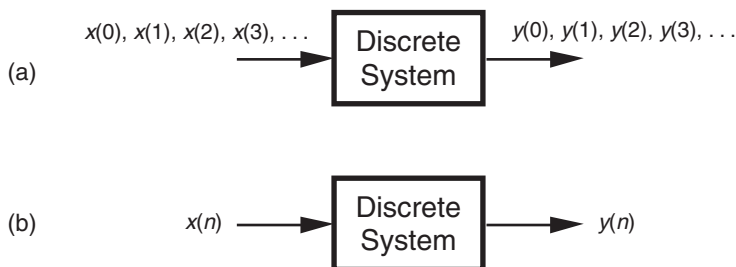


Figure 1-2 With an input applied, a discrete system provides an output: (a) the input and output are sequences of individual values; (b) input and output using the abbreviated notation of $x(n)$ and $y(n)$.

Given that the t_s sample period is, say, 0.05 milliseconds/sample, the period of the sinewave is

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

Because the frequency of a sinewave is the reciprocal of its period, we now know that the sinewave's absolute frequency is $1/(1 \text{ ms})$, 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 when dealing with discrete systems, absolute frequency determination in Hz is dependent on the sampling frequency

$$f_s = 1/t_s. \quad (1-5')$$

We'll be reminded of this dependence throughout the remainder 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

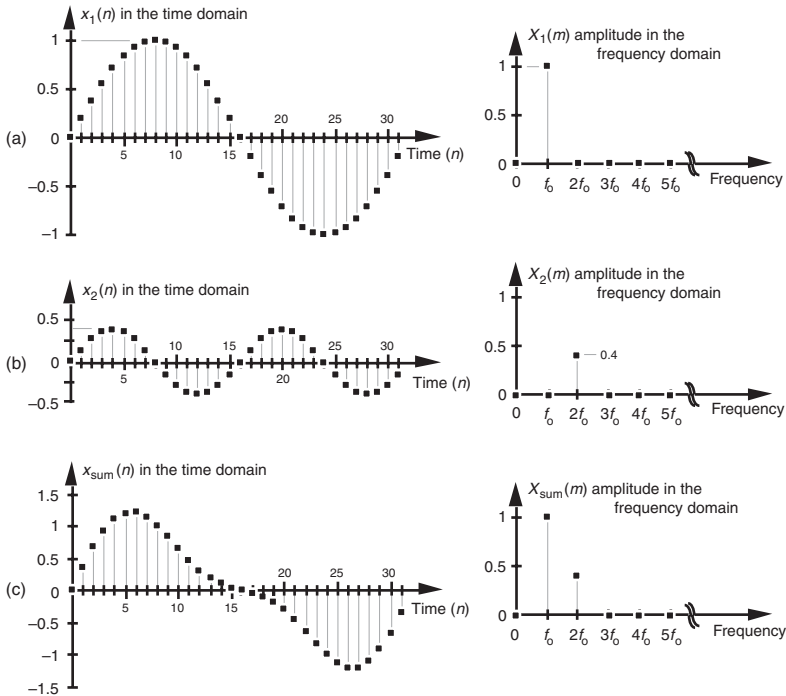


Figure 1-3 Time- and frequency-domain graphical representations: (a) sinewave of frequency f_0 ; (b) reduced amplitude sinewave of frequency $2f_0$; (c) sum of the two sinewaves.

way of example, let's say we have a discrete sinewave sequence $x_1(n)$ with an arbitrary frequency f_0 Hz as shown on the left side of Figure 1-3(a). We can also characterize $x_1(n)$ by showing its spectral content, the $X_1(m)$ sequence on the right side of Figure 1-3(a), indicating that it has a single spectral component, and no other frequency content. Although we won't dwell on 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 $2f_0$. The discrete sample values of $x_2(n)$ are expressed by the equation

$$x_2(n) = 0.4 \cdot \sin(2\pi 2f_0 n t_s). \quad (1-6)$$

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

$$x_{\text{sum}}(n) = x_1(n) + x_2(n) = \sin(2\pi f_0 n t_s) + 0.4 \cdot \sin(2\pi 2f_0 n t_s), \quad (1-7)$$

and its time- and frequency-domain representations are those given in Figure 1-3(c). We interpret the $X_{\text{sum}}(m)$ frequency-domain depiction, the *spectrum*, in Figure 1-3(c) to indicate that $x_{\text{sum}}(n)$ has a frequency component of f_0 Hz and a reduced-amplitude frequency component of $2f_0$ 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, $X_{\text{sum}}(m)$ is listed as

$$\begin{aligned} X_{\text{sum}}(0) &= 0 && \text{(1st } X_{\text{sum}}(m) \text{ value, index } m = 0) \\ X_{\text{sum}}(1) &= 1.0 && \text{(2nd } X_{\text{sum}}(m) \text{ value, index } m = 1) \\ X_{\text{sum}}(2) &= 0.4 && \text{(3rd } X_{\text{sum}}(m) \text{ value, index } m = 2) \\ X_{\text{sum}}(3) &= 0 && \text{(4th } X_{\text{sum}}(m) \text{ value, index } m = 3) \\ &\dots && \dots \end{aligned}$$

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 $X_{\text{sum}}(m)$ 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). 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 12—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 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.

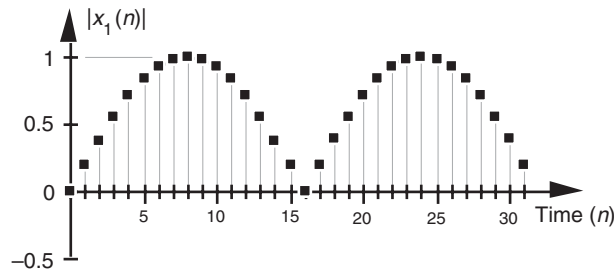


Figure 1-4 Magnitude samples, $|x_1(n)|$, of the time waveform in Figure 1-3(a).

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

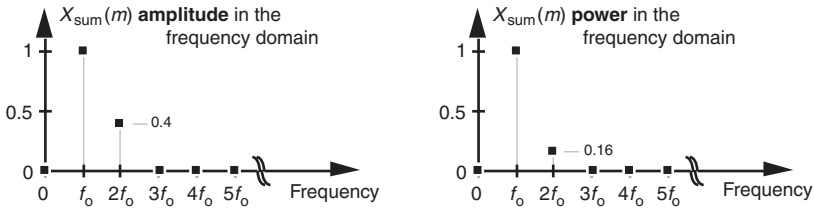


Figure 1-5 Frequency-domain amplitude and frequency-domain power of the $x_{\text{sum}}(n)$ time waveform in Figure 1-3(c).

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 ($| \cdot |$) 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

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

or

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

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, 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 $X_{\text{sum}}(m)$ 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

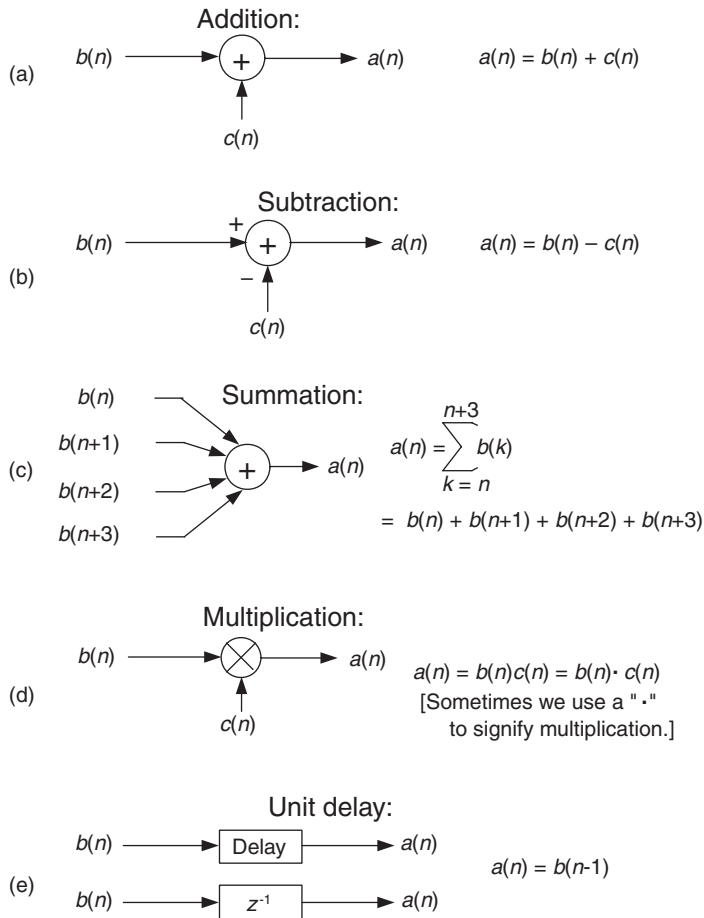


Figure 1–6 Terminology and symbols used in digital signal processing block diagrams.

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 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

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

This means that

$$\begin{array}{ll} \text{when } n = 0, \text{ index } k \text{ goes from 0 to 3, so} & a(0) = b(0) + b(1) + b(2) + b(3) \\ \text{when } n = 1, \text{ index } k \text{ goes from 1 to 4, so} & a(1) = b(1) + b(2) + b(3) + b(4) \\ \text{when } n = 2, \text{ index } k \text{ goes from 2 to 5, so} & a(2) = b(2) + b(3) + b(4) + b(5) \\ \text{when } n = 3, \text{ index } k \text{ goes from 3 to 6, so} & a(3) = b(3) + b(4) + b(5) + b(6) \\ \dots & \dots \end{array} \quad (1-10)$$

and so on.

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 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:

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

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

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

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

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

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

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

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 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

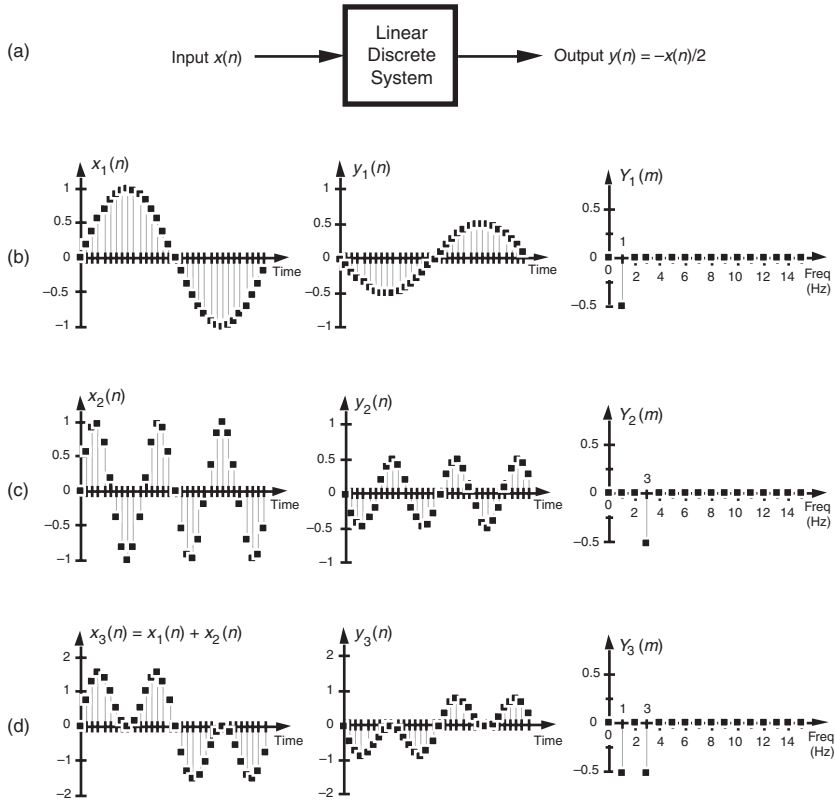


Figure 1-7 Linear system input-to-output relationships: (a) system block diagram where $y(n) = -x(n)/2$; (b) system input and output with a 1 Hz sinewave applied; (c) with a 3 Hz sinewave applied; (d) with the sum of 1 Hz and 3 Hz sinewaves applied.

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

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

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

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

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

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

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

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

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

$$\begin{aligned} y_1(n) &= \frac{\cos(2\pi \cdot 1 \cdot n t_s - 2\pi \cdot 1 \cdot n t_s)}{2} - \frac{\cos(2\pi \cdot 1 \cdot n t_s + 2\pi \cdot 1 \cdot n t_s)}{2} \\ &= \frac{\cos(0)}{2} - \frac{\cos(4\pi \cdot 1 \cdot n t_s)}{2} = \frac{1}{2} - \frac{\cos(2\pi \cdot 2 \cdot n t_s)}{2}, \end{aligned} \quad (1-20)$$

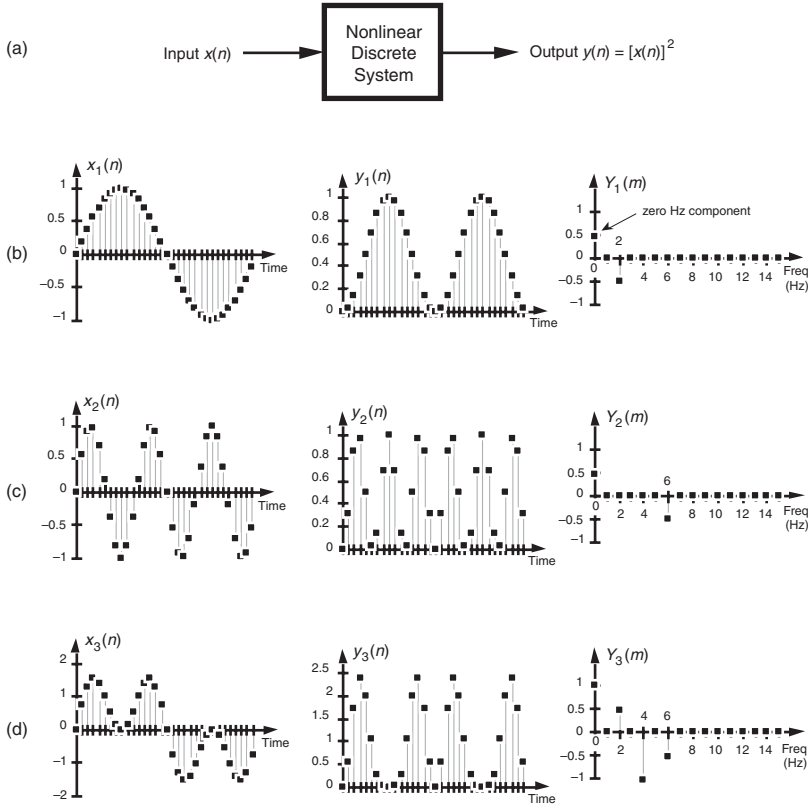


Figure 1-8 Nonlinear system input-to-output relationships: (a) system block diagram where $y(n) = (x(n))^2$; (b) system input and output with a 1 Hz sinewave applied; (c) with a 3 Hz sinewave applied; (d) with the sum of 1 Hz and 3 Hz sinewaves applied.

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

$$(a+b)^2 = a^2 + 2ab + b^2, \quad (1-21)$$

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

$$\begin{aligned} 2ab &= 2 \cdot \sin(2\pi \cdot 1 \cdot nt_s) \cdot \sin(2\pi \cdot 3 \cdot nt_s) \\ &= \frac{2 \cos(2\pi \cdot 1 \cdot nt_s - 2\pi \cdot 3 \cdot nt_s)}{2} - \frac{2 \cos(2\pi \cdot 1 \cdot nt_s + 2\pi \cdot 3 \cdot nt_s)}{2} \quad (1-22) \\ &= \cos(2\pi \cdot 2 \cdot nt_s) - \cos(2\pi \cdot 4 \cdot nt_s).^\dagger \end{aligned}$$

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 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.

[†] 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 $\cos(2\pi \cdot 2 \cdot nt_s)$.

1.6 TIME-INVARIANT SYSTEMS

A time-invariant system is one where a time delay (or shift) in the input sequence causes an 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

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

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

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

where k is some integer representing k sample period time delays. For a system to be time invariant, Eq. (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

$$x'(n) = x(n-4). \quad (1-25)$$

Equation (1-25) tells us that the input sequence $x'(n)$ is equal to sequence $x(n)$ shifted to the right by $k = -4$ samples. That is, $x'(4) = x(0)$, $x'(5) = x(1)$, $x'(6) = x(2)$, and so on as shown in 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 right by four samples, or $y'(n) = y(n-4)$. We can see that $y'(4) = y(0)$, $y'(5) = y(1)$, $y'(6) = y(2)$, and so on as shown in Figure 1-9(c). For time-invariant systems, the time shifts in $x'(n)$ and $y'(n)$ are equal. Take careful notice of the minus sign in Eq. (1-25). In later chapters, that is the notation we'll use to algebraically describe a time-delayed discrete sequence.

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 Chapter 10.

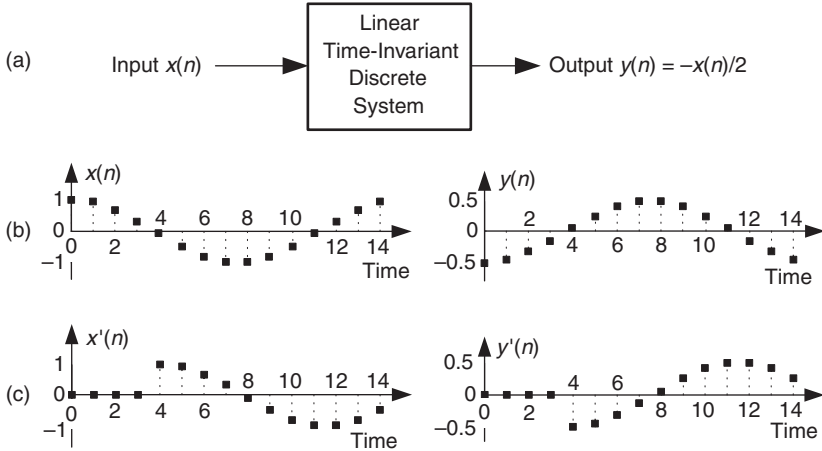


Figure 1-9 Time-invariant system input/output relationships: (a) system block diagram, $y(n) = -x(n)/2$; (b) system input/output with a sinewave input; (c) input/output when a sinewave, delayed by four samples, is the input.

1.7 THE COMMUTATIVE PROPERTY OF LINEAR TIME-INVARIANT SYSTEMS

Although we don't substantiate this fact until we reach Section 6.11, 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 iden-

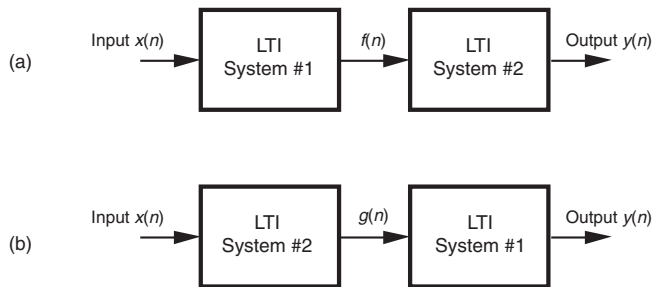


Figure 1-10 Linear time-invariant (LTI) systems in series: (a) block diagram of two LTI systems; (b) swapping the order of the two systems does not change the resultant output $y(n)$.

tical $y(n)$ 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 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]. The concepts in the two previous sentences are among the most important principles in all of digital signal processing!

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

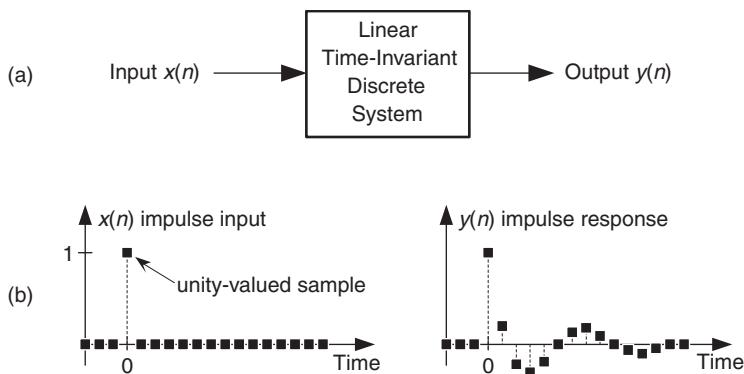


Figure 1-11 LTI system unit impulse response sequences: (a) system block diagram; (b) impulse input sequence $x(n)$ and impulse response output sequence $y(n)$.

our signal processing toolboxes as we journey through the subject of digital signal processing.

In the testing (analyzing) of continuous linear systems, engineers often use a narrow-in-time impulsive signal as an input signal to their systems. Mechanical engineers give their systems a little whack with a hammer, and electrical engineers working with analog-voltage systems generate a very narrow voltage spike as an impulsive input. Audio engineers, who need an impulsive acoustic test signal, sometimes generate an audio impulse by firing a starter pistol.

In the world of DSP, an impulse sequence called a *unit impulse* takes the form

$$x(n) = \dots 0, 0, 0, 0, 0, A, 0, 0, 0, 0, \dots \quad (1-26)$$

The value A is often set equal to one. The leading sequence of zero-valued samples, before the A -valued sample, must be a bit longer than the length of the transient response of the system under test in order to initialize the system to its zero state. The trailing sequence of zero-valued samples, following the A -valued sample, must be a bit longer than the transient response of the system under test in order to capture the system's entire $y(n)$ impulse response output sequence.

Let's further explore this notion of impulse response testing to determine the frequency response of a discrete system (and take an opportunity to start using the operational symbols introduced in Section 1.3). Consider the block diagram of a 4-point moving averager shown in Figure 1-12(a). As the $x(n)$ input samples march their way through the system, at each time index n four successive input samples are averaged to compute a single $y(n)$ output. As we'll learn in subsequent chapters, a *moving averager* behaves like a digital lowpass filter. However, we can quickly illustrate that fact now.

If we apply an impulse input sequence to the system, we'll obtain its $y(n)$ impulse response output shown in Figure 1-12(b). The $y(n)$ output is computed using the following difference equation:

$$y(n) = \frac{1}{4}[x(n) + x(n-1) + x(n-2) + x(n-3)] = \frac{1}{4} \sum_{k=n-3}^n x(k). \quad (1-27)$$

If we then perform a discrete Fourier transform (a process we cover in much detail in Chapter 3) on $y(n)$, we obtain the $Y(m)$ frequency-domain information, allowing us to plot the frequency magnitude response of the 4-point moving averager as shown in Figure 1-12(c). So we see that a moving averager indeed has the characteristic of a lowpass filter. That is, the averager attenuates (reduces the amplitude of) high-frequency signal content applied to its input.

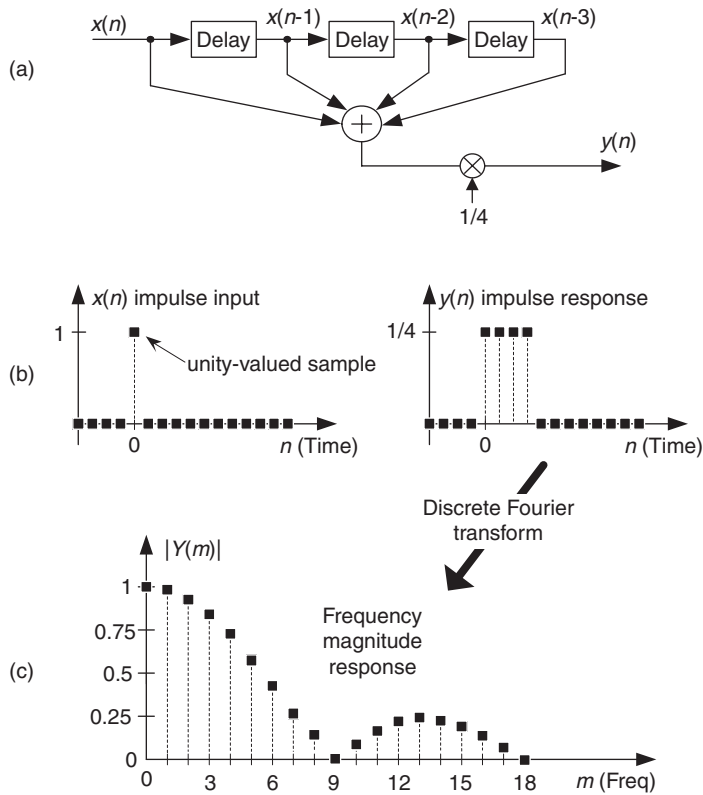


Figure 1-12 Analyzing a moving averager: (a) averager block diagram; (b) impulse input and impulse response; (c) averager frequency magnitude response.

OK, this concludes our brief introduction to discrete sequences and systems. In later chapters we'll learn the details of discrete Fourier transforms, discrete system impulse responses, and digital filters.

REFERENCES

[1] Karplus, W. J., and Soroka, W. W. *Analog Methods*, 2nd ed., 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 Proc.*, 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 1 PROBLEMS

- 1.1 This problem gives us practice in thinking about sequences of numbers. For centuries mathematicians have developed clever ways of computing π . In 1671 the Scottish mathematician James Gregory proposed the following very simple series for calculating π :

$$\pi \approx 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \dots \right).$$

Thinking of the terms inside the parentheses as a sequence indexed by the variable n , where $n = 0, 1, 2, 3, \dots, 100$, write Gregory's algorithm in the form

$$\pi \approx 4 \cdot \sum_{n=0}^{100} (-1)^? \cdot ?$$

replacing the “?” characters with expressions in terms of index n .

- 1.2 One of the ways to obtain discrete sequences, for follow-on processing, is to digitize a continuous (analog) signal with an analog-to-digital (A/D) converter. A 6-bit A/D converter's output words (6-bit binary words) can only represent $2^6=64$ different numbers. (We cover this digitization, *sampling*, and A/D converters in detail in upcoming chapters.) Thus we say the A/D converter's “digital” output can only represent a finite number of amplitude values. Can you think of a continuous time-domain electrical signal that only has a finite number of amplitude values? If so, draw a graph of that continuous-time signal.
- 1.3 On the Internet, the author once encountered the following line of C-language code

$$\text{PI} = 2 * \text{asin}(1.0);$$

whose purpose was to define the constant π . In standard mathematical notation, that line of code can be described by

$$\pi = 2 \cdot \sin^{-1}(1).$$

Under what assumption does the above expression correctly define the constant π ?

- 1.4 Many times in the literature of signal processing you will encounter the identity

$$x^0 = 1.$$

That is, x raised to the zero power is equal to one. Using the Laws of Exponents, prove the above expression to be true.

- 1.5 Recall that for discrete sequences the t_s sample period (the time period between samples) is the reciprocal of the sample frequency f_s . Write the equations, as we did in the text's Eq. (1–3), describing time-domain sequences for unity-amplitude cosine waves whose f_o frequencies are

(a) $f_o = f_s/2$, one-half the sample rate,

(b) $f_o = f_s/4$, one-fourth the sample rate,

(c) $f_o = 0$ (zero) Hz.

- 1.6 Draw the three time-domain cosine wave sequences, where a sample value is represented by a dot, described in Problem 1.5. The correct solution to Part (a) of this problem is a useful sequence used to convert some lowpass digital filters into highpass filters. (Chapter 5 discusses that topic.) The correct solution to Part (b) of this problem is an important discrete sequence used for *frequency translation* (both for signal *down-conversion* and *up-conversion*) in modern-day wireless communications systems. The correct solution to Part (c) of this problem should convince us that it's perfectly valid to describe a cosine sequence whose frequency is zero Hz.

- 1.7 Draw the three time-domain sequences of unity-amplitude sinewaves (not cosine waves) whose frequencies are

(a) $f_o = f_s/2$, one-half the sample rate,

(b) $f_o = f_s/4$, one-fourth the sample rate,

(c) $f_o = 0$ (zero) Hz.

The correct solutions to Parts (a) and (c) show us that the two frequencies, 0 Hz and $f_s/2$ Hz, are special frequencies in the world of discrete signal processing. What is *special* about the sinewave sequences obtained from the above Parts (a) and (c)?

- 1.8 Consider the infinite-length time-domain sequence $x(n)$ in Figure P1–8. Draw the first eight samples of a shifted time sequence defined by

$$x_{\text{shift}}(n) = x(n+1).$$

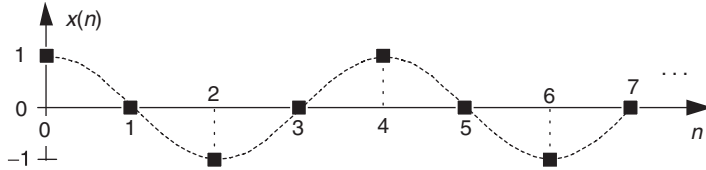


Figure P1-8

- 1.9 Assume, during your reading of the literature of DSP, you encounter the process shown in Figure P1-9. The $x(n]$ input sequence, whose f_s sample rate is 2500 Hz, is multiplied by a sinusoidal $m(n]$ sequence to produce the $y(n]$ output sequence. What is the frequency, measured in Hz, of the sinusoidal $m(n]$ sequence?

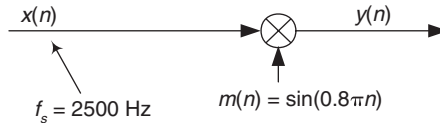


Figure P1-9

- 1.10 There is a process in DSP called an “ N -point running sum” (a kind of digital lowpass filter, actually) that is described by the following equation:

$$y(n) = \sum_{p=0}^{N-1} x(n-p).$$

Write out, giving the indices of all the $x()$ terms, the algebraic expression that describes the computations needed to compute $y(9)$ when $N=6$.

- 1.11 A 5-point *moving averager* can be described by the following difference equation:

$$y(n) = \frac{1}{5} [x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)] = \frac{1}{5} \sum_{k=n-4}^n x(k). \quad (\text{P1-1})$$

The averager’s signal-flow block diagram is shown in Figure P1-11, where the $x(n]$ input samples flow through the averager from left to right.

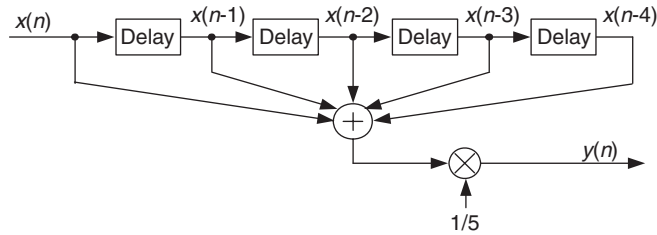


Figure P1-11

Equation (P1-1) is equivalent to

$$y(n) = \frac{x(n)}{5} + \frac{x(n-1)}{5} + \frac{x(n-2)}{5} + \frac{x(n-3)}{5} + \frac{x(n-4)}{5} \tag{P1-2}$$
$$= \sum_{k=n-4}^n \frac{x(k)}{5}.$$

- (a) Draw the block diagram of the discrete system described by Eq. (P1-2).
- (b) The *moving average* processes described by Eqs. (P1-1) and (P1-2) have identical impulse responses. Draw that impulse response.
- (c) If you had to implement (using programmable hardware or assembling discrete hardware components) either Eq. (P1-1) or Eq. (P1-2), which would you choose? Explain why.

1.12 In this book we will look at many two-dimensional drawings showing the value of one variable (y) plotted as a function of another variable (x). Stated in different words, we'll graphically display what are the values of a y axis variable for various values of an x axis variable. For example, Figure P1-12(a) plots the weight of a male child as a function of the child's age. The dimension of the x axis is years

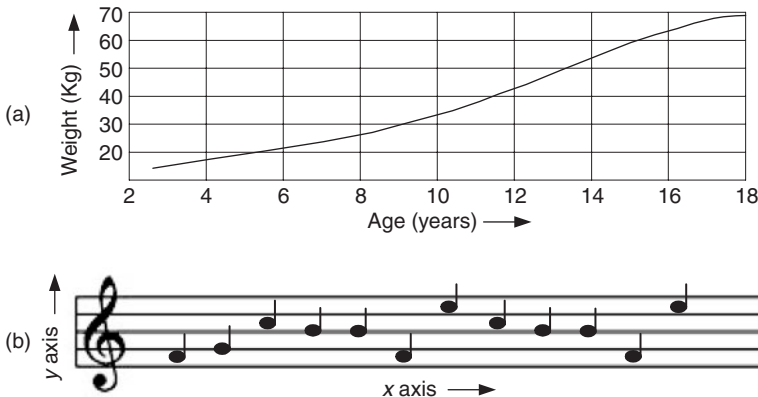


Figure P1-12

and the dimension of the y axis is kilograms. What are the dimensions of the x and y axes of the familiar two-dimensional plot given in Figure P1–12(b)?

- 1.13 Let's say you are writing software code to generate an $x(n)$ test sequence composed of the sum of two equal-amplitude discrete cosine waves, as

$$x(n) = \cos(2\pi f_0 n t_s + \phi) + \cos(2\pi f_0 n t_s)$$

where t_s is the time between your $x(n)$ samples, and ϕ is a constant phase shift measured in radians. An example $x(n)$ when $\phi = \pi/2$ is shown in Figure P1–13 where the $x(n)$ sequence, represented by the circular dots, is a single sinusoid whose frequency is f_0 Hz.

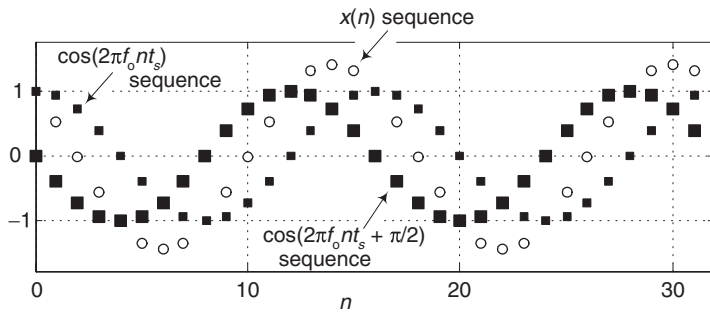


Figure P1–13

Using the trigonometric identity $\cos(\alpha+\beta) + \cos(\alpha-\beta) = 2\cos(\alpha)\cos(\beta)$, derive an equation for $x(n)$ that is of the form

$$x(n) = 2\cos(\alpha)\cos(\beta)$$

where variables α and β are in terms of $2\pi f_0 n t_s$ and ϕ .

- 1.14 In your engineering education you'll often read in some mathematical derivation, or hear someone say, "For small α , $\sin(\alpha) = \alpha$." (In fact, you'll encounter that statement a few times in this book.) Draw two curves defined by

$$x = \alpha, \text{ and } y = \sin(\alpha)$$

over the range of $\alpha = -\pi/2$ to $\alpha = \pi/2$, and discuss why that venerable "For small α , $\sin(\alpha) = \alpha$ " statement is valid.

- 1.15 Considering two continuous (analog) sinusoids, having initial phase angles of α radians at time $t = 0$, replace the following "?" characters with the correct angle arguments:

(a) $\sin(2\pi f_0 t + \alpha) = \cos(?)$.

(b) $\cos(2\pi f_0 t + \alpha) = \sin(?)$.

- 1.16 National Instruments Corp. manufactures an A/D converter, Model #NI USB-5133, that is capable of sampling an analog signal at an f_s sample rate of 100 megasamples per second (100 MHz). The A/D converter has internal memory that can store up to 4×10^6 discrete samples. What is the maximum number of cycles of a 25 MHz analog sinewave that can be stored in the A/D converter's memory? Show your work.
- 1.17 In the first part of the text's Section 1.5 we stated that for a process (or system) to be *linear* it must satisfy a scaling property that we called the *proportionality* characteristic in the text's Eq. (1-14). Determine if the following processes have that proportionality characteristic:
- (a) $y_a(n) = x(n-1)/6$,
- (b) $y_b(n) = 3 + x(n)$,
- (c) $y_c(n) = \sin[x(n)]$.

This problem is *not* "busy work." Knowing if a process (or system) is linear tells us what signal processing principles, and algorithms, can be applied in the analysis of that process (or system).

- 1.18 There is an often-used process in DSP called *decimation*, and in that process we retain some samples of an $x(n)$ input sequence and discard other $x(n)$ samples. Decimation by a factor of two can be described algebraically by

$$y(m) = x(2n) \quad (\text{P1-3})$$

where index $m=0,1,2,3, \dots$. The decimation defined by Eq. (P1-3) means that $y(m)$ is equal to alternate samples (every other sample) of $x(n)$. For example:

$$y(0) = x(0), y(1) = x(2), y(2) = x(4), y(3) = x(6), \dots$$

and so on. Here is the question: Is that decimation process time invariant? Illustrate your answer by decimating a simple sinusoidal $x(n)$ time-domain sequence by a factor of two to obtain $y(m)$. Next, create a shifted-by-one-sample version of $x(n)$ and call it $x_{\text{shift}}(n)$. That new sequence is defined by

$$x_{\text{shift}}(n) = x(n+1). \quad (\text{P1-4})$$

Finally, decimate $x_{\text{shift}}(n)$ according to Eq. (P1-3) to obtain $y_{\text{shift}}(m)$. The decimation process is time invariant if $y_{\text{shift}}(m)$ is equal to a time-shifted version of $y(m)$. That is, decimation is time invariant if

$$y_{\text{shift}}(m) = y(m+1).$$

- 1.19 In Section 1.7 of the text we discussed the commutative property of linear time-invariant systems. The two networks in Figure P1-19 exhibit that prop-

erty. Prove this to be true by showing that, given the same $x(n)$ input sequence, outputs $y_1(n)$ and $y_2(n)$ will be equal.

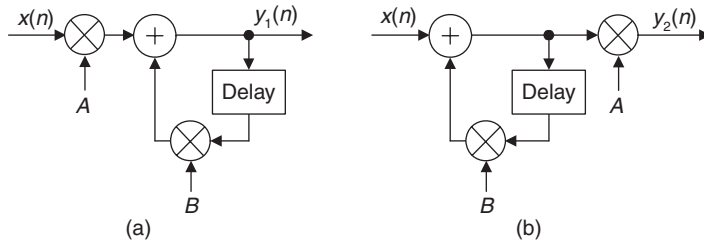


Figure P1-19

1.20 Here we investigate several simple discrete processes that turn out to be useful in a number of DSP applications. Draw the block diagrams, showing their inputs as $x(n)$, of the processes described by the following difference equations:

- (a) a 4th-order comb filter: $y_C(n) = x(n) - x(n-4)$,
- (b) an integrator: $y_I(n) = x(n) + y_I(n-1)$,
- (c) a leaky integrator: $y_{LI}(n) = Ax(n) + (1-A)y_{LI}(n-1)$ [the scalar value A is a real-valued constant in the range $0 < A < 1$],
- (d) a differentiator: $y_D(n) = 0.5x(n) - 0.5x(n-2)$.

1.21 Draw the unit impulse responses (the output sequences when the input is a unit sample impulse applied at time $n=0$) of the four processes listed in Problem 1.20. Let $A = 0.5$ for the leaky integrator. Assume that all sample values within the systems are zero at time $n = 0$.

1.22 DSP engineers involved in building control systems often need to know what is the *step response* of a discrete system. The step response, $y_{\text{step}}(n)$, can be defined in two equivalent ways. One way is to say that $y_{\text{step}}(n)$ is a system's response to an input sequence of all unity-valued samples. A second definition is that $y_{\text{step}}(n)$ is the cumulative sum (the accumulation, discrete integration) of that system's unit impulse response $y_{\text{imp}}(n)$. Algebraically, this second definition of step response is expressed as

$$y_{\text{step}}(n) = \sum_{k=-\infty}^n y_{\text{imp}}(k).$$

In words, the above $y_{\text{step}}(n)$ expression tells us: "The step response at time index n is equal to the sum of all the previous impulse response samples up to and including $y_{\text{imp}}(n)$." With that said, what are the step responses of the

four processes listed in Problem 1.20? (Let $A = 0.5$ for the leaky integrator.) Assume that all sample values within the system are zero at time $n=0$.

- 1.23 Thinking about the spectra of signals, the *ideal* continuous (analog) squarewave $s(t)$ in Figure P1–23, whose fundamental frequency is f_0 Hz, is equal to the sum of an f_0 Hz sinewave and all sinewaves whose frequencies are odd multiples of f_0 Hz. We call $s(t)$ “ideal” because we assume the amplitude transitions from plus and minus A occur instantaneously (zero seconds!). Continuous Fourier analysis of the $s(t)$ squarewave allows us to describe this sum of frequencies as the following infinite sum:

$$s(t) = \frac{4A}{\pi} \left[\sin(2\pi f_0 t) + \frac{\sin(6\pi f_0 t)}{3} + \frac{\sin(10\pi f_0 t)}{5} + \frac{\sin(14\pi f_0 t)}{7} + \dots \right].$$

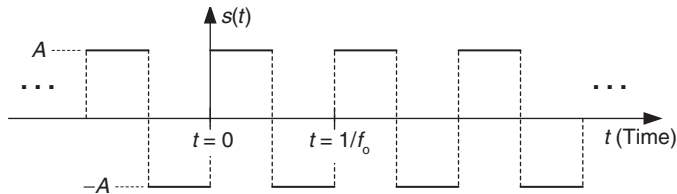


Figure P1-23

Using a summation symbol, we can express squarewave $s(t)$ algebraically as

$$s(t) = \frac{4A}{\pi} \sum_{n=1}^{\infty} \sin(2\pi n f_0 t) / n,$$

for $n = \text{odd integers only}$, showing $s(t)$ to be an infinite sum of sinusoids.

- (a) Imagine applying $s(t)$ to a filter that completely removes $s(t)$'s lowest-frequency spectral component. Draw the time-domain waveform at the output of such a filter.
 - (b) Assume $s(t)$ represents a voltage whose f_0 fundamental frequency is 1 Hz, and we wish to amplify that voltage to peak amplitudes of $\pm 2A$. Over what frequency range must an amplifier operate (that is, what must be the amplifier's *passband width*) in order to exactly double the ideal 1 Hz squarewave's peak-peak amplitude?
- 1.24 This interesting problem illustrates an *illegal* mathematical operation that we must learn to avoid in our future algebraic activities. The following claims to

be a mathematical proof that $4 = 5$. Which of the following steps is illegal? Explain why.

Proof that $4 = 5$:

Step 1: $16 - 36 = 25 - 45$

Step 2: $4^2 - 9 \cdot 4 = 5^2 - 9 \cdot 5$

Step 3: $4^2 - 9 \cdot 4 + 81/4 = 5^2 - 9 \cdot 5 + 81/4$

Step 4: $(4 - 9/2)^2 = (5 - 9/2)^2$

Step 5: $4 - 9/2 = 5 - 9/2$

Step 6: $4 = 5$

This page intentionally left blank

Index

A

- Absolute value, 9. *See also* Magnitude.
- A/D converters, quantization noise
 - clipping, 706
 - crest factor, 640
 - dithering, 706–709
 - effective bits, 641
 - fixed-point binary word length, effects
 - of, 634–642
 - oversampling, 704–706
 - reducing, 704–709
 - SNR (signal-to-noise ratio), 637–642, 711–714
 - triangular dither, 708
- A/D converters, testing techniques
 - A/D dynamic range, estimating, 714–715
 - histogram testing, 711
 - missing codes, detecting, 715–716
 - quantization noise, estimating with the FFT, 709–714
 - SFDR (spurious free dynamic range), 714–715
 - SINAD (signal-to-noise-and-distortion), 711–714
 - SNR (signal-to-noise ratio), 711–714
- Adaptive filters, 184
- Addition
 - block diagram symbol, 10
 - complex numbers, 850
- Additive white noise (AWN), 380
- AGC (automatic gain control), 783–784
- Aliasing
 - definition, 36
 - frequency-domain ambiguity, 33–38
 - in IIR filters, 304–305
- All-ones rectangular functions
 - DFT for, 115–118
 - Dirichlet kernel, 115–118, 120
- Allpass filters, definition, 893
- AM demodulation
 - filtering narrowband noise, 792–797
 - Hilbert transforms, 484–485
- Amplitude
 - definition, 8
 - loss. *See* Attenuation.
- Amplitude response, DFT
 - complex input, 73
 - real cosine input, 83–84
- Analog, definition, 2
- Analog filters
 - approximating, 302
 - vs.* digital, 169
- Analog signal processing, 2
- Analog-to-digital (A/D) converters.
 - See* A/D converters.
- Analytic signals
 - bandpass quadrature, 455
 - definition, 483
 - generation methods, comparing, 497–498
 - half-band FIR filters, 497
 - time-domain, generating, 495–497

Anti-aliasing filters, 42, 555–558
 Anti-imaging filters, 555–558
 Arctangent
 approximation, 756–758
 vector rotation. *See* Vector rotation with arctangents.
 Argand, Jean Robert, 848
 Argand diagrams of complex numbers, 848
 Argand plane, 440–441
 Attenuation
 CIC filters, improving, 557–558
 definition, 894
 Automatic gain control (AGC), 783–784
 Average, statistical measures of noise, 868–870
 Average power in electrical circuits, calculating, 874–875
 Averaging signals. *See* Signal averaging.
 AWN (additive white noise), 380

B

Band reject filters, 894
 Band-limited signals, 38
 Bandpass design, for FIR filters, 201–203
 Bandpass filters
 comb filters, 400
 definition, 895
 from half-band FIR filters, 497
 multisection complex FSFs, 398–403
 Bandpass sampling
 1st-order sampling, 46
 definition, 43
 optimum sampling frequency, 46
 positioning sampled spectra, 48
 real signals, 46
 sampling translation, 44
 SNR (signal-to-noise) ratio, 48–49
 spectral inversion, 46–47
 spectral replication, 44–45
 Bandpass signals
 in the frequency-domain, 454–455
 interpolating, 728–730
 Bandwidth, definition, 895
 Bartlett windows. *See* Triangular windows.
 Base 8 (octal) numbers, 624–625
 Base 16 (hexadecimal) numbers, 625
 Bell, Alexander Graham, 885
 Bels, definition, 885
 Bessel functions
 definition, 895
 Bessel-derived filters, ripples, 901
 Bessel's correction, 870–871
 Bias
 DC, sources and removal, 761
 in estimates, 870–871
 fixed-point binary formats, 628
 in signal variance, computing, 797–799
 Bilateral Laplace transforms, 258
 Bilinear transform method, designing IIR filters
 analytical methods, 302
 definition, 257
 example, 326–330
 frequency warping, 319, 321–325, 328–330
 mapping complex variables, 320–324
 process description, 324–326
 Bin centers, calculating absolute frequency, 139–140
 Binary points, 629
 Binary shift multiplication/division, polynomial evaluation, 773–774
 Biquad filters, 299
 Bit normalization, 653
 Bit reversals
 avoiding, 158
 fast Fourier transform input/output data index, 149–151
 Bits, definition, 623
 Blackman windows
 in FIR filter design, 195–201
 spectral leakage reduction, 686
 Blackman windows (exact), 686, 733
 Blackman-Harris windows, 686, 733
 Block averaging, SNR (signal-to-noise ratio), 770
 Block convolution. *See* Fast convolution.
 Block diagrams
 filter structure, 172–174
 quadrature sampling, 459–462
 symbols, 10–11
 uses for, 10
 Block floating point, 656–657
 Boxcar windows. *See* Rectangular windows.

- Butterfly patterns in FFTs
 - description, 145–149
 - optimized, 156
 - radix-2 structures, 151–154
 - single butterfly structures, 154–158
 - wingless, 156
- Butterworth function
 - definition, 895
 - derived filters, ripples, 901
- C**
- Cardano, Girolamo, 439
- Carrier frequency, 44
- Cartesian form, quadrature signals, 442
- Cascaded filters, 295–299, 895
- Cascaded integrators, 563
- Cascaded-comb subfilters, 412–413
- Cascade/parallel filter combinations, 295–297
- Cauer filters, 896
- Causal systems, 258
- Center frequency, definition, 895
- Central Limit Theory, 723
- Central-difference differentiators, 363–366
- CFT (continuous Fourier transform), 59, 98–102
- Chebyshev function, definition, 895
- Chebyshev windows, 197–201, 927–930
- Chebyshev-derived filters, ripples, 900
- CIC (cascaded integrator-comb) filters
 - cascaded integrators, 563
 - comb section, 553
 - compensation FIR filters, 563–566
 - definition, 895
 - implementation issues, 558–563
 - nonrecursive, 765–768
 - recursive running sum filters, 551–552
 - structures, 553–557
 - substructure sharing, 765–770
 - transposed structures, 765–770
 - two's complement overflow, 559–563
- Circular buffers, IFIR filters, 388–389
- Clipping A/D converter quantization noise, 706
- Coefficients. *See* Filter coefficients.
- Coherent sampling, 711
- Coherent signal averaging. *See* Signal averaging, coherent.
- Comb filters. *See also* Differentiators.
 - alternate FSF structures, 416–418
 - bandpass FIR filtering, 400
 - cascaded-comb subfilters, 412–413
 - with complex resonators, 392–398
 - frequency response, 903–904
 - second-order comb filters, 412–413
- Comb section. CIC filters, 553
- Commutative property, LTI, 18–19
- Commutator model, polyphase filters, 524
- Compensation FIR filters, CIC filters, 563–566
- Complex conjugate, DFT symmetry, 73
- Complex down-conversion
 - decimation, in frequency translation, 782
 - quadrature signals, 455, 456–462
- Complex exponentials, quadrature signals, 447
- Complex frequency, Laplace variable, 258
- Complex frequency response, filters, 277
- Complex mixing, quadrature signals, 455
- Complex multipliers, down-converting quadrature signals, 458
- Complex number notation, quadrature signals, 440–446
- Complex numbers. *See also* Quadrature signals.
 - Argand diagrams, 848
 - arithmetic of, 848–858
 - definition, 439
 - as a function of time, 446–450
 - graphical representation of, 847–848
 - rectangular form, definition, 848–850
 - rectangular form, *vs.* polar, 856–857
 - roots of, 853–854
 - trigonometric form, 848–850
- Complex phasors, quadrature signals, 446–450
- Complex plane, quadrature signals, 440–441, 446
- Complex resonators
 - with comb filters, 392–398
 - FSF (frequency sampling filters), 394–398
- Complex signals. *See* Quadrature signals.
- Conditional stability, Laplace transform, 268
- Conjugation, complex numbers, 851–852

- Constant-coefficient transversal FIR filters, 184
 - Continuous Fourier transform (CFT), 59, 98–102
 - Continuous lowpass filters, 41
 - Continuous signal processing
 - definition, 2
 - frequency in, 5–6
 - Continuous signals, definition, 2
 - Continuous systems, time representation, 5
 - Continuous time-domain, Laplace transform, 258–259
 - Converting analog to digital. *See* A/D converters.
 - Convolution. *See also* FIR (finite impulse response) filters, convolution.
 - fast, 716–722
 - LTI, 19
 - overlap-and-add, 720–722
 - overlap-and-save, 718–720
 - Cooley, J., 135
 - CORDIC (COordinate Rotation DIgital Computer), 756–758
 - Coupled quadrature oscillator, 787
 - Coupled-form IIR filter, 834–836
 - Crest factor, 640
 - Critical Nyquist, 37
 - Cutoff frequencies
 - definition, 896
 - designing FIR filters, 186
- D**
- Data formats
 - base systems, 624
 - definition, 623
 - place value system, 624
 - Data formats, binary numbers. *See also*
 - Fixed-point binary formats;
 - Floating-point binary formats.
 - 1.15 fixed-point, 630–632
 - block floating point, 656–657
 - converting to hexadecimal, 625
 - converting to octal, 624–625
 - definition, 623
 - dynamic range, 632–634
 - precision, 632–634
 - representing negative values, 625–626
- Data overflow. *See* Overflow.
 - dB (decibels), definition, 886, 896
 - dBm (decibels), definition, 892
 - DC
 - bias, sources of, 761
 - block-data DC removal, 762
 - defined, 62
 - from a time-domain signal, 812–815
 - DC removal, real-time
 - using filters, 761–763
 - noise shaping property, 765
 - with quantization, 763–765
 - Deadband effects, 293
 - DEC (Digital Equipment Corp.), floating-point binary formats, 654–655
 - Decibels
 - bels, definition, 885
 - common constants, 889–891
 - dB, definition, 886, 896
 - dBm, definition, 892
 - Decimation. *See also* Interpolation.
 - combining with interpolation, 521–522
 - definition, 508
 - to implement down-conversion, 676–679
 - multirate filters, 521–522
 - sample rate converters, 521–522
 - drawing downsampled spectra, 515–516
 - frequency properties, 514–515
 - magnitude loss in the frequency-domain, 515
 - overview, 508–510
 - time invariance, 514
 - time properties, 514–515
 - example, 512–513
 - overview, 510–511
 - polyphase decomposition, 514
 - Decimation filters
 - choosing, 510
 - definition, 896
 - Decimation-in-frequency algorithms, FFTs
 - radix-2 butterfly structures, 151–154, 734–735
 - Decimation-in-time algorithms, FFTs
 - index bit reversal, 149–151
 - radix-2 butterfly structures, 151–154

- single butterfly structures, 154–158, 735–737
- Demodulation
 - AM, 484–485
 - FM, 486
 - quadrature signals, 453–455, 456–462
- Descartes, René, 439
- Detection
 - envelope, 784–786
 - peak threshold, with matched filters, 377, 379–380
 - quadrature signals, 453–454
 - signal transition, 820–821
 - single tone. *See* Single tone detection.
- DFT (discrete Fourier transform). *See also*
 - DTFT (discrete-time Fourier transform); SDFT (sliding DFT).
 - analyzing FIR filters, 228–230
 - computing large DFTs from small FFTs, 826–829
 - definition, 60
 - examples, 63–73, 78–80
 - versus* FFT, 136–137
 - frequency axis, 77
 - frequency granularity, improving. *See* Zero padding.
 - frequency spacing, 77
 - frequency-domain sampling, 98–102
 - inverse, 80–81
 - linearity, 75
 - magnitudes, 75–76
 - picket fence effect, 97
 - rectangular functions, 105–112
 - resolution, 77, 98–102
 - scaloping loss, 96–97
 - shifting theorem, 77–78
 - spectral estimation, improving. *See* Zero padding.
 - time reversal, 863–865
 - zero padding, 97–102
- DFT leakage. *See also* Spectral leakage, FFTs.
 - cause, 82–84
 - definition, 81
 - description, 81–82
 - predicting, 82–84
 - sinc functions, 83, 89
 - wraparound, 86–88
- DFT leakage, minimizing
 - Chebyshev windows, 96
 - Hamming windows, 89–93
 - Hanning windows, 89–97
 - Kaiser windows, 96
 - rectangular windows, 89–97
 - triangular windows, 89–93
 - windowing, 89–97
- DFT processing gain
 - average output noise-power level, 103–104
 - inherent gain, 102–105
 - integration gain, 105
 - multiple DFTs, 105
 - output signal-power level, 103–104
 - single DFT, 102–105
 - SNR (signal-to-noise ratio), 103–104
- DIF (decimation-in-frequency), 734–735
- Difference equations
 - example, 5
 - IIR filters, 255–256
- Differentiators
 - central-difference, 363–366
 - differentiating filters, 364
 - first-difference, 363–366
 - narrowband, 366–367
 - optimized wideband, 369–370
 - overview, 361–363
 - performance improvement, 810–812
 - wideband, 367–369
- Digital differencer. *See* Differentiators.
- Digital Equipment Corp. (DEC), floating-point binary formats, 654–655
- Digital filters. *See also specific filters.*
 - vs.* analog, 169
 - definition, 896
- Digital signal processing, 2
- Direct Form I filters, 275–278, 289
- Direct Form II filters, 289–292
- Direct Form implementations, IIR filters, 292–293
- Dirichlet, Peter, 108
- Dirichlet kernel
 - all-ones rectangular functions, 115–118, 120
 - general rectangular functions, 108–112
 - symmetrical rectangular functions, 113–114

- Discrete convolution in FIR filters.
 - See also* FIR (finite impulse response) filters, convolution.
 - description, 214–215
 - in the time domain, 215–219
 - Discrete Fourier transform (DFT).
 - See* DFT (discrete Fourier transform).
 - Discrete Hilbert transforms. *See* Hilbert transforms.
 - Discrete linear systems, 12–16
 - Discrete systems
 - definition, 4
 - example, 4–5
 - time representation, 5
 - Discrete-time expression, 4
 - Discrete-time Fourier transform (DTFT), 101, 120–123
 - Discrete-time signals
 - example of, 2
 - frequency in, 5–6
 - sampling, frequency-domain
 - ambiguity, 33–38
 - use of term, 2
 - Discrete-time waveforms, describing, 8
 - Dispersion, statistical measures of noise, 869
 - DIT (decimation-in-time), 735–737
 - Dithering
 - A/D converter quantization noise, 706–709
 - with filters, 294
 - triangular, 708
 - Dolph-Chebyshev windows in FIR filter design, 197
 - Down-conversion
 - Delay/Hilbert transform filter, 817–818, 819–820
 - filtering and decimation, 676–679
 - folded FIR filters, 818
 - frequency translation, without multiplication, 676–679
 - half-band filters, 817–818
 - single-decimation technique, 819–820
 - Down-conversion, quadrature signals
 - complex, 455, 456–462
 - complex multipliers, 458
 - sampling with digital mixing, 462–464
 - Downsampling, decimation
 - drawing downsampled spectra, 515–516
 - frequency properties, 514–515
 - magnitude loss in the frequency-domain, 515
 - overview, 508–510
 - time invariance, 514
 - time properties, 514–515
 - DTFT (discrete-time Fourier transform), 101, 120–123. *See also* DFT (discrete Fourier transform).
 - Dynamic range
 - binary numbers, 632–634
 - floating-point binary formats, 656–658
 - SFDR (spurious free dynamic range), 714–715
- E**
- Elliptic functions, definition, 896
 - Elliptic-derived filters, ripples, 900
 - Envelope delay. *See* Group delay.
 - Envelope detection
 - approximate, 784–786
 - Hilbert transforms, 483–495
 - Equiripple filters, 418, 901
 - Estrin’s Method, polynomial evaluation, 774–775
 - Euler, Leonhard, 442, 444
 - Euler’s equation
 - bilinear transform design of IIR filters, 322
 - DFT equations, 60, 108
 - impulse invariance design of IIR filters, 315
 - quadrature signals, 442–443, 449, 453
 - Exact Blackman windows, 686
 - Exact interpolation, 778–781
 - Exponent, floating-point binary format, 652
 - Exponential averagers, 608–612
 - Exponential moving averages, 801–802
 - Exponential signal averaging. *See* Signal averaging, exponential.
 - Exponential variance computation, 801–802

F

- Fast convolution, 716–722
- FFT (fast Fourier transform)
 - averaging multiple, 139
 - constant-geometry algorithms, 158
 - convolution. *See* Fast convolution.
 - decimation-in-frequency algorithms, 151–154
 - decimation-in-time algorithms, 149–158
 - versus* DFT, 136–137
 - exact interpolation, 778–781
 - fast FIR filtering, 716–722
 - hints for using, 137–141
 - history of, 135
 - interpolated analytic signals,
 - computing, 781
 - interpolated real signals, interpolating, 779–780
 - interpreting results, 139–141
 - inverse, computing, 699–702, 831–833
 - in place algorithm, 157
 - radix-2 algorithm, 141–149
 - radix-2 butterfly structures, 151–158
 - signal averaging, 600–603
 - single tone detection, 737–738, 740–741
 - vs.* single tone detection, 740–741
 - software programs, 141
 - time-domain interpolation, 778–781
 - Zoom FFT, 749–753
- FFT (fast Fourier transform), real sequences
 - a $2N$ -point real FFT, 695–699
 - two N -point real FFTs, 687–694
- FFT (fast Fourier transform), twiddle factors
 - derivation of the radix-2 FFT algorithm, 143–149
 - DIF (decimation-in-frequency), 734–735
 - DIT (decimation-in-time), 735–737
- Fibonacci, 450–451
- Filter coefficients
 - definition, 897
 - for FIRs. *See* Impulse response.
 - flipping, 493–494
 - for FSF (frequency sampling filters), 913–926
 - quantization, 293–295
- Filter order, 897
- Filter taps, estimating, 234–235, 386–387
- Filters. *See also* FIR (finite impulse response) filters; IIR (infinite impulse response) filters; Matched filters; *specific filters.*
 - adaptive filters, 184
 - allpass, 893
 - analog *vs.* digital, 169
 - band reject, 894
 - bandpass, 895
 - cascaded, 895
 - Cauer, 896
 - CIC, 895
 - DC-removal, 762–763
 - decimation, 896
 - differentiating, 364. *See also* Differentiators.
 - digital, 896
 - down-conversion, 676–679
 - equiripple, 418
 - highpass, 898
 - linear phase, 899
 - lowpass, 899
 - narrowband noise, 792–797
 - nonrecursive, 226–230, 290–291, 899
 - optimal FIR, 418
 - overview, 169–170
 - parallel, 295–297
 - passband, 900
 - process description, 169–170
 - prototype, 303
 - quadrature, 900
 - real-time DC removal, 762–763
 - recursive, 290–291, 900
 - recursive running sum, 551–552
 - Remez Exchange, 418
 - sharpening, 726–728
 - structure, diagramming, 172–174
 - time-domain slope detection, 820–821
 - transposed structure, 291–292
 - transversal, 173–174. *See also* FIR (finite impulse response) filters.
 - zero-phase, 725, 902
- Filters, analytic signals
 - half-band FIR filters, 497
 - I-channel filters, 496
 - in-phase filters, 496

- Filters, analytic signals (*con't*)
 - Q-channel filters, 496
 - quadrature phase filters, 496
 - time-domain FIR filter implementation, 489–494
- Finite-word-length errors, 293–295
- FIR (finite impulse response) filters. *See also* FSF (frequency sampling filters); IFIR (interpolated FIR) filters; IIR (infinite impulse response) filters.
 - coefficients. *See* Impulse response.
 - constant coefficients, 184
 - definition, 897
 - fast FIR filtering using the FFT, 716–722
 - folded structure. *See* Folded FIR filters.
 - frequency magnitude response, determining, 179
 - frequency-domain response, determining, 179
 - group delay, 211–212
 - half-band. *See* Half-band FIR filters.
 - vs.* IIR filters, 332–333
 - impulse response, 177–179
 - narrowband lowpass. *See* IFIR (interpolated FIR) filters.
 - nonrecursive, analyzing, 226–230
 - phase response in, 209–214
 - phase unwrapping, 210
 - phase wrapping, 209, 900
 - polyphase filters, 522–527
 - sharpening, 726–728
 - signal averaging. *See* Signal averaging, with FIR filters.
 - signal averaging with, 178, 180–184
 - stopband attenuation, improving, 726–728
 - tapped delay, 181–182
 - transient response, 181–182
 - z-transform of, 288–289
- FIR (finite impulse response) filters, analyzing
 - with DFTs, 228–230
 - estimating number of, 234–235
 - fractional delay, 233
 - group delay, 230–233
 - passband gain, 233–234
 - stopband attenuation, 234–235
 - symmetrical-coefficient FIR filters, 232–233
- FIR (finite impulse response) filters, convolution
 - description, 175–186
 - discrete, description, 214–215
 - discrete, in the time domain, 215–219
 - fast convolution, 716–722
 - impulse response, 177–178
 - inputs, time order reversal, 176
 - signal averaging, 175–176
 - theorem, applying, 222–226
 - theorem, description, 219–222
 - time-domain aliasing, avoiding, 718–722
 - time-domain convolution *vs.* frequency-domain multiplication, 191–194
- FIR (finite impulse response) filters, designing
 - bandpass method, 201–203
 - cutoff frequencies, 186
 - with forward FFT software routines, 189
 - Fourier series design method. *See* Window design method, FIR filters.
 - Gibbs's phenomenon, 193
 - highpass method, 203–204
 - low-pass design, 186–201
 - magnitude fluctuations, 190–194
 - Optimal design method, 204–207
 - Parks-McClellan Exchange method, 204–207
 - passband ripples, minimizing, 190–194, 204–207. *See also* Windows.
 - Remez method, 204–207
 - stopband ripples, minimizing, 204–207
 - time-domain coefficients, determining, 186–194
 - time-domain convolution *vs.* frequency-domain multiplication, 191–194
 - very high performance filters, 775–778
 - window design method, 186–194
 - windows used in, 194–201
- 1st-order IIR filters, signal averaging, 612–614
- 1st-order sampling, 46
- First-difference differentiators, 363–366
- Fixed-point binary formats. *See also* Floating-point binary formats.

- 1.15 format, 630–632
- bias, 628
- binary points, 629
- decimal numbers, converting to 1.5
 - binary, 632
- fractional binary numbers, 629–632
- hexadecimal (base 16) numbers, 625
- integer plus fraction, 629
- lsb (least significant bit), 624
- msb (most significant bit), 624
- octal (base 8) numbers, 624–625
- offset, 627–628
- overflow, 629
- Q30 format, 629
- radix points, 629
- representing negative values, 625–626
- sign extend operations, 627
- sign-magnitude, 625–626
- two's complement, 626–627, 629
- Fixed-point binary formats, finite word lengths
 - A/D converter best estimate values, 635
 - A/D converter quantization noise, 634–642
 - A/D converter *vs.* SNR, 640–642
 - convergent rounding, 651
 - crest factor, 640
 - data overflow, 642–646
 - data rounding, 649–652
 - effective bits, 641
 - round off noise, 636–637
 - round to even method, 651
 - round-to-nearest method, 650–651
 - truncation, 646–649
- Floating-point binary formats. *See also* Fixed-point binary formats.
 - bit normalization, 653
 - common formats, 654–655
 - DEC (Digital Equipment Corp.), 654–655
 - description, 652
 - dynamic range, 656–658
 - evaluating, 652
 - exponent, 652
 - fractions, 653
 - gradual underflow, 656
 - hidden bits, 653
 - IBM, 654–655
 - IEEE Standard P754, 654–655
 - mantissa, 652
 - MIL-STD 1750A, 654–655
 - min/max values, determining, 656–657
 - unnormalized fractions, 656
 - word lengths, 655
- FM demodulation
 - algorithms for, 758–761
 - filtering narrowband noise, 792–797
 - Hilbert transforms, 486
- Folded FIR filters
 - designing Hilbert transforms, 493
 - down-conversion, 818
 - frequency translation, without multiplication, 678
 - half-band filters, sample rate conversion, 548
 - Hilbert transforms, designing, 493
 - multipliers, reducing, 702–704
 - nonrecursive, 419–420
 - tapped-delay line, 389
- Folding frequencies, 40
- Forward FFT
 - computing, 831–833
 - software routines for designing FIR filters, 189
- Fourier series design FIR filters. *See* Window design method, FIR filters.
- Fourier transform pairs, FIR filters, 178–179
- Fractional binary numbers, 629–632
- Fractional delay, FIR filters, 233
- Frequency
 - continuous *vs.* discrete systems, 5
 - of discrete signals, determining. *See* DFT (discrete Fourier transform).
 - discrete-time signals, 5–6
 - properties, interpolation, 519
 - resolution, improving with FIR filters, 228–230
 - units of measure, 2–3
- Frequency attenuation, FIR filters, 182
- Frequency axis
 - definition, 77
 - DFT, 77
 - in Hz, 118
 - normalized angle variable, 118
 - in radians/seconds, 118–119
 - rectangular functions, 118–120
 - with zero padding, 100

- Frequency domain
 - definition, 6
 - Hamming windows, 683–686
 - Hanning windows, 683–686
 - listing sequences, 7
 - performance. IIR filters, 282–289
 - quadrature signals, 451–454
 - spectral leak reduction, 683–686
 - windowing in, 683–686
 - windows, 683–686
 - Frequency magnitude response
 - definition, 897
 - determining with FIR filters, 179
 - Frequency response
 - LTI, determining, 19
 - for Mth-order IIR filter, 275–276
 - Frequency response, FIR filters
 - determining, 179–186
 - factors affecting, 174
 - modifying, 184–186
 - Frequency sampling design method *vs.* FSF, 393–394
 - Frequency sampling filters. *See* FSF (frequency sampling filters).
 - Frequency translation, bandpass sampling, 44
 - Frequency translation, with decimation
 - complex down-conversion, 782
 - complex signals, 781–783
 - real signals, 781
 - Frequency translation, without multiplication
 - by 1/2 the sampling rate, 671–673
 - by 1/4 the sampling rate, 674–676
 - down-conversion, 676–679
 - inverting the output spectrum, 678–679
 - Frequency translation to baseband, quadrature signals, 455
 - Frequency warping, 319, 321–325, 328–330
 - FSF (frequency sampling filters). *See also* FIR (finite impulse response) filters.
 - complex resonators, 394–398
 - designing, 423–426
 - frequency response, single complex FSF, 904–905
 - history of, 392–394
 - linear-phase multisection real-valued, 409–410
 - modeling, 413–414
 - multisection complex, 398–403
 - multisection real-valued, 406–409
 - vs.* Parks-McClellan filters, 392
 - real FSF transfer function, 908–909
 - stability, 403–406
 - stopband attenuation, increasing, 414–416
 - stopband sidelobe level suppression, 416
 - transition band coefficients, 414–416
 - Type IV example, 419–420, 423–426
- G**
- Gain. *See also* DFT processing gain.
 - AGC (automatic gain control), 783–784
 - IIR filters, scaling, 300–302
 - integration, signal averaging, 600–603
 - passband, 233–234
 - windows, 92
 - Gauss, Karl, 439, 444
 - Gaussian PDFs, 882–883
 - General numbers, 446. *See also* Complex numbers.
 - Geometric series, closed form, 107, 859–861
 - Gibbs’s phenomenon, 193
 - Goertzel algorithm, single tone detection
 - advantages of, 739
 - description, 738–740
 - example, 740
 - vs.* the FFT, 740–741
 - stability, 838–840
 - Gold-Rader filter, 834–836
 - Gradual underflow, floating-point binary formats, 656
 - Gregory, James, 23
 - Group delay
 - definition, 897–898
 - differentiators, 365
 - filters, computing, 830–831
 - FIR filters, 211–212, 230–233
- H**
- Half Nyquist, 37
 - Half-band FIR filters
 - analytic signals, 497

- as complex bandpass filters, 497
 - definition, 898
 - description, 207–209
 - down-conversion, 817–818
 - frequency translation, 802–804
 - Half-band FIR filters, sample rate conversion
 - conversion
 - fundamentals, 544–546
 - implementation, 546–548
 - overview, 543
 - Hamming, Richard, 366
 - Hamming windows
 - in the frequency domain, 683–686
 - spectral peak location, 733
 - Hann windows. *See* Hanning windows.
 - Hanning windows
 - description, 89–97
 - DFT leakage, minimizing, 89–97
 - in the frequency domain, 683–686
 - spectral peak location, 733
 - Harmonic sampling. *See* Bandpass sampling.
 - Harmonics of discrete signals, determining. *See* DFT (discrete Fourier transform).
 - Harris, Fred, 791
 - Heaviside, Oliver, 257
 - Hertz, 3
 - Hertz, Heinrich, 3
 - Hexadecimal (base 16) numbers, 625
 - Hidden bits, floating-point binary formats, 653
 - Highpass filters, definition, 898
 - Highpass method, designing FIR filters, 203–204
 - Hilbert, David, 479
 - Hilbert transformers, designing
 - common mistake, 493–494
 - even-tap transformers, 493
 - frequency-domain transformers, 494–495
 - half-band filter coefficient modification, 804–805
 - half-band filter frequency translation, 802–804
 - odd-tap transformers, 493
 - time-domain FIR filter implementation, 489–494
 - time-domain transformers, 489–494
 - Hilbert transforms
 - AM demodulation, 484–485
 - definition, 480
 - envelope detection, 483–495
 - example, 481–482
 - FM demodulation, 486
 - impulse response, 487–489
 - one-sided spectrum, 483
 - signal envelope, 483–495
 - Hilbert transforms, analytic signals
 - definition, 483
 - generation methods, comparing, 497–498
 - half-band FIR filters, 497
 - time-domain, generating, 495–497
 - Histogram testing, A/D converter techniques, 711
 - Homogeneity property, 12
 - Horner, William, 773
 - Horner's Rule, 772–774
 - Human ear, sensitivity to decibels, 886
- I**
- IBM, floating-point binary formats, 654–655
 - I-channel filters, analytic signals, 496
 - IDFT (inverse discrete Fourier transform), 80–81
 - IEEE Standard P754, floating-point binary formats, 654–655
 - IF sampling. *See* Bandpass sampling.
 - IFIR (interpolated FIR) filters. *See also* FIR (finite impulse response) filters.
 - computational advantage, 384–385, 391
 - definition, 381
 - expansion factor M , 381, 385–386
 - filter taps, estimating, 386–387
 - image-reject subfilter, 382–384, 390
 - implementation issues, 388–389
 - interpolated, definition, 384
 - interpolators. *See* Image-reject subfilter.
 - lowpass design example, 389–391
 - optimum expansion factor, 386
 - performance modeling, 387–388
 - prototype filters, 382
 - shaping subfilters, 382, 385

- IIR (infinite impulse response) filters. *See also* FIR (finite impulse response) filters; FSF (frequency sampling filters).
- allpass, 893
 - analytical design methods, 302
 - coupled-form, 834–836
 - definition, 899
 - design techniques, 257. *See also specific techniques.*
 - difference equations, 255–256
 - vs.* FIR filters, 253, 332–333
 - frequency domain performance, 282–289
 - infinite impulse response, definition, 280
 - interpolated, example, 837–838
 - phase equalizers. *See* Allpass filters.
 - poles, 284–289
 - recursive filters, 290–291
 - scaling the gain, 300–302
 - SNR (signal-to-noise ratio), 302
 - stability, 263–270
 - z-domain transfer function, 282–289
 - zeros, 284–289
 - z-plane pole / zero properties, 288–289
 - z-transform, 270–282
- IIR (infinite impulse response) filters, pitfalls in building
- coefficient quantization, 293–295
 - deadband effects, 293
 - Direct Form implementations, 292–293
 - dither sequences, 294
 - finite word length errors, 293–295
 - limit cycles, 293
 - limited-precision coefficients, 293
 - overflow, 293–295
 - overflow oscillations, 293
 - overview, 292–293
 - rounding off, 293
- IIR (infinite impulse response) filters, structures
- biquad filters, 299
 - cascade filter properties, 295–297
 - cascaded, 295–299
 - cascade/parallel combinations, 295–297
 - changing, 291–292
 - Direct Form 1, 275–278, 289
 - Direct Form II, 289–292
 - optimizing partitioning, 297–299
 - parallel filter properties, 295–297
 - transposed, 291–292
 - transposed Direct Form II, 289–290
 - transposition theorem, 291–292
- Imaginary numbers, 439, 446
- Imaginary part, quadrature signals, 440, 454–455
- Impulse invariance method, designing IIR filters
- aliasing, 304–305
 - analytical methods, 302
 - definition, 257
 - Method 1, description, 305–307
 - Method 1, example, 310–313
 - Method 2, description, 307–310
 - Method 2, example, 313–319
 - preferred method, 317
 - process description, 303–310
 - prototype filters, 303
- Impulse response
- convolution in FIR filters, 177–178
 - definition, 898–899
 - FIR filters, 177–179
 - Hilbert transforms, 487–489
- Incoherent signal averaging. *See* Signal averaging, incoherent.
- Infinite impulse response (IIR) filters. *See* IIR (infinite impulse response) filters.
- Integer plus fraction fixed-point binary formats, 629
- Integration gain, signal averaging, 600–603
- Integrators
- CIC filters, 553
 - overview, 370
 - performance comparison, 373–376
 - rectangular rule, 371–372
 - Simpson's rule, 372, 373–376
 - Tick's rule, 373–376
 - trapezoidal rule, 372
- Intermodulation distortion, 16
- Interpolated analytic signals, computing, 781
- Interpolated FIR (IFIR) filters. *See* IFIR (interpolated FIR) filters.
- Interpolated real signals, interpolating, 779–780
- Interpolation. *See also* Decimation.
- accuracy, 519

- bandpass signals, 728–730
 - combining with decimation, 521–522
 - definition, 384, 508
 - drawing upsampled spectra, 520–521
 - exact, 778–781
 - frequency properties, 519
 - history of, 519
 - linear, 815–816
 - multirate filters, 521–522
 - overview, 516–518
 - sample rate converters, 521–522
 - time properties, 519
 - time-domain, 778–781
 - unwanted spectral images, 519
 - upsampling, 517–518, 520–521
 - zero stuffing, 518
 - Interpolation filters, 518
 - Inverse DFT, 80–81
 - Inverse discrete Fourier transform (IDFT), 80–81
 - Inverse FFT, 699–702, 831–833
 - Inverse of complex numbers, 853
 - Inverse sinc filters, 563–566
 - I/Q demodulation, quadrature signals, 459–462
- J**
- Jacobsen, Eric, 775
 - j -operator, quadrature signals, 439, 444–450
- K**
- Kaiser, James, 270
 - Kaiser windows, in FIR filter design, 197–201
 - Kaiser-Bessel windows, in FIR filter design, 197
 - Kelvin, Lord, 60
 - Kootsookos, Peter, 603, 724
 - Kotelnik, V., 42
- L**
- Lanczos differentiators, 366–367
 - Laplace transfer function
 - conditional stability, 268
 - description, 262–263
 - determining system stability, 263–264, 268
 - impulse invariance design, Method 1, 305–307, 310–313
 - impulse invariance design, Method 2, 307–310, 313–319
 - in parallel filters, 295–297
 - second order, 265–268
 - Laplace transform. *See also* Z-transform.
 - bilateral transform, 258
 - causal systems, 258
 - conditional stability, 268
 - for continuous time-domain, 258–259
 - description, 257–263
 - development of, 257
 - one-sided transform, 258
 - one-sided/causal, 258
 - poles on the s -plane, 263–270
 - stability, 263–270
 - two-sided transform, 258
 - zeros on the s -plane, 263–270
 - Laplace variable, complex frequency, 258
 - Leakage. *See* DFT leakage.
 - Leaky integrator, 614
 - Least significant bit (lsb), 624
 - L'Hopital's Rule, 110
 - Limit cycles, 293
 - Linear, definition, 12
 - Linear differential equations, solving.
 - See* Laplace transform.
 - Linear interpolation, 815–816
 - Linear phase filters, 899
 - Linear systems, example, 13–14
 - Linear time-invariant (LTI) systems. *See* LTI (linear time-invariant) systems.
 - Linearity, DFT, 75
 - Linear-phase filters
 - DC removal, 812–815
 - definition, 899
 - Logarithms
 - and complex numbers, 854–856
 - measuring signal power, 191
 - Lowpass design
 - designing FIR filters, 186–201
 - IFIR filters, example, 389–391
 - Lowpass filters, definition, 899
 - Lowpass signals
 - definition, 38
 - sampling, 38–42

- lsb (least significant bit), 624
- LTI (linear time-invariant) systems
 - analyzing, 19–21
 - commutative property, 18–19
 - convolution, 19
 - DFT (discrete Fourier transform), 19
 - discrete linear systems, 12–16
 - frequency response, determining, 19
 - homogeneity property, 12
 - intermodulation distortion, 16
 - internally generated sinusoids, 16
 - linear, definition, 12
 - linear system, example, 13–14
 - nonlinear system, example, 14–16
 - output sequence, determining, 19
 - overview, 12
 - proportionality characteristic, 12
 - rearranging sequential order, 18–19
 - time-invariant systems, 17–18
 - unit impulse response, 19–20
- M**
- MAC (multiply and accumulate)
 - architecture
 - polynomial evaluation, 773
 - programmable DSP chips, 333
- Magnitude
 - approximation (vector), 679–683
 - of complex numbers, 848
 - definition, 8–9
 - DFT, 75–76
- Magnitude and angle form of complex numbers, 848–850
- Magnitude response of DFTs
 - aliased sinc function, 108
 - all-ones rectangular functions, 115–118
 - fluctuations. *See* Scalloping.
 - general rectangular functions, 106–112
 - overview, 105–106
 - sidelobe magnitudes, 110–111
 - symmetrical rectangular functions, 112–115
- Magnitude response of DFTs, Dirichlet kernel
 - all-ones rectangular functions, 115–118, 120
 - general rectangular functions, 108–112
 - symmetrical rectangular functions, 113–114
- Magnitude-angle form, quadrature signals, 442
- Mantissa, floating-point binary formats, 652
- Matched filters
 - definition, 376
 - example, 378–380
 - implementation considerations, 380
 - peak detection threshold, 377, 379–380
 - properties, 376–378
 - purpose, 376
 - SNR (signal-power-to-noise-power ratio), maximizing, 376
- McClellan, James, 206. *See also* Parks-McClellan algorithm.
- Mean (statistical measure of noise)
 - definition, 868–869
 - PDF (probability density function), 879–882
 - of random functions, 879–882
- Mean (statistical average), of random functions, 879–882
- Mehrnia, A., 386
- MIL-STD 1750A, floating-point binary formats, 654–655
- Missing
 - A/D conversion codes, checking, 715–716
 - sample data, recovering, 823–826. *See also* Interpolation.
- Mixing. *See* Frequency translation.
- Modeling FSF (frequency sampling filters), 413–414
- Modulation, quadrature signals, 453–454
- Modulus of complex numbers, 848
- Most significant bit (msb), 624
- Moving averages
 - CIC filters, 551–552
 - as digital lowpass filters, 20–21, 173, 231
 - sample rate conversion, CIC filters, 551–552
- Moving averages, coherent signal averaging
 - exponential moving averages, computing, 801–802
 - exponential signal averaging, 801–802
 - moving averages, computing, 799–801
 - nonrecursive moving averagers, 606–608

- recursive moving averagers, 606–608
- time-domain averaging, 604–608
- msb (most significant bit), 624
- Multiplication
 - block diagram symbol, 10
 - CIC filters, simplified, 765–770
 - complex numbers, 850–851
- Multirate filters
 - decimation, 521–522
 - interpolation, 521–522
- Multirate systems, sample rate conversion
 - filter mathematical notation, 534–535
 - signal mathematical notation, 533–534
 - z-transform analysis, 533–535
- Multirate systems, two-stage decimation, 511

N

- Narrowband differentiators, 366–367
- Narrowband noise filters, 792–797
- Natural logarithms of complex numbers, 854
- Negative frequency, quadrature signals, 450–451
- Negative values in binary numbers, 625–626
- Newton, Isaac, 773
- Newton's method, 372
- Noble identities, polyphase filters, 536
- Noise
 - definition, 589
 - measuring. *See* Statistical measures of noise.
 - random, 868
- Noise shaping property, 765
- Nonlinear systems, example, 14–16
- Nonrecursive CIC filters
 - description, 765–768
 - prime-factor-R technique, 768–770
- Nonrecursive filters. *See* FIR filters
- Nonrecursive moving averagers, 606–608
- Normal distribution of random data, generating, 722–724
- Normal PDFs, 882–883
- Normalized angle variable, 118–119
- Notch filters. *See* Band reject filters.
- Nyquist, H., 42
- Nyquist criterion, sampling lowpass signals, 40

O

- Octal (base 8) numbers, 624–625
- Offset fixed-point binary formats, 627–628
- 1.15 fixed-point binary format, 630–632
- Optimal design method, designing FIR filters, 204–207
- Optimal FIR filters, 418
- Optimization method, designing IIR filters
 - definition, 257
 - description, 302
 - iterative optimization, 330
 - process description, 330–332
- Optimized butterflies, 156
- Optimized wideband differentiators, 369–370
- Optimum sampling frequency, 46
- Order
 - of filters, 897
 - polyphase filters, swapping, 536–537
- Orthogonality, quadrature signals, 448
- Oscillation, quadrature signals, 459–462
- Oscillator, quadrature
 - coupled, 787
 - overview, 786–789
 - Taylor series approximation, 788
- Overflow
 - computing the magnitude of complex numbers, 815
 - fixed-point binary formats, 629, 642–646
 - two's complement, 559–563
- Overflow errors, 293–295
- Overflow oscillations, 293
- Oversampling A/D converter
 - quantization noise, 704–706

P

- Parallel filters, Laplace transfer function, 295–297
- Parks-McClellan algorithm
 - designing FIR filters, 204–207
 - vs.* FSF (frequency sampling filters), 392
 - optimized wideband differentiators, 369–370
- Parzen windows. *See* Triangular windows.
- Passband, definition, 900

- Passband filters, definition, 900
- Passband gain, FIR filters, 233–234
- Passband ripples
 - cascaded filters, estimating, 296–297
 - definition, 296, 900
 - IFIR filters, 390
 - minimizing, 190–194, 204–207
- PDF (probability density function)
 - Gaussian, 882–883
 - mean, calculating, 879–882
 - mean and variance of random functions, 879–882
 - normal, 882–883
 - variance, calculating, 879–882
- Peak correlation, matched filters, 379
- Peak detection threshold, matched filters, 377, 379–380
- Periodic sampling
 - aliasing, 33–38
 - frequency-domain ambiguity, 33–38
- Periodic sampling
 - 1st-order sampling, 46
 - anti-aliasing filters, 42
 - bandpass, 43–49
 - coherent sampling, 711
 - definition, 43
 - folding frequencies, 40
 - Nyquist criterion, 40
 - optimum sampling frequency, 46
 - real signals, 46
 - sampling translation, 44
 - SNR (signal-to-noise) ratio, 48–49
 - spectral inversion, 46–47
 - undersampling, 40
- Phase angles, signal averaging, 603–604
- Phase delay. *See* Phase response.
- Phase response
 - definition, 900
 - in FIR filters, 209–214
- Phase unwrapping, FIR filters, 210
- Phase wrapping, FIR filters, 209, 900
- Pi, calculating, 23
- Picket fence effect, 97
- Pisa, Leonardo da, 450–451
- Polar form
 - complex numbers, *vs.* rectangular, 856–857
 - quadrature signals, 442, 443–444
- Poles
 - IIR filters, 284–289
 - on the s-plane, Laplace transform, 263–270
- Polynomial curve fitting, 372
- Polynomial evaluation
 - binary shift multiplication/division, 773–774
 - Estrin’s Method, 774–775
 - Horner’s Rule, 772–774
 - MAC (multiply and accumulate) architecture, 773
- Polynomial factoring, CIC filters, 765–770
- Polynomials, finding the roots of, 372
- Polyphase decomposition
 - CIC filters, 765–770
 - definition, 526
 - diagrams, 538–539
 - two-stage decimation, 514
- Polyphase filters
 - benefits of, 539
 - commutator model, 524
 - implementing, 535–540
 - issues with, 526
 - noble identities, 536
 - order, swapping, 536–537
 - overview, 522–528
 - polyphase decomposition, 526, 538–539
 - prototype FIR filters, 522
 - uses for, 522
- Power, signal. *See also* Decibels.
 - absolute, 891–892
 - definition, 9
 - relative, 885–889
- Power spectrum, 63, 140–141
- Preconditioning FIR filters, 563–566
- Prewarp, 329
- Prime decomposition, CIC filters, 768–770
- Prime factorization, CIC filters, 768–770
- Probability density function (PDF). *See* PDF (probability density function).
- Processing gain or loss. *See* DFT processing gain; Gain; Loss.
- Prototype filters
 - analog, 303
 - FIR polyphase filters, 522
 - IFIR filters, 382

Q

Q30 fixed-point binary formats, 629
 Q-channel filters, analytic signals, 496
 Quadratic factorization formula, 266, 282
 Quadrature component, 454–455
 Quadrature demodulation, 455, 456–462
 Quadrature filters, definition, 900
 Quadrature mixing, 455
 Quadrature oscillation, 459–462
 Quadrature oscillator
 coupled, 787
 overview, 786–789
 Taylor series approximation, 788
 Quadrature phase, 440
 Quadrature processing, 440
 Quadrature sampling block diagram, 459–462
 Quadrature signals. *See also* Complex numbers.
 analytic, 455
 Argand plane, 440–441
 bandpass signals in the frequency-domain, 454–455
 Cartesian form, 442
 complex exponentials, 447
 complex mixing, 455
 complex number notation, 440–446
 complex phasors, 446–450
 complex plane, 440–441, 446
 decimation, in frequency translation, 781–783
 definition, 439
 demodulation, 453–454
 detection, 453–454
 down-conversion. *See* Down-conversion, quadrature signals.
 Euler's identity, 442–443, 449, 453
 exponential form, 442
 in the frequency domain, 451–454
 generating from real signals. *See* Hilbert transforms.
 generation, 453–454
 imaginary part, 440, 454–455
 in-phase component, 440, 454–455
 I/Q demodulation, 459–462
 j-operator, 439, 444–450
 magnitude-angle form, 442

 mixing to baseband, 455
 modulation, 453–454
 negative frequency, 450–451
 orthogonality, 448
 polar form, 442, 443–444
 positive frequency, 451
 real axis, 440
 real part, 440, 454–455
 rectangular form, 442
 representing real signals, 446–450
 sampling scheme, advantages of, 459–462
 simplifying mathematical analysis, 443–444
 three-dimensional frequency-domain representation, 451–454
 trigonometric form, 442, 444
 uses for, 439–440

Quantization

 coefficient/errors, 293–295
 noise. *See* A/D converters, quantization noise.
 real-time DC removal, 763–765

R

Radix points, fixed-point binary formats, 629
 Radix-2 algorithm, FFT
 butterfly structures, 151–154
 computing large DFTs, 826–829
 decimation-in-frequency algorithms, 151–154
 decimation-in-time algorithms, 151–154
 derivation of, 141–149
 FFT (fast Fourier transform), 151–158
 twiddle factors, 143–149
 Raised cosine windows. *See* Hanning windows.
 Random data
 Central Limit Theory, 723
 generating a normal distribution of, 722–724
 Random functions, mean and variance, 879–882
 Random noise, 868. *See also* SNR (signal-to-noise ratio).

- Real numbers
 - definition, 440
 - graphical representation of, 847–848
 - Real sampling, 46
 - Real signals
 - bandpass sampling, 46
 - decimation, in frequency translation, 781
 - generating complex signals from. *See* Hilbert transforms.
 - representing with quadrature signals, 446–450
 - Rectangular form of complex numbers
 - definition, 848–850
 - vs.* polar form, 856–857
 - Rectangular form of quadrature signals, 442
 - Rectangular functions
 - all ones, 115–118
 - DFT, 105–112
 - frequency axis, 118–120
 - general, 106–112
 - overview, 105–106
 - symmetrical, 112–115
 - time axis, 118–120
 - Rectangular windows, 89–97, 686
 - Recursive filters. *See* IIR filters
 - Recursive moving averagers, 606–608
 - Recursive running sum filters, 551–552
 - Remez Exchange, 204–207, 418
 - Replications, spectral. *See* Spectral replications.
 - Resolution, DFT, 77, 98–102
 - Ripples
 - in Bessel-derived filters, 901
 - in Butterworth-derived filters, 901
 - in Chebyshev-derived filters, 900
 - definition, 900–901
 - designing FIR filters, 190–194
 - in Elliptic-derived filters, 900
 - equiripple, 418, 901
 - out-of-band, 901
 - in the passband, 900
 - in the stopband, 901
 - rms value of continuous sinewaves, 874–875
 - Roll-off, definition, 901
 - Roots of
 - complex numbers, 853–854
 - polynomials, 372
 - Rosetta Stone, 450
 - Rounding fixed-point binary numbers
 - convergent rounding, 651
 - data rounding, 649–652
 - effective bits, 641
 - round off noise, 636–637
 - round to even method, 651
 - round-to-nearest method, 650–651
 - Roundoff errors, 293
- S**
- Sample rate conversion. *See also* Polyphase filters.
 - decreasing. *See* Decimation.
 - definition, 507
 - with IFIR filters, 548–550
 - increasing. *See* Interpolation.
 - missing data, recovering, 823–826. *See also* Interpolation.
 - by rational factors, 540–543
 - Sample rate conversion, multirate systems
 - filter mathematical notation, 534–535
 - signal mathematical notation, 533–534
 - z-transform analysis, 533–535
 - Sample rate conversion, with half-band filters
 - folded FIR filters, 548
 - fundamentals, 544–546
 - implementation, 546–548
 - overview, 543
 - Sample rate converters, 521–522
 - Sampling, periodic. *See* Periodic sampling.
 - Sampling translation, 44
 - Sampling with digital mixing, 462–464
 - Scaling IIR filter gain, 300–302
 - Scalloping loss, 96–97
 - SDFT (sliding DFT)
 - algorithm, 742–746
 - overview, 741
 - stability, 746–747
 - SFDR (spurious free dynamic range), 714–715
 - Shannon, Claude, 42
 - Shape factor, 901
 - Sharpened FIR filters, 726–728
 - Shifting theorem, DFT, 77–78

- Shift-invariant systems. *See* Time-invariant systems.
- Sidelobe magnitudes, 110–111
- Sidelobes
 - Blackman window and, 194–197
 - DFT leakage, 83, 89
 - FIR (finite impulse response) filters, 184
 - ripples, in low-pass FIR filters, 193–194
- Sign extend operations, 627
- Signal averaging. *See also* SNR (signal-to-noise ratio).
 - equation, 589
 - frequency-domain. *See* Signal averaging, incoherent.
 - integration gain, 600–603
 - mathematics, 592–594, 599
 - multiple FFTs, 600–603
 - phase angles, 603–604
 - postdetection. *See* Signal averaging, incoherent.
 - quantifying noise reduction, 594–597
 - rms. *See* Signal averaging, incoherent.
 - scalar. *See* Signal averaging, incoherent.
 - standard deviation, 590
 - time-domain. *See* Signal averaging, coherent.
 - time-synchronous. *See* Signal averaging, coherent.
 - variance, 589–590
 - video. *See* Signal averaging, incoherent.
- Signal averaging, coherent
 - exponential averagers, 608–612
 - exponential moving averages, computing, 801–802
 - exponential smoothing, 608
 - filtering aspects, 604–608
 - moving averagers, 604–608
 - moving averages, computing, 799–801
 - nonrecursive moving averagers, 606–608
 - overview, 590–597
 - recursive moving averagers, 606–608
 - reducing measurement uncertainty, 593, 604–608
 - time-domain filters, 609–612
 - true signal level, 604–608
 - weighting factors, 608, 789
- Signal averaging, exponential
 - 1st-order IIR filters, 612–614
 - dual-mode technique, 791
 - example, 614
 - exponential smoothing, 608
 - frequency-domain filters, 612–614
 - moving average, computing, 801–802
 - multiplier-free technique, 790–791
 - overview, 608
 - single-multiply technique, 789–790
- Signal averaging, incoherent
 - 1st-order IIR filters, 612–614
 - example, 614
 - frequency-domain filters, 612–614
 - overview, 597–599
- Signal averaging, with FIR filters
 - convolution, 175–176
 - example, 170–174, 183–184
 - as a lowpass filter, 180–182
 - performance improvement, 178
- Signal envelope, Hilbert transforms, 483–495
- Signal power. *See also* Decibels.
 - absolute, 891–892
 - relative, 885–889
- Signal processing
 - analog, 2. *See also* Continuous signals.
 - definition, 2
 - digital, 2
 - operational symbols, 10–11
- Signal transition detection, 820–821
- Signal variance
 - biased and unbiased, computing, 797–799, 799–801
 - definition, 868–870
 - exponential, computing, 801–802
 - PDF (probability density function), 879–882
 - of random functions, 879–882
 - signal averaging, 589–590
- Signal-power-to-noise-power ratio (SNR), maximizing, 376
- Signal-to-noise ratio (SNR). *See* SNR (signal-to-noise ratio).
- Sign-magnitude, fixed-point binary formats, 625–626
- Simpson, Thomas, 372
- SINAD (signal-to-noise-and-distortion), 711–714
- Sinc filters. *See* CIC (cascaded integrator-comb) filters.

- Sinc functions, 83, 89, 116
- Single tone detection, FFT method
 - drawbacks, 737–738
 - vs.* Goertzel algorithm, 740–741
- Single tone detection, Goertzel algorithm
 - advantages of, 739
 - description, 738–740
 - example, 740
 - vs.* the FFT, 740–741
 - stability, 838–840
- Single tone detection, spectrum analysis, 737–741
- Single-decimation down-conversion, 819–820
- Single-multiply technique, exponential signal averaging, 789–790
- Single-stage decimation, *vs.* two-stage, 514
- Single-stage interpolation, *vs.* two-stage, 532
- Sliding DFT (SDFT). *See* SDFT (sliding DFT).
- Slope detection, 820–821
- Smoothing impulsive noise, 770–772
- SNDR. *See* SINAD (signal-to-noise-and-distortion).
- SNR (signal-to-noise ratio)
 - vs.* A/D converter, fixed-point binary finite word lengths, 640–642
 - A/D converters, 711–714
 - bandpass sampling, 48–49
 - block averaging, 770
 - corrected mean, 771
 - DFT processing gain, 103–104
 - IIR filters, 302
 - measuring. *See* Statistical measures of noise.
 - reducing. *See* Signal averaging.
 - smoothing impulsive noise, 770–772
- SNR (signal-power-to-noise-power ratio), maximizing, 376
- Software programs, fast Fourier transform, 141
- Someya, I., 42
- Spectral inversion
 - around signal center frequency, 821–823
 - bandpass sampling, 46–47
- Spectral leakage, FFTs, 138–139, 683–686. *See also* DFT leakage.
- Spectral leakage reduction
 - A/D converter testing techniques, 710–711
 - Blackman windows, 686
 - frequency domain, 683–686
- Spectral peak location
 - estimating, algorithm for, 730–734
 - Hamming windows, 733
 - Hanning windows, 733
- Spectral replications
 - bandpass sampling, 44–45
 - sampling lowpass signals, 39–40
- Spectral vernier. *See* Zoom FFT.
- Spectrum analysis. *See also* SDFT (sliding DFT); Zoom FFT.
 - center frequencies, expanding, 748–749
 - with SDFT (sliding DFT), 748–749
 - single tone detection, 737–741
 - weighted overlap-add, 755
 - windowed-presum FFT, 755
 - Zoom FFT, 749–753
- Spectrum analyzer, 753–756
- Spurious free dynamic range (SFDR), 714–715
- Stability
 - comb filters, 403–404
 - conditional, 268
 - FSF (frequency sampling filters), 403–406
 - IIR filters, 263–270
 - Laplace transfer function, 263–264, 268
 - Laplace transform, 263–270
 - SDFT (sliding DFT), 746–747
 - single tone detection, 838–840
 - z-transform and, 272–274, 277
- Stair-step effect, A/D converter quantization noise, 637
- Standard deviation
 - of continuous sinewaves, 874–875
 - definition, 870
 - signal averaging, 590
- Statistical measures of noise
 - average, 868–870
 - average power in electrical circuits, 874–875
 - Bessel's correction, 870–871
 - biased estimates, 870–871
 - dispersion, 869
 - fluctuations around the average, 869

- overview, 867–870. *See also* SNR (signal-to-noise ratio).
 - of real-valued sequences, 874
 - rms value of continuous sinewaves, 874–875
 - of short sequences, 870–871
 - standard deviation, definition, 870
 - standard deviation, of continuous sinewaves, 874–875
 - summed sequences, 872–874
 - unbiased estimates, 871
 - Statistical measures of noise, estimating SNR
 - for common devices, 876
 - controlling SNR test signals, 879
 - in the frequency domain, 877–879
 - overview, 875–876
 - in the time domain, 876–877
 - Statistical measures of noise, mean
 - definition, 868–869
 - PDF (probability density function), 879–882
 - of random functions, 879–882
 - Statistical measures of noise, variance. *See also* Signal variance.
 - definition, 868–870
 - PDF (probability density function), 879–882
 - of random functions, 879–882
 - Steinmetz, Charles P., 446
 - Stockham, Thomas, 716
 - Stopband, definition, 901
 - Stopband ripples
 - definition, 901
 - minimizing, 204–207
 - Stopband sidelobe level suppression, 416
 - Structure, definition, 901
 - Structures, IIR filters
 - biquad filters, 299
 - cascade filter properties, 295–297
 - cascaded, 295–299
 - cascade/parallel combinations, 295–297
 - changing, 291–292
 - Direct Form 1, 275–278, 289
 - Direct Form II, 289–292
 - optimizing partitioning, 297–299
 - parallel filter properties, 295–297
 - transposed, 291–292
 - transposed Direct Form II, 289–290
 - transposition theorem, 291–292
 - Sub-Nyquist sampling. *See* Bandpass sampling.
 - Substructure sharing, 765–770
 - Subtraction
 - block diagram symbol, 10
 - complex numbers, 850
 - Summation
 - block diagram symbol, 10
 - description, 11
 - equation, 10
 - notation, 11
 - Symbols
 - block diagram, 10–11
 - signal processing, 10–11
 - Symmetrical rectangular functions, 112–115
 - Symmetrical-coefficient FIR filters, 232–233
 - Symmetry, DFT, 73–75
- T**
- Tacoma Narrows Bridge collapse, 263
 - Tap, definition, 901
 - Tap weights. *See* Filter coefficients.
 - Tapped delay, FIR filters, 174, 181–182
 - Taylor series approximation, 788
 - Tchebyshev function, definition, 902
 - Tchebyshev windows, in FIR filter design, 197
 - Time data, manipulating in FFTs, 138–139
 - Time invariance, decimation, 514
 - Time properties
 - decimation, 514–515
 - interpolation, 519
 - Time representation, continuous *vs.* discrete systems, 5
 - Time reversal, 863–865
 - Time sequences, notation syntax, 7
 - Time-domain
 - aliasing, avoiding, 718–722
 - analytic signals, generating, 495–497
 - coefficients, determining, 186–194
 - convolution, matched filters, 380
 - convolution *vs.* frequency-domain multiplication, 191–194
 - equations, example, 7

- Time-domain (*cont.*)
 - FIR filter implementation, 489–494
 - Hilbert transforms, designing, 489–494
 - interpolation, 778–781
 - slope filters, 820–821
 - Time-domain data, converting
 - from frequency-domain data. *See* IDFT (inverse discrete Fourier transform).
 - to frequency-domain data. *See* DFT (discrete Fourier transform).
 - Time-domain filters
 - coherent signal averaging, 609–612
 - exponential signal averaging, 609–612
 - Time-domain signals
 - amplitude, determining, 140
 - continuous, Laplace transform for, 258
 - DC removal, 812–815
 - definition, 4
 - vs.* frequency-domain, 120–123
 - Time-invariant systems. *See also* LTI (linear time-invariant) systems.
 - analyzing, 19–21
 - commutative property, 18–19
 - definition, 17–18
 - example of, 17–18
 - Tone detection. *See* Single tone detection.
 - Transfer functions. *See also* Laplace transfer function.
 - definition, 902
 - real FSF, 908–909
 - z-domain, 282–289
 - Transient response, FIR filters, 181–182
 - Transition region, definition, 902
 - Translation, sampling, 44
 - Transposed Direct Form II filters, 289–290
 - Transposed Direct Form II structure, 289–290
 - Transposed filters, 291–292
 - Transposed structures, 765–770
 - Transposition theorem, 291–292
 - Transversal filters, 173–174. *See also* FIR (finite impulse response) filters.
 - Triangular dither, 708
 - Triangular windows, 89–93
 - Trigonometric form, quadrature signals, 442, 444
 - Trigonometric form of complex numbers, 848–850
 - Truncation, fixed-point binary numbers, 646–649
 - Tukey, J., 135
 - Two's complement
 - fixed-point binary formats, 626–627, 629
 - overflow, 559–563
 - Two-sided Laplace transform, 258
 - Type-IV FSF
 - examples, 419–420, 423–426
 - frequency response, 910–912
 - optimum transition coefficients, 913–926
- ## U
- Unbiased estimates, 871
 - Unbiased signal variance, computing, 797–799, 799–801
 - Undersampling lowpass signals, 40. *See also* Bandpass sampling.
 - Uniform windows. *See* Rectangular windows.
 - Unit circles
 - definition, 271
 - z-transform, 271
 - Unit circles, FSF
 - forcing poles and zeros inside, 405
 - pole / zero cancellation, 395–398
 - Unit delay
 - block diagram symbol, 10
 - description, 11
 - Unit impulse response, LTI, 19–20
 - Unnormalized fractions, floating-point
 - binary formats, 656
 - Unwrapping, phase, 210
 - Upsampling, interpolation, 517–518, 520–521
- ## V
- Variance. *See* Signal variance.
 - Vector, definition, 848
 - Vector rotation with arctangents
 - to the 1st octant, 805–808
 - division by zero, avoiding, 808
 - jump address index bits, 807
 - overview, 805

- by $\pm \pi/8$, 809–810
- rotational symmetries, 807
- Vector-magnitude approximation, 679–683
- von Hann windows. *See* Hanning windows.

W

- Warping, frequency, 319, 321–325, 328–330
- Weighted overlap-add spectrum analysis, 755
- Weighting factors, coherent signal averaging, 608, 789
- Wideband compensation, 564
- Wideband differentiators, 367–370
- Willson, A., 386
- Window design method, FIR filters, 186–194
- Windowed-presum FFT spectrum analysis, 755
- Windows
 - Blackman, 195–201, 686, 733
 - Blackman-Harris, 686, 733
 - exact Blackman, 686
 - FFTs, 139
 - in the frequency domain, 683–686
 - magnitude response, 92–93
 - mathematical expressions of, 91
 - minimizing DFT leakage, 89–97
 - processing gain or loss, 92
 - purpose of, 96
 - rectangular, 89–97, 686
 - selecting, 96
 - triangular, 89–93
- Windows, Hamming
 - description, 89–93
 - DFT leakage reduction, 89–93
 - in the frequency domain, 683–686
 - spectral peak location, 733
- Windows, Hanning
 - description, 89–97
 - DFT leakage, minimizing, 89–97
 - in the frequency domain, 683–686
 - spectral peak location, 733
- Windows used in FIR filter design
 - Bessel functions, 198–199
 - Blackman, 195–201

- Chebyshev, 197–201, 927–930
 - choosing, 199–201
 - Dolph-Chebyshev, 197
 - Kaiser, 197–201
 - Kaiser-Bessel, 197
 - Tchebyschev, 197
 - Wingless butterflies, 156
 - Wraparound leakage, 86–88
 - Wrapping, phase, 209, 900
- ## Z
- z-domain expression for Mth-order IIR filter, 275–276
 - z-domain transfer function, IIR filters, 282–289
 - Zero padding
 - alleviating scalloping loss, 97–102
 - FFTs, 138–139
 - FIR filters, 228–230
 - improving DFT frequency granularity, 97–102
 - spectral peak location, 731
 - Zero stuffing
 - interpolation, 518
 - narrowband lowpass filters, 834–836
 - Zero-overhead looping
 - DSP chips, 333
 - FSF (frequency sampling filters), 422–423
 - IFIR filters, 389
 - Zero-phase filters
 - definition, 902
 - techniques, 725
 - Zeros
 - IIR filters, 284–289
 - on the s-plane, Laplace transform, 263–270
 - Zoom FFT, 749–753
 - z-plane, 270–273
 - z-plane pole / zero properties, IIR filters, 288–289
 - z-transform. *See also* Laplace transform.
 - definition, 270
 - description of, 270–272
 - FIR filters, 288–289
 - IIR filters, 270–282
 - infinite impulse response, definition, 280

- z-transform (*cont.*)
 - polar form, 271
 - poles, 272–274
 - unit circles, 271
 - zeros, 272–274
- z-transform, analyzing IIR filters
 - digital filter stability, 272–274, 277
 - Direct Form 1 structure, 275–278
 - example, 278–282
 - frequency response, 277–278
 - overview, 274–275
 - time delay, 274–278
 - z-domain transfer function, 275–278, 279–280