

SECOND EDITION

Digital Signal Processing

SYSTEM ANALYSIS AND DESIGN



**Paulo S. R. Diniz
Eduardo A. B. da Silva
Sergio L. Netto**

CAMBRIDGE

CAMBRIDGE

www.cambridge.org/9780521887755

This page intentionally left blank

Digital Signal Processing

This new, fully revised edition covers all the major topics of digital signal processing (DSP) design and analysis in a single, all-inclusive volume, interweaving theory with real-world examples and design trade-offs.

Building on the success of the original, this edition includes new material on random signal processing, a new chapter on spectral estimation, greatly expanded coverage of filter banks and wavelets, and new material on the solution of difference equations. Additional steps in mathematical derivations make them easier to follow, and an important new feature is the Do-it-Yourself section at the end of each chapter, where readers get hands-on experience of solving practical signal processing problems in a range of MATLAB® experiments.

With 120 worked examples, 20 case studies, and almost 400 homework exercises, the book is essential reading for anyone taking digital signal processing courses. Its unique blend of theory and real-world practical examples also makes it an ideal reference for practitioners.

Paulo S. R. Diniz is a Professor in the Department of Electronics and Computer Engineering at Poli/Federal University of Rio de Janeiro (UFRJ), and the Graduate Program of Electrical Engineering at COPPE/UFRJ. He is also a Fellow of the IEEE.

Eduardo A. B. da Silva is an Associate Professor in the Department of Electronics and Computer Engineering at Poli/UFRJ, and in the Graduate Program of Electrical Engineering at COPPE/UFRJ.

Sergio I. Netto is an Associate Professor in the Department of Electronics and Computer Engineering at Poli/UFRJ, and in the Graduate Program of Electrical Engineering at COPPE/UFRJ.

Digital Signal Processing

System Analysis and Design

Second Edition

Paulo S. R. Diniz
Eduardo A. B. da Silva
and
Sergio L. Netto
Federal University of Rio de Janeiro



CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,
São Paulo, Delhi, Dubai, Tokyo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org

Information on this title: www.cambridge.org/9780521887755

© Cambridge University Press 2002, 2010

This publication is in copyright. Subject to statutory exception and to the provision of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published in print format 2010

ISBN-13 978-0-511-78983-0 eBook (NetLibrary)

ISBN-13 978-0-521-88775-5 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy of urls for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

**To our families,
our parents,
and our students.**

Contents

<i>Preface</i>	<i>page</i>	xvi
Introduction		1
1 Discrete-time signals and systems	5	
1.1 Introduction	5	
1.2 Discrete-time signals	6	
1.3 Discrete-time systems	10	
1.3.1 Linearity	10	
1.3.2 Time invariance	11	
1.3.3 Causality	11	
1.3.4 Impulse response and convolution sums	14	
1.3.5 Stability	16	
1.4 Difference equations and time-domain response	17	
1.4.1 Recursive \times nonrecursive systems	21	
1.5 Solving difference equations	22	
1.5.1 Computing impulse responses	31	
1.6 Sampling of continuous-time signals	33	
1.6.1 Basic principles	34	
1.6.2 Sampling theorem	34	
1.7 Random signals	53	
1.7.1 Random variable	54	
1.7.2 Random processes	58	
1.7.3 Filtering a random signal	60	
1.8 Do-it-yourself: discrete-time signals and systems	62	
1.9 Discrete-time signals and systems with MATLAB	67	
1.10 Summary	68	
1.11 Exercises	68	
2 The z and Fourier transforms	75	
2.1 Introduction	75	
2.2 Definition of the z transform	76	
2.3 Inverse z transform	83	
2.3.1 Computation based on residue theorem	84	
2.3.2 Computation based on partial-fraction expansions	87	
2.3.3 Computation based on polynomial division	90	

2.3.4	Computation based on series expansion	92
2.4	Properties of the z transform	94
2.4.1	Linearity	94
2.4.2	Time reversal	94
2.4.3	Time-shift theorem	95
2.4.4	Multiplication by an exponential	95
2.4.5	Complex differentiation	95
2.4.6	Complex conjugation	96
2.4.7	Real and imaginary sequences	97
2.4.8	Initial-value theorem	97
2.4.9	Convolution theorem	98
2.4.10	Product of two sequences	98
2.4.11	Parseval's theorem	100
2.4.12	Table of basic z transforms	101
2.5	Transfer functions	104
2.6	Stability in the z domain	106
2.7	Frequency response	109
2.8	Fourier transform	115
2.9	Properties of the Fourier transform	120
2.9.1	Linearity	120
2.9.2	Time reversal	120
2.9.3	Time-shift theorem	120
2.9.4	Multiplication by a complex exponential (frequency shift, modulation)	120
2.9.5	Complex differentiation	120
2.9.6	Complex conjugation	121
2.9.7	Real and imaginary sequences	121
2.9.8	Symmetric and antisymmetric sequences	122
2.9.9	Convolution theorem	123
2.9.10	Product of two sequences	123
2.9.11	Parseval's theorem	123
2.10	Fourier transform for periodic sequences	123
2.11	Random signals in the transform domain	125
2.11.1	Power spectral density	125
2.11.2	White noise	128
2.12	Do-it-yourself: the z and Fourier transforms	129
2.13	The z and Fourier transforms with MATLAB	135
2.14	Summary	137
2.15	Exercises	137
3	Discrete transforms	143
3.1	Introduction	143
3.2	Discrete Fourier transform	144
3.3	Properties of the DFT	153

3.3.1	Linearity	153
3.3.2	Time reversal	153
3.3.3	Time-shift theorem	153
3.3.4	Circular frequency-shift theorem (modulation theorem)	156
3.3.5	Circular convolution in time	157
3.3.6	Correlation	158
3.3.7	Complex conjugation	159
3.3.8	Real and imaginary sequences	159
3.3.9	Symmetric and antisymmetric sequences	160
3.3.10	Parseval's theorem	162
3.3.11	Relationship between the DFT and the z transform	163
3.4	Digital filtering using the DFT	164
3.4.1	Linear and circular convolutions	164
3.4.2	Overlap-and-add method	168
3.4.3	Overlap-and-save method	171
3.5	Fast Fourier transform	175
3.5.1	Radix-2 algorithm with decimation in time	176
3.5.2	Decimation in frequency	184
3.5.3	Radix-4 algorithm	187
3.5.4	Algorithms for arbitrary values of N	192
3.5.5	Alternative techniques for determining the DFT	193
3.6	Other discrete transforms	194
3.6.1	Discrete transforms and Parseval's theorem	195
3.6.2	Discrete transforms and orthogonality	196
3.6.3	Discrete cosine transform	199
3.6.4	A family of sine and cosine transforms	203
3.6.5	Discrete Hartley transform	205
3.6.6	Hadamard transform	206
3.6.7	Other important transforms	207
3.7	Signal representations	208
3.7.1	Laplace transform	208
3.7.2	The z transform	208
3.7.3	Fourier transform (continuous time)	209
3.7.4	Fourier transform (discrete time)	209
3.7.5	Fourier series	210
3.7.6	Discrete Fourier transform	210
3.8	Do-it-yourself: discrete transforms	211
3.9	Discrete transforms with MATLAB	215
3.10	Summary	216
3.11	Exercises	217
4	Digital filters	222
4.1	Introduction	222
4.2	Basic structures of nonrecursive digital filters	222

4.2.1	Direct form	223
4.2.2	Cascade form	224
4.2.3	Linear-phase forms	225
4.3	Basic structures of recursive digital filters	232
4.3.1	Direct forms	232
4.3.2	Cascade form	236
4.3.3	Parallel form	237
4.4	Digital network analysis	241
4.5	State-space description	244
4.6	Basic properties of digital networks	246
4.6.1	Tellegen's theorem	246
4.6.2	Reciprocity	248
4.6.3	Interreciprocity	249
4.6.4	Transposition	249
4.6.5	Sensitivity	250
4.7	Useful building blocks	257
4.7.1	Second-order building blocks	257
4.7.2	Digital oscillators	260
4.7.3	Comb filter	261
4.8	Do-it-yourself: digital filters	263
4.9	Digital filter forms with MATLAB	266
4.10	Summary	270
4.11	Exercises	270
5	FIR filter approximations	277
5.1	Introduction	277
5.2	Ideal characteristics of standard filters	277
5.2.1	Lowpass, highpass, bandpass, and bandstop filters	278
5.2.2	Differentiators	280
5.2.3	Hilbert transformers	281
5.2.4	Summary	283
5.3	FIR filter approximation by frequency sampling	283
5.4	FIR filter approximation with window functions	291
5.4.1	Rectangular window	294
5.4.2	Triangular windows	295
5.4.3	Hamming and Hann windows	296
5.4.4	Blackman window	297
5.4.5	Kaiser window	299
5.4.6	Dolph–Chebyshev window	306
5.5	Maximally flat FIR filter approximation	309
5.6	FIR filter approximation by optimization	313
5.6.1	Weighted least-squares method	317
5.6.2	Chebyshev method	321
5.6.3	WLS–Chebyshev method	327

5.7	Do-it-yourself: FIR filter approximations	333
5.8	FIR filter approximation with MATLAB	336
5.9	Summary	342
5.10	Exercises	343
6	IIR filter approximations	349
6.1	Introduction	349
6.2	Analog filter approximations	350
6.2.1	Analog filter specification	350
6.2.2	Butterworth approximation	351
6.2.3	Chebyshev approximation	353
6.2.4	Elliptic approximation	356
6.2.5	Frequency transformations	359
6.3	Continuous-time to discrete-time transformations	368
6.3.1	Impulse-invariance method	368
6.3.2	Bilinear transformation method	372
6.4	Frequency transformation in the discrete-time domain	378
6.4.1	Lowpass-to-lowpass transformation	379
6.4.2	Lowpass-to-highpass transformation	380
6.4.3	Lowpass-to-bandpass transformation	380
6.4.4	Lowpass-to-bandstop transformation	381
6.4.5	Variable-cutoff filter design	381
6.5	Magnitude and phase approximation	382
6.5.1	Basic principles	382
6.5.2	Multivariable function minimization method	387
6.5.3	Alternative methods	389
6.6	Time-domain approximation	391
6.6.1	Approximate approach	393
6.7	Do-it-yourself: IIR filter approximations	394
6.8	IIR filter approximation with MATLAB	399
6.9	Summary	403
6.10	Exercises	404
7	Spectral estimation	409
7.1	Introduction	409
7.2	Estimation theory	410
7.3	Nonparametric spectral estimation	411
7.3.1	Periodogram	411
7.3.2	Periodogram variations	413
7.3.3	Minimum-variance spectral estimator	416
7.4	Modeling theory	419
7.4.1	Rational transfer-function models	419
7.4.2	Yule–Walker equations	423

7.5	Parametric spectral estimation	426
7.5.1	Linear prediction	426
7.5.2	Covariance method	430
7.5.3	Autocorrelation method	431
7.5.4	Levinson–Durbin algorithm	432
7.5.5	Burg’s method	434
7.5.6	Relationship of the Levinson–Durbin algorithm to a lattice structure	438
7.6	Wiener filter	438
7.7	Other methods for spectral estimation	441
7.8	Do-it-yourself: spectral estimation	442
7.9	Spectral estimation with MATLAB	449
7.10	Summary	450
7.11	Exercises	451
8	Multirate systems	455
8.1	Introduction	455
8.2	Basic principles	455
8.3	Decimation	456
8.4	Interpolation	462
8.4.1	Examples of interpolators	464
8.5	Rational sampling-rate changes	465
8.6	Inverse operations	466
8.7	Noble identities	467
8.8	Polyphase decompositions	469
8.9	Commutator models	471
8.10	Decimation and interpolation for efficient filter implementation	474
8.10.1	Narrowband FIR filters	474
8.10.2	Wideband FIR filters with narrow transition bands	477
8.11	Overlapped block filtering	479
8.11.1	Nonoverlapped case	480
8.11.2	Overlapped input and output	483
8.11.3	Fast convolution structure I	487
8.11.4	Fast convolution structure II	487
8.12	Random signals in multirate systems	490
8.12.1	Interpolated random signals	491
8.12.2	Decimated random signals	492
8.13	Do-it-yourself: multirate systems	493
8.14	Multirate systems with MATLAB	495
8.15	Summary	497
8.16	Exercises	498
9	Filter banks	503
9.1	Introduction	503
9.2	Filter banks	503

9.2.1	Decimation of a bandpass signal	504
9.2.2	Inverse decimation of a bandpass signal	505
9.2.3	Critically decimated M -band filter banks	506
9.3	Perfect reconstruction	507
9.3.1	M -band filter banks in terms of polyphase components	507
9.3.2	Perfect reconstruction M -band filter banks	509
9.4	Analysis of M -band filter banks	517
9.4.1	Modulation matrix representation	518
9.4.2	Time-domain analysis	520
9.4.3	Orthogonality and biorthogonality in filter banks	529
9.4.4	Transmultiplexers	534
9.5	General two-band perfect reconstruction filter banks	535
9.6	QMF filter banks	540
9.7	CQF filter banks	543
9.8	Block transforms	548
9.9	Cosine-modulated filter banks	554
9.9.1	The optimization problem in the design of cosine-modulated filter banks	559
9.10	Lapped transforms	563
9.10.1	Fast algorithms and biorthogonal LOT	573
9.10.2	Generalized LOT	576
9.11	Do-it-yourself: filter banks	581
9.12	Filter banks with MATLAB	594
9.13	Summary	594
9.14	Exercises	595
10	Wavelet transforms	599
10.1	Introduction	599
10.2	Wavelet transforms	599
10.2.1	Hierarchical filter banks	599
10.2.2	Wavelets	601
10.2.3	Scaling functions	605
10.3	Relation between $x(t)$ and $x(n)$	606
10.4	Wavelet transforms and time–frequency analysis	607
10.4.1	The short-time Fourier transform	607
10.4.2	The continuous-time wavelet transform	612
10.4.3	Sampling the continuous-time wavelet transform: the discrete wavelet transform	614
10.5	Multiresolution representation	617
10.5.1	Biorthogonal multiresolution representation	620
10.6	Wavelet transforms and filter banks	623
10.6.1	Relations between the filter coefficients	629
10.7	Regularity	633
10.7.1	Additional constraints imposed on the filter banks due to the regularity condition	634

10.7.2	A practical estimate of regularity	635
10.7.3	Number of vanishing moments	636
10.8	Examples of wavelets	638
10.9	Wavelet transforms of images	641
10.10	Wavelet transforms of finite-length signals	646
10.10.1	Periodic signal extension	646
10.10.2	Symmetric signal extensions	648
10.11	Do-it-yourself: wavelet transforms	653
10.12	Wavelets with MATLAB	659
10.13	Summary	664
10.14	Exercises	665
11	Finite-precision digital signal processing	668
11.1	Introduction	668
11.2	Binary number representation	670
11.2.1	Fixed-point representations	670
11.2.2	Signed power-of-two representation	672
11.2.3	Floating-point representation	673
11.3	Basic elements	674
11.3.1	Properties of the two's-complement representation	674
11.3.2	Serial adder	674
11.3.3	Serial multiplier	676
11.3.4	Parallel adder	684
11.3.5	Parallel multiplier	684
11.4	Distributed arithmetic implementation	685
11.5	Product quantization	691
11.6	Signal scaling	697
11.7	Coefficient quantization	706
11.7.1	Deterministic sensitivity criterion	708
11.7.2	Statistical forecast of the wordlength	711
11.8	Limit cycles	715
11.8.1	Granular limit cycles	715
11.8.2	Overflow limit cycles	717
11.8.3	Elimination of zero-input limit cycles	719
11.8.4	Elimination of constant-input limit cycles	725
11.8.5	Forced-response stability of digital filters with nonlinearities due to overflow	729
11.9	Do-it-yourself: finite-precision digital signal processing	732
11.10	Finite-precision digital signal processing with MATLAB	735
11.11	Summary	735
11.12	Exercises	736
12	Efficient FIR structures	740
12.1	Introduction	740
12.2	Lattice form	740

12.2.1	Filter banks using the lattice form	742
12.3	Polyphase form	749
12.4	Frequency-domain form	750
12.5	Recursive running sum form	750
12.6	Modified-sinc filter	752
12.7	Realizations with reduced number of arithmetic operations	753
12.7.1	Prefilter approach	753
12.7.2	Interpolation approach	756
12.7.3	Frequency-response masking approach	760
12.7.4	Quadrature approach	771
12.8	Do-it-yourself: efficient FIR structures	776
12.9	Efficient FIR structures with MATLAB	781
12.10	Summary	782
12.11	Exercises	782
13	Efficient IIR structures	787
13.1	Introduction	787
13.2	IIR parallel and cascade filters	787
13.2.1	Parallel form	788
13.2.2	Cascade form	790
13.2.3	Error spectrum shaping	795
13.2.4	Closed-form scaling	797
13.3	State-space sections	800
13.3.1	Optimal state-space sections	801
13.3.2	State-space sections without limit cycles	806
13.4	Lattice filters	815
13.5	Doubly complementary filters	822
13.5.1	QMF filter bank implementation	826
13.6	Wave filters	828
13.6.1	Motivation	829
13.6.2	Wave elements	832
13.6.3	Lattice wave digital filters	848
13.7	Do-it-yourself: efficient IIR structures	855
13.8	Efficient IIR structures with MATLAB	857
13.9	Summary	857
13.10	Exercises	858
<i>References</i>		863
<i>Index</i>		877

Preface

This book originated from a training course for engineers at the research and development center of TELEBRAS, the former Brazilian telecommunications holding. That course was taught by the first author back in 1987, and its main goal was to present efficient digital filter design methods suitable for solving some of their engineering problems. Later on, this original text was used by the first author as the basic reference for the digital filters and digital signal processing courses of the Electrical Engineering Program at COPPE/Federal University of Rio de Janeiro.

For many years, former students asked why the original text was not transformed into a book, as it presented a very distinct view that they considered worth publishing. Among the numerous reasons not to attempt such task, we could mention that there were already a good number of well-written texts on the subject; also, after many years of teaching and researching on this topic, it seemed more interesting to follow other paths than the painful one of writing a book; finally, the original text was written in Portuguese and a mere translation of it into English would be a very tedious task.

In later years, the second and third authors, who had attended the signal processing courses using the original material, were continuously giving new ideas on how to proceed. That was when we decided to go through the task of completing and updating the original text, turning it into a modern textbook. The book then took on its first-edition form, updating the original text, and including a large amount of new material written for other courses taught by the three authors up to 2002.

This second edition barely resembles the original lecture notes for several reasons. The original material was heavily concentrated on filter design and realization, whereas the present version includes a large amount of material on discrete-time systems, discrete transforms, spectral estimation, multirate systems, filter banks, and wavelets.

This book is mainly written for use as a textbook on a digital signal processing course for undergraduate students who have had previous exposure to basic linear systems, or to serve as a textbook on a graduate-level course where the most advanced topics of some chapters are covered. This reflects the structure we have at the Federal University of Rio de Janeiro, as well as at a number of other universities we have contact with. The second edition has a special feature designed for readers to test their learning by hands-on experience through so-called Do-it-yourself sections, with the aid of MATLAB®. A Do-it-yourself section is included in all chapters of the book. The book also includes, at the end of most chapters, a brief section aimed at giving a start to the reader on how to use MATLAB as a tool for the analysis and design of digital signal processing systems. As in the first edition, we decided that having explanations about MATLAB inserted in the main text would in some cases distract the readers, making them lose focus on the subject.

A distinctive feature of this book is to present a wide range of topics in digital signal processing design and analysis in a concise and complete form, while allowing the reader to fully develop practical systems. Although this book is primarily intended as an undergraduate and graduate textbook, its origins on training courses for industry warrant its potential usefulness to engineers working in the development of signal processing systems. In fact, our objective is to equip the readers with the tools that enable them to understand why and how to use digital signal processing systems; to show them how to approximate a desired transfer function characteristic using polynomials and ratios of polynomials; to teach them why an appropriate mapping of a transfer function into a suitable structure is important for practical applications; and to show how to analyze, represent, and explore the trade-off between the time and frequency representations of deterministic and stochastic signals. For all that, each chapter includes a number of examples and end-of-chapter problems to be solved. These are aimed at assimilating the concepts, as well as complementing the text. In particular, the second edition includes many new examples and exercises to be solved.

Chapters 1 and 2 review the basic concepts of discrete-time signal processing and z transforms. Although many readers may be familiar with these subjects, they could benefit from reading these chapters, getting used to the notation and the authors' way of presenting the subject. In Chapter 1 we review the concepts of discrete-time systems, including the representation of discrete-time signals and systems, as well as their time-domain responses. Most important, we present the sampling theorem, which sets the conditions for the discrete-time systems to solve practical problems related to our real continuous-time world. The basic concepts of random signals are also introduced in this chapter, followed by the Do-it-yourself section aiding the reader to test their progress in discrete-time signals and systems. Chapter 2 is concerned with the z and Fourier transforms, which are useful mathematical tools for representation of discrete-time signals and systems. The basic properties of the z and Fourier transforms are discussed, including a stability test in the z transform domain. The chapter also shows how the analysis of random signals can benefit from the z -domain formulation.

Chapter 3 discusses discrete transforms, with special emphasis given to the discrete Fourier transform (DFT), which is an invaluable tool in the frequency analysis of discrete-time signals. The DFT allows a discrete representation of discrete-time signals in the frequency domain. Since the sequence representation is natural for digital computers, the DFT is a very powerful tool, because it enables us to manipulate frequency-domain information in the same way as we can manipulate the original sequences. The importance of the DFT is further increased by the fact that computationally efficient algorithms, the so-called fast Fourier transforms (FFTs), are available to compute the DFT. This chapter also presents real coefficient transforms, such as cosine and sine transforms, which are widely used in modern audio and video coding, as well as in a number of other applications. A discussion about orthogonality in transforms is also included. This section also includes a discussion on the several forms of representing the signals, in order to aid the reader with the available choices.

Chapter 4 addresses the basic structures for mapping a transfer function into a digital filter. It is also devoted to some basic analysis methods and properties of digital filter structures.

The chapter also introduces some simple and useful building blocks widely utilized in some designs and applications.

Chapter 5 introduces several approximation methods for filters with finite-duration impulse response (FIR), starting with the simpler frequency sampling method and the widely used windows method. This method also provides insight to the windowing strategy used in several signal processing applications. Other approximation methods included are the maximally flat filters and those based on the weighted least-squares (WLS) method. This chapter also presents the Chebyshev approximation based on a multivariable optimization algorithm called the Remez exchange method. This approach leads to linear-phase transfer functions with minimum order given a prescribed set of frequency response specifications. This chapter also discusses the WLS–Chebyshev method which leads to transfer functions where the maximum and the total energy of the approximation error are prescribed. This approximation method is not widely discussed in the open literature but appears to be very useful for a number of applications.

Chapter 6 discusses the approximation procedures for filters with infinite-duration impulse response (IIR). We start with the classical continuous-time transfer-function approximations, namely the Butterworth, Chebyshev, and elliptic approximations, that can generate discrete-time transfer functions by using appropriate transformations. Two transformation methods are then presented: the impulse-invariance and the bilinear transformation methods. The chapter also includes a section on frequency transformations in the discrete-time domain. The simultaneous magnitude and phase approximation of IIR digital filters using optimization techniques is also included, providing a tool to design transfer functions satisfying more general specifications. The chapter closes by addressing the issue of time-domain approximations.

Chapter 7 introduces the basic concepts of classical estimation theory. It starts by describing the nonparametric spectral estimation methods based on a periodogram, followed by the minimum-variance spectral estimator. The chapter continues with a discussion on modeling theory, addressing the rational transfer function models and presenting the Yule–Walker equations. Several parametric spectral estimation methods are also presented, namely: the linear prediction method; the covariance method; the autocorrelation method; the Levinson–Durbin algorithm; and Burg’s method. The chapter also discusses the Wiener filter as an extension of the linear prediction method.

Chapter 8 deals with basic principles of discrete-time systems with multiple sampling rates. In this chapter we emphasize the basic properties of multirate systems, thoroughly addressing the decimation and interpolation operations, giving examples of their use for efficient digital filter design. The chapter discusses many key properties of multirate systems, such as inverse operations and noble identities, and introduces some analytical tools, such as polyphase decomposition and the commutator models. In addition, we discuss the concepts of overlapped block filtering, which can be very useful in some fast implementations of digital signal processing building blocks. The chapter also includes some discussion on how decimators and interpolators affect the properties of random signals.

Chapter 9 discusses some properties pertaining to the internal structure of filter banks, followed by the concept and construction of perfect reconstruction filter banks. The chapter also

includes some analysis tools and classifications for the filter banks and transmultiplexers. This chapter presents several design techniques for multirate filter banks, including several forms of two-band filter banks, cosine-modulated filter banks, and lapped transforms.

Chapter 10 introduces the concepts of time–frequency analysis and the discrete wavelet transform. It also presents the multiresolution representation of signals through wavelet transforms and discusses the design of wavelet transforms using filter banks. In addition, some design techniques to generate orthogonal (as well as biorthogonal) bases for signal representation are presented. Several properties of wavelets required for their classification, design, and implementation are discussed in this chapter.

Chapter 11 provides a brief introduction to the binary number representations most widely used in the implementation of digital signal processing systems. The chapter also explains how the basic elements utilized in these systems work and discusses a particular, and yet instructive, type of implementation based on distributed arithmetic. Chapter 11 also includes the models that account for quantization effects in digital filters. We discuss several approaches to analyze and deal with the effects of representing signals and filter coefficients with finite wordlength. In particular, we study the effects of quantization noise in products, signal scaling that limits the internal signal dynamic range, coefficient quantization in the designed transfer function, and the nonlinear oscillations which may occur in recursive realizations. These analyses are used to indicate the filter realizations that lead to practical finite-precision implementations of digital filters.

In Chapter 12 we present some techniques to reduce the computational complexity of FIR filters with demanding specifications or specialized requirements. The first structure discussed is the lattice form, which finds application in a number of areas, including the design of filter banks. Several useful implementation forms of FIR filters, such as polyphase, frequency-domain, recursive running sum, and modified-sinc forms, are presented to be employed as building blocks in several design methods. In particular, we introduce the prefilter and interpolation methods which are mainly useful in designing narrowband lowpass and highpass filters. In addition, we present the frequency-response masking approach, for designing filters with narrow transition bands satisfying more general specifications, and the quadrature method, for narrow bandpass and bandstop filters.

Chapter 13 presents a number of efficient realizations for IIR filters. For these filters, a number of realizations considered efficient from the finite-precision effects point of view are presented and their salient features are discussed in detail. These realizations will equip the reader with a number of choices for the design of good IIR filters. Several families of structures are considered in this chapter, namely: parallel and cascade designs using direct-form second-order sections; parallel and cascade designs using section-optimal and limit-cycle-free state-space sections; lattice filters; and several forms of wave digital filters. In addition, this chapter includes a discussion on doubly complementary filters and their use in the implementation of quadrature mirror filter banks.

This book contains enough material for an undergraduate course on digital signal processing and a first-year graduate course. There are many alternative ways to compose these courses; in the following we describe some recommendations that have been employed successfully in signal processing courses.

- An undergraduate course in discrete-time systems or digital signal processing at junior level. This should include most parts of Chapters 1, 2, 3, 4, and the nonparametric methods of Chapter 7. It could also include the noniterative approximation methods of Chapters 5 and 6, namely the frequency sampling and window methods described in Chapter 5, the analog-based approximation methods, and also the continuous-time to discrete-time transformation methods for IIR filtering of Chapter 6.
- An undergraduate course in digital signal processing at senior level. This should briefly review parts of Chapters 1 and 2 and cover Chapters 3, 4, and 7. It could also include the noniterative approximation methods of Chapters 5 and 6, namely the frequency sampling and window methods described in Chapter 5, the analog-based approximation methods, and also the continuous-time to discrete-time transformation methods for IIR filtering of Chapter 6. Chapters 8 and 11 could complement the course.
- An undergraduate course in digital filtering at senior level or first-year graduate. This should cover Chapter 4 and the iterative approximation methods of Chapters 5 and 6. The course could also cover selected topics from Chapters 11, 12, and 13. At the instructor's discretion, the course could also include selected parts of Chapter 8.
- As a graduate course textbook on multirate systems, filter banks and wavelets. The course could cover Chapters 8, 9, and 10, as well as the lattice form in Chapter 12 and doubly complementary filters in Chapter 13.

Obviously, there are several other choices for courses based on the material of this book which will depend on the course length and the judicious choice of the instructor.

This book would never be written if people with a wide vision of how an academic environment should be were not around. In fact, we were fortunate to have Professors L. P. Calôba and E. H. Watanabe as colleagues and advisors. The staff of COPPE, in particular Ms Michelle A. Nogueira and Ms F. J. Ribeiro, supported us in all possible ways to make this book a reality. Also, the first author's early students J. C. Cabezas, R. G. Lins, and J. A. B. Pereira (in memoriam) wrote, with him, a computer package that generated several of the examples of the first edition of this book. The engineers of CPqD helped us to correct the early version of this text. In particular, we would like to thank the engineer J. Sampaio for his complete trust in this work. We benefited from working in an environment with a large signal-processing group where our colleagues always helped us in various ways. Among them, we should mention Professors L. W. P. Biscainho, M. L. R. de Campos, G. V. Mendonça, A. C. M. de Queiroz, F. G. V. de Resende Jr, J. M. de Seixas, and the entire staff of the Signal Processing Lab (www.1ps.ufrj.br). Professor Biscainho superbly translated the first edition of this book to our mother tongue; he is indeed our inspirational fourth author. We would like to thank our colleagues at the Federal University of Rio de Janeiro, in particular at the Department of Electronics and Computer Engineering of the Polytechnic School of Engineering, the undergraduate studies department, and at the Electrical Engineering Program of COPPE, the graduate studies department, for their constant support during the preparation of this book.

We would like to thank many friends from other institutions whose influence helped in shaping this book. In particular, we may mention Professor A. S. de la Vega of Fluminense Federal University; Professor M. Sarcinelli Filho of the Federal University of Espírito

Santo; Professors P. Agathoklis, A. Antoniou, and W.-S. Lu of the University of Victoria; Professors I. Hartimo and T. I. Laakso and Dr. V. Välimäki of the Helsinki University of Technology; Professors T. Saramäki and Markku Renfors of the Tampere University of Technology; Professor Y. Lian of the National University of Singapore; Professor Y. C. Lim of Nanyang Technological University; Dr. R. L. de Queiroz of the University of Brasília; Dr. H. S. Malvar of Microsoft Corporation; Professor Y.-F. Huang of the University of Notre Dame; Professor J. E. Cousseau of Universidad Nacional del Sur; Professor B. Nowrouzian of University of Alberta; Dr. M. G. de Siqueira of Cisco Systems; Professors R. Miscow Filho and E. Viegas of the Military Institute of Engineering in Rio de Janeiro; Professor T. Q. Nguyen of the University of California, San Diego; and Professor Massimiliano Laddomada of Texas A&M University, Texarkana.

This acknowledgment list would be incomplete without mentioning the staff of Cambridge University Press, in particular our editor, Dr. Philip Meyler. Phil is an amazing person who knows how to stimulate people to write and read books.

We would like to thank our families for their endless patience and support. In particular, Paulo would like to express his deepest gratitude to Mariza, Paula, and Luiza, and to his mother Hirlene. Eduardo would like to mention that the continuing love and friendship from his wife Cláudia and his children Luis Eduardo and Isabella, as well as the strong and loving background provided by his parents, Zélia and Bismarck, were in all respects essential to the completion of this task. Sergio would like to express his deepest gratitude to his parents, “Big” Sergio and Maria Christina, his sincere love and admiration to his wife, Isabela, and the greatest affection to his offspring, Bruno and the twins, Renata and Manuela (see Figure 10.21). We all would also like to thank our families for bearing with us working together.

We sincerely hope that the book reflects the harmony, pleasure, friendship, and tenderness that we experience working together. Our partnership was written in the stars and heaven sent.

Introduction

When we hear the word “signal” we may first think of a phenomenon that occurs continuously over time that carries some information. Most phenomena observed in nature are continuous in time, such as for instance our speech or heart beating, the temperature of the city where we live, the car speed during a given trip, the altitude of the airplane we are traveling in – these are typical continuous-time signals. Engineers are always devising ways to design systems, which are in principle continuous time, for measuring and interfering with these and other phenomena.

One should note that, although continuous-time signals pervade our daily lives, there are also several signals which are originally discrete time; for example, the stock-market weekly financial indicators, the maximum and minimum daily temperatures in our cities, and the average lap speed of a racing car.

If an electrical or computer engineer has the task of designing systems to interact with natural phenomena, their first impulse is to convert some quantities from nature into electric signals through a transducer. Electric signals, which are represented by voltages or currents, have a continuous-time representation. Since digital technology constitutes an extremely powerful tool for information processing, it is natural to think of processing the electric signals generated using it. However, continuous-time signals cannot be processed using computer technology (digital machines), which are especially suited to deal with sequential computation involving numbers. Fortunately, this fact does not prevent the use of digital integrated circuits (which is the technology behind the computer technology revolution we witness today) in signal processing systems designs. This is because many signals taken from nature can be fully represented by their sampled versions, where the sampled signals coincide with the original continuous-time signals at some instants in time. If we know how fast the important information changes, then we can always sample and convert continuous-time information into discrete-time information which, in turn, can be converted into a sequence of numbers and transferred to a digital machine.

The main advantages of digital systems relative to analog systems are high reliability, suitability for modifying the system’s characteristics, and low cost. These advantages motivated the digital implementation of many signal processing systems which used to be implemented with analog circuit technology. In addition, a number of new applications became viable once the very large scale integration (VLSI) technology was available. Usually in the VLSI implementation of a digital signal processing system the concern is to reduce power consumption and/or area, or to increase the circuit’s speed in order to meet the demands of high-throughput applications.

Currently, a single digital integrated circuit may contain millions of logic gates operating at very high speeds, allowing very fast digital processors to be built at a reasonable cost. This technological development opened avenues to the introduction of powerful general-purpose computers which can be used to design and implement digital signal processing systems. In addition, it allowed the design of microprocessors with special features for signal processing, namely the digital signal processors (DSPs). As a consequence, there are several tools available to implement very complex digital signal processing systems. In practice, a digital signal processing system is implemented either by software on a general-purpose digital computer or DSP, or by using application-specific hardware, usually in the form of an integrated circuit.

For the reasons explained above, the field of digital signal processing has developed so fast in recent decades that it has been incorporated into the graduate and undergraduate programs of virtually all universities. This is confirmed by the number of good textbooks available in this area: Oppenheim & Schafer (1975, 1989); Rabiner & Gold (1975); Peled & Liu (1985); Roberts & Mullis (1987); Ifeachor & Jervis (1993); Jackson (1996); Antoniou (2006); Mitra (2006); Proakis & Manolakis (2007). The present book is aimed at equipping readers with tools that will enable them to design and analyze most digital signal processing systems. The building blocks for digital signal processing systems considered here are used to process signals which are discrete in time and in amplitude. The main tools emphasized in this text are:

- discrete-time signal representations
- discrete transforms and their fast algorithms
- spectral estimation
- design and implementation of digital filters and digital signal processing systems
- multirate systems and filter banks
- wavelets.

Transforms and filters are the main parts of linear signal processing systems. Although the techniques we deal with are directly applicable to processing deterministic signals, many statistical signal processing methods employ similar building blocks in some way, as will be clear in the text.

Digital signal processing is extremely useful in many areas. In the following, we enumerate a few of the disciplines where the topics covered by this book have found application.

- (a) *Image processing*: An image is essentially a space-domain signal; that is, it represents a variation of light intensity and color in space. Therefore, in order to process an image using an analog system, it has to be converted into a time-domain signal, using some form of scanning process. However, to process an image digitally, there is no need to perform this conversion, for it can be processed directly in the spatial domain, as a matrix of numbers. This lends the digital image processing techniques enormous power. In fact, in image processing, two-dimensional signal representation, filtering, and transforms play a central role (Jain, 1989).

- (b) *Multimedia systems*: Such systems deal with different kinds of information sources, such as image, video, audio, and data. In such systems, the information is essentially represented in digital form. Therefore, it is crucial to remove redundancy from the information, allowing compression, and thus efficient transmission and storage (Jayant & Noll, 1984; Gersho & Gray, 1992; Bhaskaran & Konstantinides, 1997). The Internet is a good application example where information files are transferred in a compressed manner. Most of the compression standards for images use transforms and quantizers.

The transforms, filter banks, and wavelets are very popular in compression applications because they are able to exploit the high correlation of signal sources such as audio, still images, and video (Malvar, 1992; Fliege, 1994; Vetterli and Kovačević, 1995; Strang and Nguyen, 1996; Mallat, 1999).

- (c) *Communication systems*: In communication systems, the compression and coding of the information sources are also crucial, since services can be provided at higher speed or to more users by reducing the amount of data to be transmitted. In addition, channel coding, which consists of inserting redundancy in the signal to compensate for possible channel distortions, may also use special types of digital filtering (Stüber, 1996).

Communication systems usually include fixed filters, as well as some self-designing filters for equalization and channel modeling which fall in the class of adaptive filtering systems (Diniz, 2008). Although these filters employ a statistical signal processing framework (Hayes, 1996; Kay, 1988; Manolakis *et al.*, 2000) to determine how their parameters should change, they also use some of the filter structures and in some cases the transforms introduced in this book.

Many filtering concepts take part on modern multiuser communication systems employing code-division multiple access (Verdu, 1998).

Wavelets, transforms, and filter banks also play a crucial role in the conception of orthogonal frequency-division multiplexing (OFDM) (Akansu & Medley, 1999), which is used in digital audio and TV broadcasting.

- (d) *Audio signal processing*: In statistical signal processing the filters are designed based on observed signals, which might imply that we are estimating the parameters of the model governing these signals (Kailath *et al.*, 2000). Such estimation techniques can be employed in digital audio restoration (Godsill & Rayner, 1998), where the resulting models can be used to restore lost information. However, these estimation models can be simplified and made more effective if we use some kind of sub-band processing with filter banks and transforms (Kahrs & Brandenburg, 1998). In the same field, digital filters were found to be suitable for reverberation algorithms and as models for musical instruments (Kahrs & Brandenburg, 1998).

In addition to the above applications, digital signal processing is at the heart of modern developments in speech analysis and synthesis, mobile radio, sonar, radar, biomedical engineering, seismology, home appliances, and instrumentation, among others. These developments occurred in parallel with the advances in the technology of transmission,

processing, recording, reproduction, and general treatment of signals through analog and digital electronics, as well as other means such as acoustics, mechanics, and optics.

We expect that, with the digital signal processing tools described in this book, the reader will be able to proceed further, not only exploring and understanding some of the applications described above, but developing new ones as well.

1.1 Introduction

Digital signal processing is the discipline that studies the rules governing the behavior of discrete signals, as well as the systems used to process them. It also deals with the issues involved in processing continuous signals using digital techniques. Digital signal processing pervades modern life. It has applications in compact disc players, computer tomography, geological processing, mobile phones, electronic toys, and many others.

In analog signal processing, we take a continuous signal, representing a continuously varying physical quantity, and pass it through a system that modifies this signal for a certain purpose. This modification is, in general, continuously variable by nature; that is, it can be described by differential equations.

Alternatively, in digital signal processing, we process sequences of numbers using some sort of digital hardware. We usually call these sequences of numbers digital or discrete-time signals. The power of digital signal processing comes from the fact that, once a sequence of numbers is available to appropriate digital hardware, we can carry out any form of numerical processing on it. For example, suppose we need to perform the following operation on a continuous-time signal:

$$y(t) = \frac{\cosh \left[\ln(|x(t)|) + x^3(t) + \cos^3 \left(\sqrt{|x(t)|} \right) \right]}{5x^5(t) + e^{x(t)} + \tan(x(t))}. \quad (1.1)$$

This would be clearly very difficult to implement using analog hardware. However, if we sample the analog signal $x(t)$ and convert it into a sequence of numbers $x(n)$, it can be input to a digital computer, which can perform the above operation easily and reliably, generating a sequence of numbers $y(n)$. If the continuous-time signal $y(t)$ can be recovered from $y(n)$, then the desired processing has been successfully performed.

This simple example highlights two important points. One is how powerful digital signal processing is. The other is that, if we want to process an analog signal using this sort of resource, we must have a way of converting a continuous-time signal into a discrete-time one, such that the continuous-time signal can be recovered from the discrete-time signal. However, it is important to note that very often discrete-time signals do not come from continuous-time signals, that is, they are originally discrete-time, and the results of their processing are only needed in digital form.

In this chapter, we study the basic concepts of the theory of discrete-time signals and systems. We emphasize the treatment of discrete-time systems as separate entities from

continuous-time systems. We first define discrete-time signals and, based on this, we define discrete-time systems, highlighting the properties of an important subset of these systems, namely linearity and time invariance, as well as their description by discrete-time convolutions. We then study the time-domain response of discrete-time systems by characterizing them using difference equations. We close the chapter with Nyquist's sampling theorem, which tells us how to generate, from a continuous-time signal, a discrete-time signal from which the continuous-time signal can be completely recovered. Nyquist's sampling theorem forms the basis of the digital processing of continuous-time signals.

1.2 Discrete-time signals

A discrete-time signal is one that can be represented by a sequence of numbers. For example, the sequence

$$\{x(n), n \in \mathbb{Z}\}, \quad (1.2)$$

where \mathbb{Z} is the set of integer numbers, can represent a discrete-time signal where each number $x(n)$ corresponds to the amplitude of the signal at an instant nT . If $x_a(t)$ is an analog signal, we have that

$$x(n) = x_a(nT), \quad n \in \mathbb{Z}. \quad (1.3)$$

Since n is an integer, T represents the interval between two consecutive points at which the signal is defined. It is important to note that T is not necessarily a time unit. For example, if $x_a(t)$ is the temperature along a metal rod, then if T is a length unit, $x(n) = x_a(nT)$ may represent the temperature at sensors placed uniformly along this rod.

In this text, we usually represent a discrete-time signal using the notation in Equation (1.2), where $x(n)$ is referred to as the n th sample of the signal (or the n th element of the sequence). An alternative notation, used in many texts, is to represent the signal as

$$\{x_a(nT), n \in \mathbb{Z}\}, \quad (1.4)$$

where the discrete-time signal is represented explicitly as samples of an analog signal $x_a(t)$. In this case, the time interval between samples is explicitly shown; that is, $x_a(nT)$ is the sample at time nT . Note that, using the notation in Equation (1.2), a discrete-time signal whose adjacent samples are 0.03 s apart would be represented as

$$\dots x(0), x(1), x(2), x(3), x(4), \dots, \quad (1.5)$$

whereas, using Equation (1.4) it would be represented as

$$\dots x_a(0), x_a(0.03), x_a(0.06), x_a(0.09), x_a(0.12), \dots \quad (1.6)$$

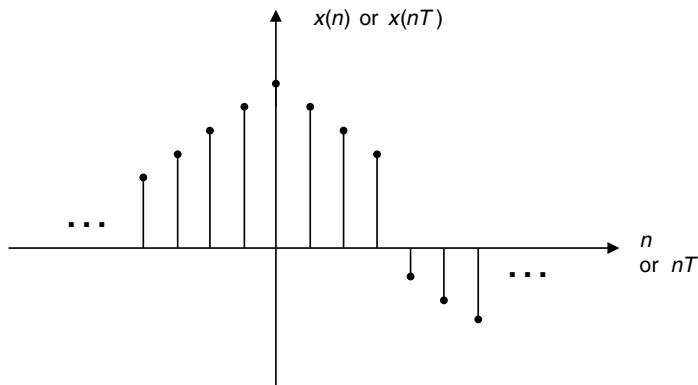


Fig. 1.1. General representation of a discrete-time signal.

The graphical representation of a general discrete-time signal is shown in Figure 1.1. In what follows, we describe some of the most important discrete-time signals.

Unit impulse (see Figure 1.2a):

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0. \end{cases} \quad (1.7)$$

Delayed unit impulse (see Figure 1.2b):

$$\delta(n - m) = \begin{cases} 1, & n = m \\ 0, & n \neq m. \end{cases} \quad (1.8)$$

Unit step (see Figure 1.2c):

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0. \end{cases} \quad (1.9)$$

Cosine function (see Figure 1.2d):

$$x(n) = \cos(\omega n). \quad (1.10)$$

The angular frequency of this sinusoid is ω rad/sample and its frequency is $\omega/2\pi$ cycles/sample. For example, in Figure 1.2d, the cosine function has angular frequency $\omega = 2\pi/16$ rad/sample. This means that it completes one cycle, that equals 2π radians, in 16 samples. If the sample separation represents time, then ω can be given in rad/(time unit). It is important to note that

$$\cos[(\omega + 2k\pi)n] = \cos(\omega n + 2kn\pi) = \cos(\omega n) \quad (1.11)$$

for $k \in \mathbb{Z}$. This implies that, in the case of discrete signals, there is an ambiguity in defining the frequency of a sinusoid. In other words, when referring to discrete sinusoids, ω and $\omega + 2k\pi$, $k \in \mathbb{Z}$, are the same frequency.

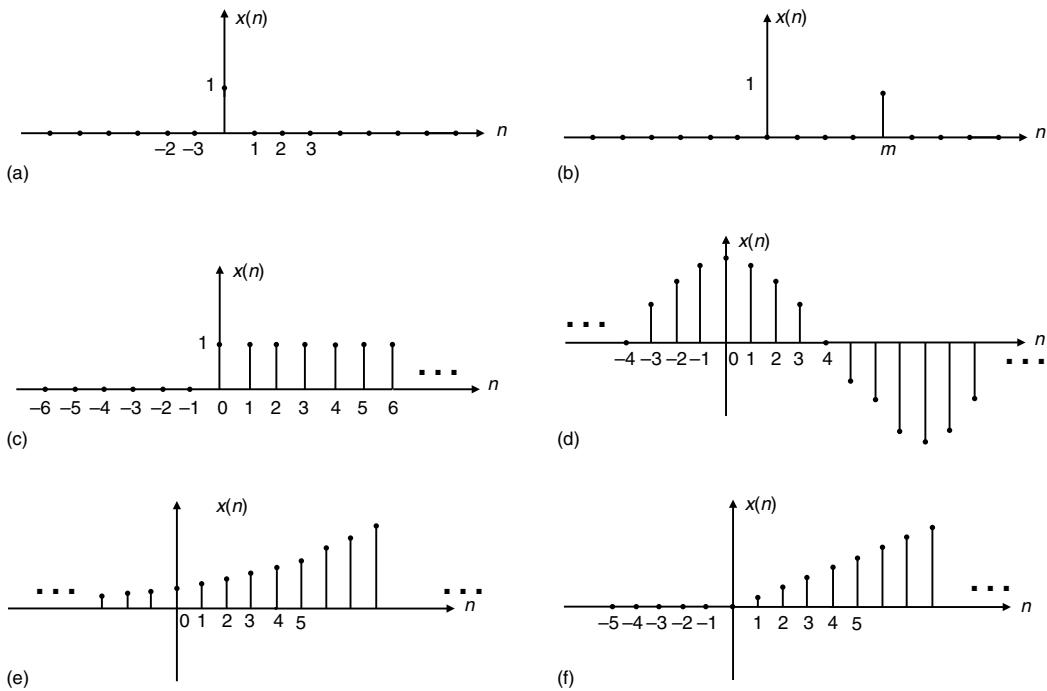


Fig. 1.2. Basic discrete-time functions: (a) unit impulse; (b) delayed unit impulse; (c) unit step; (d) cosine function with $\omega = 2\pi/16$ rad/sample; (e) real exponential function with $a = 0.2$; (f) unit ramp.

Real exponential function (see Figure 1.2e):

$$x(n) = e^{an}. \quad (1.12)$$

Unit ramp (see Figure 1.2f):

$$r(n) = \begin{cases} n, & n \geq 0 \\ 0, & n < 0 \end{cases} \quad (1.13)$$

By examining Figure 1.2b–f, we notice that any discrete-time signal is equivalent to a sum of shifted impulses multiplied by a constant; that is, the impulse shifted by k samples is multiplied by $x(k)$. This can also be deduced from the definition of a shifted impulse in Equation (1.8). For example, the unit step $u(n)$ in Equation (1.9) can also be expressed as

$$u(n) = \sum_{k=0}^{\infty} \delta(n - k). \quad (1.14)$$

Likewise, any discrete-time signal $x(n)$ can be expressed as

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n - k). \quad (1.15)$$

An important class of discrete-time signals or sequences is that of periodic sequences. A sequence $x(n)$ is periodic if and only if there is an integer $N \neq 0$ such that $x(n) = x(n + N)$ for all n . In such a case, N is called a period of the sequence. Note that, using this definition and referring to Equation (1.10), a period of the cosine function is an integer N such that

$$\cos(\omega n) = \cos[\omega(n + N)], \quad \text{for all } n \in \mathbb{Z}. \quad (1.16)$$

This happens only if there is $k \in \mathbb{N}$ such that $\omega N = 2\pi k$. The smallest period is then

$$N = \min_{\substack{k \in \mathbb{N} \\ (2\pi/\omega)k \in \mathbb{N}}} \left\{ \frac{2\pi}{\omega} k \right\}. \quad (1.17)$$

Therefore, we notice that not all discrete cosine sequences are periodic, as illustrated in Example 1.1. An example of a periodic cosine sequence with period $N = 16$ samples is given in Figure 1.2d.

Example 1.1. Determine whether the discrete signals above are periodic; if they are, determine their periods.

- (a) $x(n) = \cos[(12\pi/5)n]$
- (b) $x(n) = 10 \sin^2[(7\pi/12)n + \sqrt{2}]$
- (c) $x(n) = 2 \cos(0.02n + 3)$.

Solution

- (a) In this case, we must have

$$\frac{12\pi}{5}(n + N) = \frac{12\pi}{5}n + 2k\pi \quad \Rightarrow \quad N = \frac{5k}{6}. \quad (1.18)$$

This implies that the smallest N results for $k = 6$. Then the sequence is periodic with period $N = 5$. Note that in this case

$$\cos\left(\frac{12\pi}{5}n\right) = \cos\left(\frac{2\pi}{5}n + 2\pi n\right) = \cos\left(\frac{2\pi}{5}n\right) \quad (1.19)$$

and thus we have also that the frequency of this sinusoid, besides being $\omega = 12\pi/5$, is also $\omega = 2\pi/5$, as indicated by Equation (1.11).

- (b) In this case, periodicity implies that

$$\sin^2\left[\frac{7\pi}{12}(n + N) + \sqrt{2}\right] = \sin^2\left(\frac{7\pi}{12}n + \sqrt{2}\right) \quad (1.20)$$

and then

$$\sin\left[\frac{7\pi}{12}(n + N) + \sqrt{2}\right] = \pm \sin\left(\frac{7\pi}{12}n + \sqrt{2}\right) \quad (1.21)$$

such that

$$\frac{7\pi}{12}(n+N) = \frac{7\pi}{12}n + k\pi \Rightarrow N = \frac{12k}{7}. \quad (1.22)$$

The smallest N results for $k = 7$. Then this discrete-time signal is periodic with period $N = 12$.

- (c) The periodicity condition requires that

$$\cos[0.02(n+N) + 3] = \cos(0.02n + 3) \quad (1.23)$$

such that

$$0.02(n+N) = 0.02n + 2k\pi \Rightarrow N = 100k\pi. \quad (1.24)$$

Since no integer N satisfies the above equation, the sequence is not periodic.

△

1.3 Discrete-time systems

A discrete-time system maps an input sequence $x(n)$ to an output sequence $y(n)$, such that

$$y(n) = \mathcal{H}\{x(n)\}, \quad (1.25)$$

where the operator $\mathcal{H}\{\cdot\}$ represents a discrete-time system, as shown in Figure 1.3. Depending on the properties of $\mathcal{H}\{\cdot\}$, the discrete-time system can be classified in several ways, the most basic ones being either linear or nonlinear, either time invariant or time variant, and causal or noncausal. These classifications will be discussed in what follows.

1.3.1 Linearity

Let us suppose that there is a system that accepts as input a voice signal and outputs the voice signal modified such that its acute components (high frequencies) are enhanced. In such a system, it would be undesirable if, in the case that one increased the voice tone at the input, the output became distorted instead of enhanced. Actually, one tends to expect

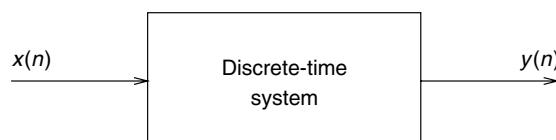


Fig. 1.3.

Representation of a discrete-time system.

that if one speaks twice as loud at the input, then the output will be just twice as loud, with its acute components enhanced in the same way. Likewise, if two people speak at the same time, one would expect the system to enhance the high frequencies of both voices, each one in the same way as if they were individually input to the system. A system with such a behavior is what is referred to as a linear system. Such systems, besides being useful in many practical applications, have very nice mathematical properties. This fact makes linear systems an important class of discrete-time systems; therefore, they constitute the primary subject of this book.

In more precise terms, a discrete-time system is linear if and only if

$$\mathcal{H}\{ax(n)\} = a\mathcal{H}\{x(n)\} \quad (1.26)$$

and

$$\mathcal{H}\{x_1(n) + x_2(n)\} = \mathcal{H}\{x_1(n)\} + \mathcal{H}\{x_2(n)\} \quad (1.27)$$

for any constant a and any sequences $x(n)$, $x_1(n)$, and $x_2(n)$.

1.3.2 Time invariance

Sometimes it is desirable to have a system whose properties do not vary in time. In other words, one wants its input–output behavior to be the same irrespective of the time instant that the input is applied to the system. Such a system is referred to as a time-invariant system. As will be seen later, when combined with linearity, time invariance gives rise to an important family of systems.

In more precise terms, a discrete-time system is time invariant if and only if, for any input sequence $x(n)$ and integer n_0 , given that

$$\mathcal{H}\{x(n)\} = y(n) \quad (1.28)$$

then

$$\mathcal{H}\{x(n - n_0)\} = y(n - n_0). \quad (1.29)$$

Some texts refer to the time-invariance property as the shift-invariance property, since a discrete system can process samples of a function not necessarily in time, as emphasized before.

1.3.3 Causality

One of the main limitations of the time domain is that time always flows from past to present and, therefore, one cannot know the future. Although this statement might seem a bit too

philosophical, this concept has a strong influence on the way discrete-time systems can be used in practice. This is so because when processing a signal in time one cannot use future values of the input to compute the output at a given time. This leads to the definition of a causal system, that is a system that cannot “see into the future.”

In more precise terms, a discrete-time system is causal if and only if, when $x_1(n) = x_2(n)$ for $n < n_0$, then

$$\mathcal{H}\{x_1(n)\} = \mathcal{H}\{x_2(n)\}, \text{ for } n < n_0. \quad (1.30)$$

In other words, causality means that the output of a system at instant n does not depend on any input occurring after n .

It is important to note that, usually, in the case of a discrete-time signal, a noncausal system is not implementable in real time. This is because we would need input samples at instants of time greater than n in order to compute the output at time n . This would be allowed only if the time samples were pre-stored, as in off-line or batch implementations. It is important to note that if the signals to be processed do not consist of time samples acquired in real time, then there might be nothing equivalent to the concepts of past or future samples. Therefore, in these cases, the role of causality is of a lesser importance. For example, in Section 1.2 we mentioned a discrete signal that corresponds to the temperature at sensors uniformly spaced along a metal rod. For this discrete signal, a processor can have access to all its samples simultaneously. Therefore, in this case, even a noncausal system can be easily implemented.

Example 1.2. Characterize the following systems as being either linear or nonlinear, either time invariant or time varying, and causal or noncausal:

- (a) $y(n) = (n + b)x(n - 4)$
- (b) $y(n) = x^2(n + 1)$.

Solution

- (a) • Linearity:

$$\begin{aligned} \mathcal{H}\{ax(n)\} &= (n + b)ax(n - 4) \\ &= a(n + b)x(n - 4) \\ &= a\mathcal{H}\{x(n)\} \end{aligned} \quad (1.31)$$

and

$$\begin{aligned} \mathcal{H}\{x_1(n) + x_2(n)\} &= (n + b)[x_1(n - 4) + x_2(n - 4)] \\ &= (n + b)x_1(n - 4) + (n + b)x_2(n - 4) \\ &= \mathcal{H}\{x_1(n)\} + \mathcal{H}\{x_2(n)\}; \end{aligned} \quad (1.32)$$

therefore, the system is linear.

- Time invariance:

$$y(n - n_0) = [(n - n_0) + b]x[(n - n_0) - 4] \quad (1.33)$$

and then

$$\mathcal{H}\{x(n - n_0)\} = (n + b)x[(n - n_0) - 4] \quad (1.34)$$

such that $y(n - n_0) \neq \mathcal{H}\{x(n - n_0)\}$, and the system is time varying.

- Causality: if

$$x_1(n) = x_2(n), \quad \text{for } n < n_0 \quad (1.35)$$

then

$$x_1(n - 4) = x_2(n - 4), \quad \text{for } n - 4 < n_0 \quad (1.36)$$

such that

$$x_1(n - 4) = x_2(n - 4), \quad \text{for } n < n_0 \quad (1.37)$$

and then

$$(n + b)x_1(n - 4) = (n + b)x_2(n - 4), \quad \text{for } n < n_0. \quad (1.38)$$

Hence, $\mathcal{H}\{x_1(n)\} = \mathcal{H}\{x_2(n)\}$ for all $n < n_0$ and, consequently, the system is causal.

- (b) • Linearity:

$$\mathcal{H}\{ax(n)\} = a^2x^2(n + 1) \neq a\mathcal{H}\{x(n)\}; \quad (1.39)$$

therefore, the system is nonlinear.

- Time invariance:

$$\mathcal{H}\{x(n - n_0)\} = x^2[(n - n_0) + 1] = y(n - n_0), \quad (1.40)$$

so the system is time invariant.

- Causality:

$$\mathcal{H}\{x_1(n)\} = x_1^2(n + 1) \quad (1.41)$$

$$\mathcal{H}\{x_2(n)\} = x_2^2(n + 1). \quad (1.42)$$

Therefore, if $x_1(n) = x_2(n)$, for $n < n_0$, and $x_1(n_0) \neq x_2(n_0)$, then, for $n = n_0 - 1 < n_0$,

$$\mathcal{H}\{x_1(n_0 - 1)\} = x_1^2(n_0) \quad (1.43)$$

$$\mathcal{H}\{x_2(n_0 - 1)\} = x_2^2(n_0) \quad (1.44)$$

and we have that $\mathcal{H}\{x_1(n)\} \neq \mathcal{H}\{x_2(n)\}$ and the system is noncausal. \triangle

1.3.4 Impulse response and convolution sums

Suppose that $\mathcal{H}\{\cdot\}$ is a linear system and we apply an excitation $x(n)$ to the system. Since, from Equation (1.15), $x(n)$ can be expressed as a sum of shifted impulses

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k), \quad (1.45)$$

we can express its output as

$$y(n) = \mathcal{H} \left\{ \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \right\} = \sum_{k=-\infty}^{\infty} \mathcal{H}\{x(k)\delta(n-k)\}. \quad (1.46)$$

Since $x(k)$ in the above equation is just a constant, the linearity of $\mathcal{H}\{\cdot\}$ also implies that

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\mathcal{H}\{\delta(n-k)\} = \sum_{k=-\infty}^{\infty} x(k)h_k(n), \quad (1.47)$$

where $h_k(n) = \mathcal{H}\{\delta(n-k)\}$ is the response of the system to an impulse at $n = k$.

If the system is also time invariant, and we define

$$\mathcal{H}\{\delta(n)\} = h_0(n) = h(n), \quad (1.48)$$

then $\mathcal{H}\{\delta(n-k)\} = h(n-k)$, and the expression in Equation (1.47) becomes

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k), \quad (1.49)$$

indicating that a linear time-invariant system is completely characterized by its unit impulse response $h(n)$. This is a very powerful and convenient result, which will be explored further, and lends great importance and usefulness to the class of linear time-invariant discrete-time systems. One should note that, when the system is linear and time varying, we would need, in order to compute $y(n)$, the values of $h_k(n)$, which depend on both n and k . This makes the computation of the summation in Equation (1.47) quite complex.

Equation (1.49) is called a convolution sum or a discrete-time convolution.¹ If we make the change of variables $l = n - k$, Equation (1.49) can be written as

$$y(n) = \sum_{l=-\infty}^{\infty} x(n-l)h(l); \quad (1.50)$$

¹ This operation is often also referred to as a discrete-time linear convolution in order to differentiate it from the discrete-time circular convolution, which will be defined in Chapter 3.

that is, we can interpret $y(n)$ as the result of the convolution of the excitation $x(n)$ with the system impulse response $h(n)$. A shorthand notation for the convolution operation, as described in Equations (1.49) and (1.50), is

$$y(n) = x(n) * h(n) = h(n) * x(n). \quad (1.51)$$

Suppose now that the output $y(n)$ of a system with impulse response $h(n)$ is the excitation for a system with impulse response $h'(n)$. In this case we have

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (1.52)$$

$$y'(n) = \sum_{l=-\infty}^{\infty} y(l)h'(n-l). \quad (1.53)$$

Substituting Equation (1.52) in Equation (1.53), we have that

$$\begin{aligned} y'(n) &= \sum_{l=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x(k)h(l-k) \right) h'(n-l) \\ &= \sum_{k=-\infty}^{\infty} x(k) \left(\sum_{l=-\infty}^{\infty} h(l-k)h'(n-l) \right). \end{aligned} \quad (1.54)$$

By performing the change of variables $l = n - r$, the above equation becomes

$$\begin{aligned} y'(n) &= \sum_{k=-\infty}^{\infty} x(k) \left(\sum_{r=-\infty}^{\infty} h(n-r-k)h'(r) \right) \\ &= \sum_{k=-\infty}^{\infty} x(k) (h(n-k) * h'(n-k)) \\ &= \sum_{k=-\infty}^{\infty} x(n-k) (h(k) * h'(k)), \end{aligned} \quad (1.55)$$

showing that the impulse response of a linear time-invariant system formed by the series (cascade) connection of two linear time-invariant subsystems is the convolution of the impulse responses of the two subsystems.

Example 1.3. Compute $y(n)$ for the system depicted in Figure 1.4, as a function of the input signal and of the impulse responses of the subsystems.

Solution

From the previous results, it is easy to conclude that

$$y(n) = (h_2(n) + h_3(n)) * h_1(n) * x(n). \quad (1.56)$$

△

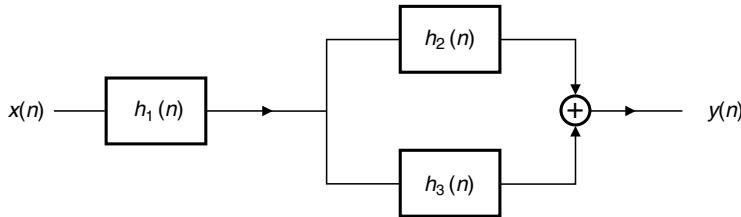


Fig. 1.4.

Linear time-invariant system composed of the connection of three subsystems.

1.3.5 Stability

A system is referred to as bounded-input bounded-output (BIBO) stable if, for every input limited in amplitude, the output signal is also limited in amplitude. For a linear time-invariant system, Equation (1.50) implies that

$$|y(n)| \leq \sum_{k=-\infty}^{\infty} |x(n-k)| |h(k)|. \quad (1.57)$$

The input being limited in amplitude is equivalent to

$$|x(n)| \leq x_{\max} < \infty, \quad \text{for all } n. \quad (1.58)$$

Therefore:

$$|y(n)| \leq x_{\max} \sum_{k=-\infty}^{\infty} |h(k)|. \quad (1.59)$$

Hence, we can conclude that a sufficient condition for a system to be BIBO stable is

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty, \quad (1.60)$$

since this condition forces $y(n)$ to be limited. To prove that this condition is also necessary, suppose that it does not hold; that is, that the summation in Equation (1.60) is infinite. If we choose an input such that

$$x(n_0 - k) = \begin{cases} +1, & \text{for } h(k) \geq 0 \\ -1, & \text{for } h(k) < 0, \end{cases} \quad (1.61)$$

we then have that

$$y(n_0) = |y(n_0)| = \sum_{k=-\infty}^{\infty} |h(k)|; \quad (1.62)$$

that is, the output $y(n)$ is unbounded, thus completing the necessity proof.

We can then conclude that the necessary and sufficient condition for a linear time-invariant system to be BIBO stable is given in Equation (1.60).

1.4 Difference equations and time-domain response

In most applications, discrete-time systems can be described by difference equations, which are the equivalent, for the discrete-time domain, to differential equations for the continuous-time domain. In fact, the systems that can be specified by difference equations are powerful enough to cover most practical applications. The input and output of a system described by a linear difference equation are generally related by (Gabel & Roberts, 1980)

$$\sum_{i=0}^N a_i y(n-i) - \sum_{l=0}^M b_l x(n-l) = 0. \quad (1.63)$$

This difference equation has an infinite number of solutions $y(n)$, like the solutions of differential equations in the continuous case. For example, suppose that a particular $y_p(n)$ satisfies Equation (1.63), that is

$$\sum_{i=0}^N a_i y_p(n-i) - \sum_{l=0}^M b_l x(n-l) = 0, \quad (1.64)$$

and that $y_h(n)$ is a solution to the homogeneous equation, that is

$$\sum_{i=0}^N a_i y_h(n-i) = 0. \quad (1.65)$$

Then, from Equations (1.63)–(1.65), we can easily infer that $y(n) = y_p(n) + y_h(n)$ is also a solution to the same difference equation.

The homogeneous solution $y_h(n)$ of a difference equation of order N , as in Equation (1.63), has N degrees of freedom (depends on N arbitrary constants). Therefore, one can only determine a solution for a difference equation if one supplies N auxiliary conditions. One example of a set of auxiliary conditions is given by the values of $y(-1), y(-2), \dots, y(-N)$. It is important to note that any N independent auxiliary conditions would be enough to solve a difference equation. In general, however, one commonly uses N consecutive samples of $y(n)$ as auxiliary conditions.

Example 1.4. Find the solution for the difference equation

$$y(n) = ay(n-1) \quad (1.66)$$

as a function of the initial condition $y(0)$.

Solution

Running the difference equation from $n = 1$ onwards, we have that

$$\left. \begin{array}{l} y(1) = ay(0) \\ y(2) = ay(1) \\ y(3) = ay(2) \\ \vdots \\ y(n) = ay(n-1) \end{array} \right\}. \quad (1.67)$$

Multiplying the above equations, we have that

$$y(1)y(2)y(3)\dots y(n) = a^n y(0)y(1)y(2)\dots y(n-1); \quad (1.68)$$

therefore, the solution of the difference equation is

$$y(n) = a^n y(0). \quad (1.69)$$

△

Example 1.5. Solve the following difference equation:

$$y(n) = e^{-\beta} y(n-1) + \delta(n). \quad (1.70)$$

Solution

By making $a = e^{-\beta}$ and $y(0) = K$ in Example 1.4, we can deduce that any function of the form $y_h(n) = K e^{-\beta n}$ satisfies

$$y_h(n) = e^{-\beta} y_h(n-1) \quad (1.71)$$

and is therefore a solution of the homogeneous difference equation. Also, one can verify by substitution that $y_p(n) = e^{-\beta n} u(n)$ is a particular solution of Equation (1.70).

Therefore, the general solution of the difference equation is given by

$$y(n) = y_p(n) + y_h(n) = e^{-\beta n} u(n) + K e^{-\beta n}, \quad (1.72)$$

where the value of K is determined by the auxiliary conditions. Since this difference equation is of first order, we need to specify only one condition. For example, if we have that $y(-1) = \alpha$, the solution to Equation (1.70) becomes

$$y(n) = e^{-\beta n} u(n) + \alpha e^{-\beta(n+1)}. \quad (1.73)$$

△

Since a linear system must satisfy equation (1.26), it is clear that $\mathcal{H}\{0\} = 0$ for a linear system; that is, the output for a zero input must be zero. If we restrict ourselves to inputs that are null prior to a certain sample (that is, $x(n) = 0$, for $n < n_0$), then there is an interesting

relation between linearity, causality, and the initial conditions of a system. If the system is causal, then the output at $n < n_0$ cannot be influenced by any sample of the input $x(n)$ for $n \geq n_0$. Therefore, if $x(n) = 0$, for $n < n_0$, then $\mathcal{H}\{0\}$ and $\mathcal{H}\{x(n)\}$ must be identical for all $n < n_0$. Since, if the system is linear, $\mathcal{H}\{0\} = 0$, then necessarily $\mathcal{H}\{x(n)\} = 0$ for $n < n_0$. This is equivalent to saying that the auxiliary conditions for $n < n_0$ must be null. Such a system is referred to as being initially relaxed. Conversely, if the system is not initially relaxed, one cannot guarantee that it is causal. This will be made clearer in Example 1.6.

Example 1.6. For the linear system described by

$$y(n) = e^{-\beta} y(n-1) + u(n), \quad (1.74)$$

determine its output for the auxiliary conditions

- (a) $y(1) = 0$
- (b) $y(-1) = 0$

and discuss the causality in both situations.

Solution

The homogeneous solution of Equation (1.74) is the same as in Example 1.5; that is

$$y_h(n) = K e^{-\beta n}. \quad (1.75)$$

By direct substitution in Equation (1.74), it can be verified that the particular solution is of the form (see Section 1.5 for a method to determine such solutions)

$$y_p(n) = (a + b e^{-\beta n}) u(n), \quad (1.76)$$

where

$$a = \frac{1}{1 - e^{-\beta}} \quad \text{and} \quad b = \frac{-e^{-\beta}}{1 - e^{-\beta}}. \quad (1.77)$$

Thus, the general solution of the difference equation is given by

$$y(n) = \left[\frac{1 - e^{-\beta(n+1)}}{1 - e^{-\beta}} \right] u(n) + K e^{-\beta n}. \quad (1.78)$$

- (a) For the auxiliary condition $y(1) = 0$, we have that

$$y(1) = \frac{1 - e^{-2\beta}}{1 - e^{-\beta}} + K e^{-\beta} = 0, \quad (1.79)$$

yielding $K = -(1 + e^\beta)$, and the general solution becomes

$$y(n) = \left[\frac{1 - e^{-\beta(n+1)}}{1 - e^{-\beta}} \right] u(n) - \left[e^{-\beta n} + e^{-\beta(n-1)} \right]. \quad (1.80)$$

Since for $n < 0$ we have that $u(n) = 0$, then $y(n)$ simplifies to

$$y(n) = -[e^{-\beta n} + e^{-\beta(n-1)}]. \quad (1.81)$$

Clearly, in this case, $y(n) \neq 0$, for $n < 0$, whereas the input $u(n) = 0$ for $n < 0$. Thus, the system is not initially relaxed and therefore is noncausal.

Another way of verifying that the system is noncausal is by noting that if the input is doubled, becoming $x(n) = 2u(n)$ instead of $u(n)$, then the particular solution is also doubled. Hence, the general solution of the difference equation becomes

$$y(n) = \left[\frac{2 - 2e^{-\beta(n+1)}}{1 - e^{-\beta}} \right] u(n) + Ke^{-\beta n}. \quad (1.82)$$

If we require that $y(1) = 0$, then $K = 2 + 2e^{\beta}$, and, for $n < 0$, this yields

$$y(n) = -2[e^{-\beta n} + e^{-\beta(n-1)}]. \quad (1.83)$$

Since this is different from the value of $y(n)$ for $u(n)$ as input, we see that the output for $n < 0$ depends on the input for $n > 0$; therefore, the system is noncausal.

- (b) For the auxiliary condition $y(-1) = 0$, we have that $K = 0$, yielding the solution

$$y(n) = \left[\frac{1 - e^{-\beta(n+1)}}{1 - e^{-\beta}} \right] u(n). \quad (1.84)$$

In this case, $y(n) = 0$, for $n < 0$; that is, the system is initially relaxed and, therefore, causal, as discussed above. \triangle

Example 1.6 shows that the system described by the difference equation is noncausal because it has nonzero auxiliary conditions prior to the application of the input to the system. To guarantee both causality and linearity for the solution of a difference equation, we have to impose zero auxiliary conditions for the samples preceding the application of the excitation to the system. This is the same as assuming that the system is initially relaxed. Therefore, an initially relaxed system described by a difference equation of the form in Equation (1.63) has the highly desirable linearity, time invariance, and causality properties. In this case, time invariance can be easily inferred if we consider that, for an initially relaxed system, the history of the system up to the application of the excitation is the same irrespective of the time sample position at which the excitation is applied. This happens because the outputs are all zero up to, but not including, the time of the application of the excitation. Therefore, if time is measured having as a reference the time sample $n = n_0$, at which the input is applied, then the output will not depend on the reference n_0 , because the history of the system prior to n_0 is the same irrespective of n_0 . This is equivalent to saying that if the input is shifted by k samples, then the output is just shifted by k samples, the rest remaining unchanged, thus characterizing a time-invariant system.

1.4.1 Recursive \times nonrecursive systems

Equation (1.63) can be rewritten, without loss of generality, considering that $a_0 = 1$, yielding

$$y(n) = -\sum_{i=1}^N a_i y(n-i) + \sum_{l=0}^M b_l x(n-l). \quad (1.85)$$

This equation can be interpreted as the output signal $y(n)$ being dependent both on samples of the input, $x(n), x(n-1), \dots, x(n-M)$, and on previous samples of the output $y(n-1), y(n-2), \dots, y(n-N)$. Then, in this general case, we say that the system is recursive, since, in order to compute the output, we need past samples of the output itself. When $a_1 = a_2 = \dots = a_N = 0$, then the output at sample n depends only on values of the input signal. In such a case, the system is called nonrecursive, being distinctively characterized by a difference equation of the form

$$y(n) = \sum_{l=0}^M b_l x(n-l). \quad (1.86)$$

If we compare Equation (1.86) with the expression for the convolution sum given in Equation (1.50), we see that Equation (1.86) corresponds to a discrete system with impulse response $h(l) = b_l$. Since b_l is defined only for l between 0 and M , we can say that $h(l)$ is nonzero only for $0 \leq l \leq M$. This implies that the system in Equation (1.86) has a finite-duration impulse response. Such discrete-time systems are often referred to as finite-duration impulse-response (FIR) filters.

In contrast, when $y(n)$ depends on its past values, as in Equation (1.85), we have that the impulse response of the discrete system, in general, might not be zero when $n \rightarrow \infty$. Therefore, recursive digital systems are often referred to as infinite-duration impulse-response (IIR) filters.

Example 1.7. Find the impulse response of the system

$$y(n) - \frac{1}{\alpha} y(n-1) = x(n) \quad (1.87)$$

supposing that it is initially relaxed.

Solution

Since the system is initially relaxed, then $y(n) = 0$, for $n \leq -1$. Hence, for $n = 0$, we have that

$$y(0) = \frac{1}{\alpha} y(-1) + \delta(0) = \delta(0) = 1. \quad (1.88)$$

For $n > 0$, we have that

$$y(n) = \frac{1}{\alpha} y(n-1) \quad (1.89)$$

and, therefore, $y(n)$ can be expressed as

$$y(n) = \left(\frac{1}{\alpha}\right)^n u(n). \quad (1.90)$$

Note that $y(n) \neq 0$ for all $n \geq 0$; that is, the impulse response has infinite length. \triangle

One should note that, in general, recursive systems have IIRs, although there are some cases when recursive systems have FIRs. Illustrations of such cases can be found in Example 1.11, Exercise 1.16, and Section 12.5.

1.5 Solving difference equations

Consider the following homogeneous difference equation:

$$\sum_{i=0}^N a_i y(n-i) = 0. \quad (1.91)$$

We start by deriving an important property of it. Let $y_1(n)$ and $y_2(n)$ be solutions to Equation (1.91). Then

$$\sum_{i=0}^N a_i y_1(n-i) = 0 \quad (1.92)$$

$$\sum_{i=0}^N a_i y_2(n-i) = 0. \quad (1.93)$$

Adding Equation (1.92) multiplied by c_1 to Equation (1.93) multiplied by c_2 we have that

$$\begin{aligned} c_1 \sum_{i=0}^N a_i y_1(n-i) + c_2 \sum_{i=0}^N a_i y_2(n-i) &= 0 \\ \Rightarrow \sum_{i=0}^N a_i c_1 y_1(n-i) + \sum_{i=0}^N a_i c_2 y_2(n-i) &= 0 \\ \Rightarrow \sum_{i=0}^N a_i (c_1 y_1(n-i) + c_2 y_2(n-i)) &= 0. \end{aligned} \quad (1.94)$$

Equation (1.94) means that $(c_1 y_1(n) + c_2 y_2(n))$ is also a solution to Equation (1.91). This implies that, if $y_i(n)$, for $i = 0, 1, \dots, (M-1)$, are solutions of an homogeneous difference

equation, then

$$y_h(n) = \sum_{i=0}^{M-1} c_i y_i(n) \quad (1.95)$$

is also a solution.

As we have seen in Example 1.4, a difference equation may have solutions of the form

$$y(n) = K\rho^n. \quad (1.96)$$

Supposing that $y(n)$ from Equation (1.96) is also a solution to the difference equation (1.91), we have that

$$\sum_{i=0}^N a_i K \rho^{n-i} = 0. \quad (1.97)$$

If we disregard the trivial solution $\rho = 0$ and divide the left-hand side of Equation (1.97) by $K\rho^n$, we get

$$\sum_{i=0}^N a_i \rho^{-i} = 0, \quad (1.98)$$

which has the same solutions as the following polynomial equation:

$$\sum_{i=0}^N a_i \rho^{N-i} = 0. \quad (1.99)$$

As a result, one can conclude that if $\rho_0, \rho_1, \dots, \rho_{M-1}$, for $M \leq N$, are distinct zeros of the so-called characteristic polynomial in Equation (1.99), then there are M solutions for the homogeneous difference equation given by

$$y(n) = c_k \rho_k^n, \quad k = 0, 1, \dots, (M-1). \quad (1.100)$$

In fact, from Equation (1.95), we have that any linear combination of these solutions is also a solution for the homogeneous difference equation. Then, a homogeneous solution can be written as

$$y_h(n) = \sum_{k=0}^{M-1} c_k \rho_k^n, \quad (1.101)$$

where c_k , for $k = 0, 1, \dots, (M-1)$, are arbitrary constants.

Example 1.8. Find the general solution for the Fibonacci equation

$$y(n) = y(n-1) + y(n-2) \quad (1.102)$$

with $y(0) = 0$ and $y(1) = 1$.

Solution

The characteristic polynomial to the Fibonacci equation is

$$\rho^2 - \rho - 1 = 0 \quad (1.103)$$

whose roots are $\rho = (1 \pm \sqrt{5})/2$, leading to the general solution

$$y(n) = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^n + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^n. \quad (1.104)$$

Applying the auxiliary conditions $y(0) = 0$ and $y(1) = 1$ to Equation (1.104), we have that

$$\begin{cases} y(0) = c_1 + c_2 = 0 \\ y(1) = \left(\frac{1 + \sqrt{5}}{2} \right) c_1 + \left(\frac{1 - \sqrt{5}}{2} \right) c_2 = 1. \end{cases} \quad (1.105)$$

Thus, $c_1 = 1/\sqrt{5}$ and $c_2 = -1/\sqrt{5}$, and the solution to the Fibonacci equation becomes

$$y(n) = \frac{1}{\sqrt{5}} \left[\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right]. \quad (1.106)$$

△

If the characteristic polynomial in Equation (1.99) has a pair of complex conjugate roots ρ and ρ^* of the form $a \pm jb = r e^{\pm j\phi}$, the associated homogeneous solution is given by

$$\begin{aligned} y_h(n) &= \hat{c}_1 (r e^{j\phi})^n + \hat{c}_2 (r e^{-j\phi})^n \\ &= r^n (\hat{c}_1 e^{j\phi n} + \hat{c}_2 e^{-j\phi n}) \\ &= r^n [(\hat{c}_1 + \hat{c}_2) \cos(\phi n) + j(\hat{c}_1 - \hat{c}_2) \sin(\phi n)] \\ &= c_1 r^n \cos(\phi n) + c_2 r^n \sin(\phi n). \end{aligned} \quad (1.107)$$

If the characteristic polynomial in Equation (1.99) has multiple roots, solutions distinct from Equation (1.100) are required. For example, if ρ is a double root, then there also exists a solution of the form

$$y_h(n) = cn\rho^n, \quad (1.108)$$

where c is an arbitrary constant. In general, if ρ is a root of multiplicity m , then the associated solution is of the form (Gabel & Roberts, 1980)

$$y_h(n) = \sum_{l=0}^{m-1} d_l n^l \rho^n, \quad (1.109)$$

where d_l , for $l = 0, 1, \dots, (m-1)$, are arbitrary constants.

Table 1.1.

Typical homogeneous solutions.

Root type [multiplicity]	Homogeneous solution $y_h(n)$
Real ρ_k [1]	$c_k \rho_k^n$
Real ρ_k [m_k]	$\sum_{l=0}^{m_k-1} d_l n^l \rho_k^n$
Complex conjugates $\rho_k, \rho_k^* = r e^{\pm j\phi}$ [1]	$r^n [c_1 \cos(\phi n) + c_2 \sin(\phi n)]$
Complex conjugates $\rho_k, \rho_k^* = r e^{\pm j\phi}$ [m_k]	$\sum_{l=0}^{m_k-1} [d_{1,l} n^l r^n \cos(\phi n) + d_{2,l} n^l r^n \sin(\phi n)]$

From the above, we can conclude that the homogeneous solutions of difference equations for each root type of the characteristic polynomial follow the rules summarized in Table 1.1.

A widely used method to find a particular solution for a difference equation of the form

$$\sum_{i=0}^N a_i y_p(n-i) = \sum_{l=0}^M b_l x(n-l) \quad (1.110)$$

is the so-called method of undetermined coefficients. This method can be used when the input sequence is the solution of a difference equation with constant coefficients. In order to do so, we define a delay operator $D\{\cdot\}$ as follows:

$$D^{-i}\{y(n)\} = y(n-i). \quad (1.111)$$

Such a delay operator is linear, since

$$\begin{aligned} D^{-i}\{c_1 y_1(n) + c_2 y_2(n)\} &= c_1 y_1(n-i) + c_2 y_2(n-i) \\ &= c_1 D^{-i}\{y_1(n)\} + c_2 D^{-i}\{y_2(n)\}. \end{aligned} \quad (1.112)$$

Also, the cascade of delay operators satisfies the following:

$$D^{-i}\{D^{-j}\{y(n)\}\} = D^{-i}\{y(n-j)\} = y(n-i-j) = D^{-(i+j)}\{y(n)\}. \quad (1.113)$$

Using delay operators, Equation (1.110) can be rewritten as

$$\left(\sum_{i=0}^N a_i D^{-i} \right) \{y_p(n)\} = \left(\sum_{l=0}^M b_l D^{-l} \right) \{x(n)\}. \quad (1.114)$$

The key idea is to find a difference operator $Q(D)$ of the form

$$Q(D) = \sum_{k=0}^R d_k D^{-k} = \prod_{r=0}^R (1 - \alpha_r D^{-1}) \quad (1.115)$$

Table 1.2.

Annihilator polynomials for different input signals.

Input $x(n)$	Polynomial $Q(D)$
s^n	$1 - sD^{-1}$
n^i	$(1 - D^{-1})^{i+1}$
$n^i s^n$	$(1 - sD^{-1})^{i+1}$
$\cos(\omega n)$ or $\sin(\omega n)$	$(1 - e^{j\omega D^{-1}})(1 - e^{-j\omega D^{-1}})$
$s^n \cos(\omega n)$ or $s^n \sin(\omega n)$	$(1 - s e^{j\omega D^{-1}})(1 - s e^{-j\omega D^{-1}})$
$n \cos(\omega n)$ or $n \sin(\omega n)$	$[(1 - e^{j\omega D^{-1}})(1 - e^{-j\omega D^{-1}})]^2$

such that it annihilates the excitation; that is:

$$Q(D)\{x(n)\} = 0. \quad (1.116)$$

Applying $Q(D)$ to Equation (1.114) we get

$$\begin{aligned} Q(D) \left\{ \left(\sum_{i=0}^N a_i D^{-i} \right) \{y_p(n)\} \right\} &= Q(D) \left\{ \left(\sum_{l=0}^M b_l D^{-l} \right) \{x(n)\} \right\} \\ &= \left(\sum_{l=0}^M b_l D^{-l} \right) \{Q(D)\{x(n)\}\} \\ &= 0. \end{aligned} \quad (1.117)$$

This allows the nonhomogeneous difference equation to be solved using the same procedures used to find the homogeneous solutions.

For example, for a sequence $x(n) = s^n$, we have that $x(n-1) = s^{n-1}$; then:

$$x(n) = s x(n-1) \Rightarrow [1 - sD^{-1}]\{x(n)\} = 0$$

and, therefore, the annihilator polynomial for $x(n) = s^n$ is $Q(D) = (1 - sD^{-1})$. The annihilator polynomials for some typical inputs are summarized in Table 1.2.

Using the concept of annihilator polynomials, we can determine the form of the particular solution for certain types of input signal, which may include some undetermined coefficients. Some useful cases are shown in Table 1.3.

It is important to notice that there are no annihilator polynomials for inputs containing $u(n-n_0)$ or $\delta(n-n_0)$. Therefore, if a difference equation has inputs such as these, the above techniques can only be used for either $n \geq n_0$ or $n < n_0$, as discussed in Example 1.9.

Table 1.3.

Typical particular solutions for different input signals.

Input $x(n)$	Particular solution $y_p(n)$
$s^n, s \neq \rho_k$	αs^n
$s^n, s = \rho_k$ with multiplicity m_k	$\alpha n^{m_k} s^n$
$\cos(\omega n + \phi)$	$\alpha \cos(\omega n + \phi)$
$\left(\sum_{i=0}^I \beta_i n^i \right) s^n$	$\left(\sum_{i=0}^I \alpha_i n^i \right) s^n$

Example 1.9. Solve the difference equation

$$y(n) + a^2 y(n - 2) = b^n \sin\left(\frac{\pi}{2}n\right) u(n) \quad (1.118)$$

assuming that $a \neq b$ and $y(n) = 0$, for $n < 0$.

Solution

Using operator notation, Equation (1.118) becomes

$$\left(1 + a^2 D^{-2}\right)\{y(n)\} = b^n \sin\left(\frac{\pi}{2}n\right) u(n). \quad (1.119)$$

The homogeneous equation is

$$y_h(n) + a^2 y_h(n - 2) = 0. \quad (1.120)$$

Then, the characteristic polynomial equation from which we derive the homogeneous solution is

$$\rho^2 + a^2 = 0. \quad (1.121)$$

Since its roots are $\rho = a e^{\pm j\pi/2}$, then two solutions for the homogeneous equation are $a^n \sin[(\pi/2)n]$ and $a^n \cos[(\pi/2)n]$, as given in Table 1.1. Then, the general homogeneous solution becomes

$$y_h(n) = a^n \left[c_1 \sin\left(\frac{\pi}{2}n\right) + c_2 \cos\left(\frac{\pi}{2}n\right) \right]. \quad (1.122)$$

If one applies the correct annihilation to the excitation signals, the original difference equation is transformed into a higher order homogeneous equation. The solutions of this higher order homogeneous equation include the homogeneous and particular solutions of the original difference equation. However, there is no annihilator polynomial for $b^n \sin[(\pi/2)n] u(n)$. Therefore, one can only compute the solution to the difference equation

for $n \geq 0$, when the term to be annihilated becomes just $b^n \sin[(\pi/2)n]$. Therefore, for $n \geq 0$, according to Table 1.2, for the given input signal the annihilator polynomial is given by

$$Q(D) = (1 - b e^{j\pi/2} D^{-1}) (1 - b e^{-j\pi/2} D^{-1}) = 1 + b^2 D^{-2}. \quad (1.123)$$

Applying the annihilator polynomial on the difference equation, we obtain²

$$(1 + b^2 D^{-2}) (1 + a^2 D^{-2}) \{y(n)\} = 0. \quad (1.124)$$

The corresponding polynomial equation is

$$(\rho^2 + b^2)(\rho^2 + a^2) = 0. \quad (1.125)$$

It has four roots, two of the form $\rho = a e^{\pm j\pi/2}$ and two of the form $\rho = b e^{\pm j\pi/2}$. Since $a \neq b$, for $n \geq 0$ the complete solution is then given by

$$y(n) = b^n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] + a^n \left[d_3 \sin\left(\frac{\pi}{2}n\right) + d_4 \cos\left(\frac{\pi}{2}n\right) \right]. \quad (1.126)$$

The constants d_i , for $i = 1, 2, 3, 4$, are computed such that $y(n)$ is a particular solution to the nonhomogeneous equation. However, we notice that the term involving a^n corresponds to the solution of the homogeneous equation. Therefore, we do not need to substitute it in the equation since it will be annihilated for every d_3 and d_4 . One can then compute d_1 and d_2 by substituting only the term involving b^n in the nonhomogeneous Equation (1.118), leading to the following algebraic development:

$$\begin{aligned} & b^n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + a^2 b^{n-2} \left\{ d_1 \sin\left[\frac{\pi}{2}(n-2)\right] + d_2 \cos\left[\frac{\pi}{2}(n-2)\right] \right\} = b^n \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + a^2 b^{-2} \left[d_1 \sin\left(\frac{\pi}{2}n - \pi\right) + d_2 \cos\left(\frac{\pi}{2}n - \pi\right) \right] = \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + a^2 b^{-2} \left[-d_1 \sin\left(\frac{\pi}{2}n\right) - d_2 \cos\left(\frac{\pi}{2}n\right) \right] = \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow d_1(1 - a^2 b^{-2}) \sin\left(\frac{\pi}{2}n\right) + d_2(1 - a^2 b^{-2}) \cos\left(\frac{\pi}{2}n\right) = \sin\left(\frac{\pi}{2}n\right). \end{aligned} \quad (1.127)$$

Therefore, we conclude that

$$d_1 = \frac{1}{1 - a^2 b^{-2}}; \quad d_2 = 0 \quad (1.128)$$

² Since the expression of the input is valid only for $n \geq 0$, in this case, technically speaking, the annihilator polynomial should have a noncausal form with only nonnegative exponents for the delay operator, resulting in $Q(D) = D^2 + b^2$. In practice, however, the two expressions present the same roots and, therefore, they are equivalent and can be readily interchanged, as suggested here.

and the overall solution for $n \geq 0$ is

$$y(n) = \frac{b^n}{1-a^2b^{-2}} \sin\left(\frac{\pi}{2}n\right) + a^n \left[d_3 \sin\left(\frac{\pi}{2}n\right) + d_4 \cos\left(\frac{\pi}{2}n\right) \right]. \quad (1.129)$$

We now compute the constants d_3 and d_4 using the auxiliary conditions generated by the condition $y(n) = 0$, for $n < 0$. This implies that $y(-1) = 0$ and $y(-2) = 0$. However, one cannot use Equation (1.129) since it is valid only for $n \geq 0$. Thus, we need to run the difference equation from the auxiliary conditions $y(-2) = y(-1) = 0$ to compute $y(0)$ and $y(1)$:

$$\begin{cases} n=0 : & y(0) + a^2y(-2) = b^0 \sin\left(\frac{\pi}{2} \times 0\right)u(0) = 0 \\ n=1 : & y(1) + a^2y(-1) = b^1 \sin\left(\frac{\pi}{2}\right)u(1) = b \end{cases} \Rightarrow \begin{cases} y(0) = 0 \\ y(1) = b. \end{cases} \quad (1.130)$$

Using these auxiliary conditions in Equation (1.129) we get

$$\begin{cases} y(0) = \frac{1}{1-a^2b^{-2}} \sin\left(\frac{\pi}{2} \times 0\right) + \left[d_3 \sin\left(\frac{\pi}{2} \times 0\right) + d_4 \cos\left(\frac{\pi}{2} \times 0\right) \right] = 0 \\ y(1) = \frac{b}{1-a^2b^{-2}} \sin\left(\frac{\pi}{2}\right) + a \left[d_3 \sin\left(\frac{\pi}{2}\right) + d_4 \cos\left(\frac{\pi}{2}\right) \right] = b \end{cases} \quad (1.131)$$

and then

$$\begin{cases} d_4 = 0 \\ \frac{b}{1-a^2b^{-2}} + ad_3 = b \end{cases} \Rightarrow \begin{cases} d_3 = -\frac{ab^{-1}}{1-a^2b^{-2}} \\ d_4 = 0. \end{cases} \quad (1.132)$$

Substituting these values in Equation (1.129), the general solution becomes

$$\begin{cases} y(n) = 0, & n < 0 \\ y(n) = \frac{b^n - a^{n+1}b^{-1}}{1-a^2b^{-2}} \sin\left(\frac{\pi}{2}n\right), & n \geq 0, \end{cases} \quad (1.133)$$

which can be expressed in compact form as

$$y(n) = \frac{b^n - a^{n+1}b^{-1}}{1-a^2b^{-2}} \sin\left(\frac{\pi}{2}n\right)u(n). \quad (1.134)$$

An interesting case arises if the excitation is a pure sinusoid; that is, if $a = 1$, then the above solution can be written as

$$y(n) = \frac{b^n}{1-b^{-2}} \sin\left(\frac{\pi}{2}n\right)u(n) - \frac{b^{-1}}{1-b^{-2}} \sin\left(\frac{\pi}{2}n\right)u(n). \quad (1.135)$$

If $b > 1$, for large values of n , the first term of the right-hand side grows without bound (since b^n tends to infinity) and, therefore, the system is unstable (see Section 1.3.5). On the other hand, if $b < 1$, then b^n tends to zero as n grows and, therefore, the solution becomes the pure sinusoid

$$y(n) = -\frac{b^{-1}}{1-b^{-2}} \sin\left(\frac{\pi}{2}n\right). \quad (1.136)$$

We refer to this as a steady-state solution of the difference equation (see Exercises 1.17 and 1.18). Such solutions are very important in practice, and in Chapter 2 other techniques to compute them will be studied. \triangle

Example 1.10. Determine the solution of the difference equation in Example 1.9 supposing that $a = b$ (observe that the annihilator polynomial has common zeros with the homogeneous equation).

Solution

For $a = b$, there are repeated roots in the difference equation, and as a result the complete solution has the following form for $n \geq 0$:

$$y(n) = na^n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] + a^n \left[d_3 \sin\left(\frac{\pi}{2}n\right) + d_4 \cos\left(\frac{\pi}{2}n\right) \right]. \quad (1.137)$$

As in the case for $a \neq b$, we notice that the right-hand side of the summation is the homogeneous solution, and thus it will be annihilated for all d_3 and d_4 . For finding d_1 and d_2 one should substitute the left-hand side of the summation in the original equation (1.118), for $n \geq 0$. This yields

$$\begin{aligned} & na^n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + a^2(n-2)a^{n-2} \left\{ d_1 \sin\left[\frac{\pi}{2}(n-2)\right] + d_2 \cos\left[\frac{\pi}{2}(n-2)\right] \right\} = a^n \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + (n-2) \left[d_1 \sin\left(\frac{\pi}{2}n - \pi\right) + d_2 \cos\left(\frac{\pi}{2}n - \pi\right) \right] = \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow n \left[d_1 \sin\left(\frac{\pi}{2}n\right) + d_2 \cos\left(\frac{\pi}{2}n\right) \right] \\ & + (n-2) \left[-d_1 \sin\left(\frac{\pi}{2}n\right) - d_2 \cos\left(\frac{\pi}{2}n\right) \right] = \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow [nd_1 - (n-2)d_1] \sin\left(\frac{\pi}{2}n\right) + [nd_2 - (n-2)d_2] \cos\left(\frac{\pi}{2}n\right) = \sin\left(\frac{\pi}{2}n\right) \\ & \Rightarrow 2d_1 \sin\left(\frac{\pi}{2}n\right) + 2d_2 \cos\left(\frac{\pi}{2}n\right) = \sin\left(\frac{\pi}{2}n\right). \end{aligned} \quad (1.138)$$

Therefore, we conclude that

$$d_1 = \frac{1}{2}; \quad d_2 = 0 \quad (1.139)$$

and the overall solution for $n \geq 0$ is

$$y(n) = \frac{na^n}{2} \sin\left(\frac{\pi}{2}n\right) + a^n \left[d_3 \sin\left(\frac{\pi}{2}n\right) + d_4 \cos\left(\frac{\pi}{2}n\right) \right]. \quad (1.140)$$

As in the case for $a \neq b$, in order to compute the constants d_3 and d_4 one must use auxiliary conditions for $n \geq 0$, since Equation (1.137) is only valid for $n \geq 0$. Since $y(n) = 0$, for $n < 0$, we need to run the difference equation from the auxiliary conditions $y(-2) = y(-1) = 0$ to compute $y(0)$ and $y(1)$:

$$\begin{cases} n = 0 : & y(0) + a^2 y(-2) = a^0 \sin\left(\frac{\pi}{2} \times 0\right) u(0) = 0 \\ n = 1 : & y(1) + a^2 y(-1) = a^1 \sin\left(\frac{\pi}{2}\right) u(1) = a \end{cases} \Rightarrow \begin{cases} y(0) = 0 \\ y(1) = a. \end{cases} \quad (1.141)$$

Using these auxiliary conditions in Equation (1.140), we get

$$\begin{cases} y(0) = d_4 = 0 \\ y(1) = a \left[\frac{1}{2} \sin\left(\frac{\pi}{2}\right) \right] + a \left[d_3 \sin\left(\frac{\pi}{2}\right) + d_4 \cos\left(\frac{\pi}{2}\right) \right] = a \end{cases} \quad (1.142)$$

and then

$$\frac{a}{2} + ad_3 = a \Rightarrow d_3 = \frac{1}{2}; \quad d_4 = 0 \quad (1.143)$$

and since $y(n) = 0$, for $n < 0$, the solution is

$$y(n) = \frac{n+1}{2} a^n \sin\left(\frac{\pi}{2} n\right) u(n). \quad (1.144)$$

△

1.5.1 Computing impulse responses

In order to find the impulse response of a system, we can start by solving the following difference equation:

$$\sum_{i=0}^N a_i y(n-i) = \delta(n). \quad (1.145)$$

As has been pointed out in the discussion preceding Example 1.6, for a linear system to be causal it must be initially relaxed; that is, the auxiliary conditions prior to the input must be zero. For causal systems, since the input $\delta(n)$ is applied at $n = 0$, we must have

$$y(-1) = y(-2) = \cdots = y(-N) = 0. \quad (1.146)$$

For $n > 0$, Equation (1.145) becomes homogeneous; that is

$$\sum_{i=0}^N a_i y(n-i) = 0. \quad (1.147)$$

This can be solved using the techniques presented earlier in Section 1.5. In order to do so, we need N auxiliary conditions. However, since Equation (1.147) is valid only for $n > 0$, we cannot use the auxiliary conditions from Equation (1.146), but need N auxiliary conditions for $n > 0$ instead. For example, these conditions can be $y(1), y(2), \dots, y(N)$, which can be found, starting from the auxiliary conditions in Equation (1.146), by running the difference Equation (1.145) from $n = 0$ to $n = N$, leading to

$$\begin{cases} n = 0 : y(0) = \frac{\delta(0)}{a_0} - \frac{1}{a_0} \sum_{i=1}^N a_i y(-i) = \frac{1}{a_0} \\ n = 1 : y(1) = \frac{\delta(1)}{a_0} - \frac{1}{a_0} \sum_{i=1}^N a_i y(1-i) = -\frac{a_1}{a_0^2} \\ \vdots \\ n = N : y(N) = \frac{\delta(N)}{a_0} - \frac{1}{a_0} \sum_{i=1}^N a_i y(N-i) = -\frac{1}{a_0} \sum_{i=1}^N a_i y(N-i). \end{cases} \quad (1.148)$$

Example 1.11. Compute the impulse response of the system governed by the following difference equation:

$$y(n) - \frac{1}{2}y(n-1) + \frac{1}{4}y(n-2) = x(n). \quad (1.149)$$

Solution

For $n > 0$ the impulse response satisfies the homogeneous equation. The corresponding polynomial equation is

$$\rho^2 - \frac{1}{2}\rho + \frac{1}{4} = 0 \quad (1.150)$$

whose roots are $\rho = \frac{1}{2} e^{\pm j\pi/3}$. Therefore, for $n > 0$, the solution is

$$y(n) = c_1 2^{-n} \cos\left(\frac{\pi}{3}n\right) + c_2 2^{-n} \sin\left(\frac{\pi}{3}n\right). \quad (1.151)$$

Considering the system to be causal, we have that $y(n) = 0$, for $n < 0$. Therefore, we need to compute the auxiliary conditions for $n > 0$ as follows:

$$\begin{cases} n = 0 : y(0) = \delta(0) + \frac{1}{2}y(-1) - \frac{1}{4}y(-2) = 1 \\ n = 1 : y(1) = \delta(1) + \frac{1}{2}y(0) - \frac{1}{4}y(-1) = \frac{1}{2} \\ n = 2 : y(2) = \delta(2) + \frac{1}{2}y(1) - \frac{1}{4}y(0) = 0. \end{cases} \quad (1.152)$$

Applying the above conditions to the solution in Equation (1.151) we have

$$\begin{cases} y(1) = c_1 2^{-1} \cos\left(\frac{\pi}{3}\right) + c_2 2^{-1} \sin\left(\frac{\pi}{3}\right) = \frac{1}{2} \\ y(2) = c_1 2^{-2} \cos\left(\frac{2\pi}{3}\right) + c_2 2^{-2} \sin\left(\frac{2\pi}{3}\right) = 0. \end{cases} \quad (1.153)$$

Hence:

$$\begin{cases} \frac{1}{4}c_1 + \frac{\sqrt{3}}{4}c_2 = \frac{1}{2} \\ -\frac{1}{8}c_1 + \frac{\sqrt{3}}{8}c_2 = 0 \end{cases} \Rightarrow \begin{cases} c_1 = 1 \\ c_2 = \frac{\sqrt{3}}{3} \end{cases} \quad (1.154)$$

and the impulse response becomes

$$y(n) = \begin{cases} 0, & n < 0 \\ 1, & n = 0 \\ \frac{1}{2}, & n = 1 \\ 0, & n = 2 \\ 2^{-n} \left\{ \cos[(\pi/3)n] + (\sqrt{3}/3) \sin[(\pi/3)n] \right\}, & n \geq 2 \end{cases} \quad (1.155)$$

which, by inspection, can be expressed in a compact form as

$$y(n) = 2^{-n} \left[\cos\left(\frac{\pi}{3}n\right) + \frac{\sqrt{3}}{3} \sin\left(\frac{\pi}{3}n\right) \right] u(n). \quad (1.156)$$

△

1.6 Sampling of continuous-time signals

In many cases, a discrete-time signal $x(n)$ consists of samples of a continuous-time signal $x_a(t)$; that is:

$$x(n) = x_a(nT). \quad (1.157)$$

If we want to process the continuous-time signal $x_a(t)$ using a discrete-time system, then we first need to convert it using Equation (1.157), process the discrete-time input digitally, and then convert the discrete-time output back to the continuous-time domain. Therefore, in order for this operation to be effective, it is essential that we have capability of restoring a continuous-time signal from its samples. In this section, we derive conditions under which a continuous-time signal can be recovered from its samples. We also devise ways of performing this recovery. To do so, we first introduce some basic concepts of analog signal processing that can be found in most standard books on linear systems (Gabel & Roberts, 1980; Oppenheim *et al.*, 1983). Following that, we derive the sampling theorem, which gives the basis for digital processing of continuous-time signals.

1.6.1 Basic principles

The Fourier transform of a continuous-time signal $f(t)$ is given by

$$F(j\Omega) = \int_{-\infty}^{\infty} f(t) e^{-j\Omega t} dt, \quad (1.158)$$

where Ω is referred to as the frequency and is measured in radians per second (rad/s). The corresponding inverse relationship is expressed as

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\Omega) e^{j\Omega t} d\Omega. \quad (1.159)$$

An important property associated with the Fourier transform is that the Fourier transform of the product of two functions equals the convolution of their Fourier transforms. That is, if $x(t) = a(t)b(t)$, then

$$X(j\Omega) = \frac{1}{2\pi} A(j\Omega) * B(j\Omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A(j\Omega - j\Omega') B(j\Omega') d\Omega', \quad (1.160)$$

where $X(j\Omega)$, $A(j\Omega)$, and $B(j\Omega)$ are the Fourier transforms of $x(t)$, $a(t)$, and $b(t)$ respectively.

In addition, if a signal $x(t)$ is periodic with period T , then we can express it by its Fourier series defined by

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{j(2\pi/T)kt}, \quad (1.161)$$

where the a_k are called the series coefficients, which are determined as

$$a_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{-jk(2\pi/T)t} dt. \quad (1.162)$$

1.6.2 Sampling theorem

Given a discrete-time signal $x(n)$ derived from a continuous-time signal $x_a(t)$ using Equation (1.157), we define a continuous-time signal $x_i(t)$ consisting of a train of impulses at $t = nT$, each of area equal to $x(n) = x_a(nT)$. Examples of signals $x_a(t)$, $x(n)$, and $x_i(t)$ are depicted in Figure 1.5, where the direct relationships between these three signals can be seen.

The signal $x_i(t)$ can be expressed as

$$x_i(t) = \sum_{n=-\infty}^{\infty} x(n) \delta(t - nT). \quad (1.163)$$

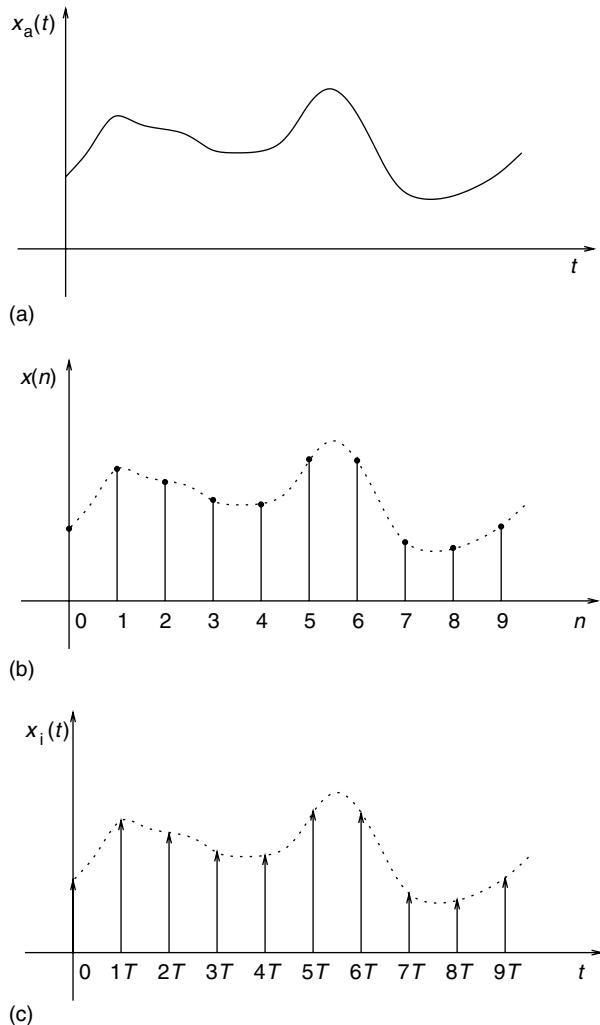


Fig. 1.5. (a) Continuous-time signal $x_a(t)$; (b) discrete-time signal $x(n)$; (c) auxiliary continuous-time signal $x_i(t)$.

Since, from Equation (1.157), $x(n) = x_a(nT)$, then Equation (1.163) becomes

$$x_i(t) = \sum_{n=-\infty}^{\infty} x_a(nT) \delta(t - nT) = x_a(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = x_a(t)p(t), \quad (1.164)$$

indicating that $x_i(t)$ can also be obtained by multiplying the continuous-time signal $x_a(t)$ by a train of impulses $p(t)$ defined as

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT). \quad (1.165)$$

In the equations above, we have defined a continuous-time signal $x_i(t)$ that can be obtained from the discrete-time signal $x(n)$ in a straightforward manner. In what follows, we relate the Fourier transforms of $x_a(t)$ and $x_i(t)$, and study the conditions under which $x_a(t)$ can be obtained from $x_i(t)$.

From equations (1.160) and (1.164), the Fourier transform of $x_i(t)$ is such that

$$X_i(j\Omega) = \frac{1}{2\pi} X_a(j\Omega) * P(j\Omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_a(j\Omega - j\Omega') P(j\Omega') d\Omega'. \quad (1.166)$$

Therefore, in order to arrive at an expression for the Fourier transform of $x_i(t)$, we must first determine the Fourier transform of $p(t)$, $P(j\Omega)$. From Equation (1.165), we see that $p(t)$ is a periodic function with period T and that we can decompose it in a Fourier series, as described in Equations (1.161) and (1.162). Since, from Equation (1.165), $p(t)$ in the interval $[-T/2, T/2]$ is equal to just an impulse $\delta(t)$, the coefficients a_k in the Fourier series of $p(t)$ are given by

$$a_k = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) e^{-jk(2\pi/T)t} dt = \frac{1}{T} \quad (1.167)$$

and the Fourier series for $p(t)$ becomes

$$p(t) = \frac{1}{T} \sum_{k=-\infty}^{\infty} e^{j(2\pi/T)kt}. \quad (1.168)$$

As the Fourier transform of $f(t) = e^{j\Omega_0 t}$ is equal to $F(j\Omega) = 2\pi\delta(\Omega - \Omega_0)$, then, from Equation (1.168), the Fourier transform of $p(t)$ becomes

$$P(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi}{T}k\right). \quad (1.169)$$

Substituting this expression for $P(j\Omega)$ in Equation (1.166), we have that

$$\begin{aligned} X_i(j\Omega) &= \frac{1}{2\pi} X_a(j\Omega) * P(j\Omega) \\ &= \frac{1}{T} X_a(j\Omega) * \sum_{k=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi}{T}k\right) \\ &= \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(j\Omega - j\frac{2\pi}{T}k\right), \end{aligned} \quad (1.170)$$

where, in the last step, we used the fact that the convolution of a function $F(j\Omega)$ with a shifted impulse $\delta(\Omega - \Omega_0)$ is the shifted function $F(j\Omega - j\Omega_0)$. Equation (1.170) shows that the spectrum of $x_i(t)$ is composed of infinite shifted copies of the spectrum of $x_a(t)$, with the shifts in frequency being multiples of the sampling frequency, $\Omega_s = 2\pi/T$. Figure 1.6 shows examples of signals $x_a(t)$, $p(t)$, and $x_i(t)$, and their respective Fourier transforms.

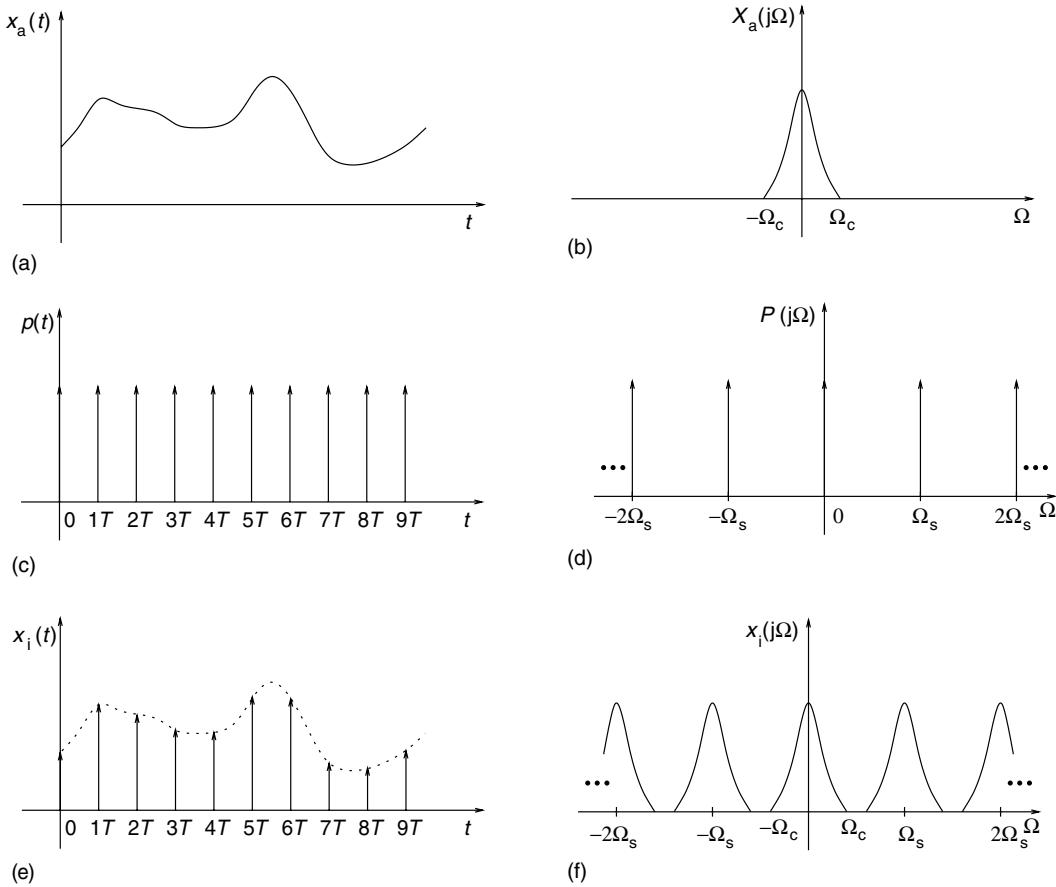


Fig. 1.6. (a) Continuous-time signal $x_a(t)$; (b) spectrum of $x_a(t)$; (c) train of impulses $p(t)$; (d) spectrum of $p(t)$; (e) auxiliary continuous-time signal $x_i(t)$; (f) spectrum of $x_i(t)$.

From Equation (1.170) and Figure 1.6f, we see that, in order to avoid the repeated copies of the spectrum of $x_a(t)$ interfering with one another, the signal should be band-limited. In addition, its bandwidth Ω_c should be such that the upper edge of the spectrum centered at zero is smaller than the lower edge of the spectrum centered at Ω_s .

Referring to the general complex case depicted in Figure 1.7, we must have $\Omega_s + \Omega_2 > \Omega_1$, or equivalently $\Omega_s > \Omega_1 - \Omega_2$.

In the case of real signals, since the spectrum is symmetric around zero, the one-sided bandwidth of the continuous-time signal Ω_c is such that $\Omega_c = \Omega_1 = -\Omega_2$, and then we must have $\Omega_s > \Omega_c - (-\Omega_c)$, implying that

$$\Omega_s > 2\Omega_c; \quad (1.171)$$

that is, the sampling frequency must be larger than double the one-sided bandwidth of the continuous-time signal. The frequency $\Omega = 2\Omega_c$ is called the Nyquist frequency of the real continuous-time signal $x_a(t)$.

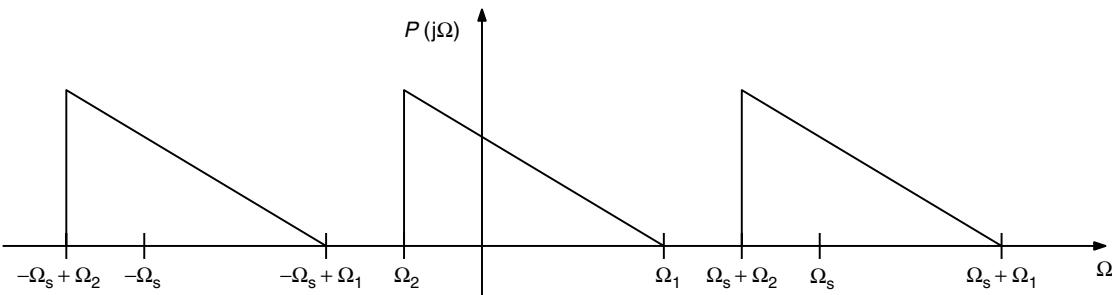


Fig. 1.7. Example of spectrum of a sampled complex signal.

In addition, if the condition in Equation (1.171) is satisfied, the original continuous signal $x_a(t)$ can be recovered by isolating the part of the spectrum of $x_i(t)$ that corresponds to the spectrum of $x_a(t)$.³ This can be achieved by filtering the signal $x_i(t)$ with an ideal lowpass filter having bandwidth $\Omega_s/2$.

On the other hand, if the condition in Equation (1.171) is not satisfied, the repetitions of the spectrum interfere with one another, and the continuous-time signal cannot be recovered from its samples. This superposition of the repetitions of the spectrum of $x_a(t)$ in $x_i(t)$, when the sampling frequency is smaller than $2\Omega_c$, is commonly referred to as aliasing. Figure 1.8b–d shows the spectra of $x_i(t)$ for Ω_s equal to, smaller than, and larger than $2\Omega_c$, respectively. The aliasing phenomenon is clearly identified in Figure 1.8c.

We are now ready to enunciate a very important result.

Theorem 1.1 (Sampling Theorem). *If a continuous-time signal $x_a(t)$ is band-limited – that is, its Fourier transform is such that $X_a(j\Omega) = 0$, for $|\Omega| > |\Omega_c|$ – then $x_a(t)$ can be completely recovered from the discrete-time signal $x(n) = x_a(nT)$ if the sampling frequency Ω_s satisfies $\Omega_s > 2\Omega_c$.*

◇

Example 1.12. Consider the discrete-time sequence

$$x(n) = \sin\left(\frac{6\pi}{4}n\right). \quad (1.172)$$

Assuming that the sampling frequency is $f_s = 40$ kHz, find two continuous-time signals that could have generated this sequence.

Solution

Supposing that the continuous-time signal is of the form

$$x_a(t) = \sin(\Omega_c t) = \sin(2\pi f_c t). \quad (1.173)$$

³ In fact, any of the spectrum repetitions have the full information about $x_a(t)$. However, if we isolate a repetition of the spectrum not centered at $\Omega = 0$, we get a modulated version of $x_a(t)$, which should be demodulated. Since its demodulation is equivalent to shifting the spectrum back to the origin, it is usually better to take the repetition of the spectrum centered at the origin in the first place.

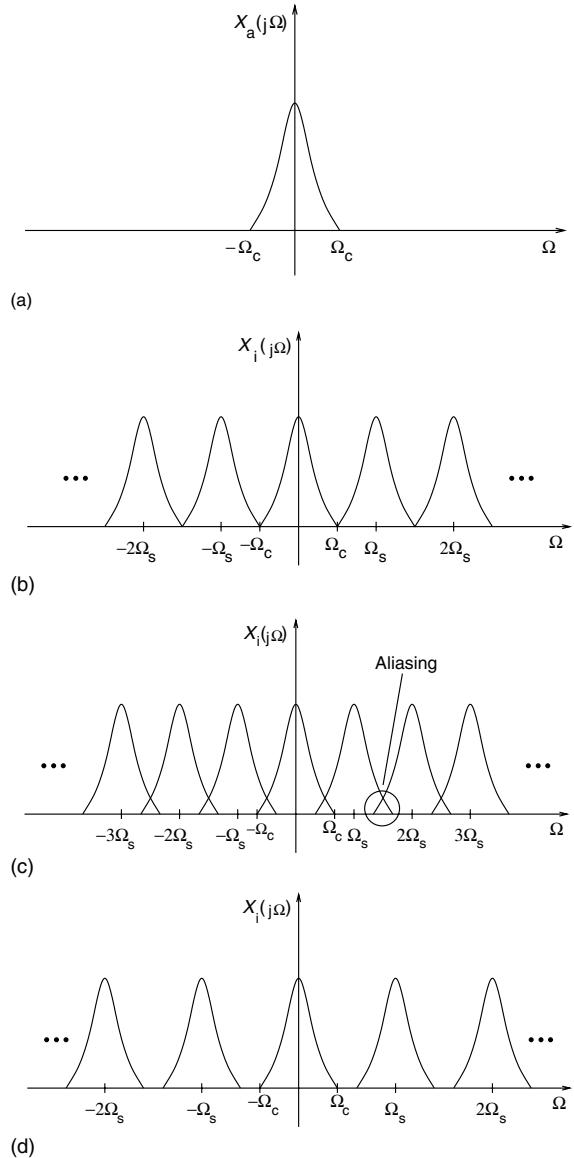


Fig. 1.8. (a) Spectrum of the continuous-time signal. Spectra of $x_i(t)$ for: (b) $\Omega_s = 2\Omega_c$; (c) $\Omega_s < 2\Omega_c$; (d) $\Omega_s > 2\Omega_c$.

We have that when sampled with sampling frequency $f_s = 1/T_s$ it generates the following discrete signal:

$$\begin{aligned} x(n) &= x_a(nT_s) \\ &= \sin(2\pi f_c n T_s) \\ &= \sin\left(2\pi \frac{f_c}{f_s} n\right) \end{aligned}$$

$$\begin{aligned}
 &= \sin\left(2\pi \frac{f_c}{f_s} n + 2k\pi n\right) \\
 &= \sin\left[2\pi \left(\frac{f_c}{f_s} + k\right) n\right]
 \end{aligned} \tag{1.174}$$

for any integer k . Therefore, in order for a sinusoid following Equation (1.173), when sampled, to yield the discrete signal in Equation (1.172), we must have that

$$2\pi\left(\frac{f_c}{f_s} + k\right) = \frac{6\pi}{4} \Rightarrow f_c = \left(\frac{3}{4} - k\right)f_s. \tag{1.175}$$

For example:

$$k = 0 \Rightarrow f_c = \frac{3}{4}f_s = 30 \text{ kHz} \Rightarrow x_1(t) = \sin(60000\pi t) \tag{1.176}$$

$$k = -1 \Rightarrow f_c = \frac{7}{4}f_s = 70 \text{ kHz} \Rightarrow x_2(t) = \sin(140000\pi t) \tag{1.177}$$

We can verify that by computing the $x_i(t)$ signals for the two above signals according to Equation (1.164):

$$\begin{aligned}
 x_{1_i}(t) &= \sum_{n=-\infty}^{\infty} x_1(t)\delta(t - nT_s) = \sum_{n=-\infty}^{\infty} \sin(60000\pi t)\delta\left(t - \frac{n}{40000}\right) \\
 &= \sum_{n=-\infty}^{\infty} \sin\left(60000\pi \frac{n}{40000}\right)\delta\left(t - \frac{n}{40000}\right) \\
 &= \sum_{n=-\infty}^{\infty} \sin\left(\frac{3\pi}{2}n\right)\delta\left(t - \frac{n}{40000}\right)
 \end{aligned} \tag{1.178}$$

$$\begin{aligned}
 x_{2_i}(t) &= \sum_{n=-\infty}^{\infty} x_2(t)\delta(t - nT_s) = \sum_{n=-\infty}^{\infty} \sin(140000\pi t)\delta\left(t - \frac{n}{40000}\right) \\
 &= \sum_{n=-\infty}^{\infty} \sin\left(140000\pi \frac{n}{40000}\right)\delta\left(t - \frac{n}{40000}\right) \\
 &= \sum_{n=-\infty}^{\infty} \sin\left(\frac{7\pi}{2}n\right)\delta\left(t - \frac{n}{40000}\right).
 \end{aligned} \tag{1.179}$$

Since

$$\sin\left(\frac{7\pi}{2}n\right) = \sin\left[\left(\frac{3\pi}{2} + 2\pi\right)n\right] = \sin\left(\frac{3\pi}{2}n\right), \tag{1.180}$$

then we have that the signals $x_{1_i}(t)$ and $x_{2_i}(t)$ are identical. \triangle

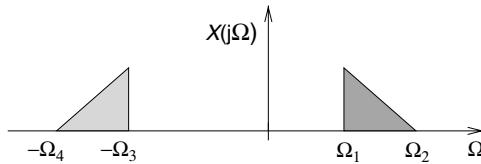


Fig. 1.9. Signal spectrum of Example 1.13.

Example 1.13. In Figure 1.9, assuming that $\Omega_2 - \Omega_1 < \Omega_1$ and $\Omega_4 - \Omega_3 < \Omega_3$:

- Using a single sampler, what would be the minimum sampling frequency such that no information is lost?
- Using an ideal filter and two samplers, what would be the minimum sampling frequencies such that no information is lost? Depict the configuration used in this case.

Solution

- By examining Figure 1.9 we see that a sampling rate $\Omega_s > \Omega_2 + \Omega_4$ would avoid aliasing. However, since it is given that $\Omega_2 - \Omega_1 < \Omega_1$ and $\Omega_4 - \Omega_3 < \Omega_3$, then in the empty spectrum between $-\Omega_3$ and Ω_1 we can accommodate one copy of the spectrum in the interval $[\Omega_1, \Omega_2]$ and one copy of the spectrum in the interval $[-\Omega_4, -\Omega_3]$. According to Equation (1.170), when a signal is sampled its spectrum is repeated at multiples of Ω_s . Therefore, we can choose Ω_s so that the spectrum of the sampled signal would be as in the lower part of Figure 1.10, where, to avoid spectrum superposition, we must have

$$\begin{cases} \Omega_1 - \Omega_s > -\Omega_3 \\ -\Omega_4 + \Omega_s > \Omega_2 - \Omega_1 \end{cases} \Rightarrow \frac{\Omega_2 + \Omega_4}{2} < \Omega_s < \Omega_1 + \Omega_3. \quad (1.181)$$

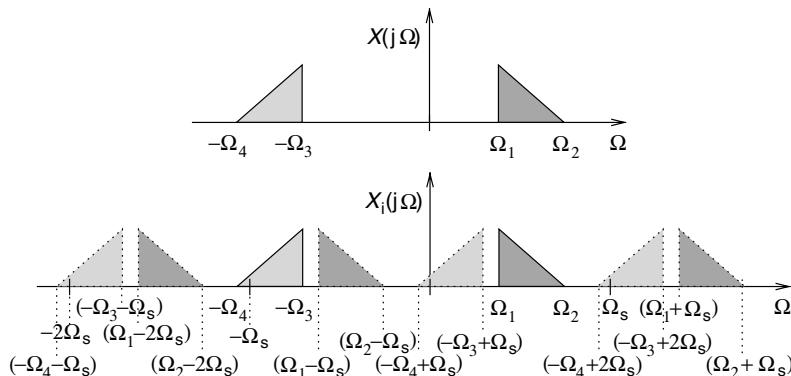


Fig. 1.10. Spectrum of the sampled signal in Example 1.13(a).

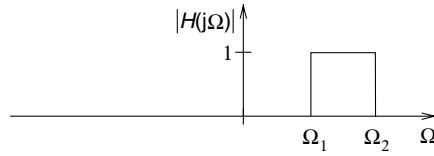


Fig. 1.11. Ideal bandpass filter.

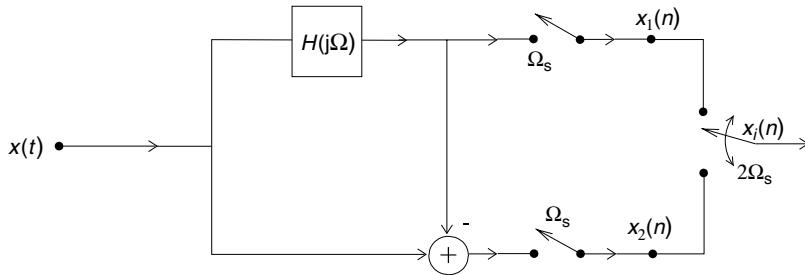


Fig. 1.12. Sampling solution in Example 1.13(b) using an ideal bandpass filter.

Therefore, the minimum sampling frequency would be $\Omega_s = (\Omega_2 + \Omega_4)/2$, provided that $\Omega_s < \Omega_1 + \Omega_3$.

- (b) If we have an ideal filter, as depicted in Figure 1.11, then we can isolate both parts of the spectrum and then sample them at a much lower rate. For example, we can sample the output of the filter in Figure 1.11 with a frequency $\Omega_{s1} > \Omega_2 - \Omega_1$. If we use the scheme in Figure 1.12, then we take the filter output and subtract it from the input signal. The result will have just the left side of the spectrum in Figure 1.9, which can be sampled with a frequency $\Omega_{s2} > \Omega_4 - \Omega_3$.

If we use a single sampling frequency, then its value should satisfy

$$\Omega_s > \max\{\Omega_2 - \Omega_1, \Omega_4 - \Omega_3\}. \quad (1.182)$$

Note that the output is composed of one sample of each signal $x_1(n)$ and $x_2(n)$; therefore, the effective sampling frequency is $2\Omega_s$. \triangle

As illustrated in the example above, for some bandpass signals $x(t)$, sampling may be performed below the limit $2\Omega_{\max}$, where Ω_{\max} represents the maximum absolute value of the frequency present in $x(t)$. Although one cannot obtain a general expression for the minimum Ω_s in these cases, it must always satisfy $\Omega_s > \Delta\Omega$, where $\Delta\Omega$ represents the net bandwidth of $x(t)$.

As mentioned in Theorem 1.1, the original continuous-time signal $x_a(t)$ can be recovered from the signal $x_i(t)$ by filtering $x_i(t)$ with an ideal lowpass filter having cutoff frequency $\Omega_s/2$. More specifically, if the signal has bandwidth Ω_c , then it suffices that the cutoff frequency of the ideal lowpass filter is Ω_{LP} , such that $\Omega_c \leq \Omega_{LP} \leq \Omega_s/2$. Therefore, the

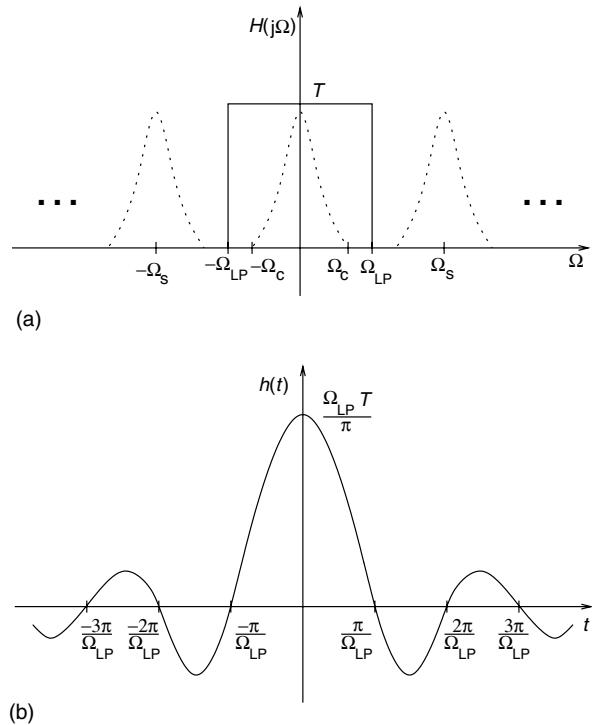


Fig. 1.13. Ideal lowpass filter: (a) frequency response; (b) impulse response.

Fourier transform of the impulse response of such a filter should be

$$H(j\Omega) = \begin{cases} T, & \text{for } |\Omega| < \Omega_{LP} \\ 0, & \text{for } |\Omega| \geq \Omega_{LP}, \end{cases} \quad (1.183)$$

where the passband gain T compensates for the factor $1/T$ in Equation (1.170). This ideal frequency response is illustrated in Figure 1.13a.

Computing the inverse Fourier transform of $H(j\Omega)$ using Equation (1.159), we see that the impulse response $h(t)$ of the filter is

$$h(t) = \frac{T \sin(\Omega_{LP} t)}{\pi t}, \quad (1.184)$$

which is depicted in Figure 1.13b.

Then, given $h(t)$, the signal $x_a(t)$ can be recovered from $x_i(t)$ by the following convolution integral (Oppenheim *et al.*, 1983):

$$x_a(t) = \int_{-\infty}^{\infty} x_i(\tau)h(t - \tau) d\tau. \quad (1.185)$$

Replacing $x_i(t)$ by its definition in Equation (1.163), we have that

$$\begin{aligned} x_a(t) &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n)\delta(\tau - nT)h(t - \tau) d\tau \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} x(n)\delta(\tau - nT)h(t - \tau) d\tau \\ &= \sum_{n=-\infty}^{\infty} x(n)h(t - nT), \end{aligned} \quad (1.186)$$

and using $h(t)$ from Equation (1.184):

$$x_a(t) = \sum_{n=-\infty}^{\infty} x(n) \frac{T \sin[\Omega_{LP}(t - nT)]}{\pi(t - nT)} \quad (1.187)$$

If we make Ω_{LP} equal to half the sampling frequency, that is, $\Omega_{LP} = \Omega_s/2 = \pi/T$, then Equation (1.187) becomes

$$x_a(t) = \sum_{n=-\infty}^{\infty} x(n) \frac{\sin\left(\frac{\Omega_s}{2}t - n\pi\right)}{\frac{\Omega_s}{2}t - n\pi} = \sum_{n=-\infty}^{\infty} x(n) \frac{\sin\left[\pi\left(\frac{t}{T} - n\right)\right]}{\pi\left(\frac{t}{T} - n\right)}. \quad (1.188)$$

Equations (1.187) and (1.188) represent interpolation formulas to recover the continuous-time signal $x_a(t)$ from its samples $x(n) = x_a(nT)$. However, since, in order to compute $x_a(t)$ at any time t_0 , all the samples of $x(n)$ have to be known, these interpolation formulas are not practical. This is the same as saying that the lowpass filter with impulse response $h(t)$ is not realizable. This is because it is noncausal and cannot be implemented via any differential equation of finite order. Clearly, the closer the lowpass filter is to the ideal, the smaller is the error in the computation of $x(t)$ using Equation (1.187). In Chapters 5 and 6, methods to approximate such ideal filters will be studied extensively.

From what we have studied in this section, we can develop a block diagram of the several phases constituting the processing of an analog signal using a digital system. Figure 1.14 depicts each step of the procedure.

The first block in the diagram of Figure 1.14 is a lowpass filter, that guarantees the analog signal is band-limited, with bandwidth $\Omega_c < \Omega_s/2$.

The second block is a sample-and-hold system, which samples the signal $x_a(t)$ at times $t = nT$ and holds the value obtained for T seconds; that is, until the value for the next time interval is sampled. More precisely, $x_a^*(t) = x_a(nT)$, for $nT < t < (n+1)T$.

The third block, the encoder, converts each sample value output by the sample and hold, $x_a^*(t)$, for $nT < t < (n+1)T$, to a number $x(n)$. Since this number is input to digital hardware, it must be represented with a finite number of bits. This operation introduces an error in the signal, which gets smaller as the number of bits used in the representation

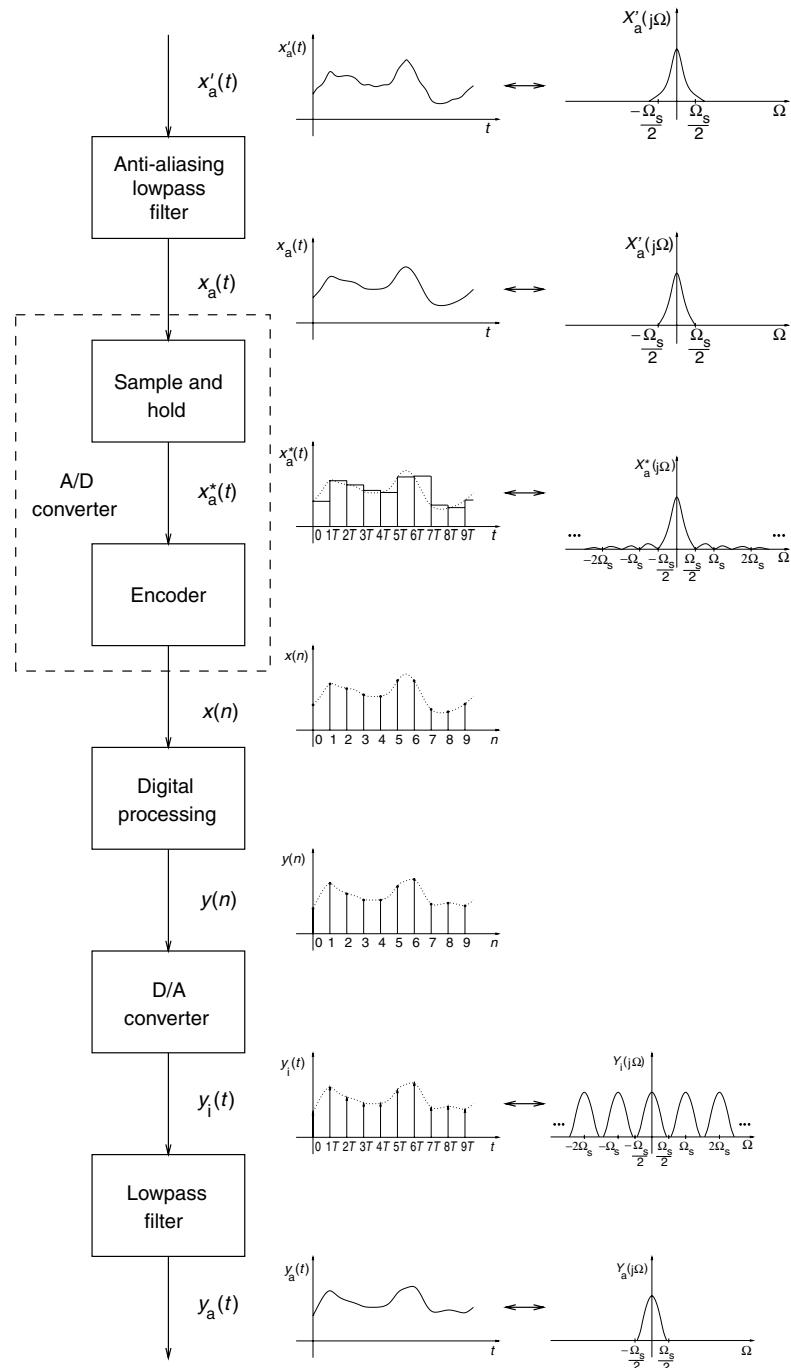


Fig. 1.14. Digital processing of analog signals (all graphs on the left represent the signals in the time domain and all graphs on the right are their corresponding Fourier transforms).

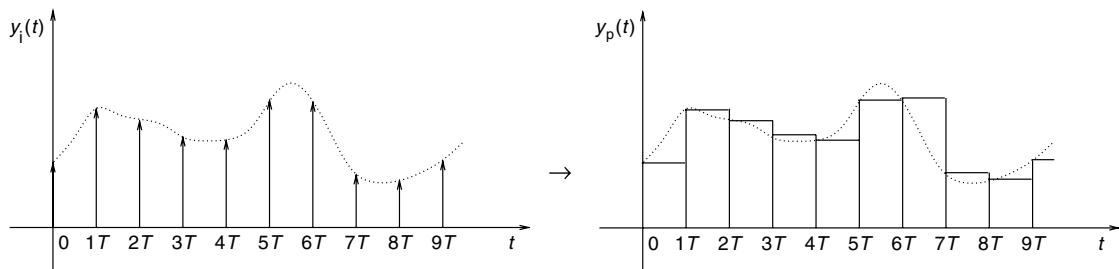


Fig. 1.15. Sample-and-hold operation.

increases. The second and third blocks constitute what we usually call the analog-to-digital (A/D) conversion.

The fourth block carries out the digital signal processing, transforming the discrete-time signal \$x(n)\$ into a discrete-time signal \$y(n)\$.

The fifth block transforms the numbers representing the samples \$y(n)\$ back into the analog domain, in the form of a train of impulses \$y_i(t)\$, constituting the process known as digital-to-analog (D/A) conversion.

The sixth block is a lowpass filter necessary to eliminate the repetitions of the spectrum contained in \$y_i(t)\$, in order to recover the analog signal \$y_a(t)\$ corresponding to \$y(n)\$. In practice, sometimes the fifth and sixth blocks are implemented in one operation. For example, we can transform the samples \$y(n)\$ into an analog signal \$y_a(t)\$ using a D/A converter plus a sample-and-hold operation, similar to the one of the second block. It is easy to show (see Figure 1.15) that the sample-and-hold operation is equivalent to filtering the train of impulses

$$y_i(t) = \sum_{n=-\infty}^{\infty} y(n)\delta(t - nT) \quad (1.189)$$

with a filter having impulse response

$$h(t) = \begin{cases} 1, & \text{for } 0 \leq t \leq T \\ 0, & \text{otherwise} \end{cases} \quad (1.190)$$

yielding the analog signal

$$y_p(t) = \sum_{n=-\infty}^{\infty} y(n)h(t - nT). \quad (1.191)$$

In this case, the recovery of the analog signal is not perfect, but is a good enough approximation in some practical cases.

Example 1.14. For the sample-and-hold operation described by Equations (1.189) to (1.191), determine:

- (a) An expression for the Fourier transform of \$y_p(t)\$ in Equation (1.191) as a function of the Fourier transform of \$x_a(t)\$ (suppose that \$y(n) = x_a(nT)\$).

- (b) The frequency response of an ideal lowpass filter that outputs $x_a(t)$ when $y_p(t)$ is applied to its input. Such a filter would compensate for the artifacts introduced by the sample-and-hold operation in a D/A conversion.

Solution

- (a) The train of pulses given by

$$y_p(t) = \sum_{n=-\infty}^{\infty} y(n) h(t - nT) \quad (1.192)$$

is the result of the convolution of an impulse train with the given pulse as follows:

$$y_p(t) = y_i(t) * h(t). \quad (1.193)$$

Using Equation (1.170), in the frequency domain the above equation becomes

$$Y_p(j\Omega) = Y_i(j\Omega)H(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(j\Omega - j\frac{2\pi}{T}k\right)H(j\Omega). \quad (1.194)$$

From Equation (1.190), since

$$\begin{aligned} H(j\Omega) &= \int_{-\infty}^{\infty} h(t) e^{-j\Omega t} dt \\ &= \int_0^T e^{-j\Omega t} dt \\ &= \frac{1}{j\Omega} \left(1 - e^{-j\Omega T}\right) \\ &= \frac{e^{-j\Omega T/2}}{j\Omega} \left(e^{j\Omega T/2} - e^{-j\Omega T/2}\right) \\ &= \frac{e^{-j\Omega T/2}}{j\Omega} [2j \sin(\Omega T/2)] \\ &= T e^{-j\Omega T/2} \left[\frac{\sin(\Omega T/2)}{\Omega T/2} \right], \end{aligned} \quad (1.195)$$

hence

$$Y_p(j\Omega) = e^{-j\Omega T/2} \left[\frac{\sin(\Omega T/2)}{\Omega T/2} \right] \sum_{k=-\infty}^{\infty} X_a\left(j\Omega - j\frac{2\pi}{T}k\right). \quad (1.196)$$

- (b) In order to recuperate signal $x_a(t)$, we have to compensate the distortion introduced by the frequency spectrum of the pulse $h(t)$. This can be done by processing $y_p(t)$ with a lowpass filter with the following desirable frequency response:

$$G(j\Omega) = \begin{cases} 0, & |\Omega| \geq \pi/T \\ \frac{T}{H(j\Omega)} = e^{j\Omega T/2} \left[\frac{\Omega T/2}{\sin(\Omega T/2)} \right], & |\Omega| < \pi/T. \end{cases} \quad (1.197)$$

△

Example 1.15. Cinema is a way of storing and displaying moving pictures. Before it was invented, all that was known was a way to store and display single images, by taking photographs. Cinema was invented when someone decided to capture a moving image as a series of pictures equally spaced in time. For example, today, in a commercial movie one captures 24 pictures/second. This scheme works because the human visual system enjoys a property called persistence of vision: when a light is flashed, one still sees the light for some time after it is gone. Because of this, when displaying the pictures in sequence, a human viewer has the illusion of continuous movement.

In mathematical terms, a moving picture is a three-dimensional signal, with two continuous spatial dimensions (representing, for instance, horizontal and vertical directions) and one continuous time dimension. A photograph is a time sample of this three-dimensional signal. When one displays this sequence of time samples (photographs), the human visual system sees it as a continuously moving picture; that is, as a continuous-time signal. From what has been said, we can note that cinema can be regarded as a discrete-time signal processing system.

By referring to Figure 1.14, identify in the cinema context which signal processing operation corresponds to each step of the processes of recording and displaying moving pictures, highlighting the associated design constraints.

Solution

The light field of a gray-scale moving picture can be described as a three-dimensional function $f_a(x, y, t)$, where x and y are spatial coordinates and t is a time coordinate, as represented in Figure 1.16.

The light intensity (luminance) of a point of coordinates (x_0, y_0) is thus the one-dimensional time signal

$$g_a(t) = f_a(x_0, y_0, t), \quad (1.198)$$

as depicted in Figure 1.17.

When one takes photographs of a moving picture every T time units, the three-dimensional signal is sampled in time, generating the discrete-time signal

$$f(x, y, n) = f_a(x, y, nT). \quad (1.199)$$

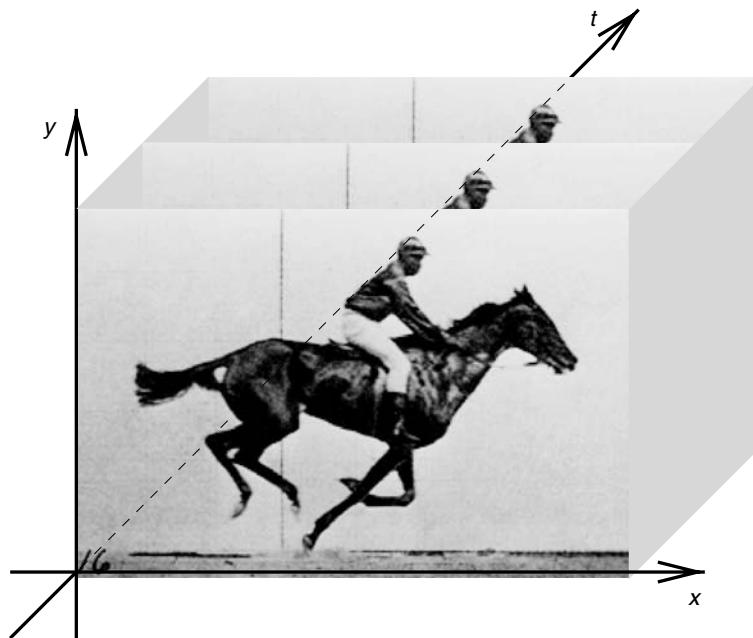


Fig. 1.16. Representation of a moving picture as a three-dimensional signal.

The discrete-time one-dimensional signal corresponding to the intensity of the point (x_0, y_0) is then

$$g(n) = g_a(nT) = f(x_0, y_0, n) = f_a(x_0, y_0, nT), \quad (1.200)$$

as represented in Figure 1.18.

Therefore, in order to avoid aliasing, one should sample the signal $g_a(t)$ with a sampling frequency larger than twice its bandwidth. Supposing that the bandwidth of $g_a(t) = f_a(x_0, y_0, t)$ is W_{x_0, y_0} , then, in order to avoid aliasing, the time interval between photographs should be determined by the largest time bandwidth among all coordinates of the picture; that is:

$$T < \frac{2\pi}{\max_{x_0, y_0} \{2W_{x_0, y_0}\}}. \quad (1.201)$$

It is reasonable to think that the faster the objects in a scene move, the larger is the time bandwidth of the light intensity of its points. Therefore, Equation (1.201) says that, in order to avoid aliasing, one should film the scene by taking enough pictures per second such that the time interval between them satisfies a maximum constraint. For example, when filming a hummingbird, if there is too long an interval between photographs, it might flap its wings many times between photographs. Depending on the speed one takes the photographs, its

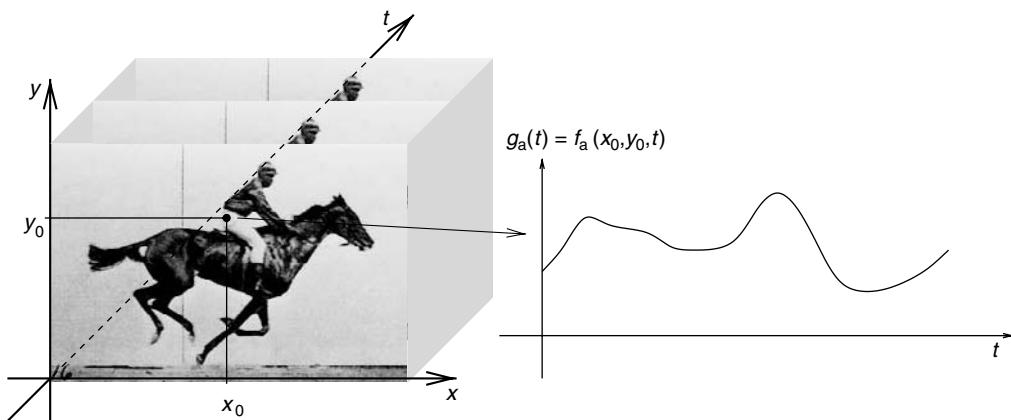


Fig. 1.17. Time signal generated by the intensity of a point (x_0, y_0) from a moving picture.

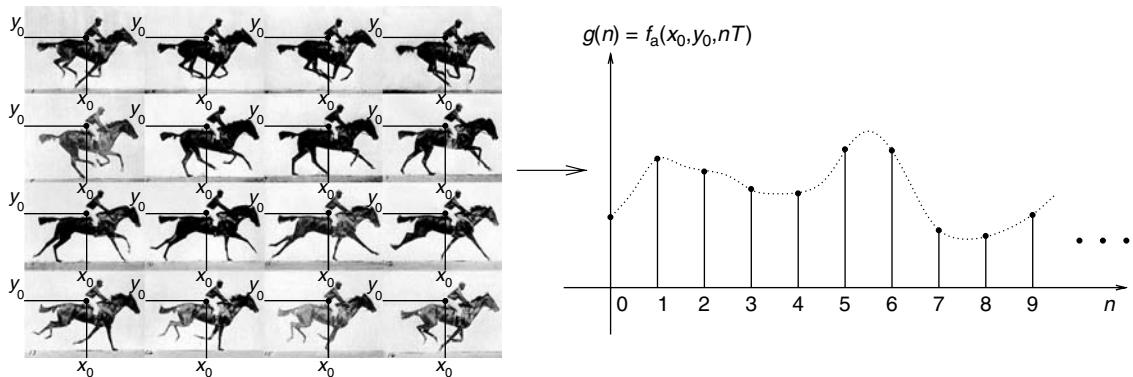


Fig. 1.18. Discrete-time signal generated by the intensity of a given coordinate from photographs of a moving picture.

wings may even be approximately in the same position in every photograph. This would have the effect that its wings would appear static when playing the film. This is a good example of the aliasing that occurs when a three-dimensional space-time signal is inadequately sampled in time.

By referring to Figure 1.14, we can see that the process of filming (that is, taking pictures of a moving scene equally spaced in time) is equivalent to the “Sample and hold” and “Encoder” blocks. It is interesting to note that in this photograph-shooting context there cannot be any anti-aliasing filter. This is so because one cannot change the way a scene varies in time before filming it. Therefore, depending on how fast a scene is moving, aliasing cannot be avoided.

As seen above, through the sampling process, one is able to represent a two-dimensional moving picture (a three-dimensional signal) as a sequence of photographs. These photographs can be stored and processed. After that, one has to display a moving picture from these photographs. In the cinema, each photograph is usually in the form of a transparency. With the help of a system of lenses, one transparency can be projected on a screen by turning on a lamp behind it. In order to display a sequence of transparencies on the screen, there must be a way of replacing the transparency currently in front of the lamp by the next one. It is easier to do this if the lamp is turned off during the transparency replacement process. This is the same as having a flashing light source projecting the transparencies on the screen such that a different transparency is in front of the light source each time it flashes. By modeling the intensity of a flashing light as an impulse, a light flashing periodically can be represented by a train of impulses, allowing one to depict the moving-picture displaying process as given in Figure 1.19. There, the light intensity of a given point on the screen has been modeled as a train of impulses, each impulse having as amplitude the intensity of the

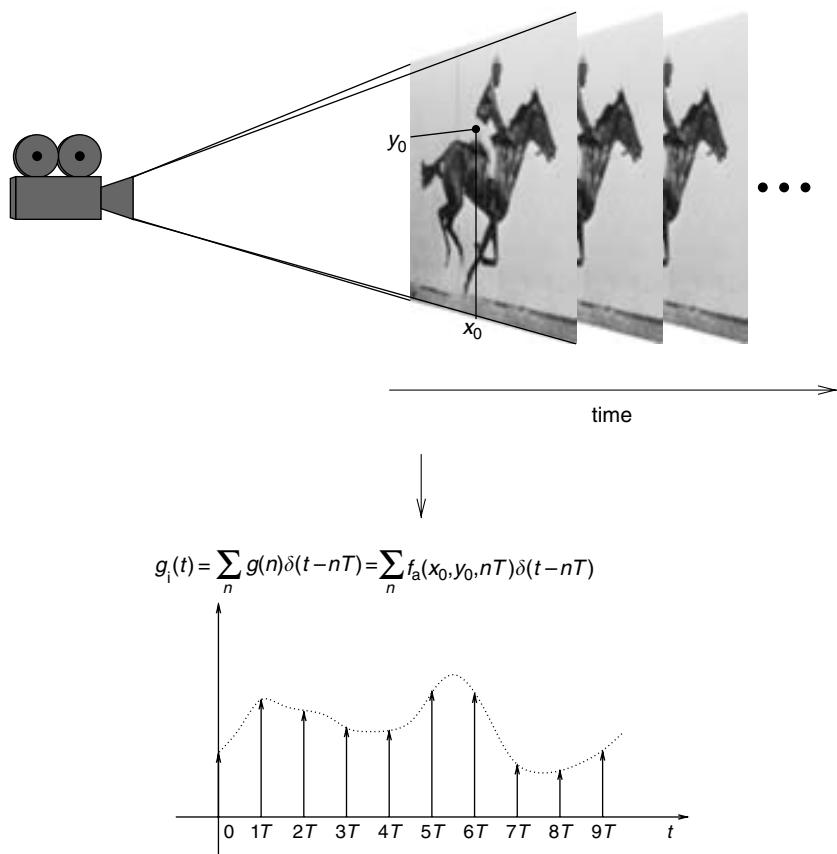


Fig. 1.19.

The film projector is equivalent to replacing the intensity of each sample in a picture by an impulse.

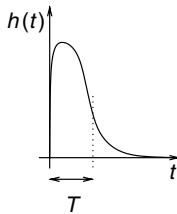


Fig. 1.20.

Response of the human visual system to a flash of light characterizing the persistence of vision phenomenon.

point at a given time. If the light field projected on the screen is $f_i(x, y, t)$, then we have that

$$f_i(x, y, t) = \sum_{n=-\infty}^{\infty} f_a(x, y, nT) \delta(t - nT). \quad (1.202)$$

Thus, using Equation (1.200), the intensity at point (x_0, y_0) is

$$g_i(t) = f_i(x_0, y_0, t) = \sum_{n=-\infty}^{\infty} f_a(x_0, y_0, nT) \delta(t - nT) = \sum_{n=-\infty}^{\infty} g(n) \delta(t - nT), \quad (1.203)$$

which is equivalent to Equations (1.163) and (1.164).

The human visual system enjoys a property called persistence of vision. In a nutshell, if a light is flashed before your eyes, persistence of vision is the reason why you keep seeing the light for some time after it goes off. This is depicted in Figure 1.20, where $h(t)$ represents the human visual system response to a flash of light at $t = 0$.

Owing to the persistence of vision, when you look at a sequence of pictures flashing in the screen, you do not actually see the pictures flashing provided that the flashes are frequent enough. Instead, you have the impression that the picture is moving continuously. In mathematical terms, the function $h(t)$ in Figure 1.20 is the response of the human visual system to an impulse of light $\delta(t)$. Therefore, from Equation (1.203), the response of the human visual system to a point (x_0, y_0) flashing on the cinema screen is given by

$$g_r(t) = g_i(t) * h(t) = \sum_{n=-\infty}^{\infty} g(n) h(t - nT) = \sum_{n=-\infty}^{\infty} f_a(x, y, nT) h(t - nT) \quad (1.204)$$

which is equivalent to Equations (1.185) and (1.186). By referring to Figure 1.21, the human visual system replaces each impulse by the function $h(t)$ in Figure 1.20, thus perceiving the sequence of light impulses as the right-hand-side signal in Figure 1.21.

At this point, we can make interesting design considerations about the cinema system. As we have seen before, the time interval between photographs is limited by the aliasing effect, and is given by Equation (1.201). In a regular signal processing system, one would choose the impulse response of the reconstructing filter according to Equations (1.183) and (1.184), so that the spectrum repetitions are filtered out from the train of impulses and the original analog signal is recovered. However, in cinema, the impulse response

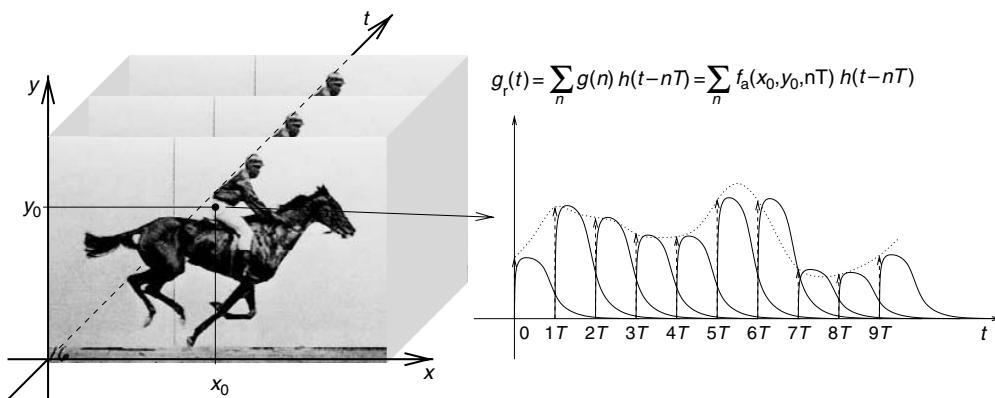


Fig. 1.21. Through persistence of vision, the human visual system replaces each flash of light by a function $h(t)$.

of the reconstructing filter is given by the persistence of vision, which is a physiological characteristic, and thus cannot be changed. Specifically in modern cinema, in order to avoid aliasing in most scenes of interest, it is enough that we take pictures at a rate of 24 pictures per second; that is, $T \leq 1/24$ s. Therefore, according to Equation (1.183), in order to filter out the spectrum repetitions, the bandwidth of the time impulse response of the human visual system $h(t)$ should be $\Omega_{LP} < 24$ Hz. However, psychovisual tests have determined that, for the human visual system, $h(t)$ is a lowpass function with $\Omega_{LP} \approx 48$ Hz. With this natural cutoff frequency, the spectrum repetitions cannot be filtered out, and thus one loses the impression of a continuous movement. In this case, one perceives that the pictures are flashing, a phenomenon referred to as flickering. In order to avoid this, there are two options. One of them is to halve the sampling period, matching T to $\Omega_{LP} = 48$ Hz. This solution has the disadvantage of requiring twice as many pictures as the ones required to avoid aliasing. This is not cost effective, since only 24 pictures per second are enough to avoid aliasing. The solution employed by modern cinema is to repeat each picture twice so that the interval between pictures is $1/48$ s. This procedure avoids aliasing, allowing one to filter out the spectrum repetitions, as determined in Exercise 1.27. The whole movie-generating process is summarized in Figure 1.22. △

1.7 Random signals

In nature, we are commonly forced to work with signals whose waveforms are not exactly known at each time instant. Some examples may include the outcome of a die, the suit of a card randomly drawn, a resistor value, or a stock price at a particular time. In such cases, even without knowing the exact signal value, one can still extract valuable information about the process of interest using the mathematical tools presented in this section.

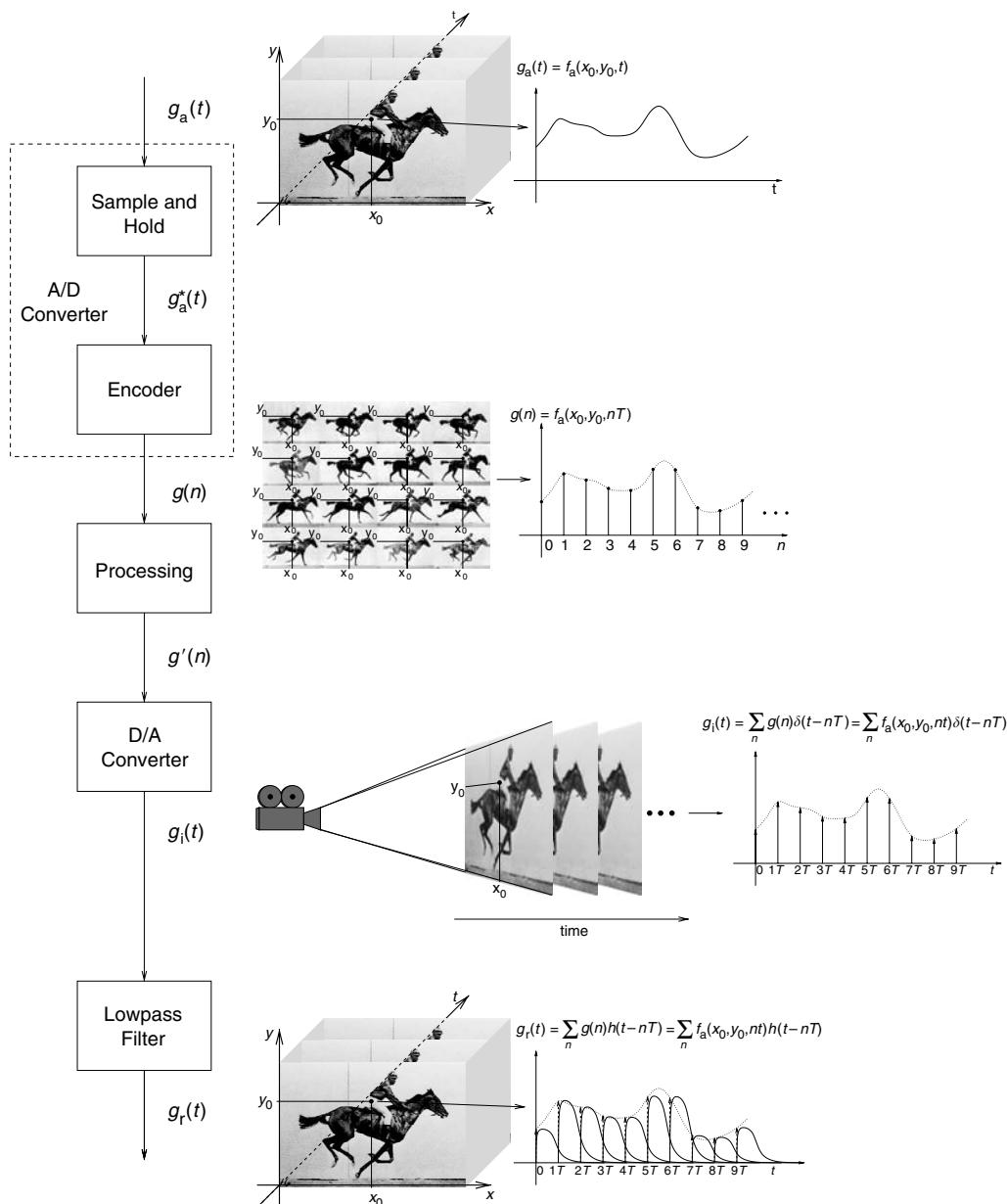


Fig. 1.22. Cinema as a digital signal processing system.

1.7.1 Random variable

A random variable is a mapping of the result of an experiment onto the set of real numbers. By doing so, we can extract numerical information about the problem at hand, as indicated below.

The cumulative distribution function (CDF) is determined by the probability of a given random variable X to be less than or equal to a particular value x ; that is:

$$F_X(x) = P\{X \leq x\}, \quad (1.205)$$

where $P\{\mathcal{E}\}$ denotes the probability of the event \mathcal{E} . The corresponding probability density function (PDF) is given by

$$f_X(x) = \frac{dF_X(x)}{dx}. \quad (1.206)$$

From their definitions, it is simple to infer that the CDF and PDF present the following properties:

- $\lim_{x \rightarrow -\infty} F_X(x) = 0$
- $\lim_{x \rightarrow +\infty} F_X(x) = 1$
- $F_X(x)$ is a nondecreasing function with x , such that $0 \leq F_X(x) \leq 1$
- $f_X(x) \geq 0$, for all x
- $\int_{-\infty}^{\infty} f_X(x) dx = 1$.

These two functions unify the statistical treatment of both discrete- and continuous-valued random variables, despite the fact that each variable type has probability functions with different characteristics (discrete random variables, for instance, present impulsive PDFs).

There is a plethora of PDFs in the associated literature (Papoulis, 1977; Peebles, 2000) to characterize all kinds of random phenomena. However, for our purposes, the most interesting PDFs are the uniform (continuous $u_{X,c}(x)$ and discrete $u_{X,d}(x)$) and Gaussian $\phi_X(x)$ distributions, defined as

$$u_{X,c}(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}, \quad (1.207)$$

$$u_{X,d}(x) = \begin{cases} \frac{1}{M} \delta(x - x_i), & x_i = x_1, x_2, \dots, x_M \\ 0, & \text{otherwise} \end{cases}, \quad (1.208)$$

and

$$\phi_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad (1.209)$$

respectively, where a, b, M, μ , and σ^2 are the corresponding parameters for these distributions. Using the PDF, we can extract meaningful measures that characterize the statistical behavior of a given random variable. In particular, we define the i th-order moment as

$$E\{X^i\} = \int_{-\infty}^{\infty} x^i f_X(x) dx, \quad (1.210)$$

where the two most important special cases are the first-order moment (statistical mean or statistical expectation) $\mu_X = E\{X\}$ and the second-order moment (energy or mean-squared value) $E\{X^2\}$. If Y is a function of the random variable X (that is, $Y = g(X)$), then for the first-order moment of Y we can write that

$$\mu_Y = E\{Y\} = \int_{-\infty}^{\infty} y f_Y(y) dy = \int_{-\infty}^{\infty} g(x) f_X(x) dx. \quad (1.211)$$

We are often interested in measures that are not influenced by a random variable's mean value. For these cases, we may also define the so-called i th-order central moments as

$$E\{(X - \mu_X)^i\} = \int_{-\infty}^{\infty} (x - \mu_X)^i f_X(x) dx. \quad (1.212)$$

By far the most important central moment is the second-order one, also known as the variance $\sigma_X^2 = E\{(X - \mu_X)^2\}$, the square-root of which is referred to as the standard deviation. Larger variances indicate that the value of the random variable is more spread around the mean, whereas smaller σ_X^2 occurs when the values of X are more concentrated around its mean. A simple algebraic development indicates that

$$\sigma_X^2 = E\{(X - \mu_X)^2\} = E\{X^2\} - 2E\{X\mu_X\} + \mu_X^2 = E\{X^2\} - \mu_X^2. \quad (1.213)$$

Example 1.16. Determine the values for the statistical mean, energy, and variance of a random variable X characterized by the discrete uniform PDF given in Equation (1.208) with $x_i = 1, 2, \dots, M$.

Solution

Using the corresponding definitions, where the integral operations on the delta functions degenerate into simple sums, we get

$$\mu_X = \sum_{i=1}^M i \frac{1}{M} = \frac{1}{M} \left[\frac{1}{2} M(M+1) \right] = \frac{M+1}{2} \quad (1.214)$$

$$E\{X^2\} = \sum_{i=1}^6 i^2 \frac{1}{M} = \frac{1}{M} \left[\frac{M(M+1)(2M+1)}{6} \right] = \frac{2M^2 + 3M + 1}{6}. \quad (1.215)$$

The variance can be determined by the relationship expressed in Equation (1.213), yielding

$$\sigma_X^2 = \frac{M^2 - 1}{12}. \quad (1.216)$$

All statistics presented above are based on the PDF of the random variable. In that sense, we are exchanging the requirement of knowing the value of a variable by the requirement of knowing its PDF. This may seem like trading one problem for another. However, in several cases, even though we are not able to determine the value of X , we can still determine its PDF. One simple example is the outcome of casting a fair die, which cannot be predicted but

whose associated PDF is easily determined. This allows one to know important statistics associated with the event, as illustrated in this example by considering $M = 6$. \triangle

When dealing with two random variables X and Y simultaneously, we may define the joint CDF by

$$F_{X,Y}(x,y) = P\{X \leq x, Y \leq y\} \quad (1.217)$$

and the corresponding joint PDF as

$$f_{X,Y}(x,y) = \frac{\partial^2 F_{X,Y}(x,y)}{\partial x \partial y}. \quad (1.218)$$

The two variables are said to be statistically independent if we may write

$$f_{X,Y}(x,y) = f_X(x)f_Y(y), \quad (1.219)$$

where independence indicates that the outcome of a given variable does not affect the value of the other. The concept of moments is easily extended to two random variables X and Y by defining the joint moment of order (i,j) as

$$E\{X^i Y^j\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^i y^j f_{X,Y}(x,y) dx dy \quad (1.220)$$

and the joint central moment of order (i,j) as

$$E\{(X - \mu_X)^i (Y - \mu_Y)^j\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_X)^i (y - \mu_Y)^j f_{X,Y}(x,y) dx dy. \quad (1.221)$$

When the orders are $(1,1)$, we get the cross-correlation $r_{X,Y}$ and the cross-covariance $c_{X,Y}$ between X and Y ; that is:

$$\begin{aligned} r_{X,Y} &= E\{XY\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy f_{X,Y}(x,y) dx dy \end{aligned} \quad (1.222)$$

$$\begin{aligned} c_{X,Y} &= E\{(X - \mu_X)(Y - \mu_Y)\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_X)(y - \mu_Y) f_{X,Y}(x,y) dx dy \end{aligned} \quad (1.223)$$

In the case of complex random variables, the cross-correlation is given by $r_{X,Y} = E\{XY^*\}$, where the superscript asterisk denotes the complex conjugation operation. For the sake of simplicity, the remainder of this chapter is restricted to real signals.

1.7.2 Random processes

A random process is an ordered collection of random variables. The most common form of ordering the set of random variables is associating each of them with different time instants, giving rise to the common interpretation of a random process as a set of random variables ordered in time.

The whole concept can be better understood with the assistance of a practical example. Consider the complete set of utterances of the vowel /A/ by a particular person. We refer to the m th discrete-time utterance as $a_m(n)$, for $m = 1, 2, \dots$. For a given m it can be regarded as a sample or a realization of the random process $\{A\}$. The entire set of realizations is referred to as the ensemble. In the context of signal processing, a realization may also be referred to as a random signal. If we consider a particular time instant n_1 , the value of each random signal at n_1 defines the random variable $A(n_1)$; that is:

$$A(n_1) = \{a_1(n_1), a_2(n_1), \dots\}. \quad (1.224)$$

Naturally, for the whole random process $\{A\}$ we can define an infinite number of random variables $A(n)$, each one associated with a particular time instant n and all of them obeying the intrinsic order established by the time variable.

In a random process $\{X\}$, each random variable $X(n)$ has its own PDF $f_{X(n)}(x)$, which can be used to determine the i th-order moments of the associated random variable. Similarly, two random variables $X(n_1)$ and $X(n_2)$ of the same random process have a joint PDF defined by

$$f_{X(n_1), X(n_2)}(x_1, x_2) = \frac{\partial^2 P\{X(n_1) \leq x_1, X(n_2) \leq x_2\}}{\partial x_1 \partial x_2}. \quad (1.225)$$

Based on this function, we can define the so-called autocorrelation function of the random process,

$$\begin{aligned} R_X(n_1, n_2) &= r_{X(n_1), X(n_2)} \\ &= E\{X(n_1)X(n_2)\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f_{X(n_1), X(n_2)}(x_1, x_2) dx_1 dx_2. \end{aligned} \quad (1.226)$$

As its name indicates, the autocorrelation function represents the statistical relation between two random variables (associated with the time-instants n_1 and n_2) of a given random process.

A random process is called wide-sense stationary (WSS) if its mean value and autocorrelation function present the following properties:

$$E\{X(n)\} = c, \text{ for all } n \quad (1.227)$$

$$R_X(n, n + v) = R_X(v), \text{ for all } n, v. \quad (1.228)$$

The first relation indicates that the mean value of all random variables $X(n)$ is constant throughout the entire process. The second property means that the autocorrelation function

of a WSS process depends only on the time interval between two random variables, not on their absolute time instants. Overall, the two relationships indicate that the first- and second-order statistics of the random process do not change over time, indicating the (wide-sense) stationary nature of the process from a statistical point of view. If the statistics of all orders are invariant in time, the process is referred to as strict-sense stationary (SSS). Hence, SSS processes are also WSS, whereas the opposite is not necessarily true. Since it is very hard to verify the invariance property for all orders, in practice we often work with the WSS characterization, which only requires first- and second-order invariance. Later, we verify that WSS random processes, for instance, can be well characterized also in the frequency domain. The stationarity concept can be extended to two distinct processes $\{X\}$ and $\{Y\}$, which are said to be jointly WSS if their first-order moments are constant for all n and if their cross-correlation function $R_{XY}(n, n + \nu) = E\{X(n)Y(n + \nu)\}$ is only a function of ν .

Example 1.17. Consider the random process $\{X\}$ described by its m th realization as

$$x_m(n) = \cos(\omega_0 n + \Theta_m), \quad (1.229)$$

where Θ is a continuous random variable with uniform PDF within the interval $[0, 2\pi]$. Determine the statistical mean and the autocorrelation function for this process, verifying whether or not it is WSS.

Solution

By writing that $X(n) = g(\Theta)$, from Equations (1.211) and (1.226) respectively we get

$$\begin{aligned} E\{X(n)\} &= \int_0^{2\pi} g(\theta) f_\Theta(\theta) d\theta \\ &= \int_0^{2\pi} \cos(\omega_0 n + \theta) \frac{1}{2\pi} d\theta \\ &= \frac{1}{2\pi} \sin(\omega_0 n + \theta) \Big|_{\theta=0}^{\theta=2\pi} \\ &= \frac{1}{2\pi} [\sin(\omega_0 n + 2\pi) - \sin(\omega_0 n)] \\ &= 0, \text{ for all } n \end{aligned} \quad (1.230)$$

$$\begin{aligned} R_X(n_1, n_2) &= E\{\cos(\omega_0 n_1 + \Theta) \cos(\omega_0 n_2 + \Theta)\} \\ &= \frac{1}{2} E\{\cos(\omega_0 n_1 - \omega_0 n_2)\} + \frac{1}{2} E\{\cos(\omega_0 n_1 + \omega_0 n_2 + 2\Theta)\}. \end{aligned} \quad (1.231)$$

In this last development for $R_X(n_1, n_2)$, the first term is not random and the corresponding expected value operator can be dropped, whereas for the second term one has that

$$\begin{aligned} \frac{1}{2} E\{\cos(\omega_0 n_1 + \omega_0 n_2 + 2\Theta)\} &= \frac{1}{2} \int_0^{2\pi} \cos(\omega_0 n_1 + \omega_0 n_2 + 2\theta) \frac{1}{2\pi} d\theta \\ &= \frac{1}{8\pi} \sin(\omega_0 n_1 + \omega_0 n_2 + 2\theta) \Big|_{\theta=0}^{\theta=2\pi} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{8\pi} [\sin(\omega_0 n_1 + \omega_0 n_2 + 4\pi) - \sin(\omega_0 n_1 + \omega_0 n_2)] \\
&= 0
\end{aligned} \tag{1.232}$$

and thus

$$R_X(n_1, n_2) = \frac{1}{2} \cos(\omega_0 n_1 - \omega_0 n_2) = \frac{1}{2} \cos[\omega_0(n_1 - n_2)], \text{ for all } n_1, n_2. \tag{1.233}$$

Therefore, from Equations (1.230) and (1.233), we conclude that the random process $\{X\}$ is WSS. \triangle

The N th-order autocorrelation matrix \mathbf{R}_X of a WSS random process $\{X\}$ is defined as

$$\mathbf{R}_X = \begin{bmatrix} R_X(0) & R_X(1) & R_X(2) & \cdots & R_X(N-1) \\ R_X(-1) & R_X(0) & R_X(1) & \cdots & R_X(N-2) \\ R_X(-2) & R_X(-1) & R_X(0) & \cdots & R_X(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_X(1-N) & R_X(2-N) & R_X(3-N) & \cdots & R_X(0) \end{bmatrix} \tag{1.234}$$

with $R_X(v) = E\{X(n)X(n+v)\}$ as before.

We classify a random process as ergodic if all statistics of the ensemble can be determined by averaging on a single realization across the different samples. For example, when the discrete variable corresponds to time, time averaging is performed in order to determine the statistics. Despite being a quite strong assumption, ergodicity is commonly resorted to in situations where only a few realizations or possibly one realization of the random process is available. In such cases, we may drop the realization index m , denoting the available random signal by just $x(n)$. The mean value, variance, autocorrelation function, and so on of the entire process $\{X\}$ are estimated based solely on $x(n)$.

1.7.3 Filtering a random signal

Consider the input–output relationship of a linear time-invariant system, described by its impulse response $h(n)$, given by the convolution sum

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k). \tag{1.235}$$

If $x(n)$ is a random signal, the nature of the output signal shall also be random. Let us proceed to characterize the output signal $y(n)$ when $x(n)$ is a WSS random signal.

Determining the mean value of $y(n)$, one gets

$$\begin{aligned} E\{y(n)\} &= E \left\{ \sum_{k=-\infty}^{\infty} x(n-k)h(k) \right\} \\ &= \sum_{k=-\infty}^{\infty} E\{x(n-k)\}h(k) \\ &= E\{x(n)\} \sum_{k=-\infty}^{\infty} h(k), \end{aligned} \quad (1.236)$$

where we have used the facts that $h(n)$ is a deterministic signal and that $E\{x(n)\}$ is constant for all n , since $\{X\}$ is assumed to be WSS. An interesting interpretation of the above equation comes from the fact that the output $w(n)$ of a system to a constant input $v(n) = c$ is

$$w(n) = \sum_{k=-\infty}^{\infty} h(k)v(n-k) = c \sum_{k=-\infty}^{\infty} h(k). \quad (1.237)$$

This means that the output of a linear system to a constant input is also constant, and equal to the input multiplied by the constant

$$H_0 = \sum_{k=-\infty}^{\infty} h(k), \quad (1.238)$$

which can be regarded as the system DC gain. Hence, Equation (1.236) indicates that the statistical mean of the output signal is the mean value of the WSS input signal multiplied by the system DC gain, which is quite an intuitive result.

The autocorrelation function of the output signal is given by

$$\begin{aligned} R_Y(n_1, n_2) &= E\{y(n_1)y(n_2)\} \\ &= E \left\{ \left(\sum_{k_1=-\infty}^{\infty} x(n_1 - k_1)h(k_1) \right) \left(\sum_{k_2=-\infty}^{\infty} x(n_2 - k_2)h(k_2) \right) \right\} \\ &= E \left\{ \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(n_1 - k_1)x(n_2 - k_2)h(k_1)h(k_2) \right\} \\ &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} E\{x(n_1 - k_1)x(n_2 - k_2)\}h(k_1)h(k_2) \\ &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} R_X(n_1 - k_1, n_2 - k_2)h(k_1)h(k_2). \end{aligned} \quad (1.239)$$

Since $\{X\}$ is WSS, then one may write that

$$R_Y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} R_X((n_1 - n_2) - k_1 + k_2)h(k_1)h(k_2), \quad (1.240)$$

which is also a function of $v = (n_1 - n_2)$ for all time instants n_1, n_2 (see Exercise 1.31). From Equations (1.236) and (1.240), one concludes that if the input $\{X\}$ is WSS, then the output process $\{Y\}$ to a linear time-invariant system is also WSS.

The scope of random variables and processes is quite vast, and this section can only present the tip of the iceberg on these matters. The main aspect that the reader should keep in mind is that even though it may not be feasible to determine the exact value of a given signal, one can still determine its PDF or autocorrelation function, and extract many pieces of information (statistical mean, variance, stationarity behavior, and so on) about it using the methodology introduced here. In several cases, the results from these analyses are all that one needs to describe a given signal or process, as will be verified in several parts of this book.

1.8 Do-it-yourself: discrete-time signals and systems

In this section, we explore the realm of discrete-time signals using MATLAB. You are encouraged to try out the experiments described below, taking note of everything you learn. Extending the experiments, following one's own curiosity, is highly desirable, since there is no better way to learn signal processing other than by doing signal processing. And that is exactly what MATLAB allows us to do in a straightforward manner.

Experiment 1.1

In Example 1.7 we were able to determine a closed-form expression for the impulse response associated with the difference Equation (1.87). A numerical solution for this problem can be determined, for $\alpha = 1.15$ and $0 \leq n \leq 30$, using the following MATLAB commands:

```
alpha = 1.15; N = 30;
x = [1 zeros(1,N)];
y = filter(1,[1 -1/alpha],x);
stem(y);
```

This yields the plot seen in Figure 1.23.

In general, the MATLAB command

```
y = filter([b_0 b_1 ... b_M], [1 a_1 ... a_N], x, z_i);
```

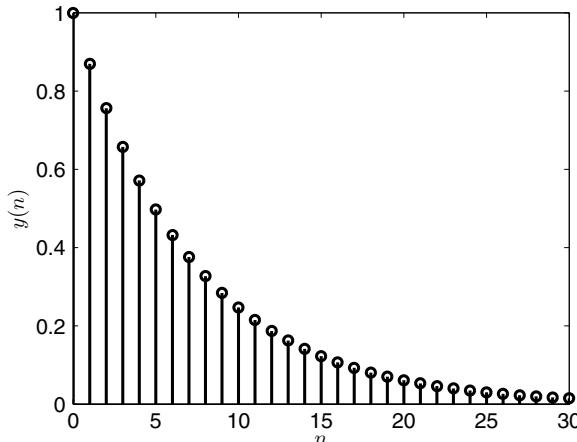


Fig. 1.23. Solution of difference Equation (1.87) in Example 1.7.

determines the solution of the general difference equation

$$y(n) + a_1 y(n-1) + \dots + a_N y(n-N) = b_0 x(n) + b_1 x(n-1) + \dots + b_M x(n-M) \quad (1.241)$$

when the input signal is provided in the vector \mathbf{x} and the vector $\mathbf{z_i}$ contains its initial conditions.

Experiment 1.2

The process of sampling can be seen as a mapping of a continuous-time function into a set of discrete-time samples. In general, however, there are infinite functions that can generate the same set of samples. To illustrate such a notion, consider a general function $f_1(t)$. Using a sampling frequency of f_s samples per second, the sampling process yields the discrete-time function $f_1(nT_s)$, with $T_s = 1/f_s$ and integer n .

Sampling any function of the form $f_2(t) = f_1(\alpha t)$, with any positive α , using a sampling frequency $f'_s = \alpha f_s$, we get $T'_s = 1/f'_s = T_s/\alpha$. Hence, $f_2(nT'_s) = f_1(\alpha(nT'_s)) = f_1(nT_s)$, which corresponds to the same set of samples as before.

Therefore, in general, a given set of samples does not specify the original continuous-time function in a unique way. To reduce this uncertainty, we must specify the sampling frequency employed to generate the given samples. By doing so, the algebraic reasoning above breaks down and we eliminate (almost) all continuous-time candidate functions for a given sample set. There is, however, one last candidate that must be eliminated to avoid ambiguity. Let us illustrate such case by emulating a sampling procedure using MATLAB.

Consider the 3-Hz cosine function $f_1(t) = \cos(2\pi 3t)$ sampled at $F_s = 10$ samples per second, for a 1s time interval, using the MATLAB command:

```
time = 0:0.1:0.9;
f_1 = cos(2*pi*3.*time)
```

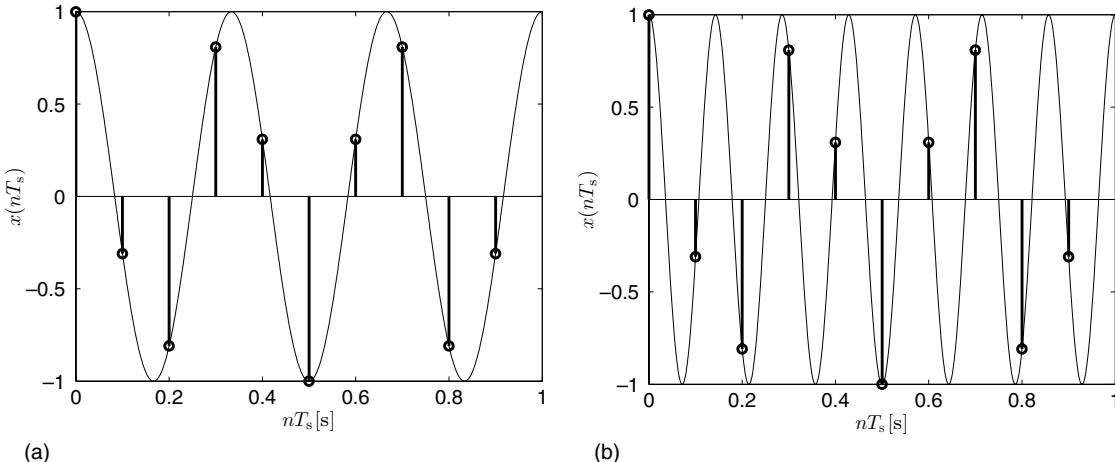


Fig. 1.24. Sampling of cosine functions of frequency f using $F_s = 10$ samples per second: (a) $f = 3$ Hz; (b) $f = 7$ Hz.

An identical sample list can be obtained, with the same sampling rate and time interval, from a 7-Hz cosine function $f_2(t) = \cos(2\pi 7t)$ as given by

```
f_2 = cos(2*pi*7.*time)
```

with the variable `time` specified as above. The resulting samples are shown in Figure 1.24, generated by the following commands:

```
time_aux = 0:0.001:(1-0.001);
figure(1);
stem(time,f_1);
hold on;
plot(time_aux, cos(2*pi*3.*time_aux));
hold off;
figure(2);
stem(time,f_2);
hold on;
plot(time_aux, cos(2*pi*7.*time_aux));
hold off;
```

In this sequence, the `hold on` commands allow plotting more than one function in the same figure and the `time_aux` variable is used to emulate a continuous time counter in the `plot` commands to draw the background functions.

To eliminate the ambiguity illustrated in Figure 1.24, we must refer to the sampling theorem introduced in Section 1.6.2. That result indicates that a 7-Hz cosine function shall not be sampled with $F_s = 10$ Hz, since the minimum sampling frequency in this case should be above $F_s = 14$ Hz. Therefore, if Nyquist's sampling criterion is satisfied, there is only one continuous-time function associated with a given set of discrete-time samples and a particular sampling frequency.

Experiment 1.3

Suppose the signal $x(t) = 5 \cos(2\pi 5t) + 2 \cos(2\pi 50t)$, sampled with $F_s = 1000$ samples per second, as shown in Figure 1.25a, is corrupted by a small amount of noise, forming the signal shown in Figure 1.25b generated by the following commands:

```
amplitude_1 = 5; freq_1 = 5;
amplitude_2 = 2; freq_2 = 50;
F_s = 1000; time = 0:1/F_s:(1-1/F_s);
sine_1 = amplitude_1*sin(2*pi*freq_1.*time);
sine_2 = amplitude_2*sin(2*pi*freq_2.*time);
noise = randn(1,length(time));
x_clean = sine_1 + sine_2;
x_noisy = x_clean + noise;
figure(1);
plot(time,x_clean);
figure(2);
plot(time,x_noisy);
```

In particular, the `randn` command generates the specified number of samples of a pseudo-random signal with Gaussian distribution with zero mean and unit variance.

We can minimize the noisy effect by averaging N successive samples of $x(n) = x_{\text{noisy}}$, implementing the following difference equation:

$$y(n) = \frac{x(n) + x(n-1) + \dots + x(n-N+1)}{N}. \quad (1.242)$$

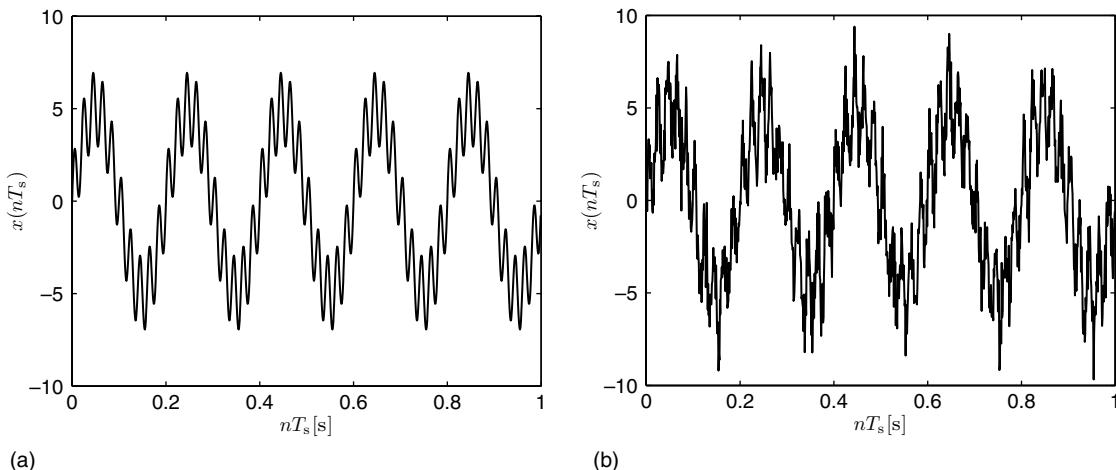


Fig. 1.25. Sum of two sinusoidal components: (a) clean signal; (b) noisy signal.

As mentioned in Experiment 1.1, we can perform such processing by specifying the value of N and using the MATLAB commands

```
b = ones(1,N);
y = filter(b,1,x_noisy);
```

which yield the plots shown in Figure 1.26 for $N = 3$, $N = 6$, $N = 10$, and $N = 20$.

Figure 1.26 indicates that the averaging technique is quite effective at reducing the amount of noise from the corrupted signal. In this case, the larger the value of N , the greater the ability to remove the noise component. If, however, N is too large, as observed in Figure 1.26d, then the averaging procedure almost eliminates the high-frequency sinusoidal component.

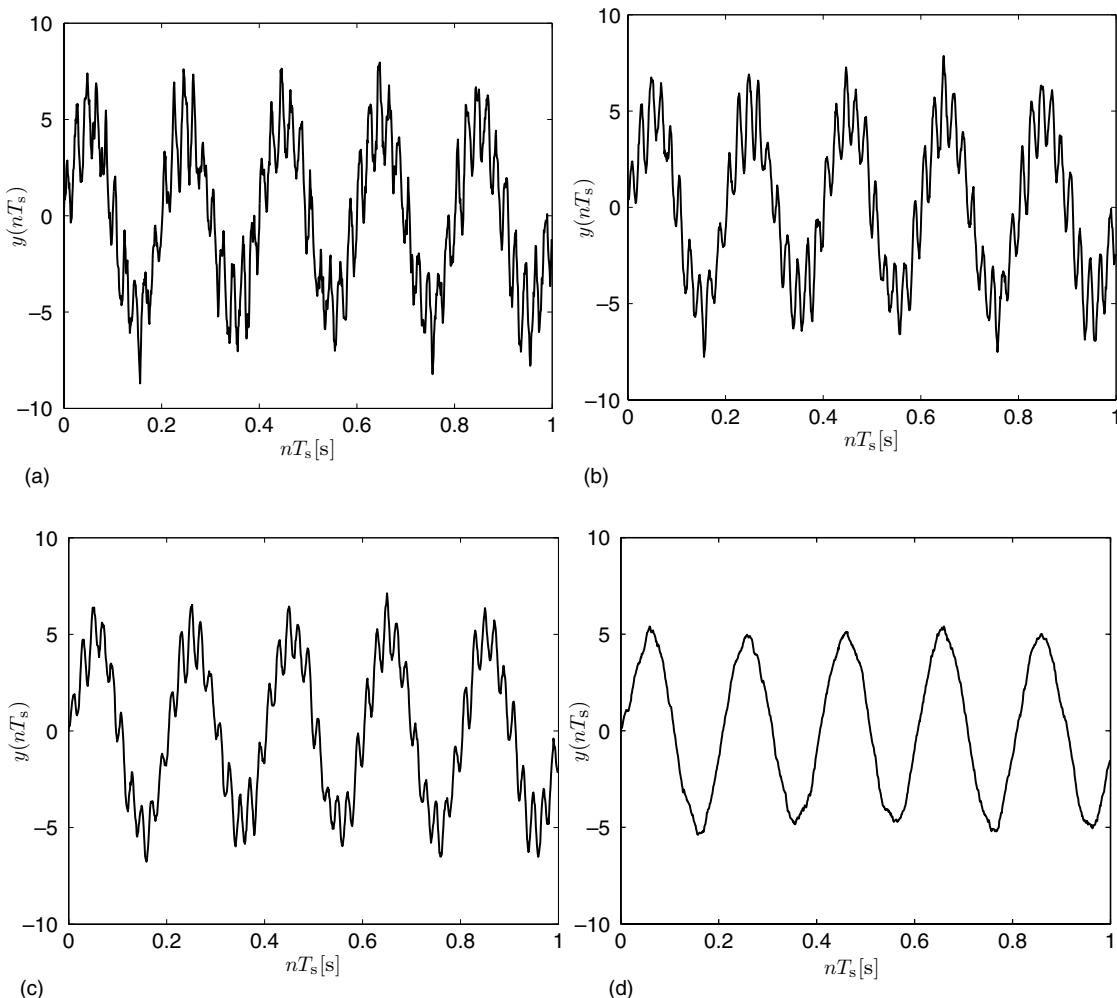


Fig. 1.26. Averaged-in-time signals using N consecutive samples: (a) $N = 3$; (b) $N = 6$; (c) $N = 10$; (d) $N = 20$.

One may then argue whether it is possible to reduce the noise component without affecting the original signal significantly. Perhaps a more elaborate processing may preserve the sinusoidal components better. The theory and design of tools for answering these types of question is the main subject of this textbook. In the following chapters, several techniques for processing a wide range of signals are investigated. Although intuition is important in some practical situations, as illustrated in this experiment, our presentation follows a formal and technically justified path. In the end, the reader may be able not only to employ the methods presented throughout the book, but also to understand them, selecting the proper tool for each particular application.

1.9 Discrete-time signals and systems with MATLAB

The Signal Processing toolbox of MATLAB has several functions that aid in the processing of discrete signals. In this section we give a brief overview of the ones that are most closely related to the material in this chapter.

- **stem:** Plots a data sequence as stems from the x axis terminated in circles, as in the plots of discrete-time signals shown in this chapter.

Input parameters:

- the ordinate values t at which the data is to be plotted;
- the data values x to be plotted.

Example:

```
t=[0:0.1:2]; x=cos(pi*t+0.6); stem(t,x);
```

- **conv:** Performs the discrete convolution of two sequences.

Input parameters: the vectors a and b holding the two sequences.

Output parameter: the vector y holding the convolution of a and b .

Example:

```
a=[1 1 1 1 1]; b=[1 2 3 4 5 6 7 8 9];
c=conv(a,b); stem(c);
```

- **impz:** Computes the impulse response of an initially relaxed system described by a difference equation.

Input parameters (refer to Equation (1.63)):

- a vector b containing the values of b_l , for $l = 0, 1, \dots, M$;
- a vector a containing the values of a_i , for $i = 0, 1, \dots, N$;
- the desired number of samples n of the impulse response.

Output parameter: a vector h holding the impulse response of the system.

Example:

```
a=[1 -0.22 -0.21 0.017 0.01]; b=[1 2 1];
h=impz(b,a,20); stem(h);
```

1.10 Summary

In this chapter we have seen the basic concepts referring to discrete-time systems and have introduced the properties of linearity, time invariance, and causality. Systems presenting these properties are of particular interest due to the numerous tools that are available to analyze and to design them. We also discussed the concept of stability. Linear time-invariant systems have been characterized using convolution sums. We have also introduced difference equations as another way to characterize discrete-time systems. The conditions necessary to carry out the discrete-time processing of continuous-time signals without losing the information they carry have been studied. Random signals have also been introduced. Finally, we presented some hands-on experiments and some MATLAB functions that are useful in the processing of discrete-time signals seen in this chapter.

1.11 Exercises

1.1 Characterize the systems below as linear/nonlinear, causal/noncausal and time invariant/time varying:

- (a) $y(n) = (n + a)^2 x(n + 4)$
- (b) $y(n) = ax(n + 1)$
- (c) $y(n) = x(n + 1) + x^3(n - 1)$
- (d) $y(n) = x(n) \sin(\omega n)$
- (e) $y(n) = x(n) + \sin(\omega n)$
- (f) $y(n) = \frac{x(n)}{x(n+3)}$
- (g) $y(n) = y(n - 1) + 8x(n - 3)$
- (h) $y(n) = 2ny(n - 1) + 3x(n - 5)$
- (i) $y(n) = n^2y(n + 1) + 5x(n - 2) + x(n - 4)$
- (j) $y(n) = y(n - 1) + x(n + 5) + x(n - 5)$
- (k) $y(n) = (2u(n - 3) - 1)y(n - 1) + x(n) + x(n - 1)$.

1.2 For each of the discrete signals below, determine whether they are periodic or not. Calculate the periods of those that are periodic.

- (a) $x(n) = \cos^2\left(\frac{2\pi}{15}n\right)$
- (b) $x(n) = \cos\left(\frac{4\pi}{5}n + \frac{\pi}{4}\right)$
- (c) $x(n) = \cos\left(\frac{\pi}{27}n + 31\right)$
- (d) $x(n) = \sin(100n)$
- (e) $x(n) = \cos\left(\frac{11\pi}{12}n\right)$
- (f) $x(n) = \sin[(5\pi + 1)n]$.

1.3 Consider the system whose output $y(m)$ is described as a function of the input $x(m)$ by the following difference equations:

- (a) $y(m) = \sum_{n=-\infty}^{\infty} x(n)\delta(m - nN)$
- (b) $y(m) = x(m) \sum_{n=-\infty}^{\infty} \delta(m - nN)$.

Determine whether the systems are linear and/or time invariant.

1.4 Compute the convolution sum of the following pairs of sequences:

$$(a) x(n) = \begin{cases} 1, & 0 \leq n \leq 4 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad h(n) = \begin{cases} a^n, & 0 \leq n \leq 7 \\ 0, & \text{otherwise} \end{cases}$$

$$(b) x(n) = \begin{cases} 1, & 0 \leq n \leq 2 \\ 0, & 3 \leq n \leq 6 \\ 1, & 7 \leq n \leq 8 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad h(n) = \begin{cases} n, & 1 \leq n \leq 4 \\ 0, & \text{otherwise} \end{cases}$$

$$(c) x(n) = \begin{cases} a(n), & n \text{ even} \\ 0, & n \text{ odd} \end{cases} \quad \text{and} \quad h(n) = \begin{cases} \frac{1}{2}, & n = -1 \\ 1, & n = 0 \\ \frac{1}{2}, & n = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Check the results of item (b) using the MATLAB function `conv`.

1.5 For the sequence

$$x(n) = \begin{cases} 1, & 0 \leq n \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

compute $y(n) = x(n) * x(n) * x(n) * x(n)$. Check your results using the MATLAB function `conv`.

- 1.6 Show that $x(n) = a^n$ is an eigenfunction of a linear time-invariant system by computing the convolution summation of $x(n)$ and the impulse response of the system $h(n)$. Determine the corresponding eigenvalue.
- 1.7 Supposing that all systems in Figure 1.27 are linear and time invariant, compute $y(n)$ as a function of the input and the impulse responses of each system.

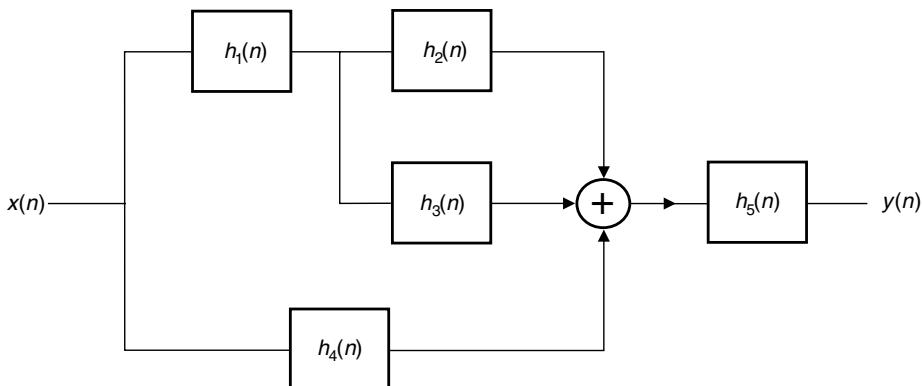


Fig. 1.27. Linear time-invariant system.

- 1.8 We define the even and odd parts of a sequence $x(n)$, $\mathcal{E}\{x(n)\}$ and $\mathcal{O}\{x(n)\}$ respectively, as

$$\mathcal{E}\{x(n)\} = \frac{x(n) + x(-n)}{2}$$

$$\mathcal{O}\{x(n)\} = \frac{x(n) - x(-n)}{2}.$$

Show that

$$\sum_{n=-\infty}^{\infty} x^2(n) = \sum_{n=-\infty}^{\infty} \mathcal{E}\{x(n)\}^2 + \sum_{n=-\infty}^{\infty} \mathcal{O}\{x(n)\}^2.$$

- 1.9 Find one solution for each of the difference equations below:

- (a) $y(n) + 2y(n-1) + y(n-2) = 0$, $y(0) = 1$ and $y(1) = 0$
 (b) $y(n) + y(n-1) + 2y(n-2) = 0$, $y(-1) = 1$ and $y(0) = 1$.

- 1.10 Find the general solution for the difference equation in Example 1.9 when $a = b$.

- 1.11 Determine the solutions of the difference equations below, supposing that the systems they represent are initially relaxed:

- (a) $y(n) - \frac{1}{\sqrt{2}}y(n-1) + y(n-2) = 2^{-n} \sin\left(\frac{\pi}{4}n\right) u(n)$
 (b) $4y(n) - 2\sqrt{3}y(n-1) + y(n-2) = \cos\left(\frac{\pi}{6}n\right) u(n)$
 (c) $y(n) + 2y(n-1) + y(n-2) = 2^n u(-n)$
 (d) $y(n) - \frac{5}{6}y(n-1) + y(n-2) = (-1)^n u(n)$
 (e) $y(n) + y(n-3) = (-1)^n u(-n)$.
- 1.12 Write a MATLAB program to plot the samples of the solutions of the difference equations in Exercise 1.9 from $n = 0$ to $n = 20$.
- 1.13 Show that a system described by Equation (1.63) is linear if and only if the auxiliary conditions are zero. Show also that the system is time invariant if the zero auxiliary conditions are defined for consecutive samples prior to the application of any input.
- 1.14 Compute the impulse responses of the systems below:
- (a) $y(n) = 5x(n) + 3x(n-1) + 8x(n-2) + 3x(n-4)$
 (b) $y(n) + \frac{1}{3}y(n-1) = x(n) + \frac{1}{2}x(n-1)$
 (c) $y(n) - 3y(n-1) = x(n)$
 (d) $y(n) + 2y(n-1) + y(n-2) = x(n)$.
- 1.15 Write a MATLAB program to compute the impulse responses of the systems described by following difference equations:
- (a) $y(n) + y(n-1) + y(n-2) = x(n)$
 (b) $4y(n) + y(n-1) + 3y(n-2) = x(n) + x(n-4)$.
- 1.16 Determine the impulse response of the following recursive system:

$$y(n) - y(n-1) = x(n) - x(n-5).$$

- 1.17 Determine the steady-state response of the system governed by the following difference equation:

$$12y(n) - 7y(n-1) + y(n-2) = \sin\left(\frac{\pi}{3}n\right)u(n).$$

- 1.18 Determine the steady-state response for the input $x(n) = \sin(\omega n)u(n)$ of the filters described by

- (a) $y(n) = x(n-2) + x(n-1) + x(n)$
- (b) $y(n) - \frac{1}{2}y(n-1) = x(n)$
- (c) $y(n) = x(n-2) + 2x(n-1) + x(n).$

What are the amplitude and phase of $y(n)$ as a function of ω in each case?

- 1.19 Write a MATLAB program to plot the solution to the three difference equations in Exercise 1.18 for $\omega = \pi/3$ and $\omega = \pi$.

- 1.20 Discuss the stability of the systems described by the impulse responses below:

- (a) $h(n) = 2^{-n}u(n)$
- (b) $h(n) = 1.5^n u(n)$
- (c) $h(n) = 0.1^n$
- (d) $h(n) = 2^{-n}u(-n)$
- (e) $h(n) = 10^n u(n) - 10^n u(n-10)$
- (f) $h(n) = 0.5^n u(n) - 0.5^n u(4-n).$

- 1.21 Show that $X_i(j\Omega)$ in Equation (1.170) is a periodic function of Ω with period $2\pi/T$.

- 1.22 Suppose we want to process the continuous-time signal

$$x_a(t) = 3 \cos(2\pi 1000t) + 7 \sin(2\pi 1100t)$$

using a discrete-time system. The sampling frequency used is 4000 samples per second. The discrete-time processing carried out on the signal samples $x(n)$ is described by the following difference equation.

$$y(n) = x(n) + x(n-2).$$

After the processing, the samples of the output $y(n)$ are converted back to continuous-time form using Equation (1.188). Give a closed-form expression for the processed continuous-time signal $y_a(t)$. Interpret the effect this processing has on the input signal.

- 1.23 Write a MATLAB program to perform a simulation of the solution of Exercise 1.22.

Hints:

- (i) Simulate the continuous-time signals $x_a(t)$ in MATLAB using sequences obtained by sampling them at 100 times the sampling frequency, that is, as $f_s(m) = f_a(m(T/100))$. The sampling process to generate the discrete-time signal is then equivalent to keeping 1 out of 100 samples of the original sequence, forming the signal $f(n) = f_s(100n) = f_a(nT)$.

- (ii) The interpolation in Equation (1.188) can be approximated by truncating each function

$$\frac{\sin\left[\pi\left(\frac{t}{T} - n\right)\right]}{\pi\left(\frac{t}{T} - n\right)},$$

considering them to be zero outside the time interval $nT - 10T \leq t \leq nT + 10T$.

Note that, since we are truncating the interpolation functions, the summation in Equation (1.188) needs to be carried out only for a few terms.

- 1.24 In the conversion of a discrete-time signal to a continuous-time signal, the practical D/A converters, instead of generating impulses at the output, generate a series of pulses $g(t)$ described by

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(n)g(t - nT), \quad (1.243)$$

where $x(n) = x_a(nT)$. For example, in the sample-and-hold operation, which is shown in Figure 1.15, the train of impulses has been replaced by a train of pulses. Assume now that the function $g(t)$ is

$$g(t) = \begin{cases} \frac{T-|t|}{T}, & -T \leq t \leq T \\ 0, & \text{otherwise} \end{cases}.$$

For the above choice of pulse $g(t)$:

- (a) Determine how could the effect of this interpolation be represented in the time domain (see Figure 1.15)?
 - (b) Determine an expression for the Fourier transform of $x_p(t)$ in Equation (1.243) as a function of the Fourier transform of $x_a(t)$.
 - (c) Determine the frequency response of an ideal lowpass filter that outputs $x_a(t)$ when $x_p(t)$ is applied to its input.
- 1.25 Suppose a discrete-time signal $x(n)$ is obtained by sampling a bandlimited, continuous-time signal $x_a(t)$ so that there is no aliasing. Prove that the energy of $x_a(t)$ is equal to the energy of $x(n)$ multiplied by the sampling period.
- 1.26 Given a sinusoid $y(t) = A \cos(\Omega_c t)$, using Equation (1.170), show that if $y(t)$ is sampled with a sampling frequency slightly above the Nyquist frequency (that is, $\Omega_s = 2\Omega_c + \epsilon$, where $\epsilon \ll \Omega_c$), then the envelope of the sampled signal will vary slowly, with a frequency of $\pi\epsilon/(2\Omega_c + \epsilon)$ rad/sample. This is often referred to as the Moiré effect. Write a MATLAB program to plot 100 samples of $y(n)$ for ϵ equal to $\Omega_s/100$ and confirm the above result.
- Hint:* $\cos(\omega_1 t) + \cos(\omega_2 t) = 2 \cos\left[\left(\frac{\omega_1 + \omega_2}{2}\right)t\right] \cos\left[\left(\frac{\omega_1 - \omega_2}{2}\right)t\right]$
- 1.27 As seen in Example 1.15, cinema is a three-dimensional spatio-temporal signal that is sampled in time. In modern cinema, the sampling frequency in order to avoid aliasing is $\Omega_s = 24$ Hz. However, the persistence of vision is equivalent to a time filter with a bandwidth of $\Omega_{LP} = 48$ Hz. In order to avoid flickering and at the same time avoiding doubling the number of samples, in cinema one repeats each picture twice. For a given

point on the screen, this is equivalent to having a train of impulses

$$g_i(t) = \sum_{n=-\infty}^{\infty} g(n)\delta(t - nT) + \sum_{n=-\infty}^{\infty} g(n)\delta\left(t - nT - \frac{T}{2}\right),$$

where $T = 1/24$ Hz. Show that with the above scheme the persistence of vision allows the human visual system to have the impression of continuous movement.

- 1.28 Show that the statistical mean and the variance for the Gaussian PDF expressed in Equation (1.209) are given by

$$\begin{aligned} E\{X\} &= \bar{\mu} \\ E\{(X - E\{X\})^2\} &= \bar{\sigma}^2. \end{aligned}$$

- 1.29 Verify that the cross-correlation $r_{X,Y}$ and the cross-covariance $c_{X,Y}$ of two random variables X and Y , as defined in Equations (1.222) and (1.223), satisfy the relation

$$c_{X,Y} = r_{X,Y} - E\{X\}E\{Y\}.$$

- 1.30 Show that the autocorrelation function of a WSS process $\{X\}$ presents the following properties:

- (a) $R_X(0) = E\{X^2(n)\}$
- (b) it is an even function; that is: $R_X(\nu) = R_X(-\nu)$, for all ν
- (c) it has a maximum at $\nu = 0$; that is: $R_X(0) \geq |R_X(\nu)|$, for all ν .

- 1.31 Show that the autocorrelation function of the output of a discrete linear system with impulse response $h(n)$ is given by

$$R_Y(n) = \sum_{k=-\infty}^{\infty} R_X(n-k)C_h(k),$$

where

$$C_h(k) = \sum_{r=-\infty}^{\infty} h(k+r)h(r).$$

- 1.32 The entropy $H(X)$ of a discrete random variable X measures the uncertainty about predicting the value of X (Cover & Thomas, 2006). If X has the probability distribution $p_X(x)$, its entropy is determined by

$$H(X) = - \sum_x p_X(x) \log_b p_X(x),$$

where the base b of the logarithm determines the entropy unit. If $b = 2$, for instance, the entropy is measured in bits/symbol. Determine in bits/symbol the entropy of a random variable X characterized by the discrete uniform distribution $u_{X,d}(x)$ given in Equation (1.208).

- 1.33 For a continuous random variable X with distribution $p_X(x)$, the uncertainty is measured by the so-called differential entropy $h(X)$ determined as (Cover & Thomas, 2006)

$$H(X) = - \int_x p_X(x) \log_b p_X(x) \, dx.$$

Determine the differential entropy of the random variables characterized by:

- (a) continuous uniform distribution $u_{X,c}(x)$ given in Equation (1.207)
- (b) the Gaussian distribution $\phi_X(x)$ given in Equation (1.209).

2.1 Introduction

In Chapter 1 we studied linear time-invariant systems, using both impulse responses and difference equations to characterize them. In this chapter we study another very useful way to characterize discrete-time systems. It is linked with the fact that, when an exponential function is input to a linear time-invariant system, its output is an exponential function of the same type, but with a different amplitude. This can be deduced by considering that, from Equation (1.50), a linear time-invariant discrete-time system with impulse response $h(n)$, when excited by an exponential $x(n) = z^n$, produces at its output a signal $y(n)$ such that

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) = \sum_{k=-\infty}^{\infty} z^{n-k}h(k) = z^n \sum_{k=-\infty}^{\infty} h(k)z^{-k}; \quad (2.1)$$

that is, the signal at the output is also an exponential z^n , but with an amplitude multiplied by the complex function

$$H(z) = \sum_{k=-\infty}^{\infty} h(k)z^{-k}. \quad (2.2)$$

In this chapter we characterize linear time-invariant systems using the quantity $H(z)$ in Equation (2.2), commonly known as the z transform of the discrete-time sequence $h(n)$. As we will see later in this chapter, with the help of the z transform, linear convolutions can be transformed into simple algebraic equations. The importance of this for discrete-time systems parallels that of the Laplace transform for continuous-time systems.

The case when z^n is a complex sinusoid with frequency ω (that is, $z = e^{j\omega}$) is of particular importance. In this case, Equation (2.2) becomes

$$H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h(k)e^{-j\omega k}, \quad (2.3)$$

which can be represented in polar form as $H(e^{j\omega}) = |H(e^{j\omega})|e^{j\Theta(\omega)}$, yielding, from Equation (2.1), an output signal $y(n)$ such that

$$y(n) = H(e^{j\omega})e^{j\omega n} = |H(e^{j\omega})|e^{j\Theta(\omega)}e^{j\omega n} = |H(e^{j\omega})|e^{j\omega n + j\Theta(\omega)}. \quad (2.4)$$

This relationship implies that the effect of a linear system characterized by $H(e^{j\omega})$ on a complex sinusoid is to multiply its amplitude by $|H(e^{j\omega})|$ and to add $\Theta(\omega)$ to its phase. For this reason, the descriptions of $|H(e^{j\omega})|$ and $\Theta(\omega)$ as functions of ω are widely used to characterize linear time-invariant systems, and are known as their magnitude and phase responses respectively. The complex function $H(e^{j\omega})$ in Equation (2.4) is also known as the Fourier transform of the discrete-time sequence $h(n)$. The Fourier transform is as important for discrete-time systems as it is for continuous-time systems.

In this chapter we will study the z and Fourier transforms for discrete-time signals. We begin by defining the z transform, discussing issues related to its convergence and its relation to the stability of discrete-time systems. Then we present the inverse z transform, as well as several z -transform properties. Next, we show how to transform discrete-time convolutions into a product of algebraic expressions and introduce the concept of a transfer function. We then present an algorithm to determine, given the transfer function of a discrete-time system, whether the system is stable or not and go on to discuss how the frequency response of a system is related to its transfer function. At this point we give a formal definition of the Fourier transform of discrete-time signals, highlighting its relations to the Fourier transform of continuous-time signals. An expression for the inverse Fourier transform is also presented. Its main properties are then shown as particular cases of those of the z transform. In a separate section, we present the main properties of random signals in the transform domain. We close the chapter by presenting some MATLAB functions which are related to z and Fourier transforms, and which aid in the analysis of transfer functions of discrete-time systems.

2.2 Definition of the z transform

The z transform of a sequence $x(n)$ is defined as

$$X(z) = \mathcal{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \quad (2.5)$$

where z is a complex variable. Note that $X(z)$ is only defined for the regions of the complex plane in which the summation on the right converges.

Very often, the signals we work with start only at $n = 0$; that is, they are nonzero only for $n \geq 0$. Because of that, some textbooks define the z transform as

$$X_U(z) = \sum_{n=0}^{\infty} x(n)z^{-n}, \quad (2.6)$$

which is commonly known as the one-sided z transform, while Equation (2.5) is referred to as the two-sided z transform. Clearly, if the signal $x(n)$ is nonzero for $n < 0$, then the one-sided and two-sided z transforms are different. In this text we work only with the two-sided z transform, which is referred to, without any risk of ambiguity, just as the z transform.

As mentioned above, the z transform of a sequence exists only for those regions of the complex plane in which the summation in Equation (2.5) converges. Example 2.1 clarifies this point.

Example 2.1. Compute the z transform of the sequence $x(n) = Ku(n)$.

Solution

By definition, the z transform of $Ku(n)$ is

$$X(z) = K \sum_{n=0}^{\infty} z^{-n} = K \sum_{n=0}^{\infty} (z^{-1})^n. \quad (2.7)$$

Thus, $X(z)$ is the sum of a power series which converges only if $|z^{-1}| < 1$. In such a case, $X(z)$ can be expressed as

$$X(z) = \frac{K}{1 - z^{-1}} = \frac{Kz}{z - 1}, \quad |z| > 1. \quad (2.8)$$

Note that, for $|z| < 1$, the n th term of the summation, z^{-n} , tends to infinity as $n \rightarrow \infty$; and therefore, $X(z)$ is not defined. For $z = 1$, the summation is also infinite. For $z = -1$, the summation oscillates between 1 and 0. In none of these cases does the z transform converge. \triangle

It is important to note that the z transform of a sequence is a Laurent series in the complex variable z (Churchill, 1975). Therefore, the properties of Laurent series apply directly to the z transform. As a general rule, given a series of the complex variable z , we can apply a result from series theory stating that

$$S(z) = \sum_{i=0}^{\infty} f_i(z) \quad (2.9)$$

such that $|f_i(z)| < \infty$, $i = 0, 1, \dots$, and given the quantity

$$\alpha(z) = \lim_{n \rightarrow \infty} |f_n(z)|^{1/n} \quad (2.10)$$

then the series converges absolutely if $\alpha(z) < 1$ and diverges if $\alpha(z) > 1$ (Kreyszig, 1979). Note that, for $\alpha(z) = 1$, the above procedure tells us nothing about the convergence of the series, which must be investigated by other means. One can justify this by noting that, if $\alpha(z) < 1$, the terms of the series are under an exponential a^n for some $a < 1$ and, therefore, their sum converges as $n \rightarrow \infty$. One should clearly note that, if $|f_i(z)| = \infty$, for some i , then the series is not convergent. Also, convergence demands that $\lim_{n \rightarrow \infty} |f_n(z)| = 0$.

The above result can be extended for the case of two-sided series:

$$S(z) = \sum_{i=-\infty}^{\infty} f_i(z), \quad (2.11)$$

if we express $S(z)$ above as the sum of two series $S_1(z)$ and $S_2(z)$ such that

$$S_1(z) = \sum_{i=0}^{\infty} f_i(z) \quad \text{and} \quad S_2(z) = \sum_{i=-\infty}^{-1} f_i(z), \quad (2.12)$$

then $S(z)$ converges if the two series $S_1(z)$ and $S_2(z)$ converge. Therefore, in this case, we have to compute the two quantities

$$\alpha_1(z) = \lim_{n \rightarrow \infty} |f_n(z)|^{1/n} \quad \text{and} \quad \alpha_2(z) = \lim_{n \rightarrow -\infty} |f_n(z)|^{1/n}. \quad (2.13)$$

Naturally, $S(z)$ converges absolutely if $\alpha_1(z) < 1$ and $\alpha_2(z) > 1$. The condition $\alpha_1(z) < 1$ is equivalent to saying that, for $n \rightarrow \infty$, the terms of the series are under a^n for some $a < 1$. The condition $\alpha_2(z) > 1$ is equivalent to saying that, for $n \rightarrow -\infty$, the terms of the series are under b^n for some $b > 1$. One should note that, for convergence, we must also have $|f_i(z)| < \infty$, for all i .

Applying these convergence results to the z -transform definition given in Equation (2.5), we conclude that the z transform converges if

$$\alpha_1 = \lim_{n \rightarrow \infty} |x(n)z^{-n}|^{1/n} = |z^{-1}| \lim_{n \rightarrow \infty} |x(n)|^{1/n} < 1 \quad (2.14)$$

$$\alpha_2 = \lim_{n \rightarrow -\infty} |x(n)z^{-n}|^{1/n} = |z^{-1}| \lim_{n \rightarrow -\infty} |x(n)|^{1/n} > 1. \quad (2.15)$$

Defining

$$r_1 = \lim_{n \rightarrow \infty} |x(n)|^{1/n} \quad (2.16)$$

$$r_2 = \lim_{n \rightarrow -\infty} |x(n)|^{1/n} \quad (2.17)$$

then Equations (2.14) and (2.15) are equivalent to

$$r_1 < |z| < r_2. \quad (2.18)$$

That is, the z transform of a sequence exists in an annular region of the complex plane defined by Equation (2.18) and illustrated in Figure 2.1. It is important to note that, for some sequences, $r_1 = 0$ or $r_2 \rightarrow \infty$. In these cases, the region of convergence may or may not include $z = 0$ or $|z| = \infty$ respectively.

We now take a closer look at the convergence of z transforms for four important classes of sequences.

- *Right-handed, one-sided sequences:* These are sequences such that $x(n) = 0$, for $n < n_0$; that is:

$$X(z) = \sum_{n=n_0}^{\infty} x(n)z^{-n}. \quad (2.19)$$

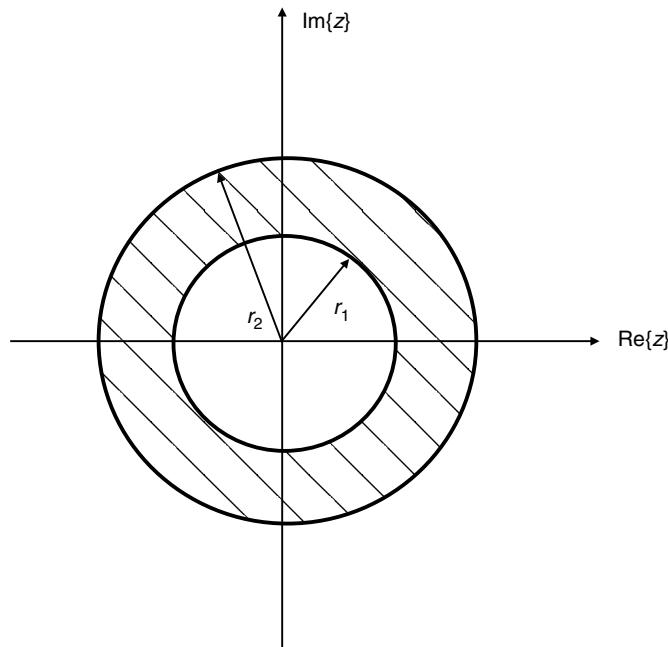


Fig. 2.1. General region of convergence of the z transform.

In this case, the z transform converges for $|z| > r_1$, where r_1 is given by Equation (2.16). Since $|x(n)z^{-n}|$ must be finite, then, if $n_0 < 0$, the convergence region excludes $|z| = \infty$.

- *Left-handed, one-sided sequences:* These are sequences such that $x(n) = 0$, for $n > n_0$; that is:

$$X(z) = \sum_{n=-\infty}^{n_0} x(n)z^{-n}. \quad (2.20)$$

In this case, the z transform converges for $|z| < r_2$, where r_2 is given by Equation (2.17).

Since $|x(n)z^{-n}|$ must be finite, then, if $n_0 > 0$, the convergence region excludes $z = 0$.

- *Two-sided sequences:* In this case:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (2.21)$$

and the z transform converges for $r_1 < |z| < r_2$, where r_1 and r_2 are given by Equations (2.16) and (2.17). Clearly, if $r_1 > r_2$, then the z transform does not exist.

- *Finite-length sequences:* These are sequences such that $x(n) = 0$, for $n < n_0$ and $n > n_1$; that is:

$$X(z) = \sum_{n=n_0}^{n_1} x(n)z^{-n}. \quad (2.22)$$

In such cases, the z transform converges everywhere except at the points such that $|x(n)z^{-n}| = \infty$. This implies that the convergence region excludes the point $z = 0$ if $n_1 > 0$ and $|z| = \infty$ if $n_0 < 0$.

Example 2.2. Compute the z transforms of the following sequences, specifying their region of convergence:

- (a) $x(n) = k2^n u(n)$
- (b) $x(n) = u(-n + 1)$
- (c) $x(n) = -k2^n u(-n - 1)$
- (d) $x(n) = 0.5^n u(n) + 3^n u(-n)$
- (e) $x(n) = 4^{-n} u(n) + 5^{-n} u(n + 1)$.

Solution

$$(a) X(z) = \sum_{n=0}^{\infty} k2^n z^{-n}.$$

This series converges if $|2z^{-1}| < 1$; that is, for $|z| > 2$. In this case, $X(z)$ is the sum of a geometric series; therefore:

$$X(z) = \frac{k}{1 - 2z^{-1}} = \frac{kz}{z - 2}, \quad \text{for } 2 < |z| \leq \infty. \quad (2.23)$$

$$(b) X(z) = \sum_{n=-\infty}^1 z^{-n}.$$

This series converges if $|z^{-1}| > 1$; that is, for $|z| < 1$. Also, in order for the term z^{-1} to be finite, $|z| \neq 0$. In this case, $X(z)$ is the sum of a geometric series, such that

$$X(z) = \frac{z^{-1}}{1 - z} = \frac{1}{z - z^2}, \quad \text{for } 0 < |z| < 1. \quad (2.24)$$

$$(c) X(z) = \sum_{n=-\infty}^{-1} -k2^n z^{-n}.$$

This series converges if $|z/2| < 1$; that is, for $|z| < 2$. In this case, $X(z)$ is the sum of a geometric series, such that

$$X(z) = \frac{-kz/2}{1 - (z/2)} = \frac{kz}{z - 2}, \quad \text{for } 0 \leq |z| < 2. \quad (2.25)$$

$$(d) X(z) = \sum_{n=0}^{\infty} 0.5^n z^{-n} + \sum_{n=-\infty}^0 3^n z^{-n}.$$

This series converges if $|0.5z^{-1}| < 1$ and $|3z^{-1}| > 1$; that is, for $0.5 < |z| < 3$. In this case, $X(z)$ is the sum of two geometric series; therefore:

$$X(z) = \frac{1}{1 - 0.5z^{-1}} + \frac{1}{1 - \frac{1}{3}z} = \frac{z}{z - 0.5} + \frac{3}{3 - z}, \quad \text{for } 0.5 < |z| < 3. \quad (2.26)$$

$$(e) X(z) = \sum_{n=0}^{\infty} 4^{-n} z^{-n} + \sum_{n=-1}^{\infty} 5^{-n} z^{-n}.$$

This series converges if $|\frac{1}{4}z^{-1}| < 1$ and $|\frac{1}{5}z^{-1}| < 1$; that is, for $|z| > \frac{1}{4}$. Also, the term for $n = -1$, $(\frac{1}{5}z^{-1})^{-1} = 5z$, is finite only for $|z| < \infty$. In this case, $X(z)$ is the sum of two geometric series, resulting in

$$X(z) = \frac{1}{1 - \frac{1}{4}z^{-1}} + \frac{5z}{1 - \frac{1}{5}z^{-1}} = \frac{4z}{4z - 1} + \frac{25z^2}{5z - 1}, \text{ for } \frac{1}{4} < |z| < \infty. \quad (2.27)$$

In this example, although the sequences in items (a) and (c) are distinct, the expressions for their z transforms are the same, the difference being only in their regions of convergence. This highlights the important fact that, in order to specify a z transform completely, its region of convergence must be supplied. In Section 2.3, when we study the inverse z transform, this issue is dealt with in more detail. \triangle

In several cases we deal with causal and stable systems. Since for a causal system its impulse response $h(n)$ is zero for $n < n_0$, $n_0 \geq 0$, then, from Equation (1.60), we have that a causal system is also BIBO stable if

$$\sum_{n=n_0}^{\infty} |h(n)| < \infty. \quad (2.28)$$

Applying the series convergence criterion seen above, we have that the system is stable only if

$$\lim_{n \rightarrow \infty} |h(n)|^{1/n} = r < 1. \quad (2.29)$$

This is equivalent to saying that $H(z)$, the z transform of $h(n)$, converges for $|z| > r$. Since, for stability, $r < 1$, then we conclude that the convergence region of the z transform of the impulse response of a stable causal system includes the region outside the unit circle and the unit circle itself (in fact, if $n_0 < 0$, then this region excludes $|z| = \infty$).

A very important case is when $X(z)$ can be expressed as a ratio of polynomials, in the form

$$X(z) = \frac{N(z)}{D(z)}. \quad (2.30)$$

We refer to the roots of $N(z)$ as the zeros of $X(z)$ and to the roots of $D(z)$ as the poles of $X(z)$. More specifically, in this case $X(z)$ can be expressed as

$$X(z) = \frac{N(z)}{\prod_{k=1}^K (z - p_k)^{m_k}}, \quad (2.31)$$

where p_k is a pole of multiplicity m_k and K is the total number of distinct poles. Since $X(z)$ is not defined at its poles, its convergence region must not include them. Therefore, given $X(z)$ as in Equation (2.31), there is an easy way of determining its convergence region, depending on the type of sequence $x(n)$:

- *Right-handed, one-sided sequences:* The convergence region of $X(z)$ is $|z| > r_1$. Since $X(z)$ is not convergent at its poles, then its poles must be inside the circle $|z| = r_1$ (except for poles at $|z| = \infty$), and $r_1 = \max_{1 \leq k \leq K} \{|p_k|\}$. This is illustrated in Figure 2.2a.
- *Left-handed, one-sided sequences:* The convergence region of $X(z)$ is $|z| < r_2$. Therefore, its poles must be outside the circle $|z| = r_2$ (except for poles at $z = 0$), and $r_2 = \min_{1 \leq k \leq K} \{|p_k|\}$. This is illustrated in Figure 2.2b.
- *Two-sided sequences:* The convergence region of $X(z)$ is $r_1 < |z| < r_2$; therefore, some of its poles are inside the circle $|z| = r_1$ and some outside the circle $|z| = r_2$. In this case, the convergence region needs to be further specified. This is illustrated in Figure 2.2c.

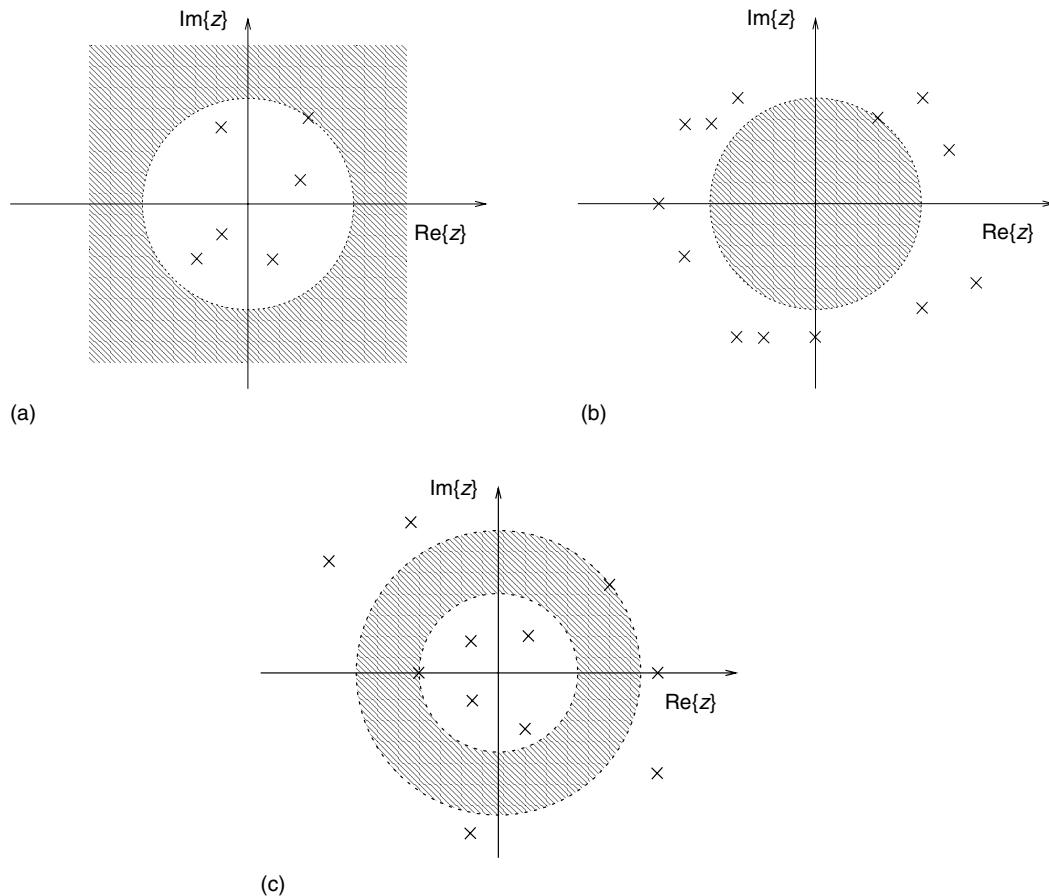


Fig. 2.2. Regions of convergence of a z transform in relation to its poles: (a) right-handed, one-sided sequences; (b) left-handed, one-sided sequences; (c) two-sided sequences.

2.3 Inverse z transform

Very often one needs to determine which sequence corresponds to a given z transform. A formula for the inverse z transform can be obtained from the residue theorem, which we state next.

Theorem 2.1 (Residue Theorem). *Let $X(z)$ be a complex function that is analytic inside a closed contour C , including the contour itself, except in a finite number of singular points p_n inside C . In this case, the following equality holds:*

$$\oint_C X(z) dz = 2\pi j \sum_{k=1}^K \text{res}_{z=p_k} \{X(z)\} \quad (2.32)$$

with the integral evaluated counterclockwise around C .

If p_k is a pole of multiplicity m_k of $X(z)$ – that is, if $X(z)$ can be written as

$$X(z) = \frac{P_k(z)}{(z - p_k)^{m_k}}, \quad (2.33)$$

where $P_k(z)$ is analytic at $z = p_k$ – then the residue of $X(z)$ with respect to p_k is given by

$$\text{res}_{z=p_k} \{X(z)\} = \frac{1}{(m_k - 1)!} \left. \frac{d^{(m_k-1)} [(z - p_k)^{m_k} X(z)]}{dz^{m_k-1}} \right|_{z=p_k}. \quad (2.34)$$

◇

Using Theorem 2.1, one can show that, if C is a counterclockwise closed contour, encircling the origin of the z plane, then

$$\frac{1}{2\pi j} \oint_C z^{n-1} dz = \begin{cases} 0, & \text{for } n \neq 0 \\ 1, & \text{for } n = 0 \end{cases} \quad (2.35)$$

and then we can derive that the inverse z transform of $X(z)$ is given by

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz, \quad (2.36)$$

where C is a closed counterclockwise contour in the convergence region of $X(z)$.

Proof Since

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}, \quad (2.37)$$

by expressing $x(n)$ using the inverse z transform as in Equation (2.36), then changing the order of the integration and summation, we have that

$$\begin{aligned} \frac{1}{2\pi j} \oint_C X(z) z^{m-1} dz &= \frac{1}{2\pi j} \oint_C \sum_{n=-\infty}^{\infty} x(n) z^{-n+m-1} dz \\ &= \frac{1}{2\pi j} \sum_{n=-\infty}^{\infty} x(n) \oint_C z^{-n+m-1} dz \\ &= x(m). \end{aligned} \quad (2.38)$$

□

In the remainder of this section we describe techniques for the computation of the inverse z transform in several practical cases.

2.3.1 Computation based on residue theorem

Whenever $X(z)$ is a ratio of polynomials, the residue theorem can be used very efficiently to compute the inverse z transform. In this case, Equation (2.36) becomes

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz = \sum_{\substack{p_k \text{ encircled by } C \\ z=p_k}} \text{res}_{z=p_k} \{X(z)z^{n-1}\}, \quad (2.39)$$

where

$$X(z)z^{n-1} = \frac{N(z)}{\prod_{k=1}^K (z - p_k)^{m_k}}. \quad (2.40)$$

Note that not all poles p_k (with multiplicity m_k) of $X(z)z^{n-1}$ enter the summation in Equation (2.39). The summation should contain only the poles that are encircled by the contour C . It is also important to note that the contour C must be contained in the convergence region of $X(z)$. In addition, in order to compute $x(n)$, for $n \leq 0$, one must consider the residues of the poles of $X(z)z^{n-1}$ at the origin.

Example 2.3. Determine the inverse z transform of

$$X(z) = \frac{z^2}{(z - 0.2)(z + 0.8)}, \quad (2.41)$$

considering that it represents the z transform of the impulse response of a causal system.

Solution

One should note that, in order to specify a z transform completely, its region of convergence must be supplied. In this example, since the system is causal, we have that its impulse

response is right handed and one sided. Therefore, as seen in Section 2.2, the convergence region of the z transform is characterized by $|z| > r_1$. This implies that its poles are inside the circle $|z| = r_1$ and, therefore, $r_1 = \max_{1 \leq k \leq K} \{|p_k|\} = 0.8$.

We then need to compute

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz = \frac{1}{2\pi j} \oint_C \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} dz \quad (2.42)$$

where C is any closed contour in the convergence region of $X(z)$; that is, encircling the poles $z = 0.2$ and $z = -0.8$ (as well as the poles at $z = 0$, for $n \leq -2$).

Since we want to use the residue theorem, there are two distinct cases. For $n \geq -1$, there are two poles inside C : at $z = 0.2$ and $z = -0.8$; for $n \leq -2$, there are three poles inside C : at $z = 0.2$, $z = -0.8$, and $z = 0$. Therefore, we have that:

- For $n \geq -1$, Equation (2.39) leads to

$$\begin{aligned} x(n) &= \underset{z=0.2}{\text{res}} \left\{ \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} \right\} + \underset{z=-0.8}{\text{res}} \left\{ \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} \right\} \\ &= \underset{z=0.2}{\text{res}} \left\{ \frac{P_1(z)}{z - 0.2} \right\} + \underset{z=-0.8}{\text{res}} \left\{ \frac{P_2(z)}{z + 0.8} \right\}, \end{aligned} \quad (2.43)$$

where

$$P_1(z) = \frac{z^{n+1}}{z + 0.8} \quad \text{and} \quad P_2(z) = \frac{z^{n+1}}{z - 0.2}. \quad (2.44)$$

From Equation (2.34):

$$\underset{z=0.2}{\text{res}} \left\{ \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} \right\} = P_1(0.2) = (0.2)^{n+1} \quad (2.45)$$

$$\underset{z=-0.8}{\text{res}} \left\{ \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} \right\} = P_2(-0.8) = -(-0.8)^{n+1} \quad (2.46)$$

and then

$$x(n) = (0.2)^{n+1} - (-0.8)^{n+1}, \quad \text{for } n \geq -1. \quad (2.47)$$

- For $n \leq -2$, we also have a pole of multiplicity $(-n - 1)$ at $z = 0$. Therefore, we have to add the residue at $z = 0$ to the two residues in Equation (2.47), such that

$$\begin{aligned} x(n) &= (0.2)^{n+1} - (-0.8)^{n+1} + \underset{z=0}{\text{res}} \left\{ \frac{z^{n+1}}{(z - 0.2)(z + 0.8)} \right\} \\ &= (0.2)^{n+1} - (-0.8)^{n+1} + \underset{z=0}{\text{res}} \left\{ P_3(z) z^{n+1} \right\}, \end{aligned} \quad (2.48)$$

where

$$P_3(z) = \frac{1}{(z - 0.2)(z + 0.8)}. \quad (2.49)$$

From Equation (2.34), since the pole $z = 0$ has multiplicity $m_k = (-n - 1)$, we have that

$$\begin{aligned} \text{res}_{z=0} \left\{ P_3(z) z^{n+1} \right\} &= \frac{1}{(-n - 2)!} \left. \frac{d^{(-n-2)} P_3(z)}{dz^{(-n-2)}} \right|_{z=0} \\ &= \frac{1}{(-n - 2)!} \left. \frac{d^{(-n-2)}}{dz^{(-n-2)}} \left\{ \frac{1}{(z - 0.2)(z + 0.8)} \right\} \right|_{z=0} \\ &= \left\{ \frac{(-1)^{-n-2}}{(z - 0.2)^{-n-1}} - \frac{(-1)^{-n-2}}{(z + 0.8)^{-n-1}} \right\} \Big|_{z=0} \\ &= (-1)^{-n-2} [(-0.2)^{n+1} - (0.8)^{n+1}] \\ &= -(0.2)^{n+1} + (-0.8)^{n+1}. \end{aligned} \quad (2.50)$$

Substituting the above result into Equation (2.48), we have that

$$x(n) = (0.2)^{n+1} - (-0.8)^{n+1} - (0.2)^{n+1} + (-0.8)^{n+1} = 0, \text{ for } n \leq -2. \quad (2.51)$$

From Equations (2.47) and (2.51), we then have that

$$x(n) = [(0.2)^{n+1} - (-0.8)^{n+1}] u(n+1). \quad (2.52)$$

△

From what we have seen in the above example, the computation of residues for the case of multiple poles at $z = 0$ involves computation of n th-order derivatives, which can very often become quite involved. Fortunately, these cases can be easily solved by means of a simple trick, which we describe next.

When the integral in

$$X(z) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz \quad (2.53)$$

involves computation of residues of multiple poles at $z = 0$, we make the change of variable $z = 1/v$. If the poles of $X(z)$ are located at $z = p_i$, then the poles of $X(1/v)$ are located at $v = 1/p_i$. Also, if $X(z)$ converges for $r_1 < |z| < r_2$, then $X(1/v)$ converges for $1/r_2 < |v| < 1/r_1$. The integral in Equation (2.36) then becomes

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz = -\frac{1}{2\pi j} \oint_{C'} X\left(\frac{1}{v}\right) v^{-n-1} dv. \quad (2.54)$$

Note that if the contour C is traversed in the counterclockwise direction by z , then the contour C' is traversed in the clockwise direction by v . Substituting the contour C' by an

equal contour C'' that is traversed in the counterclockwise direction, the sign of the integral is reversed, and the above equation becomes

$$x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1} dz = \frac{1}{2\pi j} \oint_{C''} X\left(\frac{1}{v}\right)v^{-n-1} dv. \quad (2.55)$$

If $X(z)z^{n-1}$ has multiple poles at the origin, then $X(1/v)v^{-n-1}$ has multiple poles at $|z| = \infty$, which are now outside the closed contour C'' . Therefore, the computation of the integral on the right-hand side of Equation (2.55) avoids the computation of n th-order derivatives. This fact is illustrated by Example 2.4, which revisits the computation of the inverse z transform in Example 2.3.

Example 2.4. Compute the inverse z transform of $X(z)$ in Example 2.3, for $n \leq -2$, using the residue theorem by employing the change of variables in Equation (2.55).

Solution

If we make the change of variables $z = 1/v$, Equation (2.42) becomes

$$x(n) = \frac{1}{2\pi j} \oint_C \frac{z^{n+1}}{(z-0.2)(z+0.8)} dz = \frac{1}{2\pi j} \oint_{C''} \frac{v^{-n-1}}{(1-0.2v)(1+0.8v)} dv. \quad (2.56)$$

The convergence region of the integrand on the right is $|v| < 1/0.8$; therefore, for $n \leq -2$, there are no poles inside the closed contour C'' . Then, from Equation (2.39), we conclude that

$$x(n) = 0, \quad \text{for } n \leq -2, \quad (2.57)$$

which is the result of Example 2.3 obtained in a straightforward manner. \triangle

2.3.2 Computation based on partial-fraction expansions

Using the residue theorem, one can show that the inverse z transform of

$$X(z) = \frac{1}{(z-z_0)^k}, \quad (2.58)$$

if its convergence region is $|z| > |z_0|$, is the one-sided, right-handed sequence

$$x(n) = \frac{(n-1)!}{(n-k)!(k-1)!} z_0^{n-k} u(n-k) = \binom{n-1}{k-1} z_0^{n-k} u(n-k). \quad (2.59)$$

If the convergence region of the z transform in Equation (2.58) is $|z| < |z_0|$, its inverse z transform is the one-sided, left-handed sequence

$$x(n) = -\frac{(n-1)!}{(n-k)!(k-1)!} z_0^{n-k} u(-n+k-1) = -\binom{n-1}{k-1} z_0^{n-k} u(-n+k-1). \quad (2.60)$$

Using the above relations, it becomes straightforward to compute the inverse z transform of any function $X(z)$ that can be expressed as a ratio of polynomials provided that we first compute a partial-fraction expansion of $X(z)$.

If $X(z) = N(z)/D(z)$ has K distinct poles p_k , for $k = 1, 2, \dots, K$, each of multiplicity m_k , then the partial-fraction expansion of $X(z)$ is as follows (Kreyszig, 1979):

$$X(z) = \sum_{l=0}^{M-L} g_l z^l + \sum_{k=1}^K \sum_{i=1}^{m_k} \frac{c_{ki}}{(z - p_k)^i}, \quad (2.61)$$

where M and L are respectively the degrees of the numerator and denominator of $X(z)$.

The coefficients g_l , for $l = 0, 1, \dots, (M - L)$, can be obtained from the quotient of the polynomials $N(z)$ and $D(z)$ as follows:

$$X(z) = \frac{N(z)}{D(z)} = \sum_{l=0}^{M-L} g_l z^l + \frac{C(z)}{D(z)}, \quad (2.62)$$

where the degree of $C(z)$ is smaller than the degree of $D(z)$. Clearly, if $M < L$, then $g_l = 0$, for all l .

The coefficients c_{ki} are

$$c_{ki} = \frac{1}{(m_k - i)!} \left. \frac{d^{(m_k - i)} [(z - p_k)^{m_k} X(z)]}{dz^{m_k - i}} \right|_{z=p_k}. \quad (2.63)$$

In the case of a simple pole, c_{k1} is given by

$$c_{k1} = (z - p_k) X(z)|_{z=p_k}. \quad (2.64)$$

Since the z transform is linear and the inverse z transform of each of the terms $c_{ki}/(z - p_k)^i$ can be computed using either Equation (2.59) or Equation (2.60) (depending on whether the pole is inside or outside the region of convergence of $X(z)$), then the inverse z transform follows directly from Equation (2.61).

Example 2.5. Solve Example 2.3 using the partial-fraction expansion of $X(z)$.

Solution

We form

$$X(z) = \frac{z^2}{(z - 0.2)(z + 0.8)} = g_0 + \frac{c_1}{z - 0.2} + \frac{c_2}{z + 0.8}, \quad (2.65)$$

where

$$g_0 = \lim_{|z| \rightarrow \infty} X(z) = 1 \quad (2.66)$$

and, using Equation (2.34), we find that

$$c_1 = \left. \frac{z^2}{z + 0.8} \right|_{z=0.2} = (0.2)^2 \quad (2.67)$$

$$c_2 = \left. \frac{z^2}{z - 0.2} \right|_{z=-0.8} = -(0.8)^2, \quad (2.68)$$

such that

$$X(z) = 1 + \frac{(0.2)^2}{z - 0.2} - \frac{(0.8)^2}{z + 0.8}. \quad (2.69)$$

Since $X(z)$ is the z transform of the impulse response of a causal system, then we have that the terms in the above equation correspond to a right-handed, one-sided power series. Thus, the inverse z transforms of each term are:

$$\mathcal{Z}^{-1}\{1\} = \delta(n) \quad (2.70)$$

$$\begin{aligned} \mathcal{Z}^{-1}\left\{\frac{(0.2)^2}{z - 0.2}\right\} &= (0.2)^2 \mathcal{Z}^{-1}\left\{\frac{1}{z - 0.2}\right\} \\ &= (0.2)^2 \binom{n-1}{0} (0.2)^{n-1} u(n-1) \\ &= (0.2)^{n+1} u(n-1) \end{aligned} \quad (2.71)$$

$$\begin{aligned} \mathcal{Z}^{-1}\left\{\frac{-(0.8)^2}{z + 0.8}\right\} &= -(0.8)^2 \mathcal{Z}^{-1}\left\{\frac{1}{z + 0.8}\right\} \\ &= -(0.8)^2 \binom{n-1}{0} (-0.8)^{n-1} u(n-1) \\ &= -(-0.8)^{n+1} u(n-1). \end{aligned} \quad (2.72)$$

Summing the three terms above (Equations (2.70) to (2.72)), we have that the inverse z transform of $X(z)$ is

$$\begin{aligned} x(n) &= \delta(n) + (0.2)^{n+1} u(n-1) - (-0.8)^{n+1} u(n-1) \\ &= (0.2)^{n+1} u(n) - (-0.8)^{n+1} u(n). \end{aligned} \quad (2.73)$$

△

Example 2.6. Compute the right-handed, one-sided inverse z transform of

$$X(z) = \frac{1}{z^2 - 3z + 3}. \quad (2.74)$$

Solution

Applying partial fraction expansion to $X(z)$, we have that

$$X(z) = \frac{1}{(z - \sqrt{3} e^{j\pi/6})(z - \sqrt{3} e^{-j\pi/6})} = \frac{A}{z - \sqrt{3} e^{j\pi/6}} + \frac{B}{z - \sqrt{3} e^{-j\pi/6}}, \quad (2.75)$$

where

$$A = \left. \frac{1}{z - \sqrt{3} e^{-j\pi/6}} \right|_{z=\sqrt{3} e^{j\pi/6}} = \frac{1}{\sqrt{3} e^{j\pi/6} - \sqrt{3} e^{-j\pi/6}} = \frac{1}{2j\sqrt{3} \sin \frac{\pi}{6}} = \frac{1}{j\sqrt{3}}, \quad (2.76)$$

$$B = \left. \frac{1}{z - \sqrt{3} e^{j\pi/6}} \right|_{z=\sqrt{3} e^{-j\pi/6}} = \frac{1}{\sqrt{3} e^{-j\pi/6} - \sqrt{3} e^{j\pi/6}} = \frac{1}{-2j\sqrt{3} \sin \frac{\pi}{6}} = -\frac{1}{j\sqrt{3}}, \quad (2.77)$$

and thus

$$X(z) = \frac{1}{j\sqrt{3}} \left(\frac{1}{z - \sqrt{3} e^{j\pi/6}} - \frac{1}{z - \sqrt{3} e^{-j\pi/6}} \right). \quad (2.78)$$

From Equation (2.59), we have that

$$\begin{aligned} x(n) &= \frac{1}{j\sqrt{3}} \left[(\sqrt{3} e^{j\pi/6})^{n-1} - (\sqrt{3} e^{-j\pi/6})^{n-1} \right] u(n-1) \\ &= \frac{1}{j\sqrt{3}} \left[(\sqrt{3})^{n-1} e^{j(n-1)\pi/6} - (\sqrt{3})^{n-1} e^{-j(n-1)\pi/6} \right] u(n-1) \\ &= \frac{1}{j\sqrt{3}} (\sqrt{3})^{n-1} 2j \sin \left[(n-1) \frac{\pi}{6} \right] u(n-1) \\ &= 2(\sqrt{3})^{n-2} \sin \left[(n-1) \frac{\pi}{6} \right] u(n-1). \end{aligned} \quad (2.79)$$

△

2.3.3 Computation based on polynomial division

Given $X(z) = N(z)/D(z)$, we can perform long division on the polynomials $N(z)$ and $D(z)$ and obtain that the coefficient of z^k corresponds to the value of $x(n)$ at $n = k$. One should note that this is possible only in the case of one-sided sequences. If the sequence is right handed, then the polynomials should be a function of z . If the sequence is left handed, then the polynomials should be a function of z^{-1} . This is made clear in Examples 2.7 and 2.8.

Example 2.7. Solve Example 2.3 using polynomial division.

Solution

Since $X(z)$ is the z transform of a right-handed, one-sided (causal) sequence, we can express it as a ratio of polynomials in z ; that is:

$$X(z) = \frac{z^2}{(z - 0.2)(z + 0.8)} = \frac{z^2}{z^2 + 0.6z - 0.16}. \quad (2.80)$$

Then the division becomes

$$\begin{array}{r|l} z^2 + 0.6z + 0.16 & z^2 \\ \hline 1 - 0.6z^{-1} + 0.52z^{-2} - 0.408z^{-3} + \dots & -z^2 - 0.6z + 0.16 \\ & \hline & -0.6z + 0.16 \\ & \hline & 0.6z + 0.36 - 0.096z^{-1} \\ & \hline & 0.52 - 0.096z^{-1} \\ & \hline & -0.52 - 0.312z^{-1} + 0.0832z^{-2} \\ & \hline & -0.408z^{-1} + 0.0832z^{-2} \\ & \vdots \end{array}$$

Therefore:

$$X(z) = 1 + (-0.6)z^{-1} + (0.52)z^{-2} + (-0.408)z^{-3} + \dots \quad (2.81)$$

This is the same as saying that

$$x(n) = \begin{cases} 0, & \text{for } n < 0 \\ 1, -0.6, 0.52, -0.408, \dots & \text{for } n = 0, 1, 2, \dots \end{cases} \quad (2.82)$$

The main difficulty with this method is to find a closed-form expression for $x(n)$. In the above case, we can check that indeed the above sequence corresponds to Equation (2.52).

△

Example 2.8. Find the inverse z transform of $X(z)$ in Example 2.3 using polynomial division and supposing that the sequence $x(n)$ is left handed and one sided.

Solution

Since $X(z)$ is the z transform of a left-handed, one-sided sequence, we should express it as

$$X(z) = \frac{z^2}{(z - 0.2)(z + 0.8)} = \frac{1}{-0.16z^{-2} + 0.6z^{-1} + 1}. \quad (2.83)$$

Then the division becomes

$$\begin{array}{r|l} -0.16z^{-2} + 0.6z^{-1} + 1 & 1 \\ \hline -6.25z^2 - 23.4375z^3 - 126.953125z^4 - \dots & -1 + 3.75z + 6.25z^2 \\ & \hline & 3.75z + 6.25z^2 \\ & -3.75z + 14.0625z^2 + 23.4375z^3 \\ & \hline & 20.3125z^2 + 23.4375z^3 \\ & \vdots \end{array}$$

yielding

$$X(z) = -6.25z^2 - 23.4375z^3 - 126.953125z^4 - \dots, \quad (2.84)$$

implying that

$$x(n) = \begin{cases} \dots, -126.953125, -23.4375, -6.25, & \text{for } n = \dots, -4, -3, -2 \\ 0, & \text{for } n > -2 \end{cases}. \quad (2.85)$$

△

2.3.4 Computation based on series expansion

When a z transform is not expressed as a ratio of polynomials, we can try to perform its inversion using a Taylor series expansion around either $z^{-1} = 0$ or $z = 0$, depending on whether the convergence region includes $|z| = \infty$ or $z = 0$. For right-handed, one-sided sequences, we use the expansion of $X(z)$ using the variable z^{-1} around $z^{-1} = 0$. The Taylor series expansion of $F(x)$ around $x = 0$ is given by

$$\begin{aligned} F(x) &= F(0) + x \frac{dF}{dx} \Big|_{x=0} + \frac{x^2}{2!} \frac{d^2F}{dx^2} \Big|_{x=0} + \frac{x^3}{3!} \frac{d^3F}{dx^3} \Big|_{x=0} + \dots \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \frac{d^nF}{dx^n} \Big|_{x=0}. \end{aligned} \quad (2.86)$$

If we make $x = z^{-1}$, then the expansion above has the form of a z transform of a right-handed, one-sided sequence.

Example 2.9. Find the inverse z transform of

$$X(z) = \ln \left(\frac{1}{1 - z^{-1}} \right). \quad (2.87)$$

Consider the sequence as right handed and one sided.

Solution

Expanding $X(z)$ as in Equation (2.86), using z^{-1} as the variable, we have that

$$X(z) = \sum_{n=1}^{\infty} \frac{z^{-n}}{n}. \quad (2.88)$$

One can see that the above series is convergent for $|z| > 1$, since, from Equation (2.14):

$$\lim_{n \rightarrow \infty} \left| \frac{z^{-n}}{n} \right|^{1/n} = z^{-1} \lim_{n \rightarrow \infty} \left| \frac{1}{n} \right|^{1/n} = z^{-1}. \quad (2.89)$$

Therefore, the inverse z transform of $X(z)$ is, by inspection:

$$x(n) = \frac{1}{n} u(n-1). \quad (2.90)$$

△

Example 2.10. (a) Calculate the right-hand unilateral inverse z transform related to the function described below:

$$X(z) = \arctan(z^{-1}) \quad (2.91)$$

knowing that

$$\frac{d^k \arctan(x)}{dx^k}(0) = \begin{cases} 0, & k = 2l \\ (-1)^{(k-1)/2}(k-1)!, & k = 2l+1. \end{cases} \quad (2.92)$$

(b) Could the resulting sequence represent the impulse response of a stable system? Why?

Solution

(a) Given the series definition in Equation (2.86) and Equation (2.92), the series for the arctan function can be expressed as

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} + \cdots + \frac{(-1)^l x^{(2l+1)}}{2l+1} + \cdots \quad (2.93)$$

and thus

$$\arctan(z^{-1}) = z^{-1} - \frac{z^{-3}}{3} + \frac{z^{-5}}{5} + \cdots + \frac{(-1)^l z^{-(2l+1)}}{2l+1} \cdots \quad (2.94)$$

As a result, the corresponding time-domain sequence is given by

$$y(k) = \begin{cases} 0, & k = 2l \\ \frac{(-1)^{(k-1)/2}}{k}, & k = 2l+1. \end{cases} \quad (2.95)$$

(b) For a sequence $h(n)$ to represent an impulse of a stable system, then it should be absolutely summable. From Equation (2.29), this is equivalent to

$$\lim_{n \rightarrow \infty} |h(n)|^{1/n} < 1. \quad (2.96)$$

In our case, from Equation (2.95):

$$\lim_{k \rightarrow \infty} |y(k)|^{1/k} = \lim_{k \rightarrow \infty} \left| \frac{(-1)^{(k-1)/2}}{k} \right|^{1/k} = \lim_{k \rightarrow \infty} \left| \frac{1}{k} \right|^{1/k} = 1. \quad (2.97)$$

In this case, the stability test above is inconclusive. However,

$$\sum_{k=1}^{\infty} \frac{1}{k} = \sum_{l=1}^{\infty} \left(\frac{1}{2l-1} + \frac{1}{2l} \right) < 2 \sum_{l=1}^{\infty} \frac{1}{2l-1}. \quad (2.98)$$

Therefore, since $\sum_{k=1}^{\infty} 1/k$ is not bounded, then $\sum_{n=0}^{\infty} |h(n)| = \sum_{l=1}^{\infty} [1/(2l-1)]$ is also not bounded, and the system is not stable. \triangle

2.4 Properties of the z transform

In this section we state some of the more important z -transform properties.

2.4.1 Linearity

Given two sequences $x_1(n)$ and $x_2(n)$ and two arbitrary constants k_1 and k_2 such that $x(n) = k_1x_1(n) + k_2x_2(n)$, then

$$X(z) = k_1X_1(z) + k_2X_2(z), \quad (2.99)$$

where the region of convergence of $X(z)$ is the intersection of the regions of convergence of $X_1(z)$ and $X_2(z)$.

Proof

$$\begin{aligned} X(z) &= \sum_{n=-\infty}^{\infty} (k_1x_1(n) + k_2x_2(n))z^{-n} \\ &= k_1 \sum_{n=-\infty}^{\infty} x_1(n)z^{-n} + k_2 \sum_{n=-\infty}^{\infty} x_2(n)z^{-n} \\ &= k_1X_1(z) + k_2X_2(z). \end{aligned} \quad (2.100)$$

\square

2.4.2 Time reversal

$$x(-n) \longleftrightarrow X(z^{-1}), \quad (2.101)$$

where if the region of convergence of $X(z)$ is $r_1 < |z| < r_2$, then the region of convergence of $\mathcal{Z}\{x(-n)\}$ is $1/r_2 < |z| < 1/r_1$.

Proof

$$\begin{aligned} \mathcal{Z}\{x(-n)\} &= \sum_{n=-\infty}^{\infty} x(-n)z^{-n} = \sum_{m=-\infty}^{\infty} x(m)z^m = \sum_{m=-\infty}^{\infty} x(m)(z^{-1})^{-m} = X(z^{-1}), \end{aligned} \quad (2.102)$$

implying that the region of convergence of $\mathcal{Z}\{x(-n)\}$ is $r_1 < |z^{-1}| < r_2$, which is equivalent to $1/r_2 < |z| < 1/r_1$. \square

2.4.3 Time-shift theorem

$$x(n+l) \longleftrightarrow z^l X(z), \quad (2.103)$$

where l is an integer. The region of convergence of $\mathcal{Z}\{x(n+l)\}$ is the same as the region of convergence of $X(z)$, except for the possible inclusion or exclusion of the regions $z = 0$ and $|z| = \infty$.

Proof By definition

$$\mathcal{Z}\{x(n+l)\} = \sum_{n=-\infty}^{\infty} x(n+l)z^{-n}. \quad (2.104)$$

Making the change of variables $m = n + l$, we have that

$$\mathcal{Z}\{x(n+l)\} = \sum_{m=-\infty}^{\infty} x(m)z^{-(m-l)} = z^l \sum_{m=-\infty}^{\infty} x(m)z^{-m} = z^l X(z), \quad (2.105)$$

noting that the multiplication by z^l can either introduce or exclude poles at $z = 0$ and $|z| = \infty$. \square

2.4.4 Multiplication by an exponential

$$\alpha^{-n}x(n) \longleftrightarrow X(\alpha z), \quad (2.106)$$

where if the region of convergence of $X(z)$ is $r_1 < |z| < r_2$, then the region of convergence of $\mathcal{Z}\{\alpha^{-n}x(n)\}$ is $r_1/|\alpha| < |z| < r_2/|\alpha|$.

Proof

$$\mathcal{Z}\{\alpha^{-n}x(n)\} = \sum_{n=-\infty}^{\infty} \alpha^{-n}x(n)z^{-n} = \sum_{n=-\infty}^{\infty} x(n)(\alpha z)^{-n} = X(\alpha z), \quad (2.107)$$

where the summation converges for $r_1 < |\alpha z| < r_2$, which is equivalent to $r_1/|\alpha| < |z| < r_2/|\alpha|$. \square

2.4.5 Complex differentiation

$$nx(n) \longleftrightarrow -z \frac{dX(z)}{dz}, \quad (2.108)$$

where the region of convergence of $\mathcal{Z}\{nx(n)\}$ is the same as the one of $X(z)$; that is, $r_1 < |z| < r_2$.

Proof

$$\begin{aligned}
 \mathcal{Z}\{nx(n)\} &= \sum_{n=-\infty}^{\infty} nx(n)z^{-n} \\
 &= z \sum_{n=-\infty}^{\infty} nx(n)z^{-n-1} \\
 &= -z \sum_{n=-\infty}^{\infty} x(n) (-nz^{-n-1}) \\
 &= -z \sum_{n=-\infty}^{\infty} x(n) \frac{d}{dz} \{z^{-n}\} \\
 &= -z \frac{dX(z)}{dz}.
 \end{aligned} \tag{2.109}$$

From Equations (2.16) and (2.17), we have that, if the region of convergence of $X(z)$ is $r_1 < |z| < r_2$, then

$$r_1 = \lim_{n \rightarrow \infty} |x(n)|^{1/n} \tag{2.110}$$

$$r_2 = \lim_{n \rightarrow -\infty} |x(n)|^{1/n}. \tag{2.111}$$

Therefore, if the region of convergence of $\mathcal{Z}\{nx(n)\}$ is given by $r'_1 < |z| < r'_2$, then

$$r'_1 = \lim_{n \rightarrow \infty} |nx(n)|^{1/n} = \lim_{n \rightarrow \infty} |n|^{1/n} \lim_{n \rightarrow \infty} |x(n)|^{1/n} = \lim_{n \rightarrow \infty} |x(n)|^{1/n} = r_1 \tag{2.112}$$

$$r'_2 = \lim_{n \rightarrow -\infty} |nx(n)|^{1/n} = \lim_{n \rightarrow -\infty} |n|^{1/n} \lim_{n \rightarrow -\infty} |x(n)|^{1/n} = \lim_{n \rightarrow -\infty} |x(n)|^{1/n} = r_2, \tag{2.113}$$

implying that the region of convergence of $\mathcal{Z}\{nx(n)\}$ is the same as that of $X(z)$. \square

2.4.6 Complex conjugation

$$x^*(n) \longleftrightarrow X^*(z^*). \tag{2.114}$$

The regions of convergence of $X(z)$ and $\mathcal{Z}\{x^*(n)\}$ are the same.

Proof

$$\begin{aligned}
 \mathcal{Z}\{x^*(n)\} &= \sum_{n=-\infty}^{\infty} x^*(n)z^{-n} \\
 &= \sum_{n=-\infty}^{\infty} \left[x(n)(z^*)^{-n} \right]^* \\
 &= \left[\sum_{n=-\infty}^{\infty} x(n)(z^*)^{-n} \right]^* \\
 &= X^*(z^*),
 \end{aligned} \tag{2.115}$$

from which it follows trivially that the region of convergence of $\mathcal{Z}\{x^*(n)\}$ is the same as that of $X(z)$. \square

2.4.7 Real and imaginary sequences

$$\text{Re}\{x(n)\} \longleftrightarrow \frac{1}{2}(X(z) + X^*(z^*)) \tag{2.116}$$

$$\text{Im}\{x(n)\} \longleftrightarrow \frac{1}{2j}(X(z) - X^*(z^*)), \tag{2.117}$$

where $\text{Re}\{x(n)\}$ and $\text{Im}\{x(n)\}$ are respectively the real and imaginary parts of the sequence $x(n)$. The regions of convergence of $\mathcal{Z}\{\text{Re}\{x(n)\}\}$ and $\mathcal{Z}\{\text{Im}\{x(n)\}\}$ contain the ones of $X(z)$.

Proof

$$\mathcal{Z}\{\text{Re}\{x(n)\}\} = \mathcal{Z}\left\{\frac{1}{2}(x(n) + x^*(n))\right\} = \frac{1}{2}(X(z) + X^*(z^*)) \tag{2.118}$$

$$\mathcal{Z}\{\text{Im}\{x(n)\}\} = \mathcal{Z}\left\{\frac{1}{2j}(x(n) - x^*(n))\right\} = \frac{1}{2j}(X(z) - X^*(z^*)), \tag{2.119}$$

with the respective regions of convergence following trivially from the above expressions. \square

2.4.8 Initial-value theorem

If $x(n) = 0$, for $n < 0$, then

$$x(0) = \lim_{z \rightarrow \infty} X(z). \tag{2.120}$$

Proof If $x(n) = 0$, for $n < 0$, then

$$\lim_{z \rightarrow \infty} X(z) = \lim_{z \rightarrow \infty} \sum_{n=0}^{\infty} x(n)z^{-n} = \sum_{n=0}^{\infty} \lim_{z \rightarrow \infty} x(n)z^{-n} = x(0). \quad (2.121)$$

□

2.4.9 Convolution theorem

$$x_1(n) * x_2(n) \longleftrightarrow X_1(z)X_2(z). \quad (2.122)$$

The region of convergence of $\mathcal{Z}\{x_1(n) * x_2(n)\}$ is the intersection of the regions of convergence of $X_1(z)$ and $X_2(z)$. If a pole of $X_1(z)$ is canceled by a zero of $X_2(z)$, or vice versa, then the region of convergence of $\mathcal{Z}\{x_1(n) * x_2(n)\}$ can be larger than those both $X_1(z)$ and $X_2(z)$.

Proof

$$\begin{aligned} \mathcal{Z}\{x_1(n) * x_2(n)\} &= \mathcal{Z} \left\{ \sum_{l=-\infty}^{\infty} x_1(l)x_2(n-l) \right\} \\ &= \sum_{n=-\infty}^{\infty} \left(\sum_{l=-\infty}^{\infty} x_1(l)x_2(n-l) \right) z^{-n} \\ &= \sum_{l=-\infty}^{\infty} x_1(l) \sum_{n=-\infty}^{\infty} x_2(n-l) z^{-n} \\ &= \left(\sum_{l=-\infty}^{\infty} x_1(l) z^{-l} \right) \left(\sum_{n=-\infty}^{\infty} x_2(n) z^{-n} \right) \\ &= X_1(z)X_2(z). \end{aligned} \quad (2.123)$$

□

2.4.10 Product of two sequences

$$x_1(n)x_2(n) \longleftrightarrow \frac{1}{2\pi j} \oint_{C_1} X_1(v)X_2\left(\frac{z}{v}\right) v^{-1} dv = \frac{1}{2\pi j} \oint_{C_2} X_1\left(\frac{z}{v}\right) X_2(v)v^{-1} dv, \quad (2.124)$$

where C_1 is a contour contained in the intersection of the regions of convergence of $X_1(v)$ and $X_2(z/v)$, and C_2 is a contour contained in the intersection of the regions of convergence of $X_1(z/v)$ and $X_2(v)$. Both C_1 and C_2 are assumed to be counterclockwise oriented.

If the region of convergence of $X_1(z)$ is $r_1 < |z| < r_2$ and the region of convergence of $X_2(z)$ is $r'_1 < |z| < r'_2$, then the region of convergence of $\mathcal{Z}\{x_1(n)x_2(n)\}$ is

$$r_1 r'_1 < |z| < r_2 r'_2. \quad (2.125)$$

Proof By expressing $x_2(n)$ as a function of its z transform, $X_2(z)$ (Equation (2.36)), then changing the order of the integration and summation and using the definition of the z transform, we have that

$$\begin{aligned} \mathcal{Z}\{x_1(n)x_2(n)\} &= \sum_{n=-\infty}^{\infty} x_1(n)x_2(n)z^{-n} \\ &= \sum_{n=-\infty}^{\infty} x_1(n) \left(\frac{1}{2\pi j} \oint_{C_2} X_2(v)v^{(n-1)} dv \right) z^{-n} \\ &= \frac{1}{2\pi j} \oint_{C_2} \sum_{n=-\infty}^{\infty} x_1(n)z^{-n} v^{(n-1)} X_2(v) dv \\ &= \frac{1}{2\pi j} \oint_{C_2} \left[\sum_{n=-\infty}^{\infty} x_1(n) \left(\frac{v}{z} \right)^n \right] X_2(v) v^{-1} dv \\ &= \frac{1}{2\pi j} \oint_{C_2} X_1\left(\frac{z}{v}\right) X_2(v) v^{-1} dv. \end{aligned} \quad (2.126)$$

If the region of convergence of $X_1(z)$ is $r_1 < |z| < r_2$, then the region of convergence of $X_1(z/v)$ is

$$r_1 < \frac{|z|}{|v|} < r_2, \quad (2.127)$$

which is equivalent to

$$\frac{|z|}{r_2} < |v| < \frac{|z|}{r_1}. \quad (2.128)$$

In addition, if the region of convergence of $X_2(v)$ is $r'_1 < |v| < r'_2$, then the contour C_2 must lie in the intersection of the two regions of convergence; that is, C_2 must be contained in the region

$$\max \left\{ \frac{|z|}{r_2}, r'_1 \right\} < |v| < \min \left\{ \frac{|z|}{r_1}, r'_2 \right\}. \quad (2.129)$$

Therefore, we must have

$$\min \left\{ \frac{|z|}{r_1}, r'_2 \right\} > \max \left\{ \frac{|z|}{r_2}, r'_1 \right\}, \quad (2.130)$$

which holds if $r_1 r'_1 < |z| < r_2 r'_2$. □

Equation (2.124) is also known as the complex convolution theorem (Oppenheim & Schafer, 1975; Antoniou, 1993). Although at first it does not have the form of a convolution, if we express $z = \rho_1 e^{j\theta_1}$ and $v = \rho_2 e^{j\theta_2}$ in polar form, then it can be rewritten as

$$\mathcal{Z}\{x_1(n)x_2(n)\}|_{z=\rho_1 e^{j\theta_1}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1 \left(\frac{\rho_1}{\rho_2} e^{j(\theta_1 - \theta_2)} \right) X_2 \left(\rho_2 e^{j\theta_2} \right) d\theta_2, \quad (2.131)$$

which has the form of a convolution in θ_1 .

2.4.11 Parseval's theorem

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi j} \oint_C X_1(v)X_2^* \left(\frac{1}{v^*} \right) v^{-1} dv, \quad (2.132)$$

where x^* denotes the complex conjugate of x and C is a contour contained in the intersection of the convergence regions of $X_1(v)$ and $X_2^*(1/v^*)$.

Proof We begin by noting that

$$\sum_{n=-\infty}^{\infty} x(n) = X(z)|_{z=1}. \quad (2.133)$$

Therefore:

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \mathcal{Z}\{x_1(n)x_2^*(n)\}|_{z=1}. \quad (2.134)$$

By using Equation (2.124) and the complex conjugation property in Equation (2.114), we have that Equation 2.134 implies that

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi j} \oint_C X_1(v)X_2^* \left(\frac{1}{v^*} \right) v^{-1} dv. \quad (2.135)$$

□

2.4.12 Table of basic z transforms

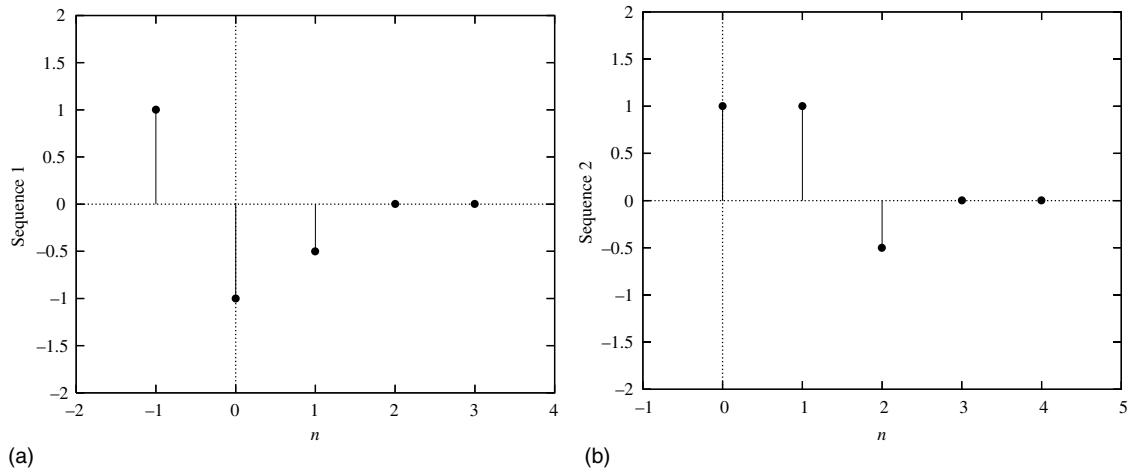
Table 2.1 contains some commonly used sequences and their corresponding z transforms, along with their regions of convergence. Although it only contains the z transforms for right-handed sequences, the results for left-handed sequences can be readily obtained by making $y(n) = x(-n)$ and applying the time-reversal property in Section 2.4.2.

Example 2.11. Calculate the linear convolution of the sequences in Figure 2.3 using the z transform. Plot the resulting sequence.

Table 2.1.

The z transforms of commonly used sequences.

$x(n)$	$X(z)$	Convergence region
$\delta(n)$	1	$z \in \mathbb{C}$
$u(n)$	$\frac{z}{z - 1}$	$ z > 1$
$(-a)^n u(n)$	$\frac{z}{z + a}$	$ z > a$
$n u(n)$	$\frac{z}{(z - 1)^2}$	$ z > 1$
$n^2 u(n)$	$\frac{z(z + 1)}{(z - 1)^3}$	$ z > 1$
$e^{an} u(n)$	$\frac{z}{z - e^a}$	$ z > e^a $
$\binom{n-1}{k-1} e^{a(n-k)} u(n-k)$	$\frac{1}{(z - e^a)^k}$	$ z > e^a $
$\cos(\omega n) u(n)$	$\frac{z[z - \cos(\omega)]}{z^2 - 2z \cos(\omega) + 1}$	$ z > 1$
$\sin(\omega n) u(n)$	$\frac{z \sin(\omega)}{z^2 - 2z \cos(\omega) + 1}$	$ z > 1$
$\frac{1}{n} u(n-1)$	$\ln\left(\frac{z}{z - 1}\right)$	$ z > 1$
$\sin(\omega n + \theta) u(n)$	$\frac{z^2 \sin(\theta) + z \sin(\omega - \theta)}{z^2 - 2z \cos(\omega) + 1}$	$ z > 1$
$e^{an} \cos(\omega n) u(n)$	$\frac{z^2 - z e^a \cos(\omega)}{z^2 - 2z e^a \cos(\omega) + e^{2a}}$	$ z > e^a $
$e^{an} \sin(\omega n) u(n)$	$\frac{z e^a \sin(\omega)}{z^2 - 2z e^a \cos(\omega) + e^{2a}}$	$ z > e^a $



(a)

(b)

Fig. 2.3. Sequences to be convolved in Example 2.11 using the z transform.**Solution**

From Figure 2.3, we can see that the z transforms of the two sequences are

$$X_1(z) = z - 1 - \frac{1}{2}z^{-1} \quad \text{and} \quad X_2(z) = 1 + z^{-1} - \frac{1}{2}z^{-2}. \quad (2.136)$$

According to the property in Section 2.4.9, the z transform of the convolution is the product of the z transforms, and then

$$\begin{aligned} Y(z) &= X_1(z)X_2(z) = \left(z - 1 - \frac{1}{2}z^{-1}\right)\left(1 + z^{-1} - \frac{1}{2}z^{-2}\right) \\ &= z + 1 - \frac{1}{2}z^{-1} - 1 - z^{-1} + \frac{1}{2}z^{-2} - \frac{1}{2}z^{-1} - \frac{1}{2}z^{-2} + \frac{1}{4}z^{-3} \\ &= z - 2z^{-1} + \frac{1}{4}z^{-3}. \end{aligned} \quad (2.137)$$

In the time domain the result is

$$y(-1) = 1, y(0) = 0, y(1) = -2, y(2) = 0, y(3) = \frac{1}{4}, y(4) = 0, \dots, \quad (2.138)$$

which is depicted in Figure 2.4. \triangle

Example 2.12. If $X(z)$ is the z transform of the sequence

$$x(0) = a_0, x(1) = a_1, x(2) = a_2, \dots, x(i) = a_i, \dots, \quad (2.139)$$

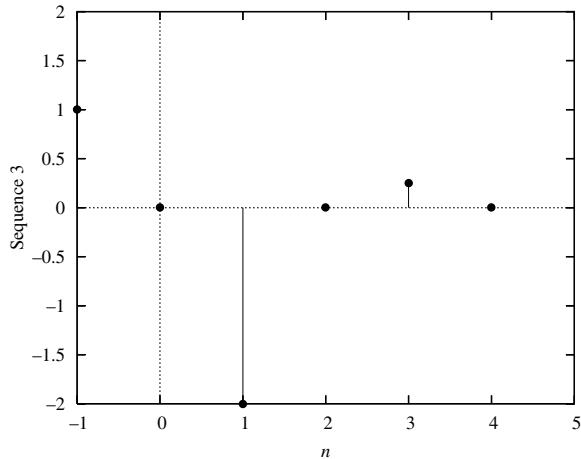


Fig. 2.4.

Resulting sequence from Example 2.11.

determine the z transform of the sequence

$$y(-2) = a_0, y(-3) = -a_1b, y(-4) = -2a_2b^2, \dots, y(-i-2) = -ia_ib^i, \dots \quad (2.140)$$

as a function of $X(z)$.

Solution

We have that $X(z)$ and $Y(z)$ are

$$X(z) = a_0 + a_1z^{-1} + a_2z^{-2} + \dots + a_iz^{-i} + \dots \quad (2.141)$$

$$Y(z) = a_0z^2 - a_1bz^3 - 2a_2b^2z^4 - \dots - ia_ib^iz^{i+2} - \dots \quad (2.142)$$

We start solving this problem by using the property in section 2.4.5; that is, if $x_1(n) = nx(n)$, then

$$\begin{aligned} X_1(z) &= -z \frac{dX(z)}{dz} \\ &= -z \left(-a_1z^{-2} - 2a_2z^{-3} - 3a_3z^{-4} - \dots - ia_iz^{-i-1} - \dots \right) \\ &= a_1z^{-1} + 2a_2z^{-2} + 3a_3z^{-3} + \dots + ia_iz^{-i} + \dots \end{aligned} \quad (2.143)$$

The next step is to create $x_2(n) = b^n x_1(n)$. From the property in section 2.4.4:

$$X_2(z) = X_1\left(\frac{z}{b}\right) = a_1bz^{-1} + 2a_2b^2z^{-2} + 3a_3b^3z^{-3} + \dots + ia_ib^iz^{-i} + \dots \quad (2.144)$$

We then generate $X_3(z) = z^{-2}X_2(z)$ as follows:

$$X_3(z) = a_1bz^{-3} + 2a_2b^2z^{-4} + 3a_3b^3z^{-5} + \dots + ia_ib^iz^{-i-2} + \dots \quad (2.145)$$

and make $X_4(z) = X_3(z^{-1})$ such that

$$X_4(z) = a_1 bz^3 + 2a_2 b^2 z^4 + 3a_3 b^3 z^5 + \cdots + ia_i b^i z^{i+2} + \cdots \quad (2.146)$$

The transform $Y(z)$ of the desired sequence is then

$$\begin{aligned} Y(z) &= a_0 z^2 - a_1 bz^3 - 2a_2 b^2 z^4 - 3a_3 b^3 z^5 - \cdots - ia_i b^i z^{i+2} - \cdots \\ &= a_0 z^2 - X_4(z). \end{aligned} \quad (2.147)$$

Using Equations (2.143) to (2.147), we can express the desired result as

$$\begin{aligned} Y(z) &= a_0 z^2 - X_4(z) \\ &= a_0 z^2 - X_3(z^{-1}) \\ &= a_0 z^2 - z^2 X_2(z^{-1}) \\ &= a_0 z^2 - z^2 X_1\left(\frac{z^{-1}}{b}\right) \\ &= a_0 z^2 - z^2 \left(-z \frac{dX(z)}{dz}\right) \Big|_{z=(z^{-1})/b} \\ &= a_0 z^2 + \frac{z}{b} \frac{dX(z)}{dz} \Big|_{z=(z^{-1})/b} \end{aligned} \quad (2.148)$$

△

2.5 Transfer functions

As we have seen in Chapter 1, a discrete-time linear system can be characterized by a difference equation. In this section, we show how the z transform can be used to solve difference equations and, therefore, to characterize linear systems.

The general form of a difference equation associated with a linear system is given by Equation (1.63), which we rewrite here for convenience:

$$\sum_{i=0}^N a_i y(n-i) - \sum_{l=0}^M b_l x(n-l) = 0. \quad (2.149)$$

Applying the z transform on both sides and using the linearity property, we find that

$$\sum_{i=0}^N a_i \mathcal{Z}\{y(n-i)\} - \sum_{l=0}^M b_l \mathcal{Z}\{x(n-l)\} = 0. \quad (2.150)$$

Applying the time-shift theorem, we obtain

$$\sum_{i=0}^N a_i z^{-i} Y(z) - \sum_{l=0}^M b_l z^{-l} X(z) = 0. \quad (2.151)$$

Therefore, for a linear system, given $X(z)$, the z -transform representation of the input, and the coefficients of its difference equation, we can use Equation (2.151) to find $Y(z)$, the z transform of the output. Applying the inverse z -transform relation in Equation (2.36), the output $y(n)$ can be computed for all n .¹

Making $a_0 = 1$, without loss of generality, we then define

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{l=0}^M b_l z^{-l}}{1 + \sum_{i=1}^N a_i z^{-i}} \quad (2.152)$$

as the transfer function of the system relating the output $Y(z)$ to the input $X(z)$.

Applying the convolution theorem to Equation (2.152), we have that

$$Y(z) = H(z)X(z) \longleftrightarrow y(n) = h(n) * x(n); \quad (2.153)$$

that is, the transfer function of the system is the z transform of its impulse response. Indeed, Equations (2.151) and (2.152) are the equivalent expressions of the convolution sum in the z -transform domain when the system is described by a difference equation.

Equation (2.152) gives the transfer function for the general case of recursive (IIR) filters. For nonrecursive (FIR) filters, all $a_i = 0$, for $i = 1, 2, \dots, N$, and the transfer function simplifies to

$$H(z) = \sum_{l=0}^M b_l z^{-l}. \quad (2.154)$$

Transfer functions are widely used to characterize discrete-time linear systems. We can describe a transfer function through its poles p_i and zeros z_l , yielding

$$H(z) = H_0 \frac{\prod_{l=1}^M (1 - z^{-1} z_l)}{\prod_{i=1}^N (1 - z^{-1} p_i)} = H_0 z^{N-M} \frac{\prod_{l=1}^M (z - z_l)}{\prod_{i=1}^N (z - p_i)}. \quad (2.155)$$

As discussed in Section 2.2, for a causal stable system, the convergence region of the z transform of its impulse response must include the unit circle. Indeed, this result is more general as, for *any* stable system, the convergence region of its transfer function must

¹ One should note that, since Equation (2.151) uses z transforms, which consist of summations for $-\infty < n < \infty$, then the system has to be describable by a difference equation for $-\infty < n < \infty$. This is the case only for initially relaxed systems; that is, systems that produce no output if the input is zero for $-\infty < n < \infty$. In our case, this does not restrict the applicability of Equation (2.151), because we are only interested in linear systems, which, as seen in Chapter 1, must be initially relaxed.

necessarily include the unit circle. We can show this by noting that, for z_0 on the unit circle ($|z_0| = 1$), we have

$$|H(z_0)| = \left| \sum_{n=-\infty}^{\infty} z_0^{-n} h(n) \right| \leq \sum_{n=-\infty}^{\infty} |z_0^{-n} h(n)| = \sum_{n=-\infty}^{\infty} |h(n)| < \infty, \quad (2.156)$$

which implies that $H(z)$ converges on the unit circle. Since in the case of a causal system the convergence region of the transfer function is $|z| > r_1$, then all the poles of a causal stable system must be inside the unit circle. For a noncausal stable system, since the convergence region is $|z| < r_2$, then all its poles must be outside the unit circle, with the possible exception of a pole at $z = 0$.

In Section 2.6 we present a numerical method for assessing the stability of a linear system without explicitly determining the positions of its poles.

2.6 Stability in the z domain

In this section we present a method for determining whether the roots of a polynomial are inside the unit circle of the complex plane. This method can be used to assess the BIBO stability of a causal discrete-time system.²

Given an N th-order polynomial in z

$$D(z) = a_N + a_{N-1}z + \cdots + a_0z^N \quad (2.157)$$

with $a_0 > 0$, we have that a necessary and sufficient condition for its zeros (the poles of the given transfer function) to be inside the unit circle of the z plane is given by the following algorithm:

- (i) Make $D_0(z) = D(z)$.
- (ii) For $k = 0, 1, \dots, (N - 2)$:
 - (a) form the polynomial $D_k^i(z)$ such that

$$D_k^i(z) = z^{N+k} D_k(z^{-1}); \quad (2.158)$$

- (b) compute α_k and $D_{k+1}(z)$, such that

$$D_k(z) = \alpha_k D_k^i(z) + D_{k+1}(z), \quad (2.159)$$

where the terms in z^j , for $j = 0, 1, \dots, k$, of $D_{k+1}(z)$ are zero. In other words, $D_{k+1}(z)$ is the remainder of the division of $D_k(z)$ by $D_k^i(z)$, when it is performed upon the terms of smallest degree.

² There are several such methods described in the literature (Jury, 1973). We have chosen to present this method in particular because it is based on polynomial division, which we consider a very important tool in the analysis and design of discrete-time systems.

- (iii) All the roots of $D(z)$ are inside the unit circle if the following conditions are satisfied:
- $D(1) > 0$
 - $D(-1) > 0$ for N even and $D(-1) < 0$ for N odd
 - $|\alpha_k| < 1$, for $k = 0, 1, \dots, (n-2)$.

Example 2.13. Test the stability of the causal system whose transfer function possesses the denominator polynomial $D(z) = 8z^4 + 4z^3 + 2z^2 - z - 1$.

Solution

If $D(z) = 8z^4 + 4z^3 + 2z^2 - z - 1$, then we have that

- $D(1) = 12 > 0$
- $N = 4$ is even and $D(-1) = 6 > 0$
- computation of α_0, α_1 , and α_2 :

$$D_0(z) = D(z) = 8z^4 + 4z^3 + 2z^2 - z - 1 \quad (2.160)$$

$$\begin{aligned} D_0^i(z) &= z^4(8z^{-4} + 4z^{-3} + 2z^{-2} - z^{-1} - 1) \\ &= 8 + 4z + 2z^2 - z^3 - z^4. \end{aligned} \quad (2.161)$$

Since $D_0(z) = \alpha_0 D_0^i(z) + D_1(z)$:

$$\begin{array}{c|l} 8 + 4z + 2z^2 - z^3 - z^4 & -1 - z + 2z^2 + 4z^3 + 8z^4 \\ \hline -\frac{1}{8} & +1 + \frac{1}{2}z + \frac{1}{4}z^2 - \frac{1}{8}z^3 - \frac{1}{8}z^4 \\ & -\frac{1}{2}z + \frac{9}{4}z^2 + \frac{31}{8}z^3 + \frac{63}{8}z^4 \end{array}$$

then $\alpha_0 = -\frac{1}{8}$ and

$$D_1(z) = -\frac{1}{2}z + \frac{9}{4}z^2 + \frac{31}{8}z^3 + \frac{63}{8}z^4 \quad (2.162)$$

$$\begin{aligned} D_1^i(z) &= z^{4+1} \left(-\frac{1}{2}z^{-1} + \frac{9}{4}z^{-2} + \frac{31}{8}z^{-3} + \frac{63}{8}z^{-4} \right) \\ &= -\frac{1}{2}z^4 + \frac{9}{4}z^3 + \frac{31}{8}z^2 + \frac{63}{8}z. \end{aligned} \quad (2.163)$$

Since $D_1(z) = \alpha_1 D_1^i(z) + D_2(z)$:

$$\begin{array}{c|l} \frac{63}{8}z + \frac{31}{8}z^2 + \frac{9}{4}z^3 - \frac{1}{2}z^4 & -\frac{1}{2}z + \frac{9}{4}z^2 + \frac{31}{8}z^3 + \frac{63}{8}z^4 \\ \hline -\frac{4}{63} & +\frac{1}{2}z + \frac{31}{126}z^2 + \frac{1}{7}z^3 - \frac{2}{63}z^4 \\ & 2.496z^2 + 4.018z^3 + 7.844z^4 \end{array}$$

then $\alpha_1 = -4/63$ and

$$D_2(z) = 2.496z^2 + 4.018z^3 + 7.844z^4 \quad (2.164)$$

$$\begin{aligned} D_2^i(z) &= z^{4+2}(2.496z^{-2} + 4.018z^{-3} + 7.844z^{-4}) \\ &= 2.496z^4 + 4.018z^3 + 7.844z^2. \end{aligned} \quad (2.165)$$

Since $D_2(z) = \alpha_2 D_2^i(z) + D_3(z)$, we have that $\alpha_2 = 2.496/7.844 = 0.3182$. Thus:

$$|\alpha_0| = \frac{1}{8} < 1, \quad |\alpha_1| = \frac{4}{63} < 1, \quad |\alpha_2| = 0.3182 < 1 \quad (2.166)$$

and, consequently, the system is stable.

△

Example 2.14. Given the polynomial $D(z) = z^2 + az + b$, determine the choices for a and b such that this polynomial represents the denominator of a stable discrete-time system. Plot $a \times b$, highlighting the stability region.

Solution

Since the order of the polynomial is even:

$$D(1) > 0 \Rightarrow 1 + a + b > 0 \Rightarrow a + b > -1 \quad (2.167)$$

$$D(-1) > 0 \Rightarrow 1 - a + b > 0 \Rightarrow -a + b > -1. \quad (2.168)$$

Since $N - 2 = 0$, there exists only α_0 . So:

$$D_0(z) = z^2 + az + b \quad (2.169)$$

$$D_0^i(z) = z^2(z^{-2} + az^{-1} + b) = 1 + az + bz^2 \quad (2.170)$$

$$\begin{array}{c} 1 + az + bz^2 \\ \hline b \\ \hline -b - abz - b^2z^2 \\ \hline (1 - b)az + (1 - b^2)z^2 \end{array}$$

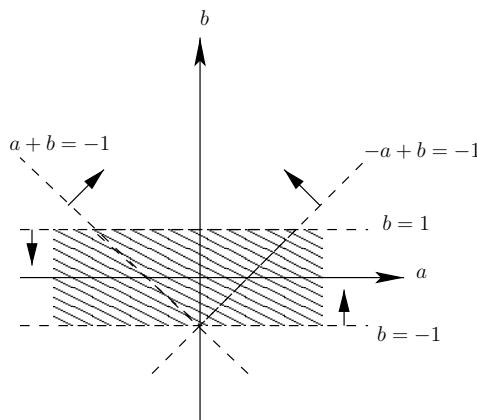


Fig. 2.5.

Stability region for Example 2.14.

then $|\alpha_0| = |b| < 1$. As such, the conditions are

$$\left. \begin{array}{l} a + b > -1 \\ -a + b > -1 \\ |b| < 1 \end{array} \right\}, \quad (2.171)$$

as depicted in Figure 2.5. \triangle

The complete derivation of the above algorithm, as well as a method to determine the number of roots a polynomial $D(z)$ has inside the unit circle, can be found in Jury (1973).

2.7 Frequency response

As mentioned in Section 2.1, when an exponential z^n is input to a linear system with impulse response $h(n)$, then its output is an exponential $H(z)z^n$. Since, as seen above, stable systems are guaranteed to have the z transform on the unit circle, it is natural to try to characterize these systems on the unit circle. Complex numbers on the unit circle are of the form $z = e^{j\omega}$, for $0 \leq \omega < 2\pi$. This implies that the corresponding exponential sequence is a sinusoid $x(n) = e^{j\omega n}$. Therefore, we can state that if we input a sinusoid $x(n) = e^{j\omega n}$ to a linear system, then its output is also a sinusoid of the same frequency; that is:

$$y(n) = H(e^{j\omega})e^{j\omega n}. \quad (2.172)$$

If $H(e^{j\omega})$ is a complex number with magnitude $|H(e^{j\omega})|$ and phase $\Theta(\omega)$, then $y(n)$ can be expressed as

$$y(n) = H(e^{j\omega})e^{j\omega n} = |H(e^{j\omega})|e^{j\Theta(\omega)}e^{j\omega n} = |H(e^{j\omega})|e^{j\omega n + j\Theta(\omega)}, \quad (2.173)$$

indicating that the output of a linear system for a sinusoidal input is a sinusoid of the same frequency, but with its amplitude multiplied by $|H(e^{j\omega})|$ and phase increased by $\Theta(\omega)$. Thus, when we characterize a linear system in terms of $H(e^{j\omega})$, we are in fact specifying, for every frequency ω , the effect that the linear system has on the input signal's amplitude and phase. Therefore, $H(e^{j\omega})$ is commonly known as the frequency response of the system.

It is important to emphasize that $H(e^{j\omega})$ is the value of the z transform $H(z)$ on the unit circle. This implies that we need to specify it only for one turn around the unit circle; that is, for $0 \leq \omega < 2\pi$. Indeed, since, for $k \in \mathbb{Z}$

$$H(e^{j(\omega+2\pi k)}) = H(e^{j2\pi k}e^{j\omega}) = H(e^{j\omega}), \quad (2.174)$$

then $H(e^{j\omega})$ is periodic with period 2π .

Another important characteristic of a linear discrete-time system is its group delay. This is defined as the derivative of the phase of its frequency response:

$$\tau(\omega) = -\frac{d\Theta(\omega)}{d\omega}. \quad (2.175)$$

When the phase $\Theta(\omega)$ is a linear function of ω , that is:

$$\Theta(\omega) = \beta\omega, \quad (2.176)$$

then the output $y(n)$ of a linear system to a sinusoid $x(n) = e^{j\omega n}$ is, according to Equation (2.173):

$$y(n) = |H(e^{j\omega})|e^{j\omega n + j\beta\omega} = |H(e^{j\omega})|e^{j\omega(n+\beta)}. \quad (2.177)$$

The above equation, together with Equation (2.175), implies that the output sinusoid is delayed by

$$-\beta = -\frac{d\Theta(\omega)}{d\omega} = \tau(\omega) \quad (2.178)$$

samples, irrespective of the frequency ω . Because of this property, the group delay is commonly used as a measure of how a linear time-invariant system delays sinusoids of different frequencies. Exercise 2.18 introduces a deeper discussion on this subject.

Example 2.15. Find the frequency response and the group delay of the FIR filter characterized by the following difference equation:

$$y(n) = \frac{x(n) + x(n-1)}{2}. \quad (2.179)$$

Solution

Taking the z transform of $y(n)$, we find

$$Y(z) = \frac{X(z) + z^{-1}X(z)}{2} = \frac{1}{2}(1 + z^{-1})X(z), \quad (2.180)$$

and then the transfer function of the system is

$$H(z) = \frac{1}{2}(1 + z^{-1}). \quad (2.181)$$

Making $z = e^{j\omega}$, the frequency response of the system becomes

$$H(e^{j\omega}) = \frac{1}{2}(1 + e^{-j\omega}) = \frac{1}{2}e^{-j\omega/2} \left(e^{j\omega/2} + e^{-j\omega/2} \right) = e^{-j\omega/2} \cos\left(\frac{\omega}{2}\right). \quad (2.182)$$

Since $\Theta(\omega) = -\frac{\omega}{2}$, then, from Equations (2.177) and (2.178), this implies that the system delays all sinusoids equally by half a sample. Also, the group delay $\tau(\omega) = \frac{1}{2}$ sample.

The magnitude and phase responses of $H(e^{j\omega})$ are depicted in Figure 2.6. Note that the frequency response is plotted for $-\pi \leq \omega < \pi$, rather than for $0 \leq \omega < 2\pi$. In practice, those two ranges are equivalent, since both comprise one period of $H(e^{j\omega})$. \triangle

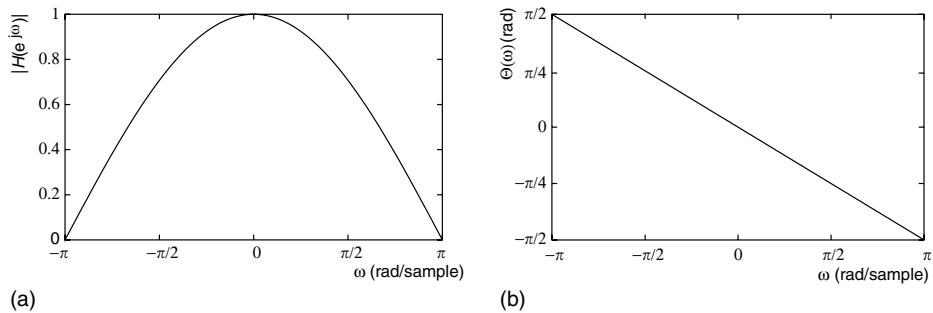


Fig. 2.6. Frequency response of the moving-average filter: (a) magnitude response; (b) phase response.

Example 2.16. A discrete-time system with impulse response $h(n) = (\frac{1}{2})^n u(n)$ is excited with $x(n) = \sin(\omega_0 n + \theta)$. Find the output $y(n)$ using the frequency response of the system.

Solution

Since

$$x(n) = \sin(\omega_0 n + \theta) = \frac{e^{j(\omega_0 n + \theta)} - e^{-j(\omega_0 n + \theta)}}{2j}, \quad (2.183)$$

the output $y(n) = \mathcal{H}\{x(n)\}$ is then,

$$\begin{aligned} y(n) &= \mathcal{H}\left\{\frac{e^{j(\omega_0 n + \theta)} - e^{-j(\omega_0 n + \theta)}}{2j}\right\} \\ &= \frac{1}{2j} \left[\mathcal{H}\{e^{j(\omega_0 n + \theta)}\} - \mathcal{H}\{e^{-j(\omega_0 n + \theta)}\} \right] \\ &= \frac{1}{2j} \left[H(e^{j\omega_0}) e^{j(\omega_0 n + \theta)} - H(e^{-j\omega_0}) e^{-j(\omega_0 n + \theta)} \right] \\ &= \frac{1}{2j} \left[|H(e^{j\omega_0})| e^{j\Theta(\omega_0)} e^{j(\omega_0 n + \theta)} - |H(e^{-j\omega_0})| e^{j\Theta(-\omega_0)} e^{-j(\omega_0 n + \theta)} \right]. \end{aligned} \quad (2.184)$$

Since $h(n)$ is real, from the property in Section 2.9.7, Equation (2.228), one has that $H(e^{j\omega}) = H^*(e^{-j\omega})$. This implies that

$$|H(e^{-j\omega})| = |H(e^{j\omega})| \quad \text{and} \quad \Theta(-\omega) = -\Theta(\omega). \quad (2.185)$$

Using this result, Equation (2.184) becomes

$$\begin{aligned} y(n) &= \frac{1}{2j} \left[|H(e^{j\omega_0})| e^{j\Theta(\omega_0)} e^{j(\omega_0 n + \theta)} - |H(e^{j\omega_0})| e^{-j\Theta(\omega_0)} e^{-j(\omega_0 n + \theta)} \right] \\ &= |H(e^{j\omega_0})| \left[\frac{e^{j(\omega_0 n + \theta + \Theta(\omega_0))} - e^{-j(\omega_0 n + \theta + \Theta(\omega_0))}}{2j} \right] \\ &= |H(e^{j\omega_0})| \sin(\omega_0 n + \theta + \Theta(\omega_0)). \end{aligned} \quad (2.186)$$

Since the system transfer function is

$$H(z) = \sum_{n=0}^{\infty} \left(\frac{1}{2}\right)^n z^{-n} = \frac{1}{1 - \frac{1}{2}z^{-1}}, \quad (2.187)$$

we have that

$$H(e^{j\omega}) = \frac{1}{1 - \frac{1}{2}e^{-j\omega}} = \frac{1}{\sqrt{\frac{5}{4} - \cos \omega}} e^{-j \arctan[\sin \omega / (2 - \cos \omega)]} \quad (2.188)$$

and then

$$|H(e^{j\omega})| = \frac{1}{\sqrt{\frac{5}{4} - \cos \omega}} \quad (2.189)$$

$$\Theta(\omega) = -\arctan\left(\frac{\sin \omega}{2 - \cos \omega}\right). \quad (2.190)$$

Substituting these values of $|H(e^{j\omega})|$ and $\Theta(\omega)$ in Equation (2.186), the output $y(n)$ becomes

$$y(n) = \frac{1}{\sqrt{\frac{5}{4} - \cos \omega_0}} \sin\left[\omega_0 n + \theta - \arctan\left(\frac{\sin \omega_0}{2 - \cos \omega_0}\right)\right]. \quad (2.191)$$

△

In general, when we design a discrete-time system, we have to satisfy predetermined magnitude, $|H(e^{j\omega})|$, and phase, $\Theta(\omega)$, characteristics. One should note that, when processing a continuous-time signal using a discrete-time system, we should translate the analog frequency Ω to the discrete-time frequency ω that is restricted to the interval $[-\pi, \pi]$. This can be done by noting that if an analog sinusoid $x_a(t) = e^{j\Omega t}$ is sampled as in Equation (1.157) to generate a sinusoid $e^{j\omega n}$; that is, if $\Omega_s = 2\pi/T$ is the sampling frequency, then

$$e^{j\omega n} = x(n) = x_a(nT) = e^{j\Omega n T}. \quad (2.192)$$

Therefore, one can deduce that the relation between the digital frequency ω and the analog frequency Ω is

$$\omega = \Omega T = 2\pi \frac{\Omega}{\Omega_s}, \quad (2.193)$$

indicating that the frequency interval $[-\pi, \pi]$ for the discrete-time frequency response corresponds to the frequency interval $[-\Omega_s/2, \Omega_s/2]$ in the analog domain.

Example 2.17. The sixth-order discrete-time lowpass elliptic filter, whose frequency response is shown in Figure 2.7, is used to process an analog signal in a similar scheme

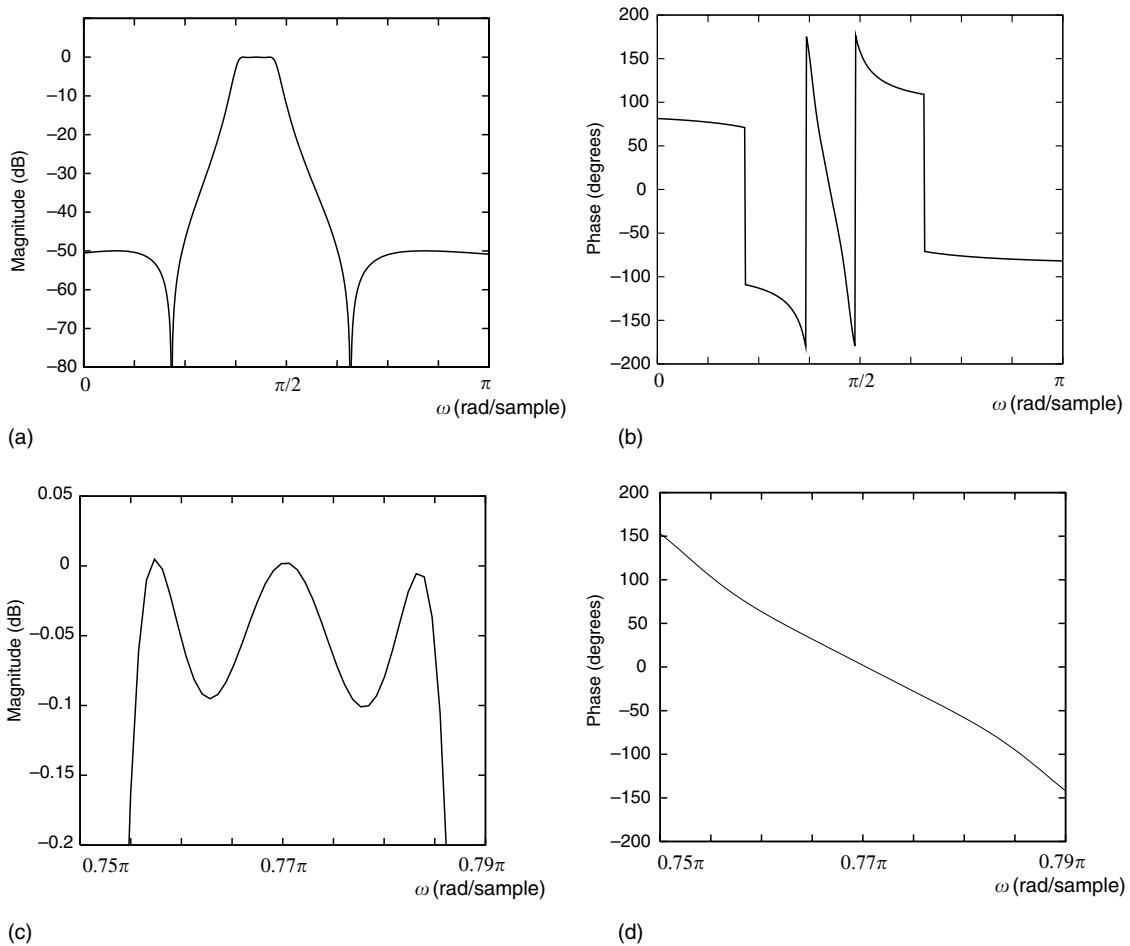


Fig. 2.7. Frequency response of a sixth-order elliptic filter: (a) magnitude response; (b) phase response; (c) magnitude response in the passband; (d) phase response in the passband.

to the one depicted in Figure 1.14. If the sampling frequency used in the analog-to-digital conversion is 8000 Hz, determine the passband of the equivalent analog filter. Consider the passband as the frequency range when the magnitude response of the filter is within 0.1 dB of its maximum value.

Solution

From Figure 2.7c, we see that the digital bandwidth in which the magnitude response of the system is within 0.1 dB of its maximum value is approximately from $\omega_{p1} = 0.755\pi$ rad/sample to $\omega_{p2} = 0.785\pi$ rad/sample. As the sampling frequency is

$$f_s = \frac{\Omega_p}{2\pi} = 8000 \text{ Hz}, \quad (2.194)$$

then the analog passband is such that

$$\Omega_{p_1} = 0.755\pi \frac{\Omega_s}{2\pi} = 0.755\pi \times 8000 = 6040\pi \text{ rad/s} \Rightarrow f_{p_1} = \frac{\Omega_{p_1}}{2\pi} = 3020 \text{ Hz}$$

(2.195)

$$\Omega_{p_2} = 0.785\pi \frac{\Omega_s}{2\pi} = 0.785\pi \times 8000 = 6280\pi \text{ rad/s} \Rightarrow f_{p_2} = \frac{\Omega_{p_2}}{2\pi} = 3140 \text{ Hz.}$$

(2.196)

△

The positions of the poles and zeros of a transfer function are very useful in the determination of its characteristics. For example, one can determine the frequency response $H(e^{j\omega})$ using a geometric method. Expressing $H(z)$ as a function of its poles and zeros as in Equation (2.155), we have that $H(e^{j\omega})$ becomes

$$H(e^{j\omega}) = H_0 e^{j\omega(N-M)} \frac{\prod_{l=1}^M (e^{j\omega} - z_l)}{\prod_{i=1}^N (e^{j\omega} - p_i)}$$

(2.197)

The magnitude and phase responses of $H(e^{j\omega})$ are then

$$|H(e^{j\omega})| = |H_0| \frac{\prod_{l=1}^M |e^{j\omega} - z_l|}{\prod_{i=1}^N |e^{j\omega} - p_i|}$$

(2.198)

$$\Theta(\omega) = \omega(N - M) + \sum_{l=1}^M \angle(e^{j\omega} - z_l) - \sum_{i=1}^N \angle(e^{j\omega} - p_i)$$

(2.199)

where $\angle z$ denotes the angle of the complex number z . The terms of the form $|e^{j\omega} - c|$ represent the distance between the point $e^{j\omega}$ on the unit circle and the complex number c . The terms of the form $\angle(e^{j\omega} - c)$ represent the angle of the line segment joining $e^{j\omega}$ and c and the real axis, measured in the counterclockwise direction.

For example, for $H(z)$ having its poles and zeros as in Figure 2.8, we have that

$$|H(e^{j\omega_0})| = \frac{D_3 D_4}{D_1 D_2 D_5 D_6}$$

(2.200)

$$\Theta(\omega_0) = \theta_3 + \theta_4 - \theta_1 - \theta_2 - \theta_5 - \theta_6.$$

(2.201)

See Experiment 2.3 in Section 2.12 for an illustration of how a pole–zero diagram can help in the design of simple discrete-time filters.

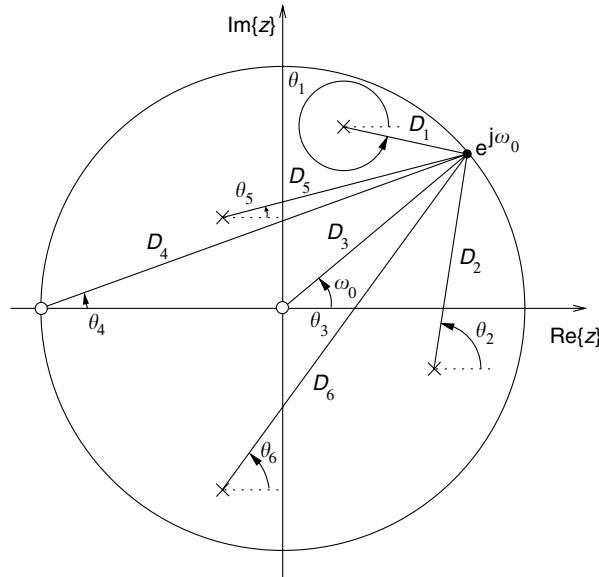


Fig. 2.8. Determination of the frequency response of $H(z)$ from the position of its poles (\times) and zeros (\circ).

2.8 Fourier transform

In the previous section we characterized linear discrete-time systems using the frequency response, which describes the behavior of a system when the input is a complex sinusoid. In this section we present the Fourier transform of discrete-time signals, which is a generalization of the concept of frequency response. It is equivalent to the decomposition of a discrete-time signal into an infinite sum of complex discrete-time sinusoids.

In Chapter 1, when deducing the sampling theorem, we formed, from the discrete-time signal $x(n)$, a continuous-time signal $x_i(t)$ consisting of a train of impulses with amplitude $x(n)$ at $t = nT$ (see Figure 1.5c). Its expression is given by Equation (1.163), which is repeated here for convenience:

$$x_i(t) = \sum_{n=-\infty}^{\infty} x(n)\delta(t - nT). \quad (2.202)$$

Since the Fourier transform of $\delta(t - Tn)$ is $e^{-j\Omega Tn}$, the Fourier transform of the continuous-time signal $x_i(t)$ becomes

$$X_i(j\Omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\Omega Tn}, \quad (2.203)$$

where Ω represents the analog frequency. Comparing this equation with the z -transform definition given in Equation (2.5), we conclude that the Fourier transform of $x_i(t)$ is such that

$$X_i(j\Omega) = X(e^{j\Omega T}). \quad (2.204)$$

This equation means that the Fourier transform at a frequency Ω of the continuous-time signal $x_i(t)$, which is generated by replacing each sample of the discrete-time signal $x(n)$ by an impulse of the same amplitude located at $t = nT$, is equal to the z transform of the signal $x(n)$ at $z = e^{j\Omega T}$. This fact implies that $X(e^{j\omega})$ holds the information about the frequency content of the signal $x_i(t)$ and, therefore, is a natural candidate to represent the frequency content of the discrete-time signal $x(n)$.

We can show that $X(e^{j\omega})$ does indeed represent the frequency content of $x(n)$ by applying the inverse z -transform formula in Equation (2.36) with C being the closed contour $z = e^{j\omega}$, for $-\pi \leq \omega < \pi$, resulting in

$$\begin{aligned} x(n) &= \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz \\ &= \frac{1}{2\pi j} \oint_{z=e^{j\omega}} X(z) z^{n-1} dz \\ &= \frac{1}{2\pi j} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega(n-1)} j e^{j\omega} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega, \end{aligned} \quad (2.205)$$

indicating that the discrete-time signal $x(n)$ can be expressed as an infinite summation of sinusoids, where $e^{j\omega n}$, the sinusoid of frequency ω , has a complex amplitude proportional to $X(e^{j\omega})$. Thus, computing $X(e^{j\omega})$ is equivalent to decomposing the discrete-time signal $x(n)$ into a sum of complex discrete-time sinusoids.

This interpretation of $X(e^{j\omega})$ parallels that of the Fourier transform of a continuous-time signal $x_a(t)$. The direct and inverse Fourier transforms of a continuous-time signal are given by

$$\left. \begin{aligned} X_a(j\Omega) &= \int_{-\infty}^{\infty} x_a(t) e^{-j\Omega t} dt \\ x_a(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X_a(j\Omega) e^{j\Omega t} d\Omega \end{aligned} \right\}. \quad (2.206)$$

This pair of equations indicates that a continuous-time signal $x_a(t)$ can be expressed as an infinite summation of continuous-time sinusoids, where $e^{j\Omega t}$, the sinusoid of frequency Ω , has a complex amplitude proportional to $X_a(j\Omega)$. Then computing $X_a(j\Omega)$ is equivalent to decomposing the continuous-time signal into a sum of complex continuous-time sinusoids.

From the above discussion, we see that $X(e^{j\omega})$ defines a Fourier transform of the discrete-time signal $x(n)$, with its inverse given by Equation (2.205). Indeed, the direct and inverse

Fourier transforms of a sequence $x(n)$ are formally defined as

$$\left. \begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\ x(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega \end{aligned} \right\}. \quad (2.207)$$

Naturally, the Fourier transform $X(e^{j\omega})$ of a discrete-time signal $x(n)$ is periodic with period 2π , since

$$X(e^{j\omega}) = X(e^{j(\omega+2\pi k)}), \text{ for all } k \in \mathbb{Z}. \quad (2.208)$$

Therefore, the Fourier transform of a discrete-time signal requires specification only for a range of 2π , as, for example, $\omega \in [-\pi, \pi]$ or $\omega \in [0, 2\pi]$.

Example 2.18. Compute the Fourier transform of the sequence

$$x(n) = \begin{cases} 1, & 0 \leq n \leq 5 \\ 0, & \text{otherwise} \end{cases}. \quad (2.209)$$

Solution

$$X(e^{j\omega}) = \sum_{k=0}^5 e^{-jk\omega} = \frac{1 - e^{-6j\omega}}{1 - e^{-j\omega}} = \frac{e^{-3j\omega}(e^{3j\omega} - e^{-3j\omega})}{e^{-j\omega/2}(e^{j\omega/2} - e^{-j\omega/2})} = e^{-j5\omega/2} \frac{\sin(3\omega)}{\sin(\omega/2)}. \quad (2.210)$$

The magnitude and phase responses of $X(e^{j\omega})$ are depicted in Figures 2.9a and 2.9b respectively. Note that in Figure 2.9b the phase has been wrapped to fit in the interval

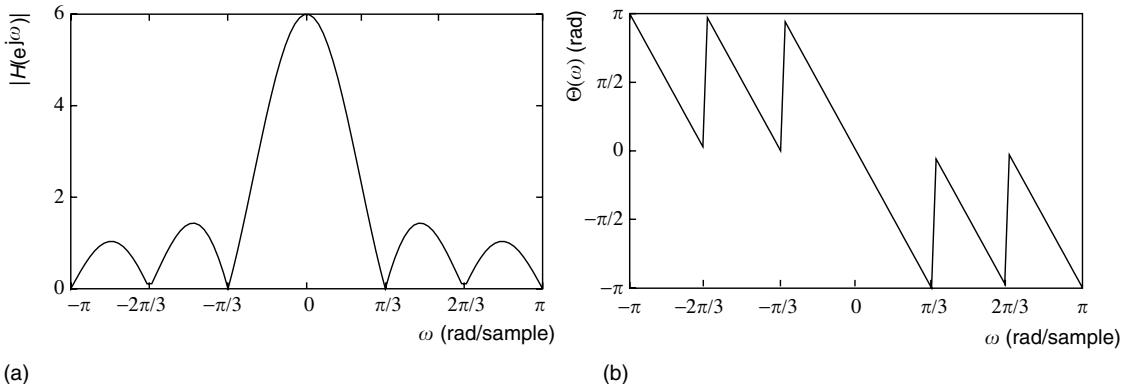


Fig. 2.9. Fourier transform of the sequence in Example 2.18: (a) magnitude response; (b) phase response.

$[-\pi, \pi]$ (that is, instead of plotting the phase $\Theta(\omega)$, we plot $\overline{\Theta}(\omega) = \Theta(\omega) + 2k(\omega)\pi$, with $k(\omega)$ an integer such that $\overline{\Theta}(\omega) \in [-\pi, \pi]$ for all ω). \triangle

One should note that, in order for the Fourier transform of a discrete-time signal to exist for all ω , its z transform must converge for $|z| = 1$. From the discussion in Section 2.5, we have seen that whenever

$$\sum_{n=-\infty}^{\infty} |x(n)| < \infty \quad (2.211)$$

then the z transform converges on the unit circle and, therefore, the Fourier transform exists for all ω . One example when Equation (2.211) does not hold and the Fourier transform does not exist for all ω is given by the sequence $x(n) = u(n)$. This case is discussed in Example 2.19. In Exercise 2.23 there is an example in which Equation (2.211) does not hold but the Fourier transform does exist for all ω . Therefore, the condition in Equation (2.211) is sufficient, but not necessary for the Fourier transform to exist. In addition, we have that not all sequences which have a Fourier transform have a z transform. For example, the z transform of any sequence is continuous in its convergence region (Churchill, 1975); hence, the sequences whose Fourier transforms are discontinuous functions of ω do not have a z transform, as also seen in Exercise 2.23.

Example 2.19. Compute the Fourier transform of the sequence $x(n) = u(n)$.

Solution

From Equation (2.8) in Example 2.1, we have that the z transform of $x(n) = u(n)$ is

$$X(z) = \frac{1}{1 - z^{-1}}, \quad |z| > 1. \quad (2.212)$$

We know that the z transform of $x(n)$ does not converge either for $|z| < 1$ or for $z = 1$ and $z = -1$. However, one can say nothing about the other values of z on the unit circle; that is, for $z = e^{j\omega}$ for any given ω . Since with Equation (2.212) we can compute $X(z)$ for $|z| > 1$ (that is, for $z = \rho e^{j\omega}$ with $\rho > 1$), we try to compute $X(e^{j\omega})$ as

$$\begin{aligned} X(e^{j\omega}) &= \lim_{\rho \rightarrow 1} X(\rho e^{j\omega}) \\ &= \lim_{\rho \rightarrow 1} \frac{1}{1 - \rho e^{-j\omega}} \\ &= \frac{1}{1 - e^{-j\omega}} \\ &= \frac{e^{j\omega/2}}{e^{j\omega/2} - e^{-j\omega/2}} \\ &= \frac{e^{j\omega/2}}{2j \sin(\omega/2)} \\ &= \frac{e^{j[\omega/2 - \pi/2]}}{2 \sin(\omega/2)}. \end{aligned} \quad (2.213)$$

From Equation (2.213), we can see that the Fourier transform of $x(n)$ does not exist for $\sin(\omega/2) = 0$; that is, for $\omega = k\pi$, which corresponds to $z = \pm 1$. However, $X(e^{j\omega})$ does exist for all $\omega \neq k\pi$. Although this result indicates that one can use $X(e^{j\omega})$ as the frequency content of $x(n)$, its implications should be considered with caution. For example, the inverse Fourier transform in Equation (2.205) is based on the convergence of $X(z)$ on the unit circle. Since $X(z)$ does not converge on the whole unit circle, then Equation (2.205) is not valid for computing $x(n)$. See Exercise 2.22 for a way to compute $x(n)$ from $X(e^{j\omega})$. \triangle

Example 2.20. Compute the Fourier transform of the sequence $x(n) = e^{j\omega_0 n}$.

Solution

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} e^{j\omega_0 n} e^{-j\omega n} = \sum_{n=-\infty}^{\infty} e^{j(\omega_0 - \omega)n}. \quad (2.214)$$

From Equations (1.165) and (1.168), we have that

$$\sum_{n=-\infty}^{\infty} \delta(t - nT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j(2\pi/T) nt}. \quad (2.215)$$

By making $T = 2\pi$ and $t = (\omega_0 - \omega)$ in Equation (2.215), one can express the transform in Equation (2.214) as

$$X(e^{j\omega}) = 2\pi \sum_{n=-\infty}^{\infty} \delta(\omega_0 - \omega - 2\pi n) = 2\pi \sum_{n=-\infty}^{\infty} \delta(\omega - \omega_0 + 2\pi n). \quad (2.216)$$

That is, the Fourier transform of a complex sinusoid of infinite duration and frequency ω_0 is an impulse centered at frequency ω_0 and repeated with period 2π . \triangle

In Chapter 1, Equation (1.170), we gave a relation between the Fourier transform of the signal $x_i(t)$ in Equation (1.163) and the one of the original analog signal $x_a(t)$, which we rewrite here for convenience:

$$X_i(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a \left(j\Omega - j\frac{2\pi}{T} k \right). \quad (2.217)$$

If the discrete-time signal $x(n)$ is such that $x(n) = x_a(nT)$, then we can use this equation to derive the relation between the Fourier transforms of the discrete-time and continuous-time signals, $X(e^{j\omega})$ and $X_a(j\Omega)$. In fact, from Equation (2.204):

$$X_i(j\Omega) = X(e^{j\Omega T}) \quad (2.218)$$

and making the change of variables $\Omega = \omega/T$ in Equation (2.217), yields

$$X(e^{j\omega}) = X_i \left(j\frac{\omega}{T} \right) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a \left(j\frac{\omega - 2\pi k}{T} \right); \quad (2.219)$$

that is, $X(e^{j\omega})$ is composed of copies of $X_a(j\omega/T)$ repeated in intervals of 2π . Also, as seen in Chapter 1, one can recover the analog signal $x_a(t)$ from $x(n)$ provided that $X_a(j\Omega) = 0$ for $|\Omega| \geq \Omega_s/2 = \pi/T$.

2.9 Properties of the Fourier transform

As seen previously, the Fourier transform $X(e^{j\omega})$ of a sequence $x(n)$ is equal to its z transform $X(z)$ at $z = e^{j\omega}$. Therefore, most properties of the Fourier transform derive from those of the z transform by simple substitution of z by $e^{j\omega}$. In what follows, we state them without proof, except in those cases where the properties do not have a straightforward z -transform correspondent.

2.9.1 Linearity

$$k_1x_1(n) + k_2x_2(n) \longleftrightarrow k_1X_1(e^{j\omega}) + k_2X_2(e^{j\omega}). \quad (2.220)$$

2.9.2 Time reversal

$$x(-n) \longleftrightarrow X(e^{-j\omega}). \quad (2.221)$$

2.9.3 Time-shift theorem

$$x(n+l) \longleftrightarrow e^{j\omega l}X(e^{j\omega}), \quad (2.222)$$

where l is an integer.

2.9.4 Multiplication by a complex exponential (frequency shift, modulation)

$$e^{j\omega_0 n}x(n) \longleftrightarrow X(e^{j(\omega-\omega_0)}). \quad (2.223)$$

2.9.5 Complex differentiation

$$nx(n) \longleftrightarrow j \frac{dX(e^{j\omega})}{d\omega}. \quad (2.224)$$

2.9.6 Complex conjugation

$$x^*(n) \longleftrightarrow X^*(e^{-j\omega}). \quad (2.225)$$

2.9.7 Real and imaginary sequences

Before presenting the properties of the Fourier transforms of real, imaginary, symmetric, and antisymmetric sequences, it is useful to give the precise definitions below:

- A symmetric (even) function is such that $f(u) = f(-u)$.
- An antisymmetric (odd) function is such that $f(u) = -f(-u)$.
- A conjugate symmetric function is such that $f(u) = f^*(-u)$.
- A conjugate antisymmetric function is such that $f(u) = -f^*(-u)$.

The following properties hold:

$$\operatorname{Re}\{x(n)\} \longleftrightarrow \frac{1}{2} \left(X(e^{j\omega}) + X^*(e^{-j\omega}) \right) \quad (2.226)$$

$$\operatorname{Im}\{x(n)\} \longleftrightarrow \frac{1}{2j} \left(X(e^{j\omega}) - X^*(e^{-j\omega}) \right). \quad (2.227)$$

If $x(n)$ is real, then $\operatorname{Im}\{x(n)\} = 0$. Hence, from Equation (2.227),

$$X(e^{j\omega}) = X^*(e^{-j\omega}); \quad (2.228)$$

that is, the Fourier transform of a real sequence is conjugate symmetric. The following properties for real $x(n)$ follow directly from Equation (2.228):

- The real part of the Fourier transform of a real sequence is even:

$$\operatorname{Re}\{X(e^{j\omega})\} = \operatorname{Re}\{X(e^{-j\omega})\}. \quad (2.229)$$

- The imaginary part of the Fourier transform of a real sequence is odd:

$$\operatorname{Im}\{X(e^{j\omega})\} = -\operatorname{Im}\{X(e^{-j\omega})\}. \quad (2.230)$$

- The magnitude of the Fourier transform of a real sequence is even:

$$|X(e^{j\omega})| = |X(e^{-j\omega})|. \quad (2.231)$$

- The phase of the Fourier transform of a real sequence is odd:

$$\angle[X(e^{j\omega})] = -\angle[X(e^{-j\omega})]. \quad (2.232)$$

Similarly, if $x(n)$ is imaginary, then $\operatorname{Re}\{x(n)\} = 0$. Hence, from Equation (2.226):

$$X(e^{j\omega}) = -X^*(e^{-j\omega}). \quad (2.233)$$

Properties similar to those in Equations (2.229) to (2.232) can be deduced for imaginary sequences, which is left as an exercise to the reader.

2.9.8 Symmetric and antisymmetric sequences

- If $x(n)$ is real and symmetric, then $X(e^{j\omega})$ is also real and symmetric.

Proof

$$\begin{aligned}
 X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\
 &= \sum_{n=-\infty}^{\infty} x(-n)e^{-j\omega n} \\
 &= \sum_{m=-\infty}^{\infty} x(m)e^{j\omega m} \\
 &= X(e^{-j\omega}) \\
 &= \sum_{m=-\infty}^{\infty} x(m)(e^{-j\omega m})^* \\
 &= \sum_{m=-\infty}^{\infty} (x(m)e^{-j\omega m})^* \\
 &= X^*(e^{j\omega}). \tag{2.234}
 \end{aligned}$$

Hence, if $X(e^{j\omega}) = X(e^{-j\omega})$, then $X(e^{j\omega})$ is even, and if $X(e^{j\omega}) = X^*(e^{j\omega})$, then $X(e^{j\omega})$ is real. \square

- If $x(n)$ is imaginary and even, then $X(e^{j\omega})$ is imaginary and even.
- If $x(n)$ is real and odd, then $X(e^{j\omega})$ is imaginary and odd.
- If $x(n)$ is imaginary and odd, then $X(e^{j\omega})$ is real and odd.
- If $x(n)$ is conjugate symmetric, then $X(e^{j\omega})$ is real.

Proof

$$\begin{aligned}
 X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \\
 &= \sum_{n=-\infty}^{\infty} x^*(-n)e^{-j\omega n} \\
 &= \sum_{m=-\infty}^{\infty} x^*(m)e^{j\omega m}
 \end{aligned}$$

$$\begin{aligned}
&= \sum_{m=-\infty}^{\infty} (x(m)e^{-j\omega m})^* \\
&= X^*(e^{j\omega})
\end{aligned} \tag{2.235}$$

and then $X(e^{j\omega})$ is real. \square

- If $x(n)$ is conjugate antisymmetric, then $X(e^{j\omega})$ is imaginary.

2.9.9 Convolution theorem

$$x_1(n) * x_2(n) \longleftrightarrow X_1(e^{j\omega})X_2(e^{j\omega}). \tag{2.236}$$

2.9.10 Product of two sequences

$$\begin{aligned}
x_1(n)x_2(n) &\longleftrightarrow \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1(e^{j\Omega})X_2(e^{j(\omega-\Omega)}) d\Omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1(e^{j(\omega-\Omega)})X_2(e^{j\Omega}) d\Omega \\
&= X_1(e^{j\omega}) \circledast X_2(e^{j\omega}).
\end{aligned} \tag{2.237}$$

2.9.11 Parseval's theorem

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1(e^{j\omega})X_2^*(e^{j\omega}) d\omega. \tag{2.238}$$

If we make $x_1(n) = x_2(n) = x(n)$, then Parseval's theorem becomes

$$\sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega. \tag{2.239}$$

The left-hand side of this equation corresponds to the energy of the sequence $x(n)$ and its right-hand side corresponds to the energy of $X(e^{j\omega})$ divided by 2π . Hence, Equation (2.239) means that the energy of a sequence is the same as the energy of its Fourier transform divided by 2π .

2.10 Fourier transform for periodic sequences

A special case of the Fourier transform that should be considered is the one of the periodic sequences. In what follows, we will obtain the expression of the Fourier transform of a periodic sequence. From it, we will define a Fourier series for discrete-time signals.

We start by considering a signal $x_f(n)$ with N nonzero samples. Without loss of generality, we make

$$x_f(n) = 0, \quad \text{for } n < 0 \text{ and } n \geq N. \quad (2.240)$$

According to Equation (2.207), its Fourier transform is given by

$$X_f(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_f(n)e^{-j\omega n} = \sum_{n=0}^{N-1} x_f(n)e^{-j\omega n}. \quad (2.241)$$

Next, we build from $x_f(n)$ a periodic signal $x(n)$ with period N composed of versions of $x_f(n)$ shifted to the positions kN , for all $k \in \mathbb{Z}$:

$$x(n) = \sum_{k=-\infty}^{\infty} x_f(n + kN). \quad (2.242)$$

Using the time-shift property in Section 2.4.3, its Fourier transform is

$$X(e^{j\omega}) = \sum_{k=-\infty}^{\infty} e^{j\omega kN} X_f(e^{j\omega}) = X_f(e^{j\omega}) \sum_{k=-\infty}^{\infty} e^{j\omega kN}. \quad (2.243)$$

From Equations (1.165) and (1.168), making $T = 2\pi/N$ and $t = \omega$, we have that Equation (2.243) becomes

$$\begin{aligned} X(e^{j\omega}) &= X_f(e^{j\omega}) \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right) \\ &= \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} X_f\left(e^{j(2\pi/N)k}\right) \delta\left(\omega - \frac{2\pi}{N}k\right) \\ &= \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} X(k) \delta\left(\omega - \frac{2\pi}{N}k\right), \end{aligned} \quad (2.244)$$

where

$$X(k) = X_f\left(e^{j(2\pi/N)k}\right) = \sum_{n=0}^{N-1} x_f(n)e^{-j(2\pi/N)k} = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)k} \quad (2.245)$$

By computing the inverse Fourier transform of Equation (2.244), we have that

$$\begin{aligned}
 x(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega \\
 &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} X(k) \delta\left(\omega - \frac{2\pi}{N} k\right) e^{j\omega n} d\omega \\
 &= \frac{1}{N} \sum_{k=-\infty}^{\infty} X(k) \int_{-\pi}^{\pi} \delta\left(\omega - \frac{2\pi}{N} k\right) e^{j\omega n} d\omega \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j(2\pi/N)kn}.
 \end{aligned} \tag{2.246}$$

This represents the expansion of a discrete-time signal $x(n)$ of period N as a sum of complex discrete-time sinusoids with frequencies that are multiples of $2\pi/N$, the fundamental frequency of $x(n)$. Therefore, Equation (2.246) can be regarded as a Fourier series expansion of $x(n)$. Note that Equations (2.245) and (2.246) define a Fourier series pair for the periodic signal, which can be used whenever one wants to avoid the impulses in frequency that come from the Fourier transform in Equation (2.244).

2.11 Random signals in the transform domain

The representation of a random process in the transform domain is not straightforward. The direct application of the z or Fourier transforms, as defined in Equations (2.5) and (2.207), respectively, on a random process does not make much sense, since in such a case the waveform $x(n)$ is by definition unknown. In addition, several types of random process present infinite energy, requiring some specific mathematical treatment. This, however, does not indicate that there are no transform-domain tools for random signal analysis. Random signal analysis can indeed benefit a great deal from transform-domain tools. This section introduces a frequency representation for the autocorrelation function and uses it to characterize the input–output relationship of a linear system when processing random signals.

2.11.1 Power spectral density

The so-called power spectral density (PSD) function $\Gamma_X(e^{j\omega})$ is defined as the Fourier transform of the autocorrelation function of a given random process $\{X\}$. For a WSS process, we have that

$$\Gamma_X(e^{j\omega}) = \sum_{v=-\infty}^{\infty} R_X(v) e^{-j\omega v} \tag{2.247}$$

in such a way that

$$R_X(v) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_X(e^{j\omega}) e^{j\omega v} d\omega. \quad (2.248)$$

Equations (2.247) and (2.248) are jointly referred to as the Wiener–Khinchin theorem³ for discrete-time signals.

In particular, by setting $v = 0$ in Equation (2.248) we get

$$R_X(0) = E\{X^2(n)\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_X(e^{j\omega}) d\omega. \quad (2.249)$$

If a random signal $x(n)$ from a WSS process $\{X\}$ is filtered by a linear time-invariant system, with impulse response $h(n)$, then the PSD function for the output signal $y(n)$, using Equations (2.247) and (1.240), is given by

$$\begin{aligned} \Gamma_Y(e^{j\omega}) &= \sum_{v=-\infty}^{\infty} R_Y(v) e^{-j\omega v} \\ &= \sum_{v=-\infty}^{\infty} \left(\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} R_X(v - k_1 + k_2) h(k_1) h(k_2) \right) e^{-j\omega v}. \end{aligned} \quad (2.250)$$

By defining the auxiliary time variable $v' = (v - k_1 + k_2)$, such that $v = (v' + k_1 - k_2)$, then

$$\begin{aligned} \Gamma_Y(e^{j\omega}) &= \sum_{v'=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} R_X(v') h(k_1) h(k_2) e^{-j\omega(v'+k_1-k_2)} \\ &= \sum_{v'=-\infty}^{\infty} R_X(v') e^{-j\omega v'} \left[\sum_{k_1=-\infty}^{\infty} h(k_1) e^{-j\omega k_1} \left(\sum_{k_2=-\infty}^{\infty} h(k_2) e^{+j\omega k_2} \right) \right] \\ &= \Gamma_X(e^{j\omega}) H(e^{j\omega}) H^*(e^{j\omega}) \end{aligned} \quad (2.251)$$

or, equivalently:

$$\Gamma_Y(e^{j\omega}) = \Gamma_X(e^{j\omega}) |H(e^{j\omega})|^2. \quad (2.252)$$

Therefore, the output PSD function is the input PSD function multiplied by the squared magnitude response of the linear system. This is the equivalent frequency-domain description for random signals of the input–output relationship of a linear time-invariant system whose input is a deterministic signal.

Consider the processing of a random signal $x(t)$ by a filter with unit gain and narrow passband, $[\omega_0 - \Delta\omega/2] \leq \omega \leq [\omega_0 + \Delta\omega/2]$, as depicted in Figure 2.10.

³ The interested reader may refer to Gardner (1987), Einstein (1987), and Yaglom (1987) for discussion on Einstein's 1914 paper on this subject.

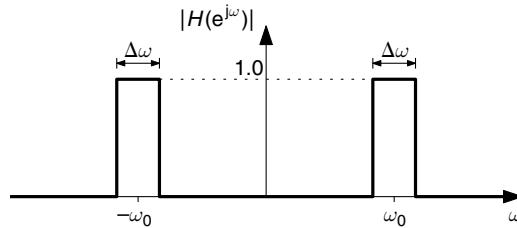


Fig. 2.10. Magnitude response of ideal filter with narrow passband.

If $\Delta\omega$ is small enough, then we may consider the input PSD constant around ω_0 in such a way that, using Equation (2.249), the output mean-squared value may be written as

$$\begin{aligned}
 E\{Y^2(n)\} &= R_Y(0) \\
 &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_Y(e^{j\omega}) d\omega \\
 &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_X(e^{j\omega}) \left| H(e^{j\omega}) \right|^2 d\omega \\
 &= \frac{2}{2\pi} \int_{\omega_0 - \Delta\omega/2}^{\omega_0 + \Delta\omega/2} \Gamma_X(e^{j\omega}) d\omega \\
 &\approx \frac{\Delta\omega}{\pi} \Gamma_X(e^{j\omega_0})
 \end{aligned} \tag{2.253}$$

and then

$$\Gamma_X(e^{j\omega_0}) \approx \frac{E\{Y^2(n)\}}{\Delta\omega/\pi}. \tag{2.254}$$

This result indicates that the value of the PSD at ω_0 is a measure of the density of signal power around that frequency, which justifies the nomenclature used for that function.

Example 2.21. Determine the PSD of the random process $\{X\}$ described by

$$x_m(n) = \cos(\omega_0 n + \theta_m), \tag{2.255}$$

where θ_m is the value of a continuous random variable with uniform PDF within the interval $[0, 2\pi]$.

Solution

As determined in Example 1.17, the random process $\{X\}$ is WSS with autocorrelation function given by

$$R_X(v) = \frac{1}{2} \cos(\omega_0 v) = \frac{e^{j\omega_0 v} + e^{-j\omega_0 v}}{4} \tag{2.256}$$

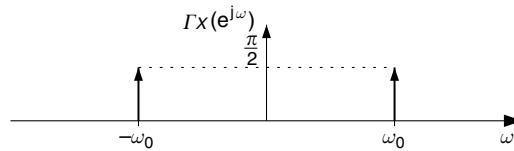


Fig. 2.11. PSD of random process $\{X\}$ in Example 2.21.

Therefore, from the definition given in Equation (2.247), and using Equation (2.216) from Example 2.20, the PSD function of $\{X\}$ is given by

$$\Gamma_X(e^{j\omega}) = \frac{\pi}{2} \sum_{n=-\infty}^{\infty} [\delta(\omega - \omega_0 + 2\pi n) + \delta(\omega + \omega_0 + 2\pi n)]. \quad (2.257)$$

This result indicates that the random process $\{X\}$ only presents signal power around the frequencies $(\pm\omega_0 + 2k\pi)$, for $k \in \mathbb{Z}$, despite its random phase component Θ . This is depicted in Figure 2.11 for frequencies in the interval $[-\pi, \pi]$. \triangle

2.11.2 White noise

A very important class of random processes includes the so-called white noise, characterized by a constant PSD function for all values of ω . Using the Wiener–Khinchin theorem, we easily infer that the autocorrelation function for the white noise is an impulse at the lag origin $v = 0$. This indicates that for any non-zero lag v , the white noise samples are statistically uncorrelated. The white noise nomenclature comes from the analogy to white light, which includes all frequency components with similar power. White noise is a powerful tool in signal analysis, due to its simple descriptions in both frequency and time (lag) domains, as visualized in Figure 2.12, making it quite suitable to model any unpredictable portion of a process.

Due to its infinite average power, a perfect white noise cannot be found in nature. It, however, can be approximated well by pseudo-random generating algorithms, such

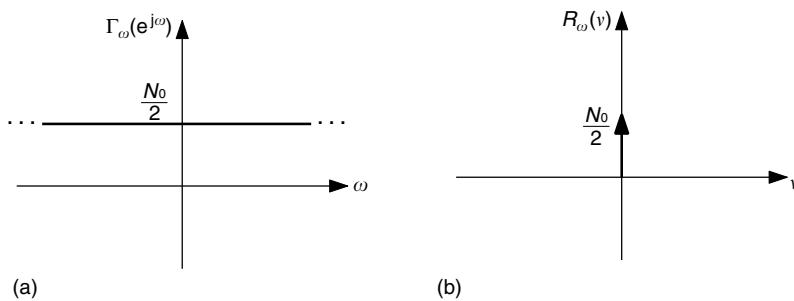


Fig. 2.12. White noise characterization: (a) frequency domain; (b) lag domain.

as the MATLAB commands `rand` and `randn`, as will be explored in the following Do-it-yourself section.

2.12 Do-it-yourself: the z and Fourier transforms

In this section we revise with the support of MATLAB some of the concepts presented throughout this chapter.

Experiment 2.1

By comparing Equation (2.34) with Equations (2.61) and (2.63), we can see that the residue of a pole p_k of $X(z)z^{n-1}$, with multiplicity m_k , is equal to the coefficient c_{k1} of its partial-fraction expansion. Therefore, the MATLAB command `residue` may be used to perform the inverse z transform of a given transfer function. In this context, the `residue` command shall receive only two input parameters containing the numerator and denominator polynomials of $X(z)z^{n-1}$ in descending powers of z . Revisiting Example 2.3, the associated impulse response $x(n)$ can be determined for $0 \leq n \leq P$, with $P = 20$, as

```
x = zeros(1,P+1);
num = [1 zeros(1,P+1)];
den = [1 0.6 -0.16];
for n = 0:P,
    [r,p,k] = residue(num(1:n+2),den);
    x(n+1) = sum(r);
end;
stem(0:P,x);
```

In this short script, the `r` variable receives the desired residue values which are summed up to determine the $x(n)$ sequence depicted in Figure 2.13.

For a pole of multiplicity $m > 1$, the `residue` command evaluates Equation (2.34) for $m_k = 1, 2, \dots, m$. In such cases, we must consider only the residue for $m_k = m$ in the summation that determines $x(n)$.

Experiment 2.2

In Experiment 1.3 we analyzed the behavior of a system whose output–input relationship is described by

$$y(n) = \frac{x(n) + x(n-1) + \cdots + x(n-N+1)}{N}. \quad (2.258)$$

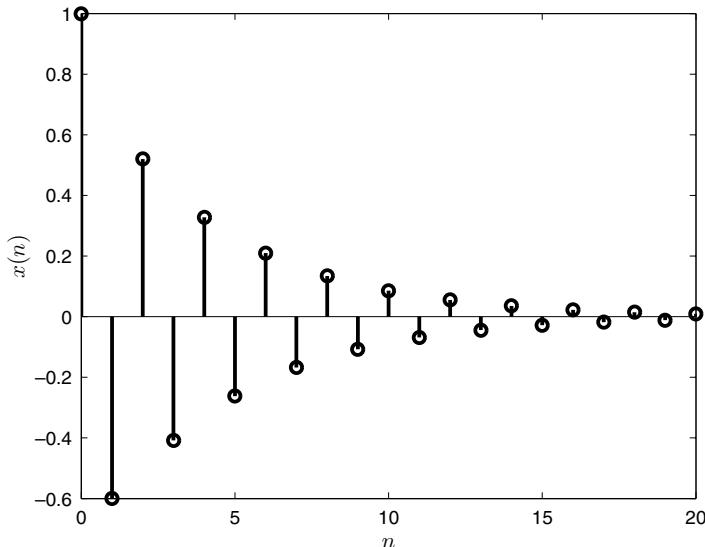


Fig. 2.13. Impulse response of Example 2.3 obtained with MATLAB command `residue`.

Taking this relation into the z domain and using the time-shift property associated with the z transform, we get the causal-form transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + z^{-1} + \cdots + z^{-N+1}}{N}. \quad (2.259)$$

As explained in Section 2.7, by letting $z = e^{j\omega}$, with $0 \leq \omega \leq 2\pi$, we determine the frequency response of the system described in Equation (2.259). In MATLAB, however, this response is easily obtained using the command `freqz`. In order to do so, we must first rewrite $H(z)$ in its rational-polynomial form with nonnegative exponents. For large values of N , the numerator and denominator polynomials of $H(z)$ are efficiently defined with the matrix commands `zeros` and `ones` as exemplified here for $N = 10$:

```
num10 = ones(1,N);
den10 = [N, zeros(1,N-1)];
[H10,W] = freqz(num10,den10);
figure(1); plot(W,abs(H10));
```

The last line generates the resulting magnitude response, which for $N = 10$ corresponds to the dashed curve in Figure 2.14. The phase response could have been obtained in a similar fashion by replacing `abs` with the command `angle`. The group delay, as defined in Equation (2.175), can be determined from the phase response or directly with the `grpdelay` command, whose input and output arguments are the same as in the `freqz` command. Repeating the script above for $N = 3, 6$, and 20 and changing the variable names accordingly, we generate Figure 2.14, which indicates that Equation (2.259) corresponds to a lowpass system whose bandwidth decreases with N .

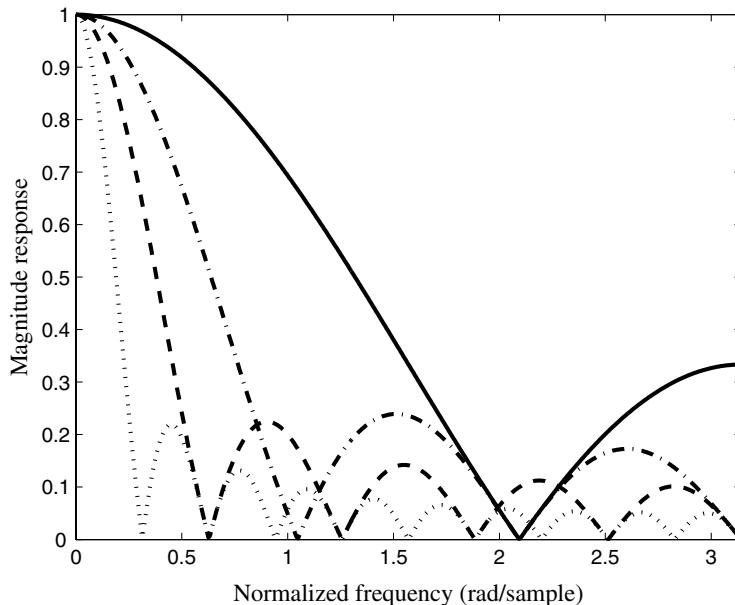


Fig. 2.14.

Magnitude responses of linear system defined in Equation (2.258) for $N = 3$ (solid line), $N = 6$ (dash-dotted line), $N = 10$ (dashed line), and $N = 20$ (dotted line).

In Experiment 1.3, two sinusoidal components of frequencies $f_1 = 1$ Hz and $f_2 = 50$ Hz were digitally processed by Equation (2.258) with $F_s = 1000$ samples/second. In the normalized-frequency scale employed in Figure 2.14, these components correspond to $\omega_1 = (2\pi f_1)/F_s \approx 0.006$ and $\omega_2 = (2\pi f_2)/F_s \approx 0.314$ rad/sample. Figure 2.14 explains how the time-averaging process of Equation (2.258) is able to reduce the amount of noise as N increases. However, as illustrated in Figure 1.26, even the sinusoidal components are affected significantly if N becomes too large. This motivates us to search for better ways to process $x(n)$ in order to reduce noise without affecting the original signal components.

Determining the causal form of Equation (2.259), we get that

$$H(z) = \frac{z^{N-1} + z^{N-2} + \dots + 1}{Nz^{N-1}}. \quad (2.260)$$

Clearly, the associated system presents $(N - 1)$ poles at the origin of the complex z plane and $(N - 1)$ zeros equally spread (except at $z = 1$) around the unit circle, as indicated in Figure 2.15 for $N = 10$, obtained with the single command line

```
zplane(num10, den10);
```

where `num10` and `den10` are line vectors as specified above. The numerical values of these zeros and poles can be determined by the `roots` command, which, as the name indicates, calculates the roots of a given polynomial, or by using the auxiliary commands `tfr2zp` and `zp2tf`, which decompose the numerator and denominator polynomials of a given transfer function into first-order factors and vice versa.

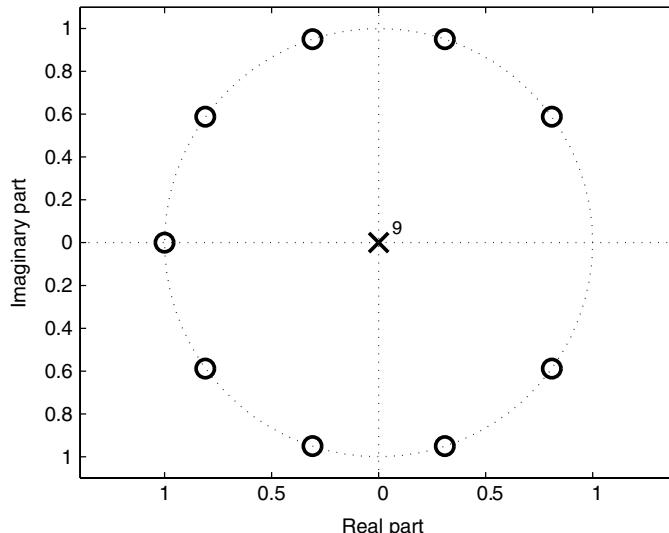


Fig. 2.15. Zero (circles) and pole (cross) constellation of transfer function given in Equation (2.260) with $N = 10$.

Experiment 2.3

The geometric computation of the magnitude and phase of a transfer function, as illustrated in Figure 2.8 and Equations (2.197)–(2.201), can be used to perform intuitive designs of digital filters. The functions `zp2tf`, which generates a transfer function given the positions of its poles and zeros, and `freqz`, which generates the magnitude and phase responses of a given transfer function (see Section 2.13), are important tools for such designs.

Suppose we want to design a filter that provides a significant magnitude response only for frequencies around $\pi/4$. One way of achieving this is to generate a transfer function that has a pole near the unit circle with phase $\pi/4$ (one should remember that, for transfer functions with real coefficients, a complex pole or zero must be accompanied by its complex conjugate). This is so because, since the denominator tends to be small around this pole, the magnitude response tends to be large. In addition, we can decrease it at the other frequencies by placing zeros at $z = 1$ and $z = -1$, forcing a zero response at the frequencies $\omega = 0$ and $\omega = \pi$ rad/sample. This pole–zero placement is depicted in Figure 2.16a, where $p_1 = 0.9e^{j\pi/4}$. The corresponding magnitude response is shown in Figure 2.16b. It can be seen that the designed filter has indeed the desired magnitude response, with a pronounced peak around $\pi/4$. Note that the closer the magnitude of the pole is to the unit circle, the more pronounced is the peak of the magnitude response. A MATLAB code that generates this example is given below:

```

p1 = 0.9*exp(j*pi/4);
z = [1 -1 ].'; P = [p1 p1'].';
[num,den] = zp2tf(z,P,1);
[h,w] = freqz(num,den);
plot(w,abs(h)/max(abs(h)));

```

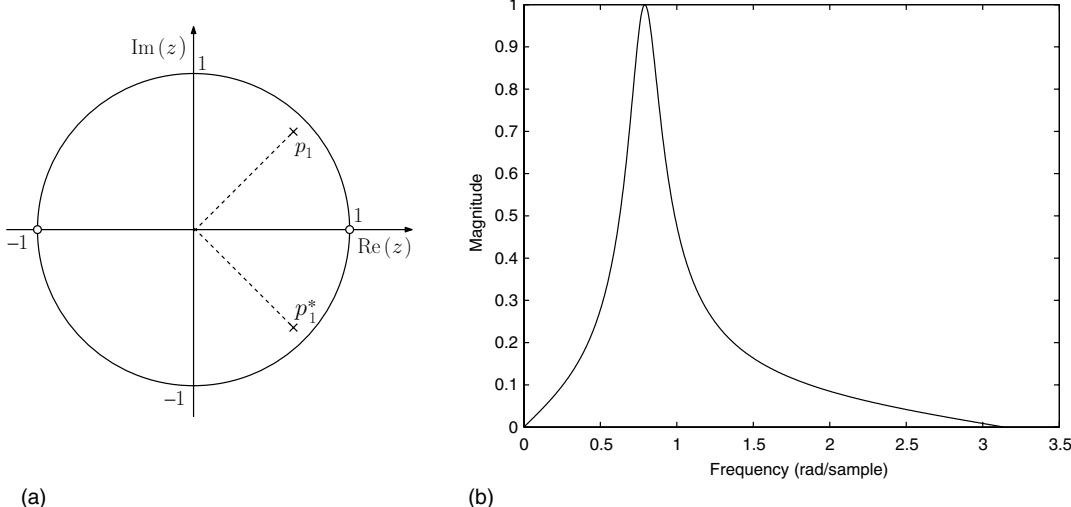


Fig. 2.16. Pole-zero placement for Experiment 2.1 and its corresponding frequency response. $p_1 = 0.9e^{j\pi/4}$.

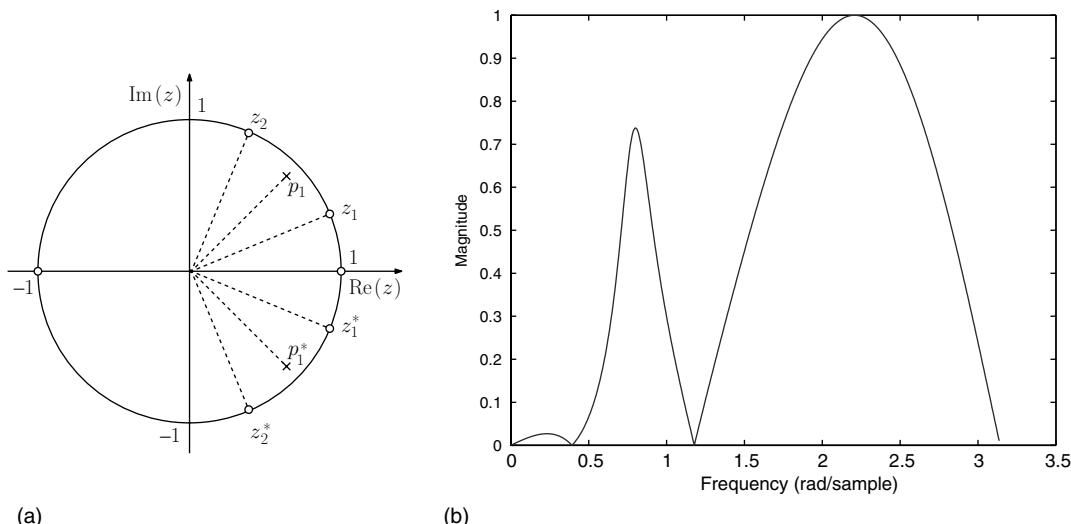


Fig. 2.17. Pole-zero placement for Experiment 2.1 and its corresponding frequency response. $p_1 = 0.9e^{j\pi/4}$, $z_1 = e^{j\pi/8}$, and $z_2 = e^{j3\pi/8}$.

The reader is encouraged to explore the effect of the pole's magnitude on the frequency response.

If we want the filter to be even more selective, then one option is to place zeros around the center frequency. In order to achieve this, we have to insert four extra zeros on the unit circle: one conjugate pair with phases $\pm\pi/8$ and another with phases $\pm3\pi/8$. The effect obtained is depicted in Figure 2.17.

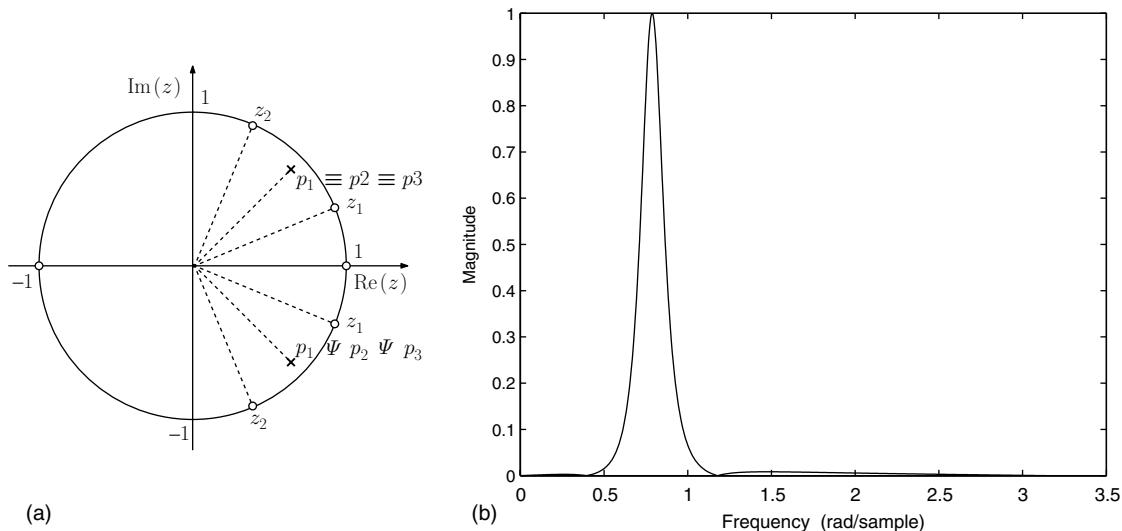


Fig. 2.18. Pole-zero placement for Experiment 2.1 and its corresponding frequency response.
 $p_1 = p_2 = p_3 = 0.9e^{j\pi/4}$, $z_1 = e^{j\pi/8}$, and $z_2 = e^{j3\pi/8}$.

We can see that the inserted zeros have produced, as an undesirable side effect, a large magnitude response around $\omega = 3\pi/2$. This draws our attention to a care that must be taken when designing transfer functions through the placement of poles and zeros. At frequencies far from the locations of the poles and zeros, all the product terms in both the numerator and the denominator of Equation (2.198) tend to be large. Then, when the number of factors in the numerator (zeros) is larger than the number of factors in the denominator (poles), the magnitude of the transfer function also tends to be large. One way to solve this problem is to have as many poles as zeros in the transfer function. In the present case, since we have six zeros (at ± 1 , $e^{\pm j\pi/8}$, and $e^{\pm 3\pi/8}$), we can achieve this by making the poles at $z = 0.9e^{\pm j\pi/4}$ triple. The magnitude response obtained is depicted in Figure 2.18. One can see that the counterbalancing of the six zeros with six poles has the desired effect.

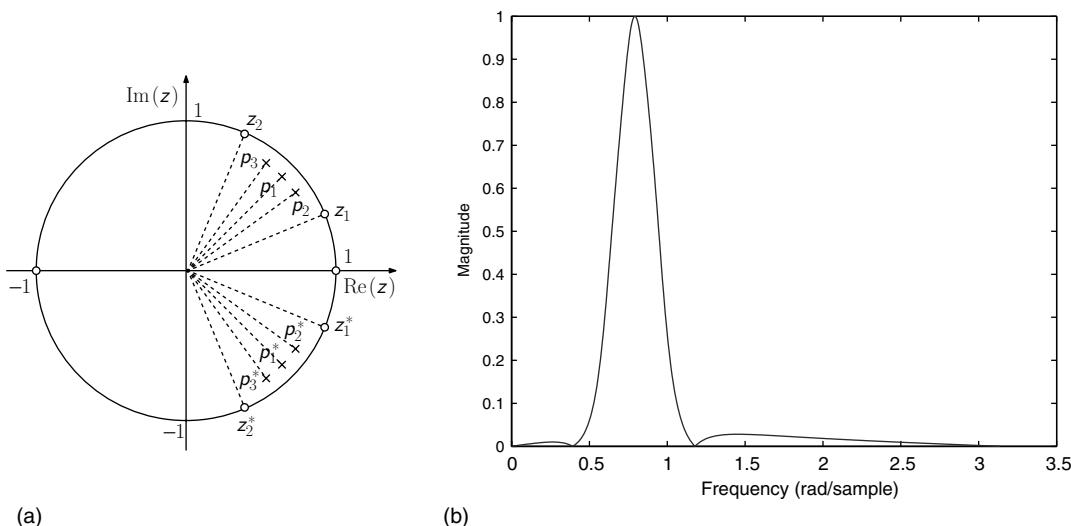
As a further illustration, we now attempt to make the magnitude response in Figure 2.18b less spiky without significantly affecting its bandwidth. In order to do so, we slightly move the two poles p_2 and p_3 away from p_1 . Figure 2.19 shows the effect obtained when p_2 and p_3 are rotated by $-\pi/20$ and $\pi/20$ respectively. One can see that the magnitude response is indeed less spiky.

The MATLAB code to generate Figure 2.19 is shown below:

```

z1 = exp(j*pi/8);
z2 = exp(j*3*pi/8);
p1 = 0.9*exp(j*pi/4);
p2 = 0.9*exp(j*pi/4 - j*pi/20);
p3 = 0.9*exp(j*pi/4 + j*pi/20);
Z = [1 -1 z1 z1' z2 z2'].';
P = [p1 p1' p2 p2' p3 p3'].';

```



(a)

(b)

Fig. 2.19. Pole-zero placement for Experiment 2.1 and its corresponding frequency response. $p_1 = 0.9e^{j\pi/4}$, $p_2 = p_1 e^{-j\pi/20}$, $p_3 = p_1 e^{j\pi/20}$, $z_1 = e^{j\pi/8}$, and $z_2 = e^{j3\pi/8}$.

```
[num,den] = zp2tf(Z,P,1);
[h,w] = freqz(num,den);
plot(w,abs(h)/max(abs(h)));
```

2.13 The z and Fourier transforms with MATLAB

The Signal Processing Toolbox of MATLAB has several functions that are related to transfer functions. In this section, we briefly describe some of them.

- **abs:** Finds the absolute value of the elements in a matrix of complex numbers.

Input parameter: the matrix of complex numbers x .

Output parameter: the matrix y with the magnitudes of the elements of x .

Example:

```
x=[0.3+0.4i -0.5+i 0.33+0.47i]; abs(x);
```

- **angle:** Computes the phase of each element of a matrix of complex numbers.

Input parameter: the matrix of complex numbers x .

Output parameter: the matrix y with the phases of the elements of x in radians.

Example:

```
x=[0.3+0.4i -0.5+i 0.33+0.47i]; angle(x);
```

- **freqz:** Gives the frequency response of a linear discrete-time system specified by its transfer function.

Input parameters (refer to Equation (2.152)):

- a vector *b* containing the coefficients of the numerator polynomial, b_l , for $l = 0, 1, \dots, M$;
- a vector *a* containing the coefficients of the denominator polynomial, a_i , for $i = 0, 1, \dots, N$;
- the number *n* of points around the upper half of the unit circle, or, alternatively, the vector *w* of points in which the response is evaluated.

Output parameters:

- a vector *h* containing the frequency response;
- a vector *w* containing the frequency points at which the frequency response is evaluated;
- with no arguments, *freqz* plots the magnitude and unwrapped phase of the transfer function.

Example:

```
a=[1 0.8 0.64]; b=[1 1]; [h,w]=freqz(b,a,200);
plot(w,abs(h)); figure(2); plot(w,angle(h));
```

- *freqspace*: Generates a vector with equally spaced points on the unit circle.

Input parameters:

- the number *n* of points on the upper half of the unit circle;
- if 'whole' is given as a parameter, then the *n* points are equally spaced on the whole unit circle.

Output parameter: the vector of frequency points in hertz.

Example:

```
a=[1 0.8 0.64]; b=[1 1]; w=pi*freqspace(200);
freqz(b,a,w);
```

- *unwrap*: Unwraps the phase so that phase jumps are always smaller than 2π .

Input parameter: the vector *p* containing the phase.

Output parameter: the vector *q* containing the unwrapped phase.

Example:

```
p1=[0:0.01:6*pi]; p=[p1 p1]; plot(p);
q=unwrap(p); figure(2); plot(q);
```

- *grpdelay*: Gives the group delay of a linear discrete-time system specified by its transfer function.

Input parameters: same as for the *freqz* command.

Output parameters:

- a vector *gd* containing the frequency response;
- a vector *w* containing the frequency points at which the frequency response is evaluated;
- with no arguments, *grpdelay* plots the group delay of the transfer function versus the normalized frequency ($\pi \rightarrow 1$).

Example:

```
a=[1 0.8 0.64]; b=[1 1]; [gd,w]=grpdelay(b,a,200);
plot(w,gd);
```

- `tf2zp`: Given a transfer function, returns the zero and pole locations. See also Section 3.9.
- `zp2tf`: Given the zero and pole locations, returns the transfer function coefficients, thus reversing the operation of the `tf2zp` command. See also Section 3.9.
- `zplane`: Given a transfer function, plots the zero and pole locations.

Input parameters:

- a row vector `b` containing the coefficients of the numerator polynomial in decreasing powers of z ;
- a row vector `a` containing the coefficients of the denominator polynomial in decreasing powers of z ;
- if we input column vectors instead, they are interpreted as the vectors containing the zeros and poles.

Example 1:

```
a=[1 0.8 0.64 0.3 0.02]; b=[1 1 0.3]; zplane(b,a);
```

Example 2:

```
a=[1 0.8 0.64 0.3 0.02]; b=[1 1 0.3];
[z p k]=tf2zp(b,a); zplane(z,p);
```

- `residuez`: Computes the partial-fraction expansion of a transfer function. See also Section 3.9.
- See also the commands `conv` and `impz` described in Section 1.9 and the command `filter` described in Section 4.9.

2.14 Summary

In this chapter we have studied the z and Fourier transforms, two very important tools in the analysis and design of linear discrete-time systems. We first defined the z transform and its inverse, and then we presented some interesting z -transform properties. Next, we introduced the concept of the transfer function as a way of characterizing linear discrete-time systems using the z transform. Then a criterion for determining the stability of discrete-time linear systems was presented. Linked with the concept of transfer functions, we defined the frequency response of a system and the Fourier transform of discrete-time signals, presenting some of its properties. We closed the chapter by describing some MATLAB experiments and functions related to z transforms, Fourier transforms, and discrete-time transfer functions.

2.15 Exercises

2.1 Compute the z transform of the following sequences, indicating their regions of convergence:

- $x(n) = \sin(\omega n + \theta)u(n)$
- $x(n) = \cos(\omega n)u(n)$

- (c) $x(n) = \begin{cases} n, & 0 \leq n \leq 4 \\ 0, & n < 0 \text{ and } n > 4 \end{cases}$
- (d) $x(n) = a^n u(-n)$
- (e) $x(n) = e^{-\alpha n} u(n)$
- (f) $x(n) = e^{-\alpha n} \sin(\omega n) u(n)$
- (g) $x(n) = n^2 u(n).$

2.2 Prove Equation (2.35).

2.3 Suppose that the transfer function of a digital filter is given by

$$\frac{(z - 1)^2}{z^2 + (m_1 - m_2)z + (1 - m_1 - m_2)}.$$

Plot a graph specifying the region of the $m_2 \times m_1$ plane in which the digital filter is stable.

2.4 Compute the impulse response of the system with transfer function

$$H(z) = \frac{z^2}{4z^2 - 2\sqrt{2}z + 1},$$

supposing that the system is stable.

2.5 Compute the time response of the causal system described by the transfer function

$$H(z) = \frac{(z - 1)^2}{z^2 - 0.32z + 0.8}$$

when the input signal is the unit step.

2.6 Determine the inverse z transform of the following functions of the complex variable z , supposing that the systems are stable:

- (a) $\frac{z}{z - 0.8}$
- (b) $\frac{z^2}{z^2 - z + 0.5}$
- (c) $\frac{z^2 + 2z + 1}{z^2 - z + 0.5}$
- (d) $\frac{z^2}{(z - a)(z - 1)}$
- (e) $\frac{1 - z^2}{(2z^2 - 1)(z - 2)}.$

2.7 Compute the inverse z transform of the functions below. Suppose that the sequences are right handed and one sided.

(a) $X(z) = \sin\left(\frac{1}{z}\right)$

(b) $X(z) = \sqrt{\frac{z}{1+z}}.$

- 2.8 An alternative condition for the convergence of a series of the complex variable z , namely

$$S(z) = \sum_{i=0}^{\infty} f_i(z),$$

is based on the function of z

$$\alpha(z) = \lim_{n \rightarrow \infty} \left| \frac{f_{n+1}(z)}{f_n(z)} \right|. \quad (2.261)$$

The series converges for $\alpha(z) < 1$ and diverges for $\alpha(z) > 1$. If $\alpha(z) = 1$ then convergence has to be investigated further.

However, for this condition to be applied, no term $f_i(z)$ can be null for all z . In the cases that any $f_i(z)$ is null, one has to create a new sequence of functions $g_j(z)$ composed only of the terms for which $f_i(z) \neq 0$ and apply the condition above to this new sequence. With this in mind, solve item (b) of Example 2.10 using the convergence condition above.

- 2.9 Given a sequence $x(n)$, form a new sequence consisting of only the even samples of $x(n)$; that is, $y(n) = x(2n)$. Determine the z transform of $y(n)$ as a function of the z transform of $x(n)$, using the auxiliary sequence $(-1)^n x(n)$.
- 2.10 Determine whether the polynomials below can be the denominator of a causal stable filter:
- (a) $z^5 + 2z^4 + z^3 + 2z^2 + z + 0.5$
 - (b) $z^6 - z^5 + z^4 + 2z^3 + z^2 + z + 0.25$
 - (c) $z^4 + 0.5z^3 - 2z^2 + 1.75z + 0.5$
- 2.11 Given the polynomial $D(z) = z^2 - (2+a-b)z + a + 1$ that represents the denominator of a discrete-time system:
- (a) Determine the range of values of a and b such that the system is stable, using the stability test described in the text.
 - (b) Plot a graph $a \times b$, highlighting the stability region.
- 2.12 For the pole-zero constellation shown in Figure 2.20, determine the stable impulse response and discuss the properties of the solution obtained.
- 2.13 Compute the frequency response of a system with the following impulse response:

$$h(n) = \begin{cases} (-1)^n, & |n| < N - 1 \\ 0, & \text{otherwise} \end{cases}.$$

- 2.14 Compute and plot the magnitude and phase of the frequency response of the systems described by the following difference equations:
- (a) $y(n) = x(n) + 2x(n-1) + 3x(n-2) + 2x(n-3) + x(n-4)$
 - (b) $y(n) = y(n-1) + x(n)$
 - (c) $y(n) = x(n) + 3x(n-1) + 2x(n-2)$.

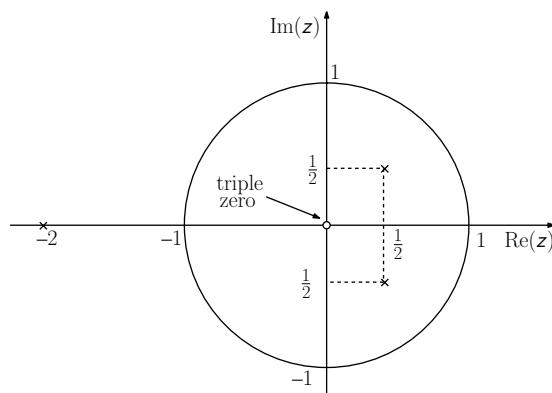


Fig. 2.20. Pole-zero constellation for Exercise 2.12.

- 2.15 Plot the magnitude and phase of the frequency response of the digital filters characterized by the following transfer functions:
- $H(z) = z^{-4} + 2z^{-3} + 2z^{-1} + 1$
 - $H(z) = \frac{z^2 - 1}{z^2 - 1.2z + 0.95}$.
- 2.16 If a digital filter has transfer function $H(z)$, compute the steady-state response of this system for an input of the type $x(n) = \sin(\omega n)u(n)$.
- 2.17 A given piece of hardware can generate filters with the following generic transfer function:

$$H(z) = \frac{\delta_0 + \delta_1 z^{-1} - \delta_2 z^{-1} f(z)}{1 - (1 + m_1)z^{-1} - m_2 z^{-1} f(z)},$$

- where $f(z) = 1/(1 - z^{-1})$. Design the filter so that it has unit gain at DC and two zeros at $\omega = \pi$.
- 2.18 Given a linear time-invariant system, prove the properties below:
- A constant group delay is a necessary but not sufficient condition for the delay introduced by the system to a sinusoid to be independent of its frequency.
 - Let $y_1(n)$ and $y_2(n)$ be the outputs of the system to two sinusoids $x_1(n)$ and $x_2(n)$ respectively. A constant group delay τ implies that if $x_1(n_0) = x_2(n_0)$, then $y_1(n_0 - \tau) = y_2(n_0 - \tau)$.
- 2.19 Suppose we have a system with transfer function having two zeros at $z = 0$, double poles at $z = a$, and a single pole at $z = -b$, where $a > 1$ and $0 < b < 1$ are real numbers. Compute the impulse response for a stable solution assuming that its DC gain is unity.
- 2.20 If the signal $x(n) = 4 \cos[(\pi/4)n - (\pi/6)]u(n)$ is input to the linear system of Exercise 2.6e, determine its steady-state response.
- 2.21 Compute the Fourier transform of each of the sequences in Exercise 2.1.

2.22 Compute the inverse Fourier transform of

$$X(e^{j\omega}) = \frac{1}{1 - e^{-j\omega}}$$

in Example 2.19.

Hint: Replace $e^{j\omega}$ by z , compute the inverse z transform using as closed contour C the one defined by $z = \rho e^{j\omega}$, $\rho > 1$, and take the limit as $\rho \rightarrow 1$.

2.23 Compute $h(n)$, the inverse Fourier transform of

$$H(e^{j\omega}) = \begin{cases} -j, & \text{for } 0 \leq \omega < \pi \\ j, & \text{for } -\pi \leq \omega < 0 \end{cases}$$

and show that:

- (a) Equation (2.211) does not hold.
- (b) The sequence $h(n)$ does not have a z transform.

2.24 Prove that the Fourier transform of $x(n) = e^{j\omega_0 n}$ is given by Equation (2.216) by computing

$$X(e^{j\omega}) = \lim_{N \rightarrow \infty} \sum_{n=-N}^N x(n)e^{-j\omega n}.$$

Hint: A function $f(t)$ is an impulse if $f(t) = 0$, for $t \neq 0$ and $\int_{-\infty}^{\infty} f(t) dt = 1$.

- 2.25 Prove the properties of the Fourier transform of real sequences given by Equations (2.229) to (2.232).
- 2.26 State and prove the properties of the Fourier transform of imaginary sequences that correspond to those given by Equations (2.229)–(2.232).
- 2.27 Show that the direct–inverse Fourier transform pair of the correlation of two sequences is

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2(n+l) \longleftrightarrow X_1(e^{-j\omega})X_2(e^{j\omega}).$$

- 2.28 Show that the Fourier transform of an imaginary and odd sequence is real and odd.
- 2.29 Show that the Fourier transform of a conjugate antisymmetric sequence is imaginary.
- 2.30 We define the even and odd parts of a complex sequence $x(n)$ as

$$\mathcal{E}\{x(n)\} = \frac{x(n) + x^*(-n)}{2} \quad \text{and} \quad \mathcal{O}\{x(n)\} = \frac{x(n) - x^*(-n)}{2}$$

respectively. Show that

$$\mathcal{F}\{\mathcal{E}\{x(n)\}\} = \operatorname{Re}\{X(e^{j\omega})\} \quad \text{and} \quad \mathcal{F}\{\mathcal{O}\{x(n)\}\} = j \operatorname{Im}\{X(e^{j\omega})\}$$

where $X(e^{j\omega}) = \mathcal{F}\{x(n)\}$.

- 2.31 Solve Exercise 1.22 using the concept of the transfer function.

2.32 Prove that

$$\mathcal{F}^{-1} \left\{ \sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right) \right\} = \frac{N}{2\pi} \sum_{p=-\infty}^{\infty} \delta(n - Np) \quad (2.262)$$

by computing its left-hand side with the inverse Fourier transform in Equation (2.207) and verify that the resulting sequence is equal to its right-hand side.

2.33 Show that the PSD function of a WSS random process $\{X\}$ satisfies the following properties:

(a) $\Gamma_X(0) = \sum_{v=-\infty}^{\infty} R_X(v).$

(b) It is an even function; that is: $\Gamma_X(e^{j\omega}) = \Gamma_X(e^{-j\omega})$, for all ω .

(c) It is a nonnegative function; that is: $\Gamma_X(e^{j\omega}) \geq 0$, for all ω .

3.1 Introduction

In Chapter 2 we saw that discrete-time signals and systems can be characterized in the frequency domain by their Fourier transforms. Also, as seen in Chapter 2, one of the main advantages of discrete-time signals is that they can be processed and represented in digital computers. However, when we examine the definition of the Fourier transform in Equation (2.207), namely

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \quad (3.1)$$

we notice that such a characterization in the frequency domain depends on the continuous variable ω . This implies that the Fourier transform, as it is, is not suitable for the processing of discrete-time signals in digital computers. We need a transform depending on a discrete-frequency variable that, if possible, preserves the idea and information expressed by Equation (3.1). This can be obtained from the Fourier transform itself in a very simple way, by sampling uniformly the continuous-frequency variable ω . With this, we obtain a mapping of a signal depending on a discrete-time variable n to a transform depending on a discrete-frequency variable k . Such a mapping is referred to as the discrete Fourier transform (DFT).

In the main part of this chapter, we will study the DFT. First, the expression for the direct and inverse DFTs will be derived. Then, the limitations of the DFT in the representation of generic discrete-time signals will be analyzed. Also, a useful matrix form of the DFT will be presented.

Next, the properties of the DFT will be analyzed, with special emphasis given to the convolution property, which allows the computation of a discrete convolution in the frequency domain. The overlap-and-add and overlap-and-save methods of performing convolution of long signals in the frequency domain will be presented.

A major drawback to the practical use of the DFT is the large number of arithmetic operations involved in its computation, especially for long sequences. This problem has been partially solved with the introduction of efficient algorithms for DFT computation, generally known as fast Fourier transforms (FFTs). The first FFT algorithms were proposed by Cooley and Tukey (1965). Since then, the DFT has been widely used in signal processing applications. In this chapter, some of the most commonly used FFT algorithms are studied.

After the FFT algorithms, discrete transforms other than the DFT will be dealt with. Among others, the discrete cosine transform and the Hartley transform will be analyzed.

Then a summary of discrete signal representations will be given, highlighting the relations between the continuous-time and discrete-time Fourier transforms, the Laplace transform, the z transform, the Fourier series, and the DFT. A Do-it-yourself section has MATLAB experiments on digital filtering and signal analysis using DFTs.

Finally, a brief description of MATLAB commands which are useful for computing fast transforms and performing fast convolutions is given.

3.2 Discrete Fourier transform

The Fourier transform of a sequence $x(n)$ is given by Equation (3.1). Since $X(e^{j\omega})$ is periodic with period 2π , it is convenient to sample it with a sampling frequency equal to an integer multiple of its period; that is, taking N uniformly spaced samples between 0 and 2π . Let us use the frequencies $\omega_k = (2\pi/N)k$, $k \in \mathbb{Z}$. This sampling process is equivalent to generating a Fourier transform $X'(e^{j\omega})$ such that

$$X'(e^{j\omega}) = X(e^{j\omega}) \sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right). \quad (3.2)$$

Applying the convolution theorem seen in Chapter 2, Equation (3.2) becomes

$$x'(n) = \mathcal{F}^{-1}\left\{X'(e^{j\omega})\right\} = x(n) * \mathcal{F}^{-1}\left\{\sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right)\right\} \quad (3.3)$$

and since (see Exercise 2.32)

$$\mathcal{F}^{-1}\left\{\sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{N}k\right)\right\} = \frac{N}{2\pi} \sum_{p=-\infty}^{\infty} \delta(n - Np), \quad (3.4)$$

Equation (3.3) becomes

$$x'(n) = x(n) * \frac{N}{2\pi} \sum_{p=-\infty}^{\infty} \delta(n - Np) = \frac{N}{2\pi} \sum_{p=-\infty}^{\infty} x(n - Np). \quad (3.5)$$

Equation (3.5) indicates that, from equally spaced samples of the Fourier transform, we are able to recover a signal $x'(n)$ consisting of a sum of periodic repetitions of the original discrete signal $x(n)$. In this case, the period of the repetitions is equal to the number N of samples of the Fourier transform in one period. It is then easy to see that if the length L of $x(n)$ is larger than N , then $x(n)$ cannot be obtained from $x'(n)$. On the other hand, if $L \leq N$, then $x'(n)$ is a precise periodic repetition of $x(n)$ and, therefore, $x(n)$ can be obtained by

isolating one complete period of N samples of $x'(n)$, as given by

$$x(n) = \frac{2\pi}{N} x'(n), \quad \text{for } 0 \leq n \leq N - 1. \quad (3.6)$$

From the above discussion, we can draw two important conclusions:

- The samples of the Fourier transform can provide an effective discrete representation, in frequency, of a finite-length discrete-time signal.
- This representation is useful only if the number of samples N of the Fourier transform in one period is greater than or equal to the original signal length L .

It is interesting to note that Equation (3.5) is of the same form as Equation (1.170), which gives the Fourier transform of a sampled continuous-time signal. In that case, Equation (1.170) implies that, in order for a continuous-time signal to be recoverable from its samples, aliasing has to be avoided. This can be done by using a sampling frequency greater than two times the bandwidth of the analog signal. Similarly, Equation (3.5) implies that one can recover a digital signal from the samples of its Fourier transform provided that the signal length L is smaller than or equal to the number of samples N taken in one complete period of its Fourier transform.

We can express $x(n)$ directly as a function of the samples of $X(e^{j\omega})$ by manipulating Equation (3.2) in a different manner to that described above. We begin by noting that Equation (3.2) is equivalent to

$$X'(e^{j\omega}) = \sum_{k=-\infty}^{\infty} X\left(e^{j(2\pi/N)k}\right) \delta\left(\omega - \frac{2\pi}{N}k\right). \quad (3.7)$$

By applying the inverse Fourier transform relation in Equation (2.207), we have

$$\begin{aligned} x'(n) &= \frac{1}{2\pi} \int_0^{2\pi} X'(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_0^{2\pi} \sum_{k=-\infty}^{\infty} X\left(e^{j(2\pi/N)k}\right) \delta\left(\omega - \frac{2\pi}{N}k\right) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \sum_{k=0}^{N-1} X\left(e^{j(2\pi/N)k}\right) e^{j(2\pi/N)kn}. \end{aligned} \quad (3.8)$$

Substituting Equation (3.8) in Equation (3.6), $x(n)$ can be expressed as

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(e^{j(2\pi/N)k}\right) e^{j(2\pi/N)kn}, \quad \text{for } 0 \leq n \leq N - 1. \quad (3.9)$$

Equation (3.9) shows how a discrete-time signal can be recovered from its discrete-frequency representation. This relation is known as the inverse discrete Fourier transform (IDFT).

An inverse expression to Equation (3.9), relating the discrete-frequency representation to the discrete-time signal, can be obtained by just rewriting Equation (3.1) for the frequencies

$\omega_k = (2\pi/N)k$, for $k = 0, 1, \dots, (N - 1)$. Since $x(n)$ has finite duration, we assume that its nonzero samples are within the interval $0 \leq n \leq N - 1$, and then

$$X(e^{j(2\pi/N)k}) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn}, \quad \text{for } 0 \leq k \leq N - 1. \quad (3.10)$$

Equation (3.10) is known as the DFT.

It is important to point out that, in Equation (3.10), if $x(n)$ has length $L < N$, it has to be padded with zeros up to length N , to adapt the sequence length when calculating its corresponding DFT. Referring to Figure 3.1, we can see that the amount of zero padding also

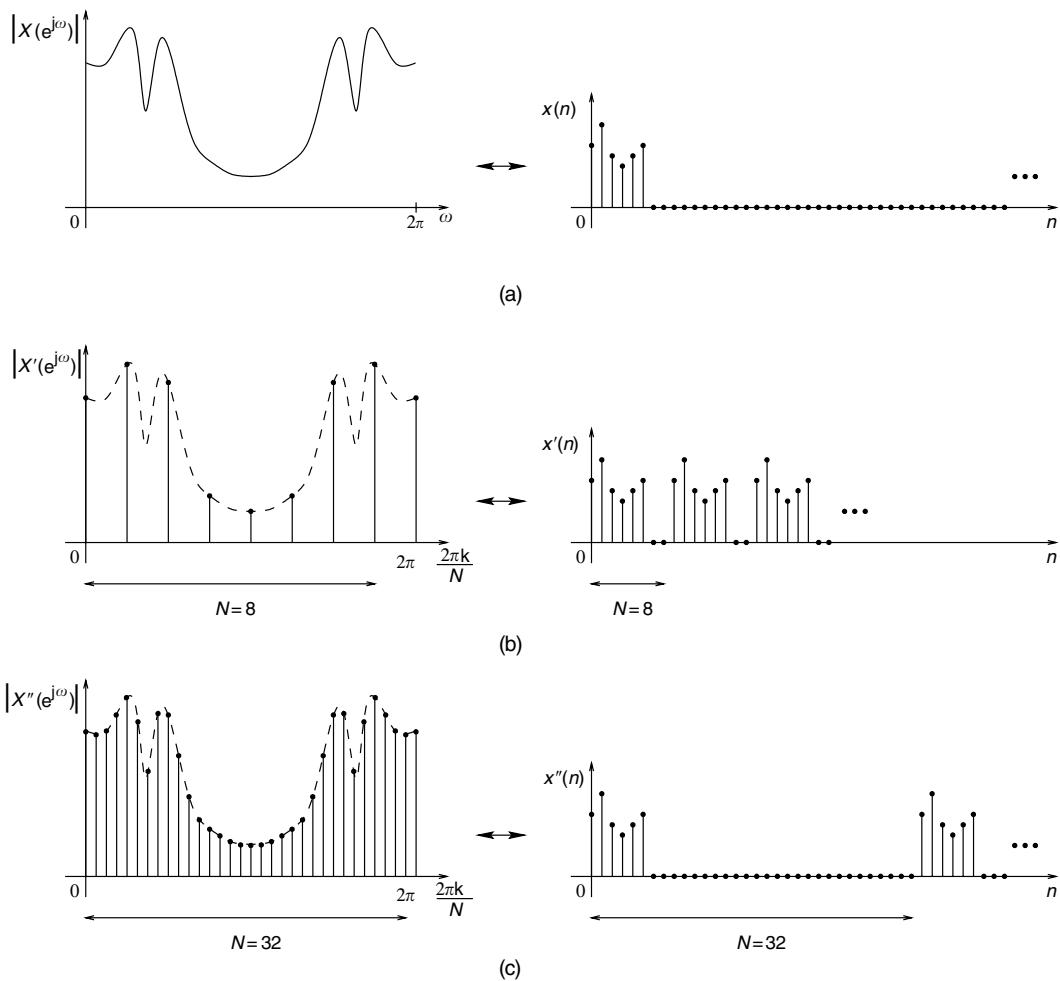


Fig. 3.1.

Equivalence between sampling the Fourier transform of a signal and its DFT: (a) example of Fourier transform of a signal $x(n)$ with only $L = 6$ nonzero samples; (b) DFT with $N = 8$ samples, along with the corresponding zero-padded $x'(n)$; (c) DFT with $N = 32$ samples, along with the corresponding zero-padded $x''(n)$.

determines the frequency resolution of the DFT. Figure 3.1a shows a signal $x(n)$ consisting of $L = 6$ samples along with its Fourier transform, which distinctively presents two pairs of close peaks within $[0, 2\pi]$. In Figure 3.1b, we see that sampling the Fourier transform with eight samples in the interval $[0, 2\pi]$ is equivalent to computing the DFT of $x(n)$ using Equation (3.10) with $N = 8$ samples, which requires $x(n)$ to be padded with $N - L = 2$ zeros. From this figure, we observe that in such a case the two close peaks of the Fourier transform of $x(n)$ could not be resolved by the DFT coefficients. This fact indicates that the resolution of the DFT should be improved by increasing the number of samples N , thus requiring $x(n)$ to be padded with even more zeros. In Figure 3.1c, we can see that the close peaks can be easily identified by the DFT coefficients when N has been increased to 32, corresponding to a zero padding of $N - L = 32 - 6 = 26$ zeros in $x(n)$.

We can summarize the issues in the above discussion as follows:

- The larger the number of zeros padded on $x(n)$ for the calculation of the DFT, the more it resembles its Fourier transform. This happens because of the larger number of samples taken within $[0, 2\pi]$.
- The amount of zero padding used depends on the arithmetic complexity allowed by a particular application, because the larger the amount of zero padding is, the greater the computational and storage requirements involved in the DFT computation are.

There is, however, an important observation that has to be made about the discussion related to Figure 3.1. We see in Figure 3.1b that for $N = 8$ the DFT could not resolve the two close peaks of the Fourier transform. However, since the signal duration is $L = 6$, $x(n)$ can be recovered from the samples of the Fourier transform in Figure 3.1b using Equation (3.9). With $x(n)$, the Fourier transform $X(e^{j\omega})$, as in Figure 3.1a, can be computed using Equation (3.1) and, therefore, the two close peaks could be fully recovered and identified. At this point one could ask why one would need to use a DFT of larger resolution if a DFT of size equal to the signal duration would be enough to fully recover the correct Fourier transform. A justification for the use of a DFT of a larger size than the signal duration is that, for example, for N large enough, one would not need to perform indirect calculations to identify the two close peaks in Figure 3.1, because they would be represented directly by the DFT coefficients, as depicted in Figure 3.1c. In what follows, we derive an expression relating the Fourier transform of a signal $x(n)$ to its DFT.

Equation (3.6), which relates the signal $x(n)$ to the signal $x'(n)$ obtainable from the samples of its Fourier transform, can be rewritten as

$$x(n) = \frac{2\pi}{N} x'(n)(u(n) - u(n - N)). \quad (3.11)$$

Using the fact that a multiplication in the time domain corresponds to a periodic convolution in the frequency domain, discussed in Chapter 2, we have

$$\begin{aligned} X(e^{j\omega}) &= \frac{1}{2\pi} \frac{2\pi}{N} X'(e^{j\omega}) \circledast \mathcal{F}\{u(n) - u(n - N)\} \\ &= \frac{1}{N} X'(e^{j\omega}) \circledast \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-j\omega(N-1)/2} \right]. \end{aligned} \quad (3.12)$$

Substituting Equation (3.7) in Equation (3.12), $X(e^{j\omega})$ becomes

$$\begin{aligned}
 X(e^{j\omega}) &= \frac{1}{N} \left[\sum_{k=-\infty}^{\infty} X(e^{j(2\pi/N)k}) \delta\left(\omega - \frac{2\pi}{N}k\right) \right] \circledast \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-j\omega(N-1)/2} \right] \\
 &= \frac{1}{N} \sum_{k=-\infty}^{\infty} X(e^{j(2\pi/N)k}) \left\{ \frac{\sin \frac{[\omega - (2\pi/N)k]N}{2}}{\sin \frac{\omega - (2\pi/N)k}{2}} e^{-j[\omega - (2\pi/N)k](N-1)/2} \right\} \\
 &= \frac{1}{N} \sum_{k=-\infty}^{\infty} X(e^{j(2\pi/N)k}) \left\{ \frac{\sin[(\omega N/2) - \pi k]}{\sin[(\omega/2) - (\pi k/N)]} e^{-j[(\omega/2) - (\pi k/N)](N-1)} \right\}.
 \end{aligned} \tag{3.13}$$

This equation corresponds to an interpolation formula that gives the Fourier transform of a signal as a function of its DFT. One should note, once again, that such a relationship only works when N is larger than the signal length L .

In order to simplify the notation, it is common practice to use $X(k)$ instead of $X(e^{j(2\pi/N)k})$ and to define

$$W_N = e^{-j2\pi/N}. \tag{3.14}$$

Using this notation, the definitions of the DFT and IDFT, as given in Equations (3.10) and (3.9), become

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad \text{for } 0 \leq k \leq N-1 \tag{3.15}$$

and

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad \text{for } 0 \leq n \leq N-1 \tag{3.16}$$

respectively.

From the development represented by Equations (3.7)–(3.10), it can be seen that Equation (3.16) is the inverse of Equation (3.15). This can be shown in an alternative way by substituting Equation (3.16) into Equation (3.15) directly (see Exercise 3.3).

One should note that if, in the above definitions, $x(n)$ and $X(k)$ are not restricted to being between 0 and $N-1$, then they should be interpreted as being periodic sequences with period N .

From the above, the discrete Fourier transform as expressed in Equation (3.15) can be interpreted in two related ways:

- As a discrete-frequency representation of finite-length signals whereby a length- N signal is mapped into N discrete-frequency coefficients, which correspond to N samples of the Fourier transform of $x(n)$.

- As the Fourier transform of a periodic signal having period N . This periodic signal may correspond to the finite-length signal $x(n)$ repeated periodically, not restricting the index n in Equation (3.16) to the interval $0 \leq n \leq N - 1$.

By referring to Equations (2.245) and (2.246) in Section 2.10, one can see that the DFT and IDFT are actually a Fourier series pair for the periodic signal.

Example 3.1. Compute the DFT of the following sequence:

$$x(n) = \begin{cases} 1, & 0 \leq n \leq 4 \\ -1, & 5 \leq n \leq 9. \end{cases} \quad (3.17)$$

Solution

Before solving this example it is worth stating a simple but important property of W_N . If N is a multiple of k , then

$$W_N^k = e^{-j(2\pi/N)k} = e^{-j2\pi/(N/k)} = W_{N/k}. \quad (3.18)$$

We have that

$$X(k) = \sum_{n=0}^4 W_{10}^{kn} - \sum_{n=5}^9 W_{10}^{kn}. \quad (3.19)$$

Since $0 \leq k \leq 9$, if $k \neq 0$, we have that

$$\begin{aligned} X(k) &= \frac{1 - W_{10}^{5k}}{1 - W_{10}^k} - \frac{W_{10}^{5k} - W_{10}^{10k}}{1 - W_{10}^k} \\ &= \frac{2(1 - W_{10}^{5k})}{1 - W_{10}^k} \\ &= \frac{2(1 - W_2^k)}{1 - W_{10}^k} \\ &= \frac{2[1 - (-1)^k]}{1 - W_{10}^k}. \end{aligned} \quad (3.20)$$

If $k = 0$, the summation in Equation (3.19) becomes

$$X(k) = \sum_{n=0}^4 W_{10}^0 - \sum_{n=5}^9 W_{10}^0 = \sum_{n=0}^4 1 - \sum_{n=5}^9 1 = 0. \quad (3.21)$$

By examining Equations (3.20) and (3.21), we can see that $X(k) = 0$ for k even. Therefore, we have that the DFT can be expressed as

$$X(k) = \begin{cases} 0, & \text{for } k \text{ even} \\ \frac{4}{1 - W_{10}^k}, & \text{for } k \text{ odd.} \end{cases} \quad (3.22)$$

△

Example 3.2. Given the sequence

$$x(n) = \alpha(-a)^n, \quad \text{for } 0 \leq n \leq N-1, \quad (3.23)$$

compute the length- N DFT, supposing that N is even.

Solution

(a) For $a \neq \pm 1$, since N is even:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} \alpha(-a)^n W_N^{kn} \\ &= \alpha \frac{1 - (-a)^N W_N^{Nk}}{1 + aW_N^k} \\ &= \alpha \frac{1 - a^N W_N^{Nk}}{1 + aW_N^k} \\ &= \alpha \frac{1 - a^N}{1 + aW_N^k}. \end{aligned} \quad (3.24)$$

(b) For $a = -1$, then $x(n) = \alpha$ for all n . Then:

$$X(k) = \sum_{n=0}^{N-1} \alpha W_N^{kn}. \quad (3.25)$$

Since $0 \leq k < N$, if $k = 0$, we have that

$$X(k) = \sum_{n=0}^{N-1} \alpha = \alpha N. \quad (3.26)$$

If $k \neq 0$:

$$X(k) = \sum_{n=0}^{N-1} \alpha W_N^{kn} = \alpha \frac{1 - W_N^{Nk}}{1 - W_N^k} = 0. \quad (3.27)$$

From Equations (3.26) and (3.27), we can write

$$X(k) = \alpha N \delta(k). \quad (3.28)$$

(c) For $a = 1$:

$$X(k) = \sum_{n=0}^{N-1} \alpha(-1)^n W_N^{kn} = \sum_{n=0}^{N-1} \alpha(-W_N^k)^n. \quad (3.29)$$

If $k = N/2$, then $W_N^k = -1$, and from the above equation we have

$$X(k) = \sum_{n=0}^{N-1} \alpha = \alpha N. \quad (3.30)$$

If $k \neq N/2$, from Equation (3.29), since N is even, we have

$$X(k) = \alpha \frac{1 - (-W_N^k)^N}{1 + W_N^k} = \alpha \frac{1 - W_N^{kN}}{1 + W_N^k} = 0. \quad (3.31)$$

From Equations (3.30) and (3.31), we can write

$$X(k) = \alpha N \delta\left(k - \frac{N}{2}\right). \quad (3.32)$$

From Equation (3.32), it can be concluded that the signal $x(n)$, for $a = 1$, has spectral contents only at $k = N/2$, which corresponds to $\omega = \pi$; that is, the maximum normalized frequency for a discrete signal. On the other hand, Equation (3.28) says that for $a = -1$ the signal $x(n)$ has spectral contents only for $k = 0$; that is, for $\omega = 0$. This is indeed clear from the fact that if $a = -1$ then $x(n) = \alpha$, which is constant and, therefore, has only the DC component. \triangle

Equations (3.15) and (3.16) can be written in matrix notation as

$$X(k) = \begin{bmatrix} W_N^0 & W_N^k & W_N^{2k} & \dots & W_N^{(N-1)k} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \quad (3.33)$$

and

$$x(n) = \frac{1}{N} \begin{bmatrix} W_N^0 & W_N^{-k} & W_N^{-2k} & \dots & W_N^{-(N-1)k} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} \quad (3.34)$$

respectively, and then

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & \cdots & W_N^{(N-1)} \\ W_N^0 & W_N^2 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \quad (3.35)$$

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} = \frac{1}{N} \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{-1} & \cdots & W_N^{-(N-1)} \\ W_N^0 & W_N^{-2} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{-(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix}. \quad (3.36)$$

By defining

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \quad (3.37)$$

and a matrix \mathbf{W}_N such that

$$\{\mathbf{W}_N\}_{ij} = W_N^{ij}, \quad \text{for } 0 \leq i, j \leq N-1, \quad (3.38)$$

Equations (3.35) and (3.36) can be rewritten more concisely as

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \quad (3.39)$$

and

$$\mathbf{x} = \frac{1}{N} \mathbf{W}_N^* \mathbf{X} \quad (3.40)$$

respectively.

Note that the matrix \mathbf{W}_N enjoys very special properties. Equation (3.38) implies that this matrix is symmetric; that is, $\mathbf{W}_N^T = \mathbf{W}_N$. Also, the direct and inverse DFT relations in Equations (3.39) and (3.40) imply that $\mathbf{W}_N^{-1} = N^{-1} \mathbf{W}_N^*$.

From either Equations (3.15) and (3.16) or Equations (3.35) and (3.36), one can easily conclude that a length- N DFT requires N^2 complex multiplications, including possible trivial multiplications by $W_N^0 = 1$. Since the first row and the first column of the matrices in Equations (3.35) and (3.36) are equal to 1, we have $(2N - 1)$ elements equal to 1. Therefore, if we discount these trivial cases, the total number of multiplications is equal to $(N^2 - 2N + 1)$. Furthermore, the total number of additions is $N(N - 1)$.

3.3 Properties of the DFT

In this section, we describe the main properties of the direct and inverse DFTs, and provide proofs for some of them. Note that since the DFT corresponds to samples of the Fourier transform, its properties closely resemble those of the Fourier transform presented in Section 2.9. However, from N samples of the Fourier transform, one can only recover a signal corresponding to the periodic repetition of the signal $x(n)$ with period N , as given by Equation (3.5). This makes the properties of the DFT slightly different from those of the Fourier transform.

One should bear in mind that the DFT, although it can be interpreted as the Fourier transform of a periodic signal, is just a mapping from a length- N signal into N frequency coefficients and vice versa. However, we often resort to this periodic signal interpretation, since some of the DFT properties follow naturally from it. Note that in this case the periodic signal is the one in Equation (3.5), which is the same one as that obtained by allowing the index n in Equation (3.16) to vary within $(-\infty, \infty)$.

3.3.1 Linearity

The DFT of a linear combination of two sequences is the linear combination of the DFT of the individual sequences; that is, if $x(n) = k_1x_1(n) + k_2x_2(n)$, then

$$X(k) = k_1X_1(k) + k_2X_2(k). \quad (3.41)$$

Note that the two DFTs must have the same length, and thus the two sequences, if necessary, should be zero padded accordingly in order to reach the same length N .

3.3.2 Time reversal

The DFT of $x(-n)$ is such that

$$x(-n) \longleftrightarrow X(-k). \quad (3.42)$$

It must be noted that if both indexes n and k are constrained to be between 0 and $N - 1$, then $-n$ and $-k$ are outside this interval. Therefore, consistency with Equations (3.15) and (3.16) requires that $x(-n) = x(N - n)$ and $X(-k) = X(N - k)$. These relations can also be deduced from the fact that we can interpret both $x(n)$ and $X(k)$ as being periodic with period N .

3.3.3 Time-shift theorem

The DFT of a sequence shifted in time is such that

$$x(n + l) \longleftrightarrow W_N^{-lk}X(k). \quad (3.43)$$

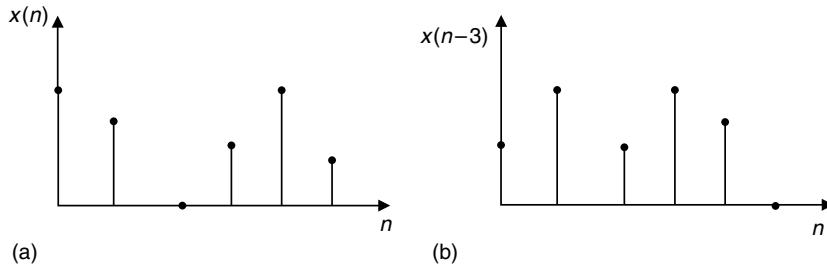


Fig. 3.2. Circular shift of three samples: (a) original signal $x(n)$; (b) resulting signal $x(n - 3)$.

In the definition of the IDFT given in Equation (3.16), if the index n is allowed to vary outside the set $0, 1, \dots, (N - 1)$, then $x(n)$ can be interpreted as being periodic with period N . This interpretation implies that the signal $x(n + l)$ obtained from the IDFT of $W_N^{-lk} X(k)$ corresponds to a circular shift of $x(n)$; that is, provided that $1 \leq l \leq N - 1$, if

$$y(n) \longleftrightarrow W_N^{-lk} X(k), \quad (3.44)$$

then

$$y(n) = \begin{cases} x(n + l), & \text{for } 0 \leq n \leq N - l - 1 \\ x(n + l - N), & \text{for } N - l \leq n \leq N - 1. \end{cases} \quad (3.45)$$

This result indicates that $y(n)$ is a sequence whose last l samples are equal to the first l samples of $x(n)$. An example of circular shift is illustrated in Figure 3.2. A formal proof of this property is provided below for the case $1 \leq l \leq N - 1$.

Proof

$$\begin{aligned} X'(k) &= \sum_{n=0}^{N-1} x(n + l) W_N^{nk} \\ &= W_N^{-lk} \sum_{n=0}^{N-1} x(n + l) W_N^{(n+l)k} \\ &= W_N^{-lk} \sum_{m=l}^{N+l-1} x(m) W_N^{mk} \\ &= W_N^{-lk} \left(\sum_{m=l}^{N-1} x(m) W_N^{mk} + \sum_{m=N}^{N+l-1} x(m) W_N^{mk} \right). \end{aligned} \quad (3.46)$$

Since W_N^k has period N and $x(n+l)$ is obtained from a circular shift of $x(n)$, then the summation from N to $(N+l-1)$ is equivalent to a summation from 0 to $(l-1)$. Therefore:

$$\begin{aligned} X'(k) &= W_N^{-lk} \left(\sum_{m=l}^{N-1} x(m) W_N^{mk} + \sum_{m=0}^{l-1} x(m) W_N^{mk} \right) \\ &= W_N^{-lk} \sum_{m=0}^{N-1} x(m) W_N^{mk} \\ &= W_N^{-lk} X(k). \end{aligned} \quad (3.47)$$

It is important to note that since both $x(n)$ and W_N^{kn} are periodic with period N , then the property is still valid for both $l < 0$ and $l \geq N$. \square

A shorthand notation for Equation (3.45) is

$$y(n) = x((n+l) \bmod N), \quad (3.48)$$

where $(n \bmod N)$ represents the remainder of the division of n by N and is always in between 0 and $(N-1)$.

Example 3.3. The DFT of a length-6 sequence $x(n)$ is

$$X(k) = \begin{cases} 4, & k = 0 \\ 2, & 1 \leq k \leq 5. \end{cases} \quad (3.49)$$

- (a) Compute $x(n)$.
- (b) Determine the length-6 sequence $y(n)$ whose DFT is $Y(k) = W_6^{-2k} X(k)$.

Solution

(a)

$$x(n) = \frac{1}{6} \sum_{k=0}^5 X(k) W_6^{-kn} = \frac{1}{6} \left(4W_6^0 + \sum_{k=1}^5 2W_6^{-kn} \right). \quad (3.50)$$

If $n = 0$, we have that

$$x(0) = \frac{1}{6} \left(4W_6^0 + \sum_{k=1}^5 2W_6^0 \right) = \frac{7}{3}. \quad (3.51)$$

For $1 \leq n \leq 5$:

$$x(n) = \frac{1}{6} \left(4 + \frac{2W_6^{-n} - 2W_6^{-6n}}{1 - W_6^{-n}} \right) = \frac{1}{6} \left(4 + 2 \frac{W_6^{-n} - 1}{1 - W_6^{-n}} \right) = \frac{1}{3}. \quad (3.52)$$

In shorthand notation:

$$x(n) = \frac{1}{3} + 2\delta(n), \quad \text{for } 0 \leq n \leq 5. \quad (3.53)$$

We can express the above equations using the matrix notation in Equation (3.37) as

$$\mathbf{x} = \left[\frac{7}{3} \quad \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \right]^T. \quad (3.54)$$

(b) Using the time-shift theorem, we have that if $Y(k) = W_6^{-2k}X(k)$, then

$$y(n) = x((n+2) \bmod 6) = \frac{1}{3} + 2\delta((n+2) \bmod 6) \quad (3.55)$$

Since $((n+2) \bmod 6) = 0$ for $n = 4$, then we can express $y(n)$ as

$$y(n) = \frac{1}{3} + 2\delta(n-4), \quad 0 \leq n \leq 5, \quad (3.56)$$

which in matrix notation is

$$\mathbf{y} = \left[\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \quad \frac{7}{3} \quad \frac{1}{3} \right]^T. \quad (3.57)$$

△

3.3.4 Circular frequency-shift theorem (modulation theorem)

$$W_N^{ln}x(n) \longleftrightarrow X(k+l). \quad (3.58)$$

The proof is analogous to that of the time-shift theorem and is left as an exercise for the reader.

Noting that

$$W_N^{ln} = \cos\left(\frac{2\pi}{N}ln\right) - j\sin\left(\frac{2\pi}{N}ln\right), \quad (3.59)$$

Equation (3.58) also implies the following properties:

$$x(n)\sin\left(\frac{2\pi}{N}ln\right) \longleftrightarrow \frac{1}{2j}(X(k-l) - X(k+l)) \quad (3.60)$$

$$x(n)\cos\left(\frac{2\pi}{N}ln\right) \longleftrightarrow \frac{1}{2}(X(k-l) + X(k+l)). \quad (3.61)$$

3.3.5 Circular convolution in time

If $x(n)$ and $h(n)$ are periodic with period N , then

$$\sum_{l=0}^{N-1} x(l)h(n-l) = \sum_{l=0}^{N-1} x(n-l)h(l) \longleftrightarrow X(k)H(k), \quad (3.62)$$

where $X(k)$ and $H(k)$ are the DFTs of the length- N signals corresponding to one period of $x(n)$ and $h(n)$ respectively.

Proof If $Y(k) = X(k)H(k)$, then

$$\begin{aligned} y(n) &= \frac{1}{N} \sum_{k=0}^{N-1} H(k)X(k)W_N^{-kn} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} H(k) \left(\sum_{l=0}^{N-1} x(l)W_N^{kl} \right) W_N^{-kn} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} H(k)x(l)W_N^{(l-n)k} \\ &= \sum_{l=0}^{N-1} x(l) \frac{1}{N} \sum_{k=0}^{N-1} H(k)W_N^{-(n-l)k} \\ &= \sum_{l=0}^{N-1} x(l)h(n-l). \end{aligned} \quad (3.63)$$

□

This result is the basis of one of the most important applications of the DFT, which is the ability to compute a discrete convolution in time through the application of the IDFT to the product of the DFTs of the two sequences. However, one should bear in mind that when computing the DFT all the sequence shifts involved are circular, as depicted in Figure 3.2. Therefore, we say that the product of two DFTs actually corresponds to a circular convolution in the time domain. In fact, the circular convolution is equivalent to one period of the convolution between one original sequence and the periodic version of the other. It is important to note that, as seen in Chapter 1, linear convolutions are usually the ones of interest in practice. In the next section we will discuss how linear convolutions can be implemented through circular convolutions and, therefore, through the computation of DFTs.

Since $x(n)$ and $h(n)$ in Equation (3.62) have period N , then their circular convolution $y(n)$ also has period N and needs to be specified only for $0 \leq n \leq N - 1$. Therefore, the circular convolution in Equation (3.62) can be expressed as a function of only those samples

of $x(n)$ and $h(n)$ between 0 and $N - 1$ as

$$\begin{aligned} y(n) &= \sum_{l=0}^{N-1} x(l)h(n-l) \\ &= \sum_{l=0}^n x(l)h(n-l) + \sum_{l=n+1}^{N-1} x(l)h(n-l+N), \quad \text{for } 0 \leq n \leq N-1, \end{aligned} \quad (3.64)$$

which can be rewritten in a compact form as

$$y(n) = \sum_{l=0}^{N-1} x(l)h((n-l) \bmod N) = x(n) \circledast h(n), \quad (3.65)$$

where $(l \bmod N)$ represents the remainder of the integer division of l by N .

Equation (3.65) can be expressed as $y(n) = \mathbf{h}^T \mathbf{x}$ with

$$\mathbf{h} = \begin{bmatrix} h(n \bmod N) \\ h((n-1) \bmod N) \\ h((n-2) \bmod N) \\ \vdots \\ h((n-N+1) \bmod N) \end{bmatrix} \quad (3.66)$$

and \mathbf{x} as before. Therefore, the circular convolution can be put in matrix form as

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(N-1) & h(N-2) & \cdots & h(1) \\ h(1) & h(0) & h(N-1) & \cdots & h(2) \\ h(2) & h(1) & h(0) & \cdots & h(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h(N-1) & h(N-2) & h(N-3) & \cdots & h(0) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix}. \quad (3.67)$$

Note that each row of the matrix in Equation (3.67) is a circular right-shift of the previous row.

In the remainder of this chapter, unless stated otherwise, it will be assumed that all the sequences are periodic and all the convolutions are circular.

3.3.6 Correlation

The DFT of the correlation in time between two sequences is such that

$$\sum_{n=0}^{N-1} h(n)x(l+n) \longleftrightarrow H(-k)X(k). \quad (3.68)$$

This result is a direct consequence of the convolution and the time-reversal properties. Its proof is left as an exercise for the reader.

3.3.7 Complex conjugation

$$x^*(n) \longleftrightarrow X^*(-k). \quad (3.69)$$

Proof

$$\sum_{n=0}^{N-1} x^*(n) W_N^{kn} = \left(\sum_{n=0}^{N-1} x(n) W_N^{-kn} \right)^* = X^*(-k). \quad (3.70)$$

□

3.3.8 Real and imaginary sequences

If $x(n)$ is a real sequence, then $X(k) = X^*(-k)$; that is:

$$\left. \begin{array}{l} \operatorname{Re}\{X(k)\} = \operatorname{Re}\{X(-k)\} \\ \operatorname{Im}\{X(k)\} = -\operatorname{Im}\{X(-k)\} \end{array} \right\}. \quad (3.71)$$

The proof can be easily obtained from the definition of the DFT, by using the expression for W_N^{kn} in Equation (3.59).

When $x(n)$ is imaginary, then $X(k) = -X^*(-k)$; that is:

$$\left. \begin{array}{l} \operatorname{Re}\{X(k)\} = -\operatorname{Re}\{X(-k)\} \\ \operatorname{Im}\{X(k)\} = \operatorname{Im}\{X(-k)\} \end{array} \right\}. \quad (3.72)$$

Example 3.4. Show how to compute the DFTs of two real sequences through the computation of only one DFT.

Solution

With the two real sequences $x_1(n)$ and $x_2(n)$ we form the sequence $y(n) = x_1(n) + jx_2(n)$. From the linearity of the DFT, we have that the DFT of $y(n)$ is

$$Y(k) = X_1(k) + jX_2(k). \quad (3.73)$$

From this equation, we have that

$$\left. \begin{array}{l} \operatorname{Re}\{Y(k)\} = \operatorname{Re}\{X_1(k)\} - \operatorname{Im}\{X_2(k)\} \\ \operatorname{Im}\{Y(k)\} = \operatorname{Re}\{X_2(k)\} + \operatorname{Im}\{X_1(k)\} \end{array} \right\}. \quad (3.74)$$

Using the properties in Equation (3.71) in Equation (3.74), we get

$$\left. \begin{aligned} \operatorname{Re}\{Y(-k)\} &= \operatorname{Re}\{X_1(k)\} + \operatorname{Im}\{X_2(k)\} \\ \operatorname{Im}\{Y(-k)\} &= \operatorname{Re}\{X_2(k)\} - \operatorname{Im}\{X_1(k)\} \end{aligned} \right\}. \quad (3.75)$$

Combining Equations (3.74) and (3.75), the DFTs of $x_1(n)$ and $x_2(n)$ can be computed as

$$\left. \begin{aligned} \operatorname{Re}\{X_1(k)\} &= \frac{1}{2} (\operatorname{Re}\{Y(k)\} + \operatorname{Re}\{Y(-k)\}) \\ \operatorname{Im}\{X_1(k)\} &= \frac{1}{2} (\operatorname{Im}\{Y(k)\} - \operatorname{Im}\{Y(-k)\}) \\ \operatorname{Re}\{X_2(k)\} &= \frac{1}{2} (\operatorname{Im}\{Y(k)\} + \operatorname{Im}\{Y(-k)\}) \\ \operatorname{Im}\{X_2(k)\} &= \frac{1}{2} (\operatorname{Re}\{Y(-k)\} - \operatorname{Re}\{Y(k)\}) \end{aligned} \right\}. \quad (3.76)$$

△

3.3.9 Symmetric and antisymmetric sequences

Symmetric and antisymmetric sequences are of special interest because their DFTs have some interesting properties. In Section 2.9 we gave the properties of the Fourier transform relating to symmetric and antisymmetric sequences. In this chapter, the meanings of symmetry and antisymmetry are slightly different. This happens because, unlike the Fourier and z transforms, which are applied to infinite-duration signals, the DFT is applied to finite-duration signals. In fact, the DFT can be interpreted as the Fourier transform of a periodic signal formed from the infinite repetition of the finite-duration signal. Therefore, before describing the properties of the DFT relating to symmetric and antisymmetric sequences, we give precise definitions for them in the context of the DFT.

- A sequence is called symmetric (or even) if $x(n) = x(-n)$. Since, for indexes outside the set $0, 1, \dots, (N - 1)$, $x(n)$ can be interpreted as periodic with period N (see Equation (3.16)), then $x(-n) = x(N - n)$. Therefore, symmetry is equivalent to $x(n) = x(N - n)$.
- A sequence is antisymmetric (or odd) if $x(n) = -x(-n) = -x(N - n)$.
- A complex sequence is said to be conjugate symmetric if $x(n) = x^*(-n) = x^*(N - n)$.
- A complex sequence is called conjugate antisymmetric if $x(n) = -x^*(-n) = -x^*(N - n)$.

Using such concepts, the following properties hold:

- If $x(n)$ is real and symmetric, $X(k)$ is also real and symmetric.

Proof From Equation (3.15), $X(k)$ is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi}{N}kn\right) - j \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi}{N}kn\right). \quad (3.77)$$

Since $x(n) = x(N - n)$, the imaginary part of the above summation is null, because, for N even, we get that

$$\begin{aligned}
 \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) &= \sum_{n=0}^{(N/2)-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) + \sum_{n=N/2}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
 &= \sum_{n=1}^{(N/2)-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) + \sum_{n=(N/2)+1}^{N-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
 &= \sum_{n=1}^{(N/2)-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) \\
 &\quad + \sum_{m=1}^{(N/2)-1} x(N - m) \sin\left[\frac{2\pi}{N} k(N - m)\right] \\
 &= \sum_{n=1}^{(N/2)-1} x(n) \sin\left(\frac{2\pi}{N} kn\right) - \sum_{m=1}^{(N/2)-1} x(m) \sin\left(\frac{2\pi}{N} km\right) \\
 &= 0.
 \end{aligned} \tag{3.78}$$

Therefore, we have that

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi}{N} kn\right), \tag{3.79}$$

which is real and symmetric (even). The proof for N odd is analogous and is left as an exercise to the reader. \square

- If $x(n)$ is imaginary and even, then $X(k)$ is imaginary and even.
- If $x(n)$ is real and odd, then $X(k)$ is imaginary and odd.
- If $x(n)$ is imaginary and odd, then $X(k)$ is real and odd.
- If $x(n)$ is conjugate symmetric, then $X(k)$ is real.

Proof A conjugate symmetric sequence $x(n)$ can be expressed as

$$x(n) = x_e(n) + jx_o(n), \tag{3.80}$$

where $x_e(n)$ is real and even and $x_o(n)$ is real and odd. Therefore:

$$X(k) = X_e(k) + jX_o(k). \tag{3.81}$$

From the above properties, $X_e(k)$ is real and even and $X_o(k)$ is imaginary and odd. Thus, $X(k) = X_e(k) + jX_o(k)$ is real. \square

- If $x(n)$ is conjugate antisymmetric, then $X(k)$ is imaginary.

The proofs of all the other properties are left as exercises for the interested reader.

Example 3.5. Given the sequence $x(n)$ represented by the vector

$$\mathbf{x} = [1 \ 2 \ 3 \ 4 \ 0 \ 0]^T, \quad (3.82)$$

find the sequence $y(n)$ whose length-6 DFT is given by $Y(k) = \text{Re}\{X(k)\}$.

Solution

$$Y(k) = \frac{X(k) + X^*(k)}{2}. \quad (3.83)$$

From the linearity of the DFT:

$$y(n) = \frac{\text{IDFT}\{X(k)\} + \text{IDFT}\{X^*(k)\}}{2}. \quad (3.84)$$

Since $x(n)$ is real and is a sequence of length 6, we have

$$y(n) = \frac{x(n) + x(-n)}{2} = \frac{x(n) + x(6-n)}{2}. \quad (3.85)$$

Then:

$$\left. \begin{aligned} y(0) &= \frac{x(0) + x(6)}{2} = 1 \\ y(1) &= \frac{x(1) + x(5)}{2} = 1 \\ y(2) &= \frac{x(2) + x(4)}{2} = \frac{3}{2} \\ y(3) &= \frac{x(3) + x(3)}{2} = 4 \\ y(4) &= \frac{x(4) + x(2)}{2} = \frac{3}{2} \\ y(5) &= \frac{x(5) + x(1)}{2} = 1 \end{aligned} \right\}. \quad (3.86)$$

\triangle

3.3.10 Parseval's theorem

$$\sum_{n=0}^{N-1} x_1(n)x_2^*(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_1(k)X_2^*(k). \quad (3.87)$$

Proof

$$\begin{aligned}
 \sum_{n=0}^{N-1} x_1(n)x_2^*(n) &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X_1(k) W_N^{-kn} \right) x_2^*(n) \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} X_1(k) \sum_{n=0}^{N-1} x_2^*(n) W_N^{-kn} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} X_1(k) X_2^*(k).
 \end{aligned} \tag{3.88}$$

□

If $x_1(n) = x_2(n)$, then we have that

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2. \tag{3.89}$$

Equation (3.89) can be interpreted as an energy conservation property, since the energy in the discrete-time domain is equal to the energy in the discrete-frequency domain, up to a scaling factor N .

3.3.11 Relationship between the DFT and the z transform

In Section 3.2 we defined the DFT as samples of the Fourier transform and showed, in Equation (3.13), how to obtain the Fourier transform directly from the DFT. Since the Fourier transform corresponds to the z transform for $z = e^{j\omega}$, then clearly the DFT can be obtained by sampling the z transform at $\omega = (2\pi/N)k$.

Mathematically, since the z transform $X_z(z)$ of a length- N sequence $x(n)$ is

$$X_z(z) = \sum_{n=0}^{N-1} x(n)z^{-n}, \tag{3.90}$$

if we make $z = e^{j(2\pi/N)k}$, then we have that

$$X_z \left(e^{j(2\pi/N)kn} \right) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn}. \tag{3.91}$$

This equation corresponds to samples of the z transform equally spaced on the unit circle, and it is identical to the definition of the DFT in Equation (3.15).

For $z \neq W_N^k$, in order to obtain the z transform from the DFT coefficients $X(k)$, we substitute Equation (3.16) into Equation (3.90), obtaining

$$\begin{aligned}
X_z(z) &= \sum_{n=0}^{N-1} x(n)z^{-n} \\
&= \sum_{n=0}^{N-1} \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-kn}z^{-n} \\
&= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \sum_{n=0}^{N-1} \left(W_N^{-k}z^{-1}\right)^n \\
&= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \frac{1 - W_N^{-kN}z^{-N}}{1 - W_N^{-k}z^{-1}} \\
&= \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{X(k)}{1 - W_N^{-k}z^{-1}}, \tag{3.92}
\end{aligned}$$

which, similar to Equation (3.13) for the Fourier transform, relates the DFT to the z transform.

3.4 Digital filtering using the DFT

3.4.1 Linear and circular convolutions

A linear time-invariant system implements the linear convolution of the input signal with the impulse response of the system. Since the Fourier transform of the convolution of two sequences is the product of their Fourier transforms, it is natural to consider the computation of time convolutions in the frequency domain. The DFT is the discrete version of the Fourier transform and, therefore, should be the transform used for such computations. However, as is described in Equation (3.5), the sampling process in the frequency domain forces the signal to be periodic in time. In Section 3.3, see Equation (3.62), we saw that this implies that the IDFT of the product of the DFTs of two length- N signals corresponds to the convolution of one sequence with the periodic version of the other. This periodic version is obtained by repeating the length- N signal with period N . As seen before, this convolution between a signal and the periodic version of the other is called the circular convolution between the two length- N signals. This means that, using the DFT, one can in principle compute only circular convolutions, but not the linear convolution necessary to implement a linear system. In this section we describe techniques to circumvent this problem and allow us to implement discrete-time linear systems in the frequency domain.

These techniques are essentially based on a very simple trick. Supposing that the DFTs are of size N , we have that the circular convolution between two sequences $x(n)$ and $h(n)$

is given by Equation (3.64), which is repeated here for the reader's convenience:

$$\begin{aligned} y(n) &= \sum_{l=0}^{N-1} x(l)h(n-l) \\ &= \sum_{l=0}^n x(l)h(n-l) + \sum_{l=n+1}^{N-1} x(l)h(n-l+N), \quad \text{for } 0 \leq n \leq N-1. \end{aligned} \quad (3.93)$$

If we want the circular convolution to be equal to the linear convolution between $x(n)$ and $h(n)$, then the second summation in the above equation must be null; that is:

$$c(n) = \sum_{l=n+1}^{N-1} x(l)h(n-l+N) = 0, \quad \text{for } 0 \leq n \leq N-1. \quad (3.94)$$

Assuming that $x(n)$ has duration L and $h(n)$ has duration K , namely that

$$x(n) = 0, \quad \text{for } n \geq L; \quad h(n) = 0, \quad \text{for } n \geq K, \quad (3.95)$$

we have that the summation $c(n)$ in Equation (3.94) is different from zero only if both $x(l)$ and $h(n-l+N)$ are nonzero; this happens if

$$l \leq L-1 \quad \text{and} \quad n-l+N \leq K-1, \quad (3.96)$$

which implies that

$$n+N-K+1 \leq l \leq L-1. \quad (3.97)$$

If we then want the summation $c(n)$ to be null for $0 \leq n \leq N-1$, then it should be impossible to satisfy Equation (3.97) for $0 \leq n \leq N-1$. This happens when

$$n+N-K+1 > L-1 \quad \text{for } 0 \leq n \leq N-1. \quad (3.98)$$

Since the most strict case of Equation (3.98) is for $n = 0$, we have that the condition for $c(n) = 0, 0 \leq n \leq N-1$ – and, therefore, for the circular convolution to be equivalent to the linear convolution – is

$$N \geq L + K - 1. \quad (3.99)$$

Thus, in order to perform a linear convolution using the IDFT of the product of the DFT of two sequences, we must choose a DFT size N satisfying Equation (3.99). This is equivalent to padding $x(n)$ using at least $K-1$ zeros and padding $h(n)$ using at least $L-1$ zeros. This zero-padding process is illustrated in Figure 3.3 for $L = 4$ and $K = 3$, where, after zero padding, the sequences $x(n)$ and $h(n)$ are denoted as $x_1(n)$ and $h_1(n)$ respectively.

The following example will help to clarify the above discussion.

Example 3.6. Referring to Figure 3.3, compute the linear convolution of the two sequences $x(n)$ and $h(n)$ and compare it with the circular convolution of $x(n)$ and $h(n)$ as well as the circular convolution of $x_1(n)$ and $h_1(n)$.

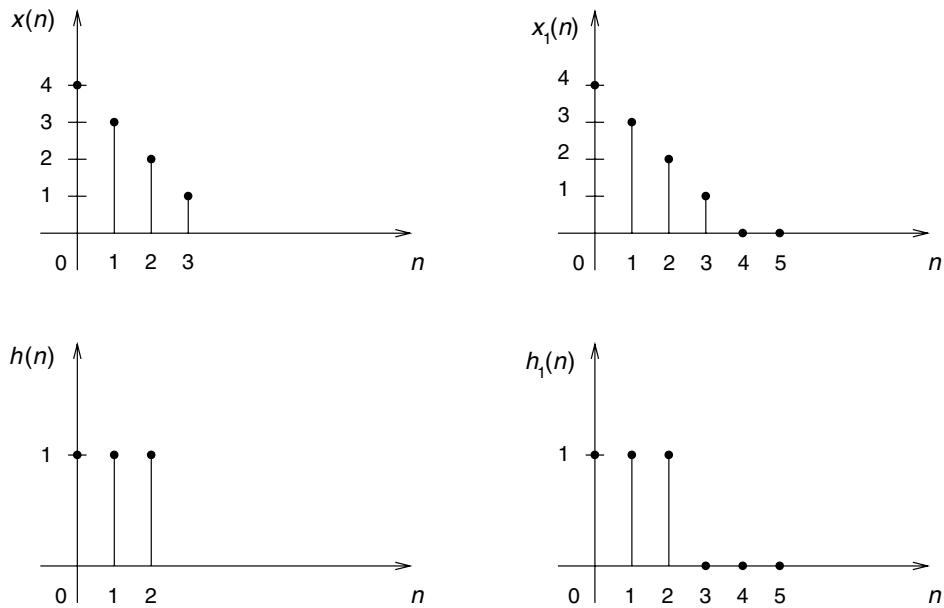


Fig. 3.3. Zero padding of two sequences in order to perform linear convolution using the DFT: \$x_1(n)\$ corresponds to \$x(n)\$, and \$h_1(n)\$ corresponds to \$h(n)\$ after appropriate padding.

Solution

First, we compute the linear convolution of \$x(n)\$ and \$h(n)\$:

$$y_l(n) = x(n) * h(n) = \sum_{l=0}^3 x(l)h(n-l) \quad (3.100)$$

such that

$$\left. \begin{aligned} y_l(0) &= x(0)h(0) = 4 \\ y_l(1) &= x(0)h(1) + x(1)h(0) = 7 \\ y_l(2) &= x(0)h(2) + x(1)h(1) + x(2)h(0) = 9 \\ y_l(3) &= x(1)h(2) + x(2)h(1) + x(3)h(0) = 6 \\ y_l(4) &= x(2)h(2) + x(3)h(1) = 3 \\ y_l(5) &= x(3)h(2) = 1 \end{aligned} \right\}. \quad (3.101)$$

The circular convolution of length \$N = 4\$ between \$x(n)\$ and \$h(n)\$, using Equation (3.65), is equal to

$$y_{c4}(n) = x(n) \circledast h(n) = \sum_{l=0}^3 x(l)h((n-l) \bmod 4) \quad (3.102)$$

such that

$$\left. \begin{aligned} y_{c_4}(0) &= x(0)h(0) + x(1)h(3) + x(2)h(2) + x(3)h(1) = 7 \\ y_{c_4}(1) &= x(0)h(1) + x(1)h(0) + x(2)h(3) + x(3)h(2) = 8 \\ y_{c_4}(2) &= x(0)h(2) + x(1)h(1) + x(2)h(0) + x(3)h(3) = 9 \\ y_{c_4}(3) &= x(0)h(3) + x(1)h(2) + x(2)h(1) + x(3)h(0) = 6 \end{aligned} \right\}. \quad (3.103)$$

We can also use Equation (3.65) to compute the circular convolution of length $N = 6$ of $x_1(n)$ and $h_1(n)$, obtaining

$$y_{c_6}(n) = x_1(n) \circledast h_1(n) = \sum_{l=0}^5 x(l)h((n-l) \bmod 6) \quad (3.104)$$

such that

$$\left. \begin{aligned} y_{c_6}(0) &= x_1(0)h_1(0) + x_1(1)h_1(5) + x_1(2)h_1(4) \\ &\quad + x_1(3)h_1(3) + x_1(4)h_1(2) + x_1(5)h_1(1) \\ &= x_1(0)h_1(0) \\ &= 4 \\ y_{c_6}(1) &= x_1(0)h_1(1) + x_1(1)h_1(0) + x_1(2)h_1(5) \\ &\quad + x_1(3)h_1(4) + x_1(4)h_1(3) + x_1(5)h_1(2) \\ &= x_1(0)h_1(1) + x_1(1)h_1(0) \\ &= 7 \\ y_{c_6}(2) &= x_1(0)h_1(2) + x_1(1)h_1(1) + x_1(2)h_1(0) \\ &\quad + x_1(3)h_1(5) + x_1(4)h_1(4) + x_1(5)h_1(3) \\ &= x_1(0)h_1(2) + x_1(1)h_1(1) + x_1(2)h_1(0) \\ &= 9 \\ y_{c_6}(3) &= x_1(0)h_1(3) + x_1(1)h_1(2) + x_1(2)h_1(1) \\ &\quad + x_1(3)h_1(0) + x_1(4)h_1(5) + x_1(5)h_1(4) \\ &= x_1(1)h_1(2) + x_1(2)h_1(1) + x_1(3)h_1(0) \\ &= 6 \\ y_{c_6}(4) &= x_1(0)h_1(4) + x_1(1)h_1(3) + x_1(2)h_1(2) \\ &\quad + x_1(3)h_1(1) + x_1(4)h_1(0) + x_1(5)h_1(5) \\ &= x_1(2)h_1(2) + x_1(3)h_1(1) \\ &= 3 \\ y_{c_6}(5) &= x_1(0)h_1(5) + x_1(1)h_1(4) + x_1(2)h_1(3) \\ &\quad + x_1(3)h_1(2) + x_1(4)h_1(1) + x_1(5)h_1(0) \\ &= x_1(3)h_1(2) \\ &= 1 \end{aligned} \right\}. \quad (3.105)$$

Comparing Equations (3.101) and (3.105), it is easy to confirm that $y_{c_6}(n)$ corresponds exactly to the linear convolution between $x(n)$ and $h(n)$. \triangle

We have now seen how it is possible to implement the linear convolution between two finite-length signals through the DFT. However, in practice, it is frequently necessary to implement the convolution of a finite-length sequence with an infinite-length sequence, or even to convolve a short-duration sequence with a long-duration sequence. In both cases, it is not feasible to compute the DFT of a very long or infinite sequence. The solution adopted, in these cases, is to divide the long sequence into blocks of duration N and perform the convolution of each block with the short sequence. In most practical cases, the long or infinite sequence corresponds to the system input and the short-length sequence to the system impulse response. However, the results of the convolution of each block must be properly combined so that the final result corresponds to the convolution of the long sequence with the short sequence. Two methods for performing this combination are the so-called overlap-and-add and overlap-and-save methods discussed below.

3.4.2 Overlap-and-add method

We can describe a signal $x(n)$ decomposed in nonoverlapping blocks $x_m(n - mN)$ of length N as

$$x(n) = \sum_{m=0}^{\infty} x_m(n - mN), \quad (3.106)$$

where

$$x_m(n) = \begin{cases} x(n + mN), & \text{for } 0 \leq n \leq N - 1 \\ 0, & \text{otherwise} \end{cases}; \quad (3.107)$$

that is, each block $x_m(n)$ is nonzero between 0 and $N - 1$, and $x(n)$ is composed of the sum of the blocks $x_m(n)$ shifted to $n = mN$.

Using Equation (3.106), the convolution of $x(n)$ with another signal $h(n)$ can be written as

$$y(n) = x(n) * h(n) = \sum_{m=0}^{\infty} (x_m(n - mN) * h(n)) = \sum_{m=0}^{\infty} y_m(n - mN). \quad (3.108)$$

Note that the above equation implies that $y_m(n)$ is the result of the convolution of $h(n)$ with the m th block $x_m(n)$.

As seen in Section 3.4.1, if we want to compute the linear convolutions leading to $y_m(n)$ using DFTs, then their lengths must be at least $(N + K - 1)$, where K is the duration of $h(n)$. Thus, if $x_m(n)$ and $h(n)$ are zero padded to length $(N + K - 1)$, then the linear convolutions in Equation (3.108) can be implemented using circular convolutions. If $x'_m(n)$ and $h'(n)$ are the zero-padded versions of $x_m(n)$ and $h(n)$, then we have that the filtered blocks $y_m(n)$ in Equation (3.108) become

$$y_m(n) = \sum_{l=0}^{N+K-2} x'_m(l)h'(n - l), \quad \text{for } 0 \leq n \leq N + K - 2. \quad (3.109)$$

Then, from Equations (3.108) and (3.109), we see that in order to convolve the long $x(n)$ with the length- K $h(n)$ it suffices to:

- (i) Divide $x(n)$ into length- N blocks $x_m(n)$.
- (ii) Zero pad $h(n)$ and each block $x_m(n)$ to length $N + K - 1$.
- (iii) Perform the circular convolution of each block using the length- $(N + K - 1)$ DFT.
- (iv) Add the results according to Equation (3.108).

Note that in the addition performed in step (iv), there is an overlap of the $K - 1$ last samples of $y_m(n - mN)$ and the $K - 1$ first samples of $y_{m+1}(n - (m + 1)N)$. This is why the above procedure is called the overlap-and-add method, which is illustrated schematically in Figure 3.4.

Example 3.7. Compute graphically the linear convolution of $x(n)$ and $h(n)$ in Figure 3.5a by splitting $x(n)$ into blocks of two samples and using the overlap-and-add method.

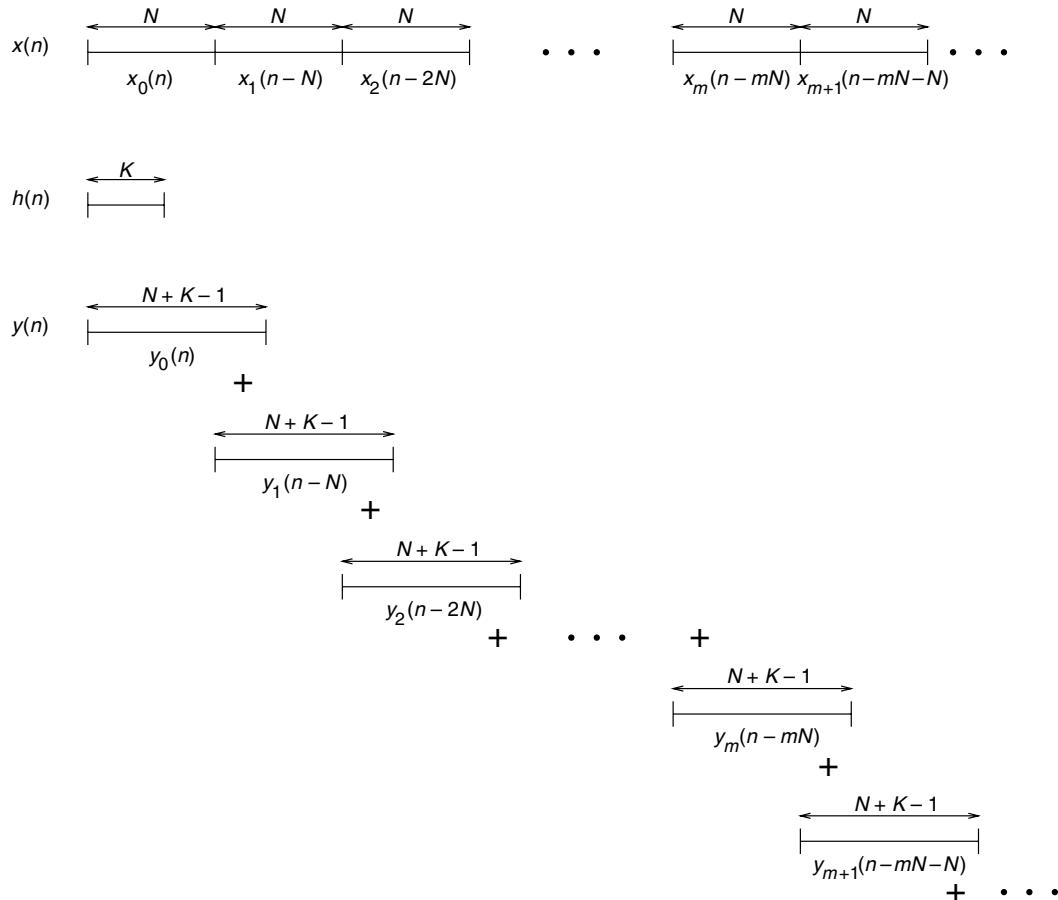


Fig. 3.4. Illustration of the overlap-and-add method.

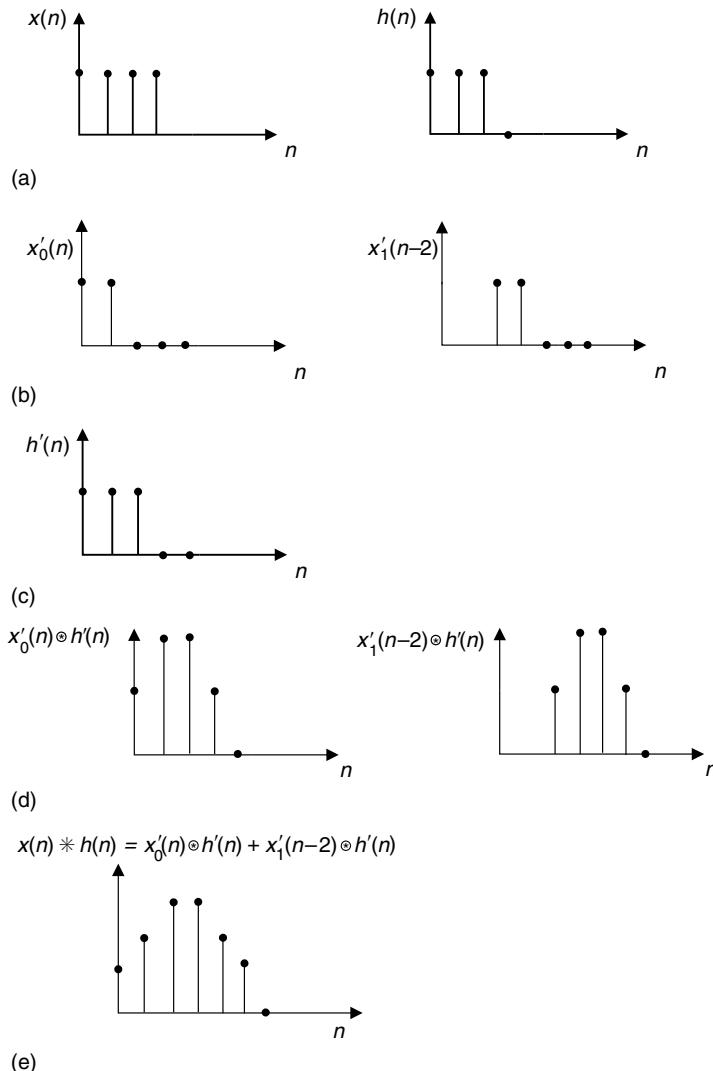


Fig. 3.5. Example of the application of the overlap-and-add method: (a) original sequences; (b) input sequence divided into length-2 blocks, where each block has been zero padded to length $N = 5$; (c) zero-padded $h(n)$; (d) circular convolution of each zero-padded block with zero-padded $h(n)$; (e) final result (normalized).

Solution

According to Equations (3.106) and (3.107), the division of the original signal in Figure 3.5a in two blocks having length $N = 2$ can be expressed as

$$x(n) = x_0(n) + x_1(n - 2), \quad (3.110)$$

where

$$\begin{aligned} x_0(n) &= \begin{cases} x(n), & \text{for } 0 \leq n \leq 1 \\ 0, & \text{otherwise} \end{cases} \\ x_1(n) &= \begin{cases} x(n+2), & \text{for } 0 \leq n \leq 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (3.111)$$

The two signals on the right-hand side of Equation (3.110) are depicted in Figure 3.5b. The desired convolution is then

$$y(n) = (x_0(n) * h(n)) + (x_1(n-2) * h(n)), \quad (3.112)$$

where each of the signals on the right-hand side of Equation (3.112) are depicted in Figure 3.5d. Their sum $y(n)$ is shown in Figure 3.5e. \triangle

3.4.3 Overlap-and-save method

In the overlap-and-add method, one divides the signal into blocks of length N and computes the convolutions using DFTs of size $(N + K - 1)$, where K is the duration of $h(n)$. In the overlap-and-save method, one uses DFTs of length N instead. This poses a problem, because the length- N circular convolution of a length- N block $x_m(n)$ and a length- K $h(n)$ is not equal to their linear convolution. To circumvent this, one only uses the samples of the circular convolution that are equal to those of the linear convolution. These valid samples can be determined by referring to the expression of the circular convolution in Section 3.4.1, Equation (3.93). From there, we see that the condition for computing a linear convolution using a circular convolution is given by Equation (3.94), repeated here for convenience, with $x(n)$ replaced by $x_m(n)$:

$$c(n) = \sum_{l=n+1}^{N-1} x_m(l)h(n-l+N) = 0, \quad \text{for } 0 \leq n \leq N-1. \quad (3.113)$$

This equation indicates that the length- N circular convolution between a length- N block $x_m(n)$ and a length- K $h(n)$ is equal to their linear convolution whenever the above summation is null. Since, for $0 \leq n \leq N-1$, we have that $x_m(n) \neq 0$, then $c(n)$ above is null only for n such that all the $h(n)$ in the summation are zero; that is, $h(n-l+N) = 0$ for l in the interval $n+1 \leq l \leq N-1$. Since $h(n)$ has length K , then $h(r) = 0$, for $r \geq K$. Since we should have $h(n-l+N) = 0$, then n should be such that $n-l+N \geq K$; that is, $n \geq K-N+l$. The most strict case in this inequality is when $l=N-1$. This implies that the condition in Equation (3.113) is satisfied only when $n \geq K-1$.

The conclusion is that the only samples of the length- N circular convolution that are equal to those of the linear convolution are for $n \geq K-1$. Therefore, when computing the convolution of the blocks $x_m(n)$ with $h(n)$, the first $K-1$ samples of the result have to be discarded. In order to compensate for the discarded samples, there must be an overlap of an extra $K-1$ samples between adjacent blocks.

Thus, the signal $x(n)$ must be divided into blocks $x_m(n)$ of length N such that

$$x_m(n) = \begin{cases} x(n + m(N - K + 1) - K + 1), & \text{for } 0 \leq n \leq N - 1 \\ 0, & \text{otherwise} \end{cases}. \quad (3.114)$$

Note that the first $K - 1$ samples of $x_m(n)$ are equal to the last $K - 1$ samples of $x_{m-1}(n)$. The filtered output of the m th block consists only of the samples of the circular convolution $y_m(n)$ of $x_m(n)$ and $h(n)$ of index larger than or equal to $K - 1$. It is important that the original signal $x(n)$ be padded with $K - 1$ zeros at the beginning, since the first $K - 1$ samples of the output are discarded. Then, if $h'(n)$ is the version of $h(n)$ zero padded to

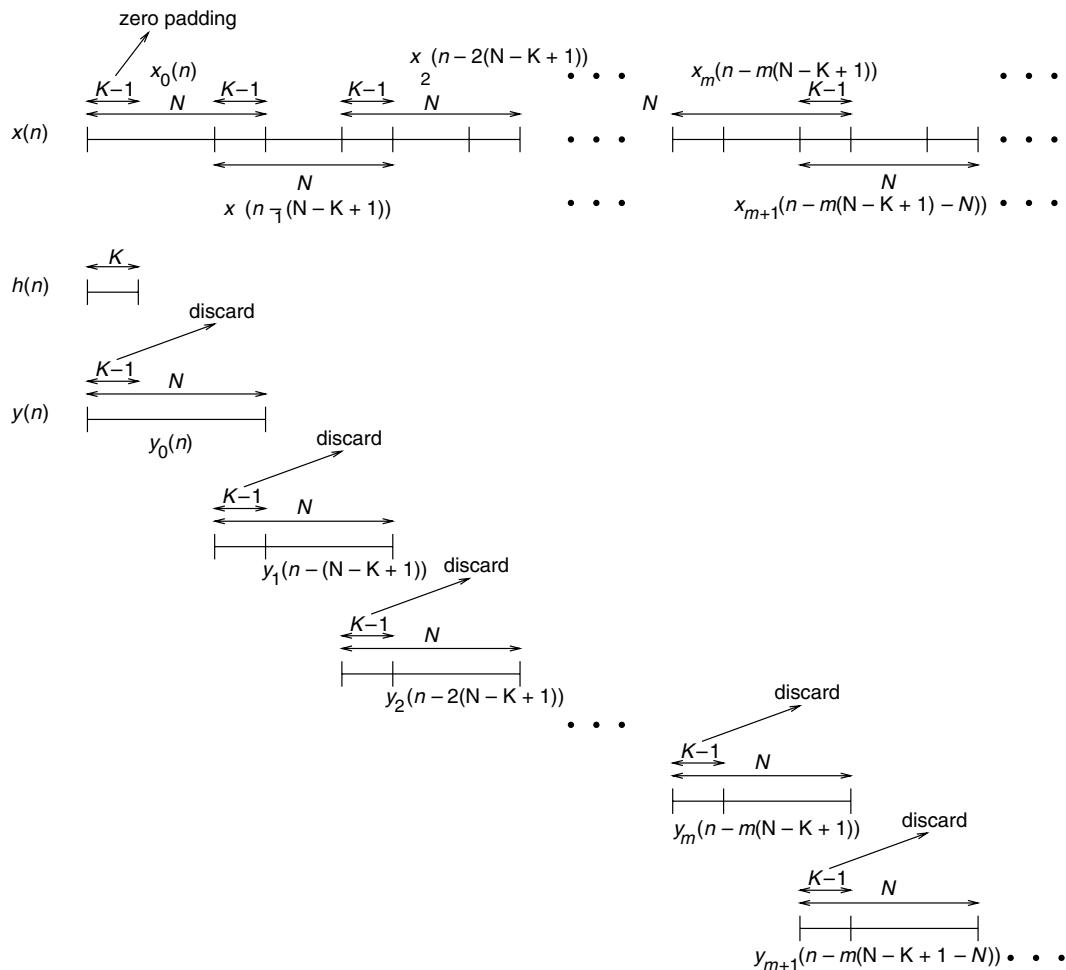


Fig. 3.6. Illustration of the overlap-and-save method.

length N , the output $y_m(n)$ of each block can be expressed as

$$y_m(n) = \sum_{l=0}^{N-1} x_m(l)h'((n-l) \bmod N), \quad (3.115)$$

where only the samples of $y_m(n)$ from $n = K - 1$ to $n = N - 1$ need to be computed.

Then the output $y(n) = x(n) * h(n)$ is built, for each m , as

$$y(n) = y_m(n - m(N - K + 1)) \quad (3.116)$$

for $m(N - K + 1) + K - 1 \leq n \leq m(N - K + 1) + N - 1$.

From Equations (3.115) and (3.116), we see that, in order to convolve the long $x(n)$ with the length- K $h(n)$ using the overlap-and-save method, it suffices to:

- (i) Divide $x(n)$ into length- N blocks $x_m(n)$ with an overlap of $K - 1$ samples as in Equation (3.114). The first block should be zero padded with $K - 1$ zeros at its beginning. If the original signal has length L , then the total number of blocks B should obey

$$B \geq \frac{L + K - 1}{N - K + 1}. \quad (3.117)$$

- (ii) Zero pad $h(n)$ to length N .
- (iii) Perform the circular convolution of each block with $h(n)$ (Equation (3.115)) using a length- N DFT.
- (iv) Build the output signal according to Equation (3.116).

Note that we can interpret step (iv) as the $K - 1$ last samples of block $y_m(n)$ being saved in order to replace the discarded $K - 1$ first samples of block $y_{m+1}(n)$, thus justifying the terminology overlap-and-save method, which is illustrated schematically in Figure 3.6.

Example 3.8. Determine, graphically, the linear convolution of $x(n)$ and $h(n)$ in Figure 3.7a using the DFT, partitioning $x(n)$ into length-6 blocks and using the overlap-and-save method.

Solution

The length of the impulse response $h(n)$ is $K = 3$, $x(n)$ has length $L = 8$, and the DFT has length $N = 6$. From Equation (3.117), the number of overlapping blocks should be

$$B \geq \frac{8 + 3 - 1}{6 - 3 + 1} = 2.5; \quad (3.118)$$

therefore, $B = 3$. The beginning of the first block will be zero padded with two zeros and the last block will be zero padded with four zeros. Therefore, from Equation (3.114), we

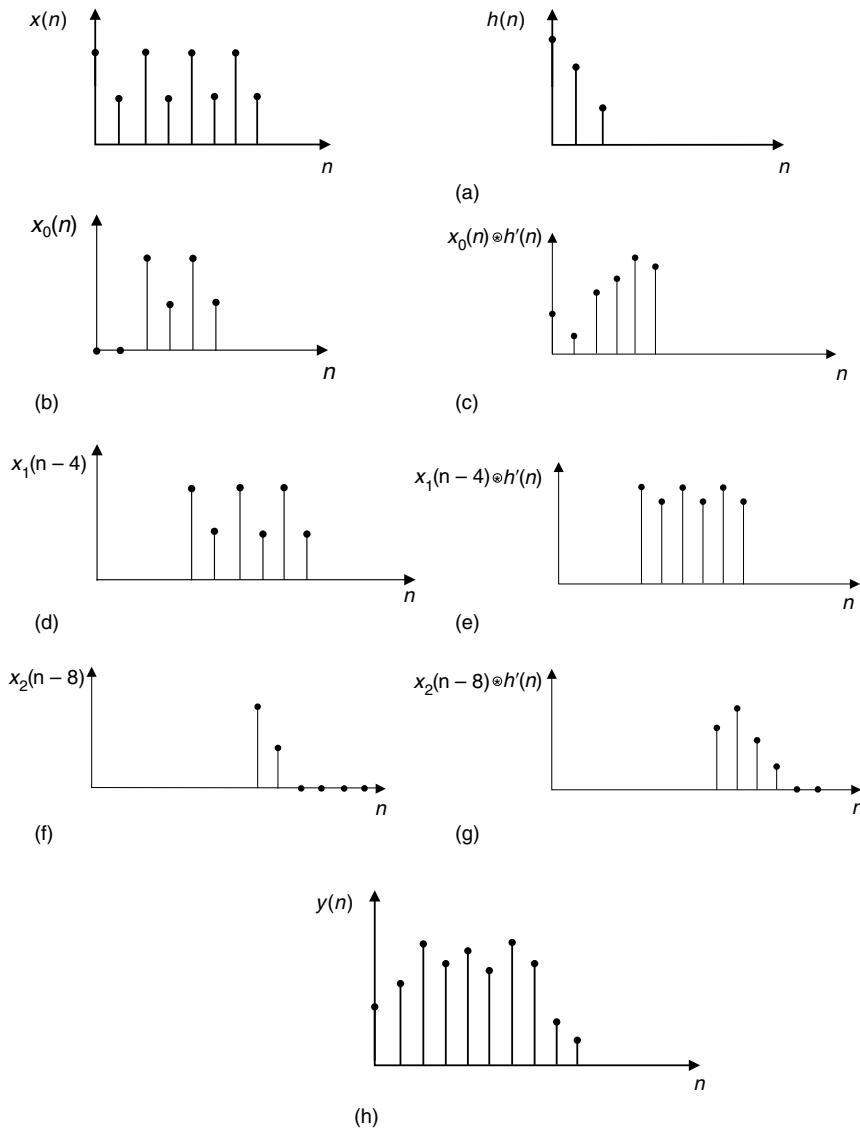


Fig. 3.7. Linear convolution using the DFT and the overlap-and-save method: (a) original sequences; (b) first block; (c) first partial convolution; (d) second block; (e) second partial convolution; (f) third block; (g) third partial convolution; (h) final result.

have that

$$\begin{aligned}x_0(n) &= \begin{cases} x(n-2), & \text{for } 0 \leq n \leq 5 \\ 0, & \text{otherwise} \end{cases} \\x_1(n) &= \begin{cases} x(n+2), & \text{for } 0 \leq n \leq 5 \\ 0, & \text{otherwise} \end{cases} \\x_2(n) &= \begin{cases} x(n+6), & \text{for } 0 \leq n \leq 5 \\ 0, & \text{otherwise} \end{cases}\end{aligned}\quad (3.119)$$

These signals are depicted in Figures 3.7b, 3.7d, and 3.7f respectively. Using Equation (3.115), the signals in Figures 3.7c, 3.7e and 3.7g are computed as

$$y_m(n) = \sum_{l=0}^5 x_m(l)h'((n-l) \bmod 6), \quad \text{for } 2 \leq n \leq 5 \quad \text{and } m = 0, 1, 2. \quad (3.120)$$

The final result in Figure 3.7h is computed using Equation (3.116), yielding

$$\left. \begin{array}{ll} y(n) = y_0(n), & \text{for } 2 \leq n \leq 5 \\ y(n) = y_1(n-4), & \text{for } 6 \leq n \leq 9 \\ y(n) = y_2(n-8), & \text{for } 10 \leq n \leq 13 \end{array} \right\}. \quad (3.121)$$

Note that in this example, we have that $K = 3$, $N = 6$, and each partial convolution generates $(N - K + 1) = 4$ new samples. \triangle

3.5 Fast Fourier transform

In the previous section, we saw that the DFT is an effective discrete representation in frequency that can be used to compute linear convolutions between two discrete sequences. However, by examining the DFT and IDFT definitions in Equations (3.15) and (3.16), repeated here for convenience

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad \text{for } 0 \leq k \leq N-1 \quad (3.122)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-kn}, \quad \text{for } 0 \leq n \leq N-1, \quad (3.123)$$

we see that, in order to compute the DFT and IDFT of a length- N sequence, one needs about N^2 complex multiplications; that is, the complexity of the DFT grows with the square of the signal length. This severely limits its practical use for lengthy signals. Fortunately, in 1965, Cooley and Tukey (1965) proposed an efficient algorithm to compute the DFT, which requires a number of complex multiplications of the order of $N \log_2 N$. This may represent a tremendous decrease in complexity. For example, even for signal lengths as low as 1024 samples, the decrease in complexity is of the order of 100 times; that is, two orders of magnitude. Needless to say, the advent of this algorithm opened up an endless list of practical applications for the DFT, ranging from signal analysis to fast linear filtering. Today, there is an enormous number of fast algorithms for the computation of the DFT, and they are collectively known as FFT algorithms (Cochran *et al.*, 1967). In this section we will study some of the most popular types of FFT algorithm.

3.5.1 Radix-2 algorithm with decimation in time

Suppose we have a sequence $x(n)$ whose length N is a power of two; that is, $N = 2^l$. Now, let us express the DFT relation in Equation (3.122) by splitting the summation into two parts, one with the even-indexed $x(n)$ and another with the odd-indexed $x(n)$, obtaining

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{(N/2)-1} x(2n) W_N^{2nk} + \sum_{n=0}^{(N/2)-1} x(2n+1) W_N^{(2n+1)k} \\ &= \sum_{n=0}^{(N/2)-1} x(2n) W_N^{2nk} + W_N^k \sum_{n=0}^{(N/2)-1} x(2n+1) W_N^{2nk}. \end{aligned} \quad (3.124)$$

If we note that for N even, we have that

$$W_N^{2nk} = e^{-j(2\pi/N)2nk} = e^{-j[2\pi/(N/2)]nk} = W_{N/2}^{nk}, \quad (3.125)$$

then Equation (3.124) becomes

$$X(k) = \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nk} \quad (3.126)$$

and we can see that each summation may represent a distinct DFT of size $N/2$. Therefore, a DFT of size N can be computed through two DFTs of size $N/2$, in addition to the multiplications by W_N^k . Note that each new DFT has only $N/2$ coefficients, and for the computation of its coefficients we now need solely $(N/2)^2$ complex multiplications. In addition, since we have one distinct coefficient W_N^k for each k between 0 and $N - 1$, then we need to perform N multiplications by W_N^k . Therefore, the computation of the DFT according to Equation (3.126) requires

$$2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N \quad (3.127)$$

complex multiplications. Since $N + (N^2/2)$ is smaller than N^2 , for $N > 2$, Equation (3.126) provides a decrease in complexity when compared with the usual DFT computation.

We should also compare the number of complex additions of the two forms of computation. The usual length- N DFT computation needs a total of $N(N - 1) = N^2 - N$ additions. In Equation (3.126), we need to compute two DFTs of length $N/2$ and then perform the N additions of the two DFTs after multiplication by W_N^k . Therefore, the total number of

complex additions in Equation (3.126) is

$$2 \left[\left(\frac{N}{2} \right)^2 - \frac{N}{2} \right] + N = \frac{N^2}{2}, \quad (3.128)$$

which also corresponds to a reduction in complexity.

From the above, exploiting the fact that N is a power of 2, it is easy to see that if the procedure shown in Equation (3.126) is recursively applied to each of the resulting DFTs, we may get a very significant reduction in complexity until all the remaining DFTs are of length 2. The overall procedure is formalized in an algorithm by first rewriting Equation (3.126) as

$$X(k) = X_e(k) + W_N^k X_o(k) \quad (3.129)$$

where $X_e(k)$ and $X_o(k)$ are, respectively, the DFTs of length $N/2$ of the even- and odd-indexed samples of $x(n)$; that is:

$$\left. \begin{aligned} X_e(k) &= \sum_{n=0}^{(N/2)-1} x(2n) W_{N/2}^{nk} = \sum_{n=0}^{(N/2)-1} x_e(n) W_{N/2}^{nk} \\ X_o(k) &= \sum_{n=0}^{(N/2)-1} x(2n+1) W_{N/2}^{nk} = \sum_{n=0}^{(N/2)-1} x_o(n) W_{N/2}^{nk} \end{aligned} \right\}. \quad (3.130)$$

The DFTs above can be computed by separating $x_e(n)$ and $x_o(n)$ into their even- and odd-indexed samples, as follows:

$$\left. \begin{aligned} X_e(k) &= \sum_{n=0}^{(N/4)-1} x_e(2n) W_{N/4}^{nk} + W_{N/2}^k \sum_{n=0}^{(N/4)-1} x_e(2n+1) W_{N/4}^{nk} \\ X_o(k) &= \sum_{n=0}^{(N/4)-1} x_o(2n) W_{N/4}^{nk} + W_{N/2}^k \sum_{n=0}^{(N/4)-1} x_o(2n+1) W_{N/4}^{nk} \end{aligned} \right\} \quad (3.131)$$

such that

$$\left. \begin{aligned} X_e(k) &= X_{ee}(k) + W_{N/2}^k X_{eo}(k) \\ X_o(k) &= X_{oe}(k) + W_{N/2}^k X_{oo}(k) \end{aligned} \right\}, \quad (3.132)$$

where $X_{ee}(k)$, $X_{eo}(k)$, $X_{oe}(k)$, and $X_{oo}(k)$ correspond now to DFTs of length $N/4$.

Generically, in each step we compute DFTs of length L using DFTs of length $L/2$ as follows:

$$X_i(k) = X_{ie}(k) + W_L^k X_{io}(k). \quad (3.133)$$

A recursive application of the above procedure can convert the computation of a DFT of length $N = 2^l$ in l steps to the computation of 2^l DFTs of length 1, because each step

converts a DFT of length L into two DFTs of length $L/2$ plus a complex product by W_L^k and a complex sum. Therefore, supposing that $\mathcal{M}(N)$ and $\mathcal{A}(N)$ are respectively the number of complex multiplications and additions to compute a DFT of length N , the following relations hold:

$$\mathcal{M}(N) = 2\mathcal{M}\left(\frac{N}{2}\right) + N \quad (3.134)$$

$$\mathcal{A}(N) = 2\mathcal{A}\left(\frac{N}{2}\right) + N. \quad (3.135)$$

In order to compute the values of $\mathcal{M}(N)$ and $\mathcal{A}(N)$, we must solve the recursive equations above. The initial conditions are $\mathcal{M}(1) = 1$ and $\mathcal{A}(1) = 0$, since a length-1 DFT needs no additions and a multiplication by W_1^0 (these multiplications are trivial but must be considered in order to maintain coherence with Equation (3.127); they will be discounted when stating the final result).

We can compute the number of multiplications employing the change of variables $N = 2^l$ and $T(l) = \mathcal{M}(N)/N$. With it, Equation (3.134) becomes

$$T(l) = T(l - 1) + 1. \quad (3.136)$$

Since $T(0) = \mathcal{M}(1) = 1$, then $T(l) = l + 1$. Therefore, we conclude that

$$\frac{\mathcal{M}(N)}{N} = 1 + \log_2 N \quad (3.137)$$

and thus

$$\mathcal{M}(N) = N + N \log_2 N. \quad (3.138)$$

Note that the above equation is coherent with Equation (3.127). However, since we do not perform the trivial multiplications necessary to compute the N DFTs of size 1, we conclude that the actual number of multiplications is

$$\mathcal{M}(N) = N \log_2 N. \quad (3.139)$$

In order to compute the number of additions, we can use the same change of variables; that is, we make $N = 2^l$ and $T(l) = \mathcal{A}(N)/N$. With it, Equation (3.135) becomes

$$T(l) = T(l - 1) + 1, \quad (3.140)$$

but this time $T(0) = \mathcal{A}(1) = 0$, and then $T(l) = l$. Therefore, we conclude that

$$\frac{\mathcal{A}(N)}{N} = \log_2 N \quad (3.141)$$

and thus

$$\mathcal{A}(N) = N \log_2 N; \quad (3.142)$$

that is, the FFT can be computed using $N \log_2 N$ complex multiplications and additions, which comprises an economy of the order of $N/\log_2 N$, when compared with the direct implementation in Equation (3.122).

This FFT algorithm is known as a decimation-in-time algorithm because it recursively divides the sequence $x(n)$ into subsequences. We can devise a graphical representation of the above procedure by noting that in the operation in Equation (3.133) each value of either $X_{ie}(k)$ or $X_{io}(k)$ is used twice. This is so because, if $X_i(k)$ has length L , then $X_{ie}(k)$ and $X_{io}(k)$ have length $L/2$ (and thus period $L/2$). The corresponding equations are

$$X_i(k) = X_{ie}(k) + W_L^k X_{io}(k) \quad (3.143)$$

$$\begin{aligned} X_i\left(k + \frac{L}{2}\right) &= X_{ie}\left(k + \frac{L}{2}\right) + W_L^{k+(L/2)} X_{io}\left(k + \frac{L}{2}\right) \\ &= X_{ie}(k) + W_L^{k+(L/2)} X_{io}(k). \end{aligned} \quad (3.144)$$

Therefore, if we have the DFTs of length $L/2$, $X_{ie}(k)$ and $X_{io}(k)$, we can compute the length- L DFT $X_i(k)$ by applying Equations (3.143) and (3.144), for $k = 0, 1, \dots, (L/2) - 1$. This process is illustrated by the graph in Figure 3.8. Since the FFT algorithm is composed of repetitions of this procedure, it is often called the basic cell of the algorithm. Owing to the appearance of its graph, the basic cell is often referred to as a butterfly.

A graph for the complete FFT algorithm above is obtained by repeating the basic cell in Figure 3.8, for $L = N, N/2, \dots, 2$ and for $k = 0, 1, \dots, (L/2) - 1$. Figure 3.9 illustrates the case when $N = 8$.

The graph in Figure 3.9 has an interesting property. The nodes of one section of the graph depend only on nodes of the previous section of the graph. For example, $X_i(k)$ depends only on $X_{ie}(k)$ and $X_{io}(k)$. In addition, $X_{ie}(k)$ and $X_{io}(k)$ are only used to compute $X_i(k)$ and $X_i(k + (L/2))$. Therefore, once computed, the values of $X_i(k)$ and $X_i(k + (L/2))$ can be stored in the same place as $X_{ie}(k)$ and $X_{io}(k)$. This implies that the intermediate results of the length- N FFT computation can be stored in a single size- N vector; that is, the results of section l can be stored in the same place as the results of section $l - 1$. This is why the computation of the intermediate results of the FFT is often said to be “in place.”

Another important aspect of the FFT algorithm relates to the ordering of the input vector. As seen in Figure 3.9, the output vector is ordered sequentially while the input vector is not. A general rule for the ordering of the input vector can be devised by referring to the

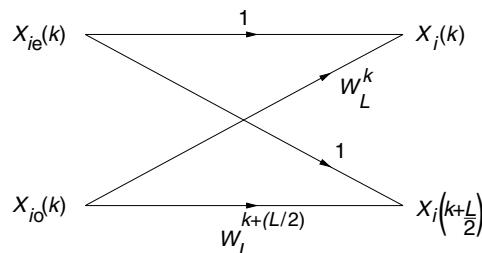


Fig. 3.8. Basic cell of the decimation-in-time FFT algorithm.

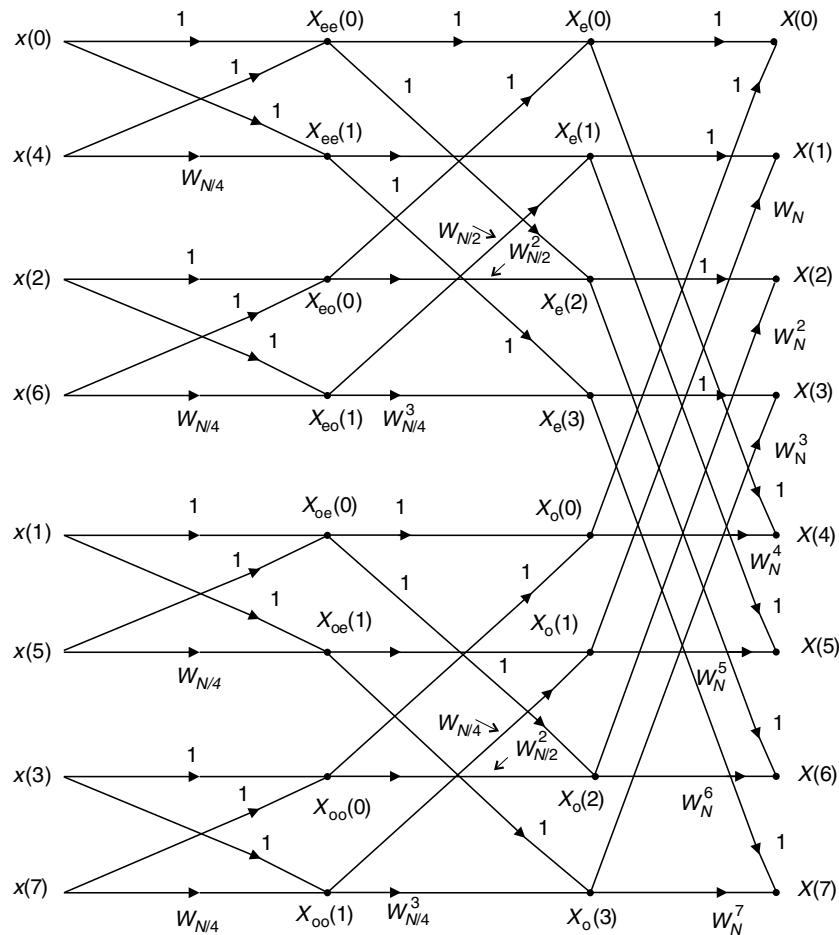


Fig. 3.9. Graph of the decimation-in-time eight-point FFT algorithm.

FFT graph in Figure 3.9. Moving from right to left, we note that, in the second section, the upper half corresponds to the DFT of the even-indexed samples and the lower half to the DFT of the odd-indexed samples. Since the even-indexed samples have the least significant bit (LSB) equal to zero and the odd-indexed samples have the LSB equal to one, then the DFT of the upper half corresponds to samples with an index having $\text{LSB} = 0$ and the lower half to the DFT of samples with an index having $\text{LSB} = 1$. Likewise, for each half, the upper half corresponds to the even-indexed samples belonging to this half; that is, the ones with the second LSB = 0. Similarly, the lower half corresponds to the second LSB = 1. If we proceed until we reach the input signal on the left, then we end up with the uppermost sample having an index with all bits equal to zero, the second one having the first bit equal to one and the others equal to zero, and so on. We note, then, that the ordering of the input vector is the ascending order of the bit-reversed indexes. For example, $x(3) = x(011)$ will be put in position 110 of the input; that is, position 6.

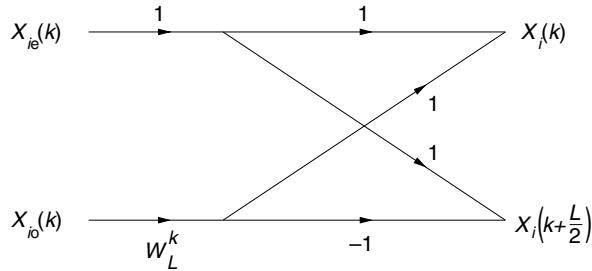


Fig. 3.10. More efficient basic cell of the decimation-in-time FFT algorithm.

An additional economy in the number of complex multiplications of the algorithm can be obtained by noting that, in Equation (3.144).

$$W_L^{k+(L/2)} = W_L^k W_L^{L/2} = W_L^k W_2 = -W_L^k. \quad (3.145)$$

Then, Equations (3.143) and (3.144) can be rewritten as

$$\left. \begin{aligned} X_i(k) &= X_{ie}(k) + W_L^k X_{io}(k) \\ X_i(k + (L/2)) &= X_{ie}(k) - W_L^k X_{io}(k) \end{aligned} \right\}. \quad (3.146)$$

This allows a more efficient implementation of the basic cell in Figure 3.8, using one complex multiplication instead of two. The resulting cell is depicted in Figure 3.10. Substituting the basic cells corresponding to Figure 3.8 by those corresponding to Figure 3.10, we have the more efficient graph for the FFT shown in Figure 3.11.

With this new basic cell, the number of complex multiplications has dropped to half; that is, there is a total of $(N/2) \log_2 N$ complex multiplications. In order to perform a more accurate calculation, we have to discount the remaining trivial multiplications. One set of them are by $W_L^0 = 1$. In Equation (3.146), when the DFTs have length L , we have N/L DFTs and thus the term W_L^0 appears N/L times. Then, we have $N/2 + N/4 + \dots + N/N = N - 1$ multiplications by 1. The other set of trivial multiplications are the ones by $W_L^{L/4} = -j$. Since, in the first stage, there are no terms equal to $-j$ and, from the second stage on, the number of times a term $-j$ appears is the same as the number of times a term 1 appears, then we have $N - 1 - (N/2) = (N/2) - 1$ multiplications by $-j$. This gives an overall number of nontrivial complex multiplications equal to

$$\mathcal{M}(N) = \frac{N}{2} \log_2 N - N + 1 - \frac{N}{2} + 1 = \frac{N}{2} \log_2 N - \frac{3}{2}N + 2. \quad (3.147)$$

Note that the number of complex additions remains $\mathcal{A}(N) = N \log_2 N$.

If we shuffle the horizontal branches of the graph in Figure 3.11 so that the input signal is in normal ordering, then the output of the graph comes in bit-reversed ordering. In this case, we have another “in place” algorithm. In fact, there are a plethora of forms for the FFT. For example, there is an algorithm in which the input and output appear in normal ordering, but where there is no possibility of “in place” computation (Cochran *et al.*, 1967; Oppenheim & Schafer, 1975).

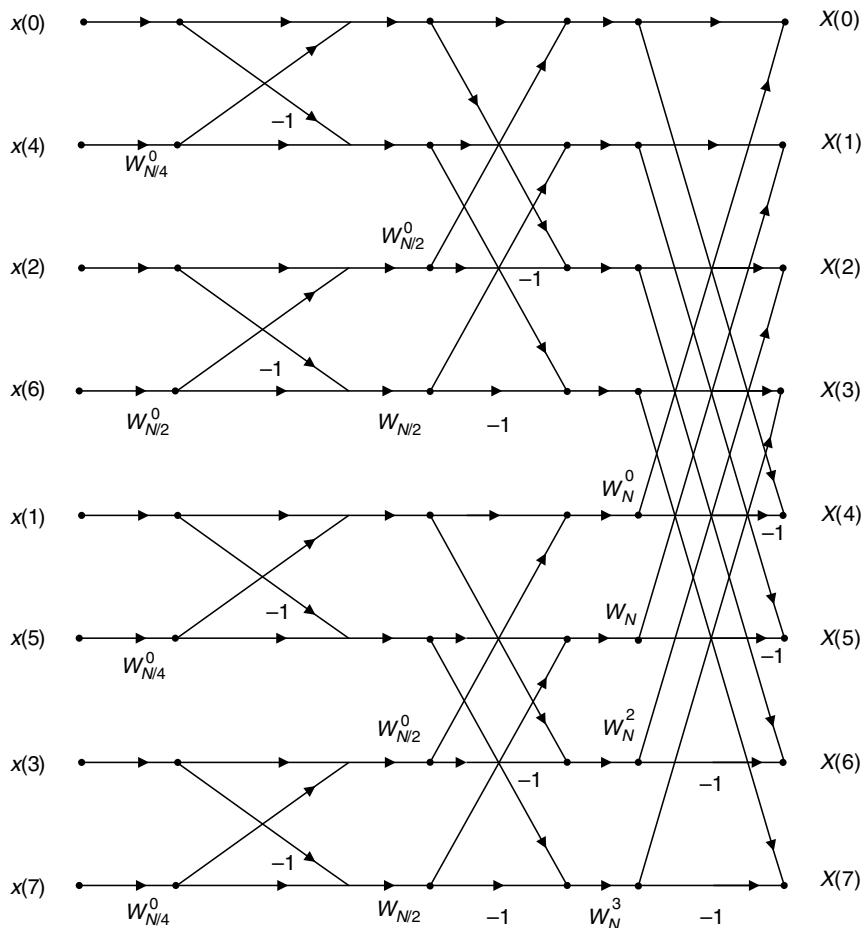


Fig. 3.11. More efficient graph of the decimation-in-time eight-point FFT algorithm. The unmarked branches are equal to 1.

It is clear from Equations (3.122) and (3.123) that, in order to compute the IDFT it suffices to change, in the graphs of Figures 3.9 or 3.11, the terms W_N^k to W_N^{-k} and divide the output of the graph by N .

An interesting interpretation of the FFT algorithms can be found if we look at the matrix form of the DFT in Equation (3.39):

$$\mathbf{X} = \mathbf{W}_N \mathbf{x}. \quad (3.148)$$

The matrix form is widely used for describing fast transform algorithms (Elliott & Rao, 1982). For instance, the decimation-in-time FFT algorithm can be represented in matrix form if we notice that Equations (3.143) and (3.144), corresponding to the basic cell in

Figure 3.8, can be expressed in matrix form as

$$\begin{bmatrix} X_i(k) \\ X_i(k + (L/2)) \end{bmatrix} = \begin{bmatrix} 1 & W_L^k \\ 1 & W_L^{k+(L/2)} \end{bmatrix} \begin{bmatrix} X_{ie}(k) \\ X_{io}(k) \end{bmatrix} \quad (3.149)$$

Then the graph in Figure 3.9 can be expressed as

$$\mathbf{X} = \mathbf{F}_8^{(8)} \mathbf{F}_8^{(4)} \mathbf{F}_8^{(2)} \mathbf{P}_8 \mathbf{x}, \quad (3.150)$$

where

$$\mathbf{P}_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.151)$$

corresponds to the bit-reversal operation onto the indexes of the input vector \mathbf{x} ,

$$\mathbf{F}_8^{(2)} = \begin{bmatrix} 1 & W_2^0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W_2^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_2^1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_2^1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_2^1 & 0 \end{bmatrix} \quad (3.152)$$

corresponds to the basic cells of the first stage,

$$\mathbf{F}_8^{(4)} = \begin{bmatrix} 1 & 0 & W_4^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & W_4^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_4^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W_4^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_4^3 & 0 \end{bmatrix} \quad (3.153)$$

corresponds to the basic cells of the second stage, and

$$\mathbf{F}_8^{(8)} = \begin{bmatrix} 1 & 0 & 0 & 0 & W_8^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & W_8^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8^5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^7 \end{bmatrix} \quad (3.154)$$

corresponds to the basic cells of the third stage.

Comparing Equations (3.148) and (3.150), one can see the FFT algorithm in Figure 3.9 as a factorization of the DFT matrix \mathbf{W}_8 such that

$$\mathbf{W}_8 = \mathbf{F}_8^{(8)} \mathbf{F}_8^{(4)} \mathbf{F}_8^{(2)} \mathbf{P}_8. \quad (3.155)$$

This factorization corresponds to a fast algorithm because the matrices $\mathbf{F}_8^{(8)}$, $\mathbf{F}_8^{(4)}$, $\mathbf{F}_8^{(2)}$, and \mathbf{P}_8 have most elements equal to zero. Because of that, the product by each matrix above can be effected at the expense of at most eight complex multiplications. An exception is the product by \mathbf{P}_8 , which, being just a permutation, requires no multiplications at all.

Equations (3.150)–(3.155) exemplify the general fact that each different FFT algorithm corresponds to a different factorization of the DFT matrix \mathbf{W}_N into sparse matrices. For example, the reduction in complexity achieved by replacing Figure 3.8 (Equations (3.143) and (3.144)) by Figure 3.10 (Equation (3.146)) is equivalent to factoring the matrix in Equation (3.149) as

$$\begin{bmatrix} 1 & W_L^k \\ 1 & W_L^{k+(L/2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & W_L^k \end{bmatrix}. \quad (3.156)$$

3.5.2 Decimation in frequency

An alternative algorithm for the fast computation of the DFT can be obtained using decimation in frequency; that is, division of $X(k)$ into subsequences. The algorithm is generated as follows:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + \sum_{n=N/2}^{N-1} x(n) W_N^{nk} \end{aligned}$$

$$\begin{aligned}
&= \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + \sum_{n=0}^{(N/2)-1} x\left(n + \frac{N}{2}\right) W_N^{(N/2)k} W_N^{nk} \\
&= \sum_{n=0}^{(N/2)-1} \left(x(n) + W_N^{(N/2)k} x\left(n + \frac{N}{2}\right) \right) W_N^{nk}.
\end{aligned} \tag{3.157}$$

We can now compute the even and odd samples of $X(k)$ separately; that is:

$$\begin{aligned}
X(2l) &= \sum_{n=0}^{(N/2)-1} \left(x(n) + W_N^{nl} x\left(n + \frac{N}{2}\right) \right) W_N^{2nl} \\
&= \sum_{n=0}^{(N/2)-1} \left(x(n) + x\left(n + \frac{N}{2}\right) \right) W_N^{2nl}
\end{aligned} \tag{3.158}$$

for $l = 0, 1, \dots, (N/2) - 1$, and

$$\begin{aligned}
X(2l+1) &= \sum_{n=0}^{(N/2)-1} \left(x(n) + W_N^{(2l+1)N/2} x\left(n + \frac{N}{2}\right) \right) W_N^{(2l+1)n} \\
&= \sum_{n=0}^{(N/2)-1} \left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_N^{(2l+1)n} \\
&= \sum_{n=0}^{(N/2)-1} \left[\left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_N^n \right] W_N^{2ln}
\end{aligned} \tag{3.159}$$

for $l = 0, 1, \dots, (N/2) - 1$.

Equations (3.158) and (3.159) can be recognized as DFTs of length $N/2$, since $W_N^{2ln} = W_{N/2}^{ln}$. Before computing these DFTs, we have to compute the two intermediate signals $S_e(n)$ and $S_o(n)$, of length $N/2$, given by

$$\left. \begin{aligned} S_e(n) &= x(n) + x\left(n + \frac{N}{2}\right) \\ S_o(n) &= \left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_N^n \end{aligned} \right\}. \tag{3.160}$$

Naturally, the above procedure can be repeated for each of the DFTs of length $N/2$, thus generating DFTs of length $N/4, N/8$, and so on. Therefore, the basic cell used in the decimation-in-frequency computation of the DFT is characterized by

$$\left. \begin{aligned} S_{ie}(n) &= S_i(n) + S_i\left(n + \frac{L}{2}\right) \\ S_{io}(n) &= \left(S_i(n) - S_i\left(n + \frac{L}{2}\right) \right) W_L^n \end{aligned} \right\}, \tag{3.161}$$

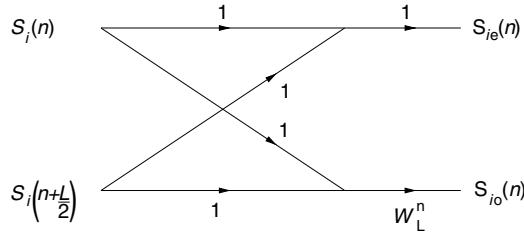


Fig. 3.12. Basic cell of the decimation-in-frequency FFT algorithm.

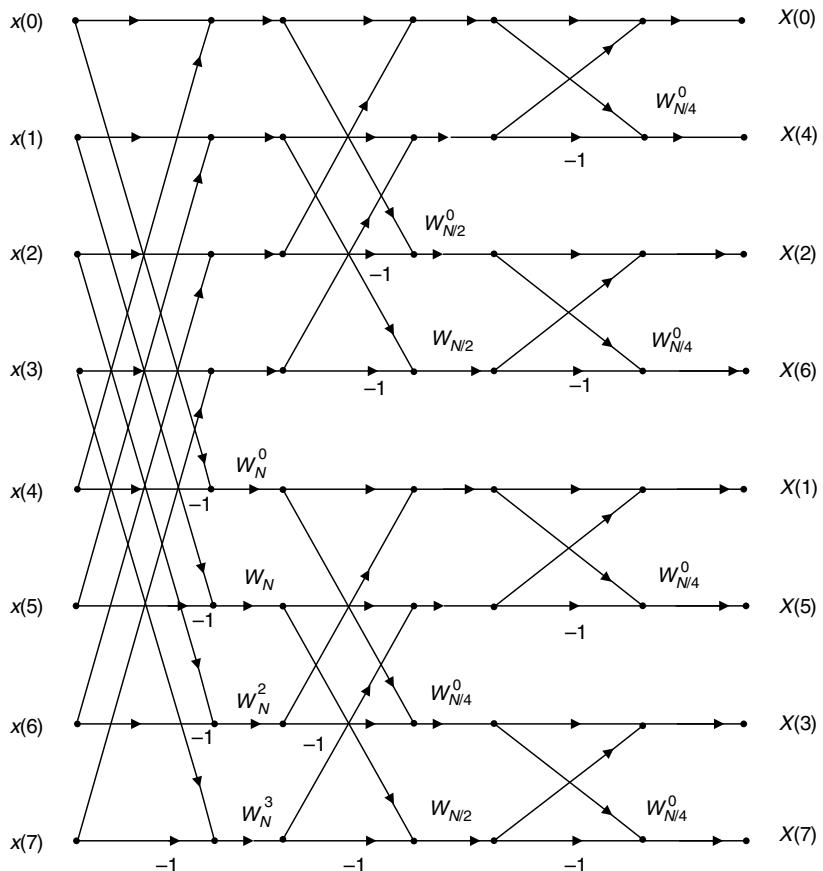


Fig. 3.13. Graph of the decimation-in-frequency eight-point FFT algorithm.

where $S_{ie}(n)$ and $S_{io}(n)$ have length $L/2$ and $S_i(n)$ has length L . The graph of such a basic cell is depicted in Figure 3.12.

Figure 3.13 shows the complete decimation-in-frequency FFT algorithm for $N = 8$. It is important to note that, in this algorithm, the input sequence $x(n)$ is in normal ordering and the output sequence $X(k)$ is in bit-reversed ordering. Also, comparing Figures 3.11 and 3.13, it is interesting to see that one graph is the transpose of the other.

3.5.3 Radix-4 algorithm

If $N = 2^{2l}$, then, instead of using radix-2 algorithms, we can use radix-4 FFT algorithms, which can give us additional economy in the required number of complex multiplications.

The derivation of the radix-4 algorithms parallels those of the radix-2 algorithms. If we use decimation-in-time, then a length- N sequence is divided into four sequences of length $N/4$ such that

$$\begin{aligned}
 X(k) &= \sum_{m=0}^{(N/4)-1} x(4m)W_N^{4mk} + \sum_{m=0}^{(N/4)-1} x(4m+1)W_N^{(4m+1)k} \\
 &\quad + \sum_{m=0}^{(N/4)-1} x(4m+2)W_N^{(4m+2)k} + \sum_{m=0}^{(N/4)-1} x(4m+3)W_N^{(4m+3)k} \\
 &= \sum_{m=0}^{(N/4)-1} x(4m)W_{N/4}^{mk} + W_N^k \sum_{m=0}^{(N/4)-1} x(4m+1)W_{N/4}^{mk} \\
 &\quad + W_N^{2k} \sum_{m=0}^{(N/4)-1} x(4m+2)W_{N/4}^{mk} + W_N^{3k} \sum_{m=0}^{(N/4)-1} x(4m+3)W_{N/4}^{mk} \\
 &= \sum_{l=0}^3 W_N^{lk} \sum_{m=0}^{(N/4)-1} x(4m+l)W_{N/4}^{mk}. \tag{3.162}
 \end{aligned}$$

We can rewrite Equation (3.162) as

$$X(k) = \sum_{l=0}^3 W_N^{lk} F_l(k), \tag{3.163}$$

where each $F_l(k)$ can be computed using four DFTs of length $N/16$, as shown below:

$$\begin{aligned}
 F_l(k) &= \sum_{m=0}^{(N/4)-1} x(4m+l)W_{N/4}^{mk} \\
 &= \sum_{q=0}^{(N/16)-1} x(16q+l)W_{N/4}^{4qk} + W_{N/4}^k \sum_{q=0}^{(N/16)-1} x(16q+4+l)W_{N/4}^{4qk} \\
 &\quad + W_{N/4}^{2k} \sum_{q=0}^{(N/16)-1} x(16q+8+l)W_{N/4}^{4qk} + W_{N/4}^{3k} \sum_{q=0}^{(N/16)-1} x(16q+12+l)W_{N/4}^{4qk} \\
 &= \sum_{r=0}^3 W_{N/16}^{rk} \sum_{q=0}^{(N/16)-1} x(16q+4r+l)W_{N/16}^{qk}. \tag{3.164}
 \end{aligned}$$

This process is recursively applied until we have to compute $N/4$ DFTs of length 4. From Equation (3.163), we can see that the basic cell implements the equations below when computing a DFT $S(k)$ of length L using four DFTs of length $L/4$, $S_l(k)$, $l = 0, 1, 2, 3$:

$$\left. \begin{aligned} S(k) &= \sum_{l=0}^3 W_L^{lk} S_l(k) \\ S(k + (L/4)) &= \sum_{l=0}^3 W_L^{l[k+(L/4)]} S_l(k) = \sum_{l=0}^3 W_L^{lk} (-j)^l S_l(k) \\ S(k + (L/2)) &= \sum_{l=0}^3 W_L^{l[k+(L/2)]} S_l(k) = \sum_{l=0}^3 W_L^{lk} (-1)^l S_l(k) \\ S(k + (3L/4)) &= \sum_{l=0}^3 W_L^{l[k+(3L/4)]} S_l(k) = \sum_{l=0}^3 W_L^{lk} (j)^l S_l(k) \end{aligned} \right\}. \quad (3.165)$$

The corresponding graph for the radix-4 butterfly is shown in Figure 3.14. As an illustration of the radix-4 algorithm, Figure 3.15 shows a sketch of the computation of a DFT of length 64 using a radix-4 FFT.

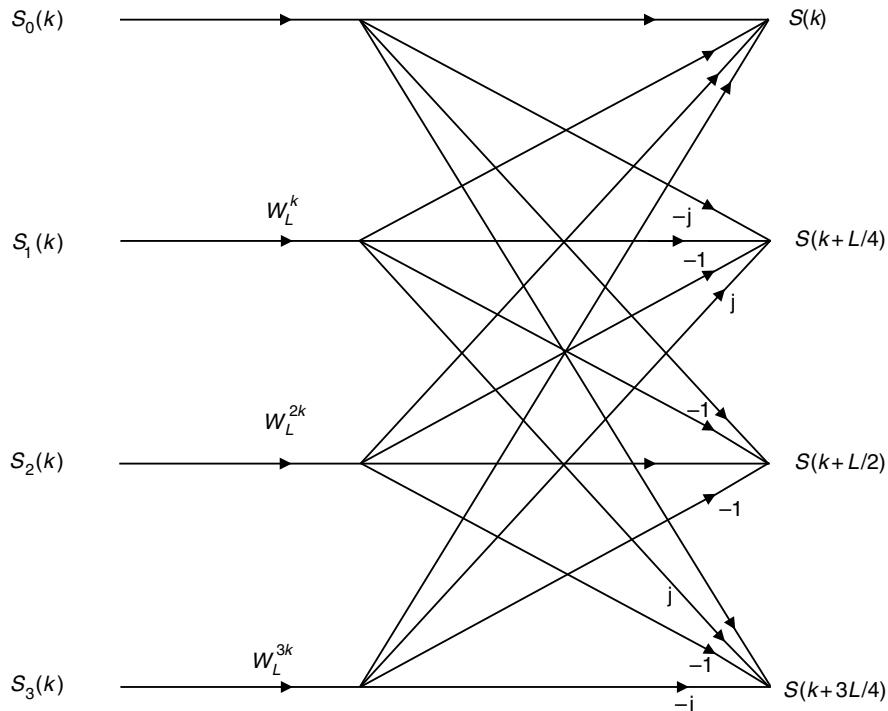


Fig. 3.14. Basic cell of the radix-4 FFT algorithm.

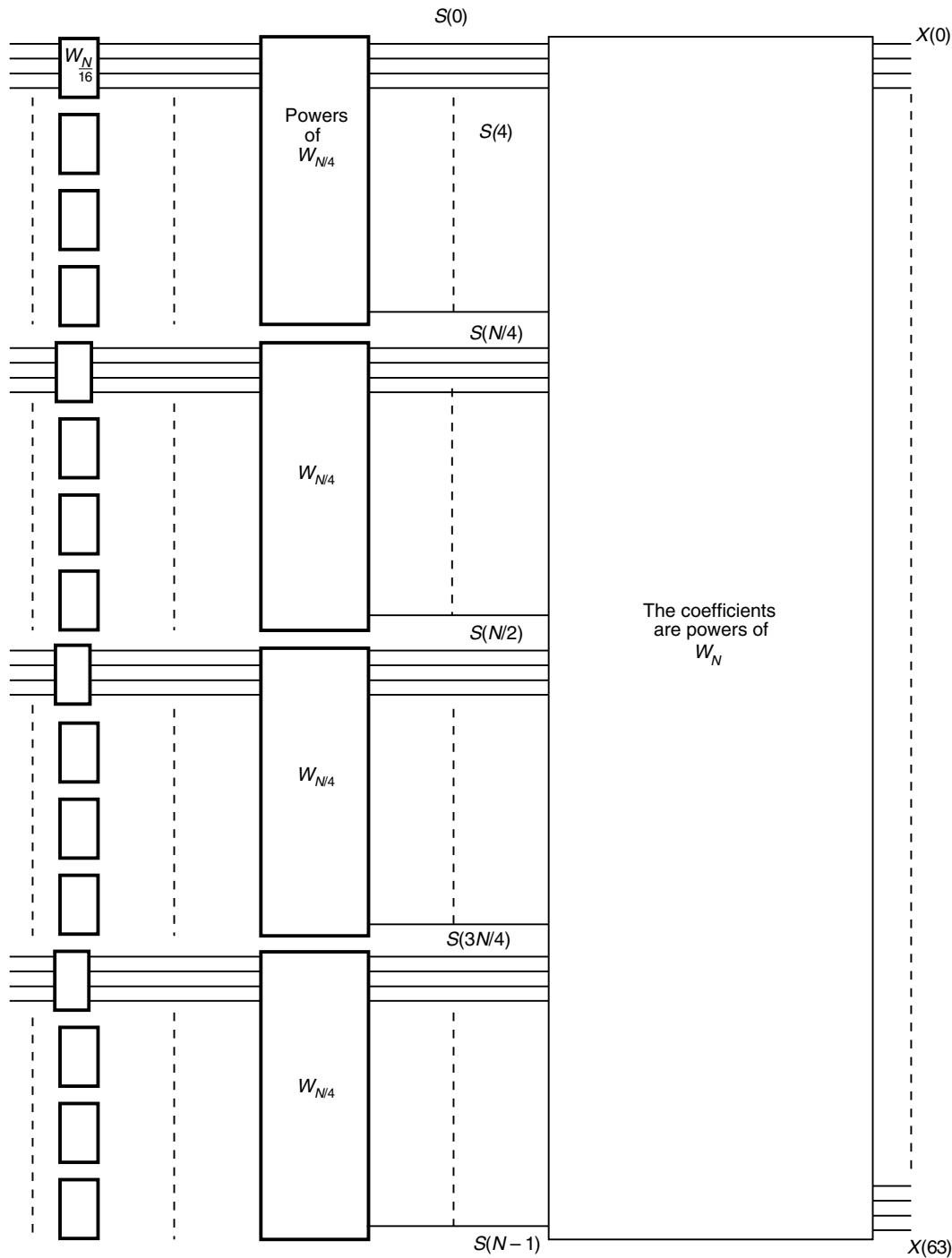


Fig. 3.15. Sketch of a length-64 DFT using a radix-4 FFT algorithm.

As can be deduced from Figure 3.14 and Equation (3.164), at each stage of the application of the radix-4 FFT algorithm we need N complex multiplications and $3N$ complex additions, giving a total number of complex operations of

$$\mathcal{M}(N) = N \log_4 N = \frac{N}{2} \log_2 N \quad (3.166)$$

$$\mathcal{A}(N) = 3N \log_4 N = \frac{3N}{2} \log_2 N. \quad (3.167)$$

Apparently, the radix-4 algorithms do not present any advantage when compared with radix-2 algorithms. However, the number of additions in the radix-4 basic cell can be decreased if we note from Figure 3.14 and Equation (3.165) that the quantities

$$\left. \begin{array}{l} W_L^0 S_0(k) + W_L^{2k} S_2(k) \\ W_L^0 S_0(k) - W_L^{2k} S_2(k) \\ W_L^k S_1(k) + W_L^{3k} S_3(k) \\ W_L^k S_1(k) - W_L^{3k} S_3(k) \end{array} \right\} \quad (3.168)$$

are each computed twice, unnecessarily. By exploiting this fact, the more economical basic cell for the radix-4 algorithm shown in Figure 3.16 results, and we decrease the number of complex additions to $2N$ per stage, instead of $3N$.

The number of multiplications can be further decreased if we do not consider the multiplications by W_N^0 . There is one of them in each basic cell, and three more of them in the basic cells corresponding to the index $k = 0$. Since there are $\log_4 N$ stages, we have, in the former case, $(N/4) \log_4 N$ elements W_N^0 , while the number of elements corresponding to $k = 0$ is $3(1+4+16+\dots+N/4) = (N-1)$. Therefore, the total number of multiplications is given by

$$\mathcal{M}(N) = N \log_4 N - \frac{N}{4} \log_4 N - N + 1 = \frac{3}{8} N \log_2 N - N + 1. \quad (3.169)$$

Some additional trivial multiplications can also be detected, as shown by Nussbaumer (1982). If we then compare Equations (3.147) and (3.169), we note that the radix-4 algorithm can be more economical, in terms of the number of overall complex multiplications, than the radix-2 algorithm.

It is important to point out that, in general, the shorter the length of the DFTs of the basic cell of a FFT algorithm, the more efficient it is. The exceptions to this rule are the radix-4, -8, -16, ..., algorithms, where we can obtain a smaller number of multiplications than in radix-2 algorithms.

Although the radix-4 algorithm introduced here was based on the decimation-in-time approach, a similar algorithm based on the decimation-in-frequency method could be readily obtained.

Example 3.9. Derive the butterfly of the radix-3 DFT, exploiting the possible savings in the number of multiplications and additions.

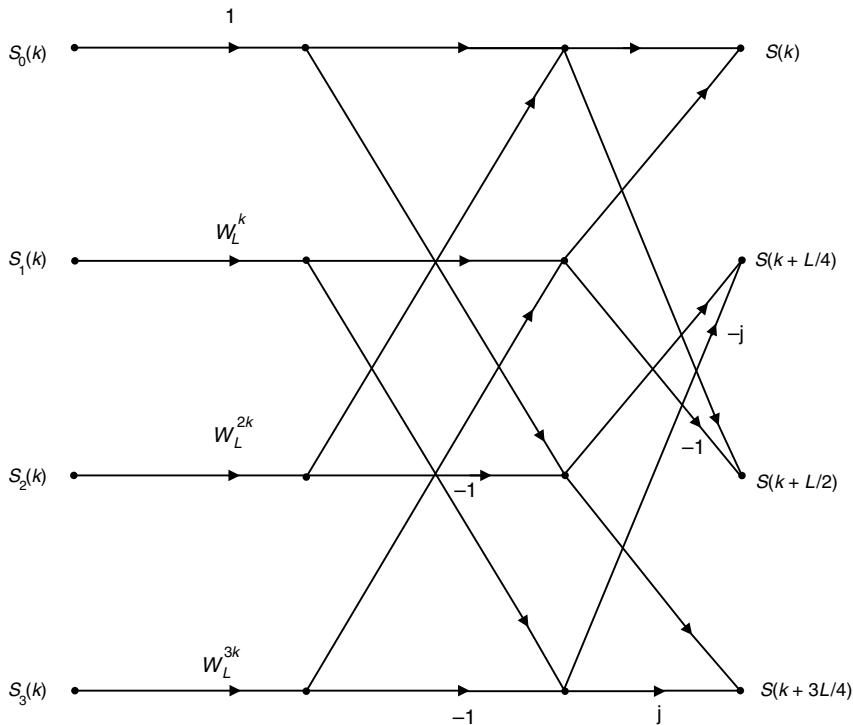


Fig. 3.16. More efficient basic cell of the radix-4 FFT algorithm.

Solution

$$\begin{aligned}
 X(k) &= \sum_{n=0}^2 x(n) W_N^{nk} \\
 &= x(0) + W_3^k x(1) + W_3^{2k} x(2) \\
 &= x(0) + e^{-j(2\pi/3)k} x(1) + e^{-j(4\pi/3)k} x(2).
 \end{aligned} \tag{3.170}$$

Using the above equation and discounting the trivial multiplications, one can compute the radix-3 butterfly using six additions (two for each value of k) and four complex multiplications (two for $k = 1$ and two for $k = 2$).

By further developing Equation (3.170), we have

$$X(0) = x(0) + x(1) + x(2) \tag{3.171}$$

$$\begin{aligned}
 X(1) &= x(0) + e^{-j2\pi/3} x(1) + e^{-j4\pi/3} x(2) \\
 &= x(0) + e^{-j2\pi/3} x(1) + e^{j2\pi/3} x(2) \\
 &= x(0) + W_3 x(1) + W_3^{-1} x(2)
 \end{aligned} \tag{3.172}$$

$$X(2) = x(0) + e^{-j4\pi/3} x(1) + e^{-j8\pi/3} x(2)$$

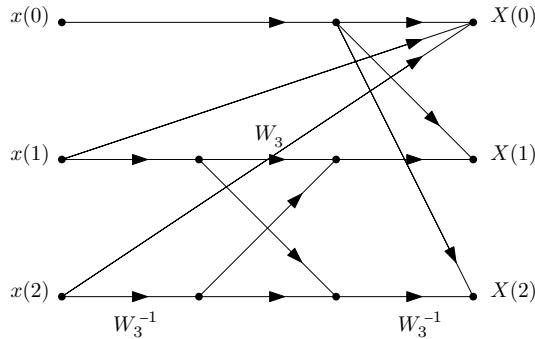


Fig. 3.17. Efficient butterfly for the radix-3 DFT in Example 3.9.

$$\begin{aligned}
 &= x(0) + e^{j2\pi/3}x(1) + e^{4\pi/3}x(2) \\
 &= x(0) + e^{j2\pi/3} (x(1) + e^{j2\pi/3}x(2)) \\
 &= x(0) + W_3^{-1} (x(1) + W_3^{-1}x(2)). \tag{3.173}
 \end{aligned}$$

From Equations (3.171) to (3.173), we can compute the radix-3 butterfly using the graph in Figure 3.17, which uses six additions and three complex multiplications, a saving of one complex multiplication relative to the solution in Equation (3.170). \triangle

3.5.4 Algorithms for arbitrary values of N

Efficient algorithms for the computation of DFTs having a generic length N are possible provided that N is not prime (Singleton, 1969; Rabiner, 1979). In such cases, we can decompose N as a product of factors

$$N = N_1 N_2 N_3 \cdots N_l = N_1 N_{2 \rightarrow l}, \tag{3.174}$$

where $N_{2 \rightarrow l} = N_2 N_3 \cdots N_l$. We can then initially divide the input sequence into N_1 sequences of length $N_{2 \rightarrow l}$, thus writing the DFT of $x(n)$ as

$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\
 &= \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m) W_N^{m N_1 k} + \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + 1) W_N^{m N_1 k + k} + \cdots \\
 &\quad + \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + N_1 - 1) W_N^{m N_1 k + (N_1 - 1)k}
 \end{aligned}$$

$$\begin{aligned}
&= \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m) W_{N_{2 \rightarrow l}}^{mk} + W_N^k \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + 1) W_{N_{2 \rightarrow l}}^{mk} \\
&\quad + W_N^{2k} \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + 2) W_{N_{2 \rightarrow l}}^{mk} + \dots \\
&\quad + W_N^{(N_1-1)k} \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + N_1 - 1) W_{N_{2 \rightarrow l}}^{mk} \\
&= \sum_{r=0}^{N_1-1} W_N^{rk} \sum_{m=0}^{N_{2 \rightarrow l}-1} x(N_1 m + r) W_{N_{2 \rightarrow l}}^{mk}.
\end{aligned} \tag{3.175}$$

This equation can be interpreted as being the computation of a length- N DFT using N_1 DFTs of length $N_{2 \rightarrow l}$. We then need $N_1 N_{2 \rightarrow l}^2$ complex multiplications to compute the N_1 mentioned DFTs, plus $N(N_1 - 1)$ complex multiplications to compute the products of W_N^{rk} with the N_1 DFTs. We can continue this process by computing each of the N_1 DFTs of length $N_{2 \rightarrow l}$ using N_2 DFTs of length $N_{3 \rightarrow l}$, where $N_{3 \rightarrow l} = N_3 N_4 \cdots N_l$, and so on, until all the DFTs have length N_l .

It can be shown that in such a case the total number of complex multiplications is given by (see Exercise 3.24)

$$\mathcal{M}(N) = N(N_1 + N_2 + \dots + N_{l-1} + N_l - l). \tag{3.176}$$

For example, if $N = 63 = 3 \times 3 \times 7$ we have that $\mathcal{M}(N) = 63(3 + 3 + 7 - 3) = 630$. It is interesting to note that in order to compute a length-64 FFT we need only 384 complex multiplications if we use a radix-2 algorithm. This example reinforces the idea that we should, as a rule of thumb, divide N into factors as small as possible. In practice, whenever possible, the input sequences are zero padded to force N to be a power of 2. As seen in the example above, this usually leads to an overall economy in the number of multiplications.

3.5.5 Alternative techniques for determining the DFT

The algorithms for efficient computation of the DFT presented in the previous subsections are generically called FFT algorithms. They were, for a long time, the only known methods to enable the efficient computation of long DFTs. In 1976, however, Winograd showed that there are algorithms with smaller complexity than the FFTs. These algorithms are based on convolution calculations exploring “number-theoretic” properties of the addresses of the data (McClellan & Rader, 1979). These algorithms are denominated Winograd Fourier transform (WFT) algorithms.

In terms of practical implementations, the WFT has a smaller number of complex multiplications than the FFT, at the expense of a more complex algorithm. This gives an advantage to the WFT in many cases. However, the FFTs are more modular, which is an advantage

in hardware implementations, especially in very large scale integration (VLSI). The main disadvantage of the WFT in this case is the complexity of the control path. Therefore, when implemented in special-purpose DSPs the FFTs have a clear advantage. This is so because multiplications are not a problem in such processors, and the more complex algorithms of the WFT make it slower than the FFTs in most cases.

Another class of techniques for the computation of convolutions and DFTs is given by the number-theoretic transform (NTT), which explores number-theoretic properties of the data. NTT techniques have practical implementations in machines with modular arithmetic. Also, they are useful for computations using hardware based on residue arithmetic (McClellan & Rader, 1979; Elliott & Rao, 1982; Nussbaumer, 1982).

3.6 Other discrete transforms

As seen in the previous sections, the DFT is a natural discrete-frequency representation for finite-length discrete signals, consisting of uniformly spaced samples of the Fourier transform. The direct and inverse DFTs can be expressed in matrix form as given by Equations (3.39) and (3.40), repeated here for the reader's convenience:

$$\mathbf{X} = \mathbf{W}_N \mathbf{x} \quad (3.177)$$

$$\mathbf{x} = \frac{1}{N} \mathbf{W}_N^* \mathbf{X}, \quad (3.178)$$

where $\{\mathbf{W}_N\}_{ij} = W_N^{ij}$.

Now, let \mathbf{A}_N be a matrix of dimensions $N \times N$, such that

$$\mathbf{A}_N^{-1} = \gamma \mathbf{A}_N^{*\top}, \quad (3.179)$$

where the superscript T and asterisk denote the matrix transposition and complex conjugate operations respectively, and γ is a constant. Using \mathbf{A}_N , we can generalize the definition in Equations (3.177) and (3.178) to

$$\mathbf{X} = \mathbf{A}_N \mathbf{x} \quad (3.180)$$

$$\mathbf{x} = \gamma \mathbf{A}_N^{*\top} \mathbf{X}. \quad (3.181)$$

Equations (3.180) and (3.181) represent several discrete transforms which are frequently employed in signal processing applications. In Section 3.6.1 we will see that Parseval's theorem is also valid for discrete transforms in general, and we discuss some of its implications.

3.6.1 Discrete transforms and Parseval's theorem

Before proceeding, it is interesting to define some important operations between two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{C}^N$:

- The inner product between \mathbf{v}_2 and \mathbf{v}_1 is defined as

$$\langle \mathbf{v}_2, \mathbf{v}_1 \rangle = \mathbf{v}_1^{*\top} \mathbf{v}_2. \quad (3.182)$$

- The norm of vector \mathbf{v} is defined as

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^{*\top} \mathbf{v}. \quad (3.183)$$

- The angle θ between two vectors \mathbf{v}_2 and \mathbf{v}_1 is defined as

$$\cos \theta = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} = \frac{\mathbf{v}_1^{*\top} \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}. \quad (3.184)$$

Given two length- N signals $x_1(n)$ and $x_2(n)$ (\mathbf{x}_1 and \mathbf{x}_2 in vector form), and their respective transforms $X_1(k)$ and $X_2(k)$ (\mathbf{X}_1 and \mathbf{X}_2 in vector form), according to Equations (3.180) and (3.181), we have that

$$\begin{aligned} \sum_{k=0}^{N-1} X_1(k) X_2^*(k) &= \mathbf{X}_2^{*\top} \mathbf{X}_1 \\ &= (\mathbf{A}_N \mathbf{x}_2)^{*\top} \mathbf{A}_N \mathbf{x}_1 \\ &= \mathbf{x}_2^{*\top} \mathbf{A}_N^{*\top} \mathbf{A}_N \mathbf{x}_1 \\ &= \mathbf{x}_2^{*\top} \left(\frac{1}{\gamma} \mathbf{A}_N^{-1} \right) \mathbf{A}_N \mathbf{x}_1 \\ &= \frac{1}{\gamma} \mathbf{x}_2^{*\top} \mathbf{x}_1 \\ &= \frac{1}{\gamma} \sum_{n=0}^{N-1} x_1(n) x_2^*(n). \end{aligned} \quad (3.185)$$

The above equation when $\gamma = 1/N$ is equivalent to Parseval's relation in Equation (3.87). If $x_1(n) = x_2(n) = x(n)$, the Equation (3.185) becomes

$$\|\mathbf{X}\|^2 = \frac{1}{\gamma} \|\mathbf{x}\|^2. \quad (3.186)$$

An interesting property of transforms as defined by Equations (3.180) and (3.181) is related to the angle between two vectors as defined by Equation (3.184). We have that the

angles θ_x between \mathbf{x}_1 and \mathbf{x}_2 and θ_X between their transforms \mathbf{X}_1 and \mathbf{X}_2 satisfy

$$\begin{aligned}\cos \theta_x &= \frac{\mathbf{x}_1^{*T} \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} \\ &= \frac{(\gamma \mathbf{A}_N^{*T} \mathbf{X}_1)^{*T} (\gamma \mathbf{A}_N^{*T} \mathbf{X}_2)}{\sqrt{\gamma} \|\mathbf{X}_1\| \sqrt{\gamma} \|\mathbf{X}_2\|} \\ &= \frac{\gamma \mathbf{X}_1^{*T} \mathbf{A}_N \gamma \left(\frac{1}{\gamma} \mathbf{A}_N^{-1} \right) \mathbf{X}_2}{\gamma \|\mathbf{X}_1\| \|\mathbf{X}_2\|} \\ &= \frac{\mathbf{X}_1^{*T} \mathbf{X}_2}{\|\mathbf{X}_1\| \|\mathbf{X}_2\|} \\ &= \cos \theta_X;\end{aligned}\tag{3.187}$$

that is, transforms do not change the angles between vectors.

A special case of transforms is when $\gamma = 1$. These are referred to as unitary transforms. For unitary transforms, Equation (3.186) means that the energy in the transform domain is equal to the energy in the time domain, or, equivalently, unitary transforms do not change the length of vectors. If we consider this property together with the one expressed by Equation (3.187), then we see that neither angles nor lengths are changed under unitary transforms. This is equivalent to saying that unitary transforms are just rotations in \mathbb{C}^N .

Note that $\gamma = 1/N$ for the DFT as defined in Equations (3.177) and (3.178). A unitary definition of the DFT would be

$$\mathbf{X} = \frac{1}{\sqrt{N}} \mathbf{W}_N \mathbf{x}\tag{3.188}$$

$$\mathbf{x} = \frac{1}{\sqrt{N}} \mathbf{W}_N^* \mathbf{X},\tag{3.189}$$

where $\{\mathbf{W}_N\}_{ij} = W_N^{ij}$. This unitary version is often used when one needs strict energy conservation between time and frequency domains.

3.6.2 Discrete transforms and orthogonality

Two vectors \mathbf{v}_1 and \mathbf{v}_2 are orthogonal if their inner product is null; that is:

$$\langle \mathbf{v}_2, \mathbf{v}_1 \rangle = \mathbf{v}_1^{*T} \mathbf{v}_2 = 0.\tag{3.190}$$

Equation (3.190), together with Equation (3.184), implies that the angle between two orthogonal vectors is $\theta = \pi/2$.

The transforms as defined in Equation (3.179) have an interesting interpretation when we consider the orthogonality concept.

We can express matrix \mathbf{A}_N as composed by its rows:

$$\mathbf{A}_N = \begin{bmatrix} \mathbf{a}_0^* \\ \mathbf{a}_1^* \\ \vdots \\ \mathbf{a}_{N-1}^* \end{bmatrix}, \quad (3.191)$$

where \mathbf{a}_k^* is the k th row of matrix \mathbf{A}_N .

As we have seen above, the transform definition in Equations (3.180) and (3.181) implies Equation (3.179), which is equivalent to

$$\mathbf{A}_N \mathbf{A}_N^{*T} = \frac{1}{\gamma} \mathbf{I}_N. \quad (3.192)$$

Expressing \mathbf{A}_N as in Equation (3.191), Equation (3.192) becomes

$$\begin{aligned} \mathbf{A}_N \mathbf{A}_N^{*T} &= \begin{bmatrix} \mathbf{a}_0^* \\ \mathbf{a}_1^* \\ \vdots \\ \mathbf{a}_{N-1}^* \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \cdots & \mathbf{a}_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}_0^* \mathbf{a}_0 & \mathbf{a}_0^* \mathbf{a}_1 & \cdots & \mathbf{a}_0^* \mathbf{a}_{N-1} \\ \mathbf{a}_1^* \mathbf{a}_0 & \mathbf{a}_1^* \mathbf{a}_1 & \cdots & \mathbf{a}_1^* \mathbf{a}_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{N-1}^* \mathbf{a}_0 & \mathbf{a}_{N-1}^* \mathbf{a}_1 & \cdots & \mathbf{a}_{N-1}^* \mathbf{a}_{N-1} \end{bmatrix} \\ &= \frac{1}{\gamma} \mathbf{I}_N \\ &= \frac{1}{\gamma} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \end{aligned} \quad (3.193)$$

which is the same as saying that

$$\mathbf{a}_k^* \mathbf{a}_l = \frac{1}{\gamma} \delta(k - l). \quad (3.194)$$

Therefore, we can conclude that the rows of a transform matrix \mathbf{A}_N are orthogonal; that is, the angle between any pair of distinct rows is equal to $\pi/2$. In addition, the constant γ is such that

$$\gamma = \frac{1}{\mathbf{a}_k^* \mathbf{a}_k} = \frac{1}{\|\mathbf{a}_k\|^2}. \quad (3.195)$$

Now, expressing the direct transform in Equation (3.180) as a function of the rows of \mathbf{A}_N , we have that

$$\mathbf{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0^{*\top} \\ \mathbf{a}_1^{*\top} \\ \vdots \\ \mathbf{a}_{N-1}^{*\top} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{a}_0^{*\top} \mathbf{x} \\ \mathbf{a}_1^{*\top} \mathbf{x} \\ \vdots \\ \mathbf{a}_{N-1}^{*\top} \mathbf{x} \end{bmatrix} \quad (3.196)$$

that is:

$$X(k) = \mathbf{a}_k^{*\top} \mathbf{x} = \langle \mathbf{x}, \mathbf{a}_k \rangle. \quad (3.197)$$

Likewise, expressing the inverse transform in Equation (3.181) as a function of the columns of $\mathbf{A}_N^{*\top}$, we have that

$$\begin{aligned} \mathbf{x} &= \gamma [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \cdots \quad \mathbf{a}_{N-1}] \mathbf{X} \\ &= \gamma [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \cdots \quad \mathbf{a}_{N-1}] \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \\ &= \sum_{k=0}^{N-1} \gamma X(k) \mathbf{a}_k. \end{aligned} \quad (3.198)$$

The above equation means that a transform expresses a vector \mathbf{x} as a linear combination of N orthogonal vectors \mathbf{a}_k , for $k = 0, 1, \dots, (N-1)$. The coefficients of this linear combination are proportional to the transform coefficients $X(k)$. Furthermore, from Equation (3.197), $X(k)$ is equal to the inner product between the vector \mathbf{x} and the vector \mathbf{a}_k .

We take this interpretation one step further by observing that orthogonality of the vectors \mathbf{a}_k implies that γ is given by Equation (3.195). Replacing this value of γ and the value of $X(k)$ from Equation (3.197) in Equation (3.198), we get

$$\begin{aligned} \mathbf{x} &= \sum_{k=0}^{N-1} \underbrace{\left(\frac{1}{\|\mathbf{a}_k\|^2} \right)}_{\gamma} \underbrace{\left(\mathbf{a}_k^{*\top} \mathbf{x} \right)}_{X(k)} \mathbf{a}_k \\ &= \sum_{k=0}^{N-1} \left(\frac{\mathbf{a}_k^{*\top} \mathbf{x}}{\|\mathbf{a}_k\|} \right) \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|} \\ &= \sum_{k=0}^{N-1} \left\langle \mathbf{x}, \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|} \right\rangle \frac{\mathbf{a}_k}{\|\mathbf{a}_k\|}. \end{aligned} \quad (3.199)$$

From Equation (3.199) one can interpret the transform as a representation of a signal using an orthogonal basis of vectors \mathbf{a}_k , for $k = 0, 1, \dots, (N-1)$. The transform coefficient

$X(k)$ is proportional to the component of vector \mathbf{x} in the direction of the basis vector \mathbf{a}_k , which corresponds to the projection \mathbf{p} of \mathbf{x} on the unit-norm vector $\mathbf{a}_k/\|\mathbf{a}_k\|$; that is, $\mathbf{p} = \langle \mathbf{x}, \mathbf{a}_k/\|\mathbf{a}_k\| \rangle$.

If the transform is unitary, then $\gamma = 1$, which implies, from Equation (3.195), that $\|\mathbf{a}_k\| = 1$. In this case, vectors \mathbf{a}_k are said to form an orthonormal basis, such that Equation (3.199) becomes

$$\mathbf{x} = \sum_{k=0}^{N-1} \langle \mathbf{x}, \mathbf{a}_k \rangle \mathbf{a}_k; \quad (3.200)$$

that is, the transform coefficient $X(k)$ is equal to the projection of vector \mathbf{x} on the unit-norm basis vector \mathbf{a}_k .

The canonical basis is formed by the vectors \mathbf{e}_k , for $k = 0, 1, \dots, (N - 1)$, such that

$$\begin{bmatrix} \mathbf{e}_0^T \\ \mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_{N-1}^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \quad (3.201)$$

The vectors \mathbf{e}_k form an orthonormal set that is an orthonormal basis of \mathbb{C}^N . In this basis, the n th component of any vector \mathbf{x} can be expressed as

$$x(n) = \mathbf{e}_n^* \mathbf{x} = \langle \mathbf{x}, \mathbf{e}_n \rangle. \quad (3.202)$$

From equation (3.202) we see that the samples of \mathbf{x} are its projections (or coordinates) on the canonical basis. Since all orthonormal basis are rotations of one another, and any unitary transform is a projection on an orthonormal basis, this confirms the statement made at the end of Section 3.6.1, that all orthonormal transforms are just rotations in \mathbb{C}^N .

In the remainder of Section 3.6 we describe some of the most commonly employed discrete transforms.

3.6.3 Discrete cosine transform

The length- N discrete cosine transform (DCT) of a signal $x(n)$ can be defined (Ahmed *et al.*, 1974) as

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\pi(n + \frac{1}{2})k}{N} \right], \quad \text{for } 0 \leq k \leq N - 1, \quad (3.203)$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{for } k = 0 \\ \sqrt{\frac{2}{N}}, & \text{for } 1 \leq k \leq N - 1. \end{cases} \quad (3.204)$$

Accordingly, the inverse DCT is given by

$$x(n) = \sum_{k=0}^{N-1} \alpha(k) C(k) \cos \left[\frac{\pi(n + \frac{1}{2})k}{N} \right], \quad \text{for } 0 \leq n \leq N - 1. \quad (3.205)$$

Note that the DCT is a real transform; that is, it maps a real signal into real DCT coefficients. From Equations (3.203)–(3.205), we can define the DCT matrix \mathbf{C}_N by

$$\{\mathbf{C}_N\}_{kn} = \alpha(k) \cos \left[\frac{\pi(n + \frac{1}{2})k}{N} \right], \quad (3.206)$$

and the matrix form of the DCT becomes

$$\mathbf{c} = \mathbf{C}_N \mathbf{x} \quad (3.207)$$

$$\mathbf{x} = \mathbf{C}_N^T \mathbf{c}. \quad (3.208)$$

Note that, for the above equations to be valid, $\mathbf{C}_N^{-1} = \mathbf{C}_N^T$, which, together with the fact that \mathbf{C}_N is a real matrix, implies that the matrix \mathbf{C}_N is unitary. As seen in Section 3.6.1, the Parseval theorem is valid, and the DCT can be considered a rotation in \mathbb{C}^N (and also in \mathbb{R}^N , since it is a real transform).

The DCT basis functions \mathbf{c}_k are the conjugate transpose of the rows of \mathbf{C}_N (see Equation (3.191)) and are thus given by

$$\begin{aligned} \mathbf{c}_k(n) &= \{\mathbf{C}_N\}_{kn}^* \\ &= \alpha(k) \cos \left[\frac{\pi(n + \frac{1}{2})k}{N} \right] \\ &= \alpha(k) \cos \left(\frac{2\pi}{2N} kn + \frac{\pi}{2N} k \right), \end{aligned} \quad (3.209)$$

which are sinusoids of frequencies $\omega_k = (2\pi/2N)kn$, for $k = 0, 1, \dots, (N - 1)$. Thus, the DCT, besides being a rotation in \mathbb{R}^N , decomposes a signal as a sum of N real sinusoids of frequencies given by ω_k above.

The DCT enjoys another very important property: when applied to signals such as voice and video, most of the transform energy is concentrated in few coefficients. For example, in Figure 3.18a, we can see a digital signal $x(n)$ corresponding to one line of a digitized television signal, and in Figure 3.18b, we show its DCT coefficients $C(k)$. It can be seen that the energy of the signal is more or less evenly spread among its samples, while it is mostly concentrated in the first transform coefficients. Owing to this property, the DCT is widely used in video compression schemes, because the coefficients with lowest energy can be discarded during transmission without introducing significant distortion in the original

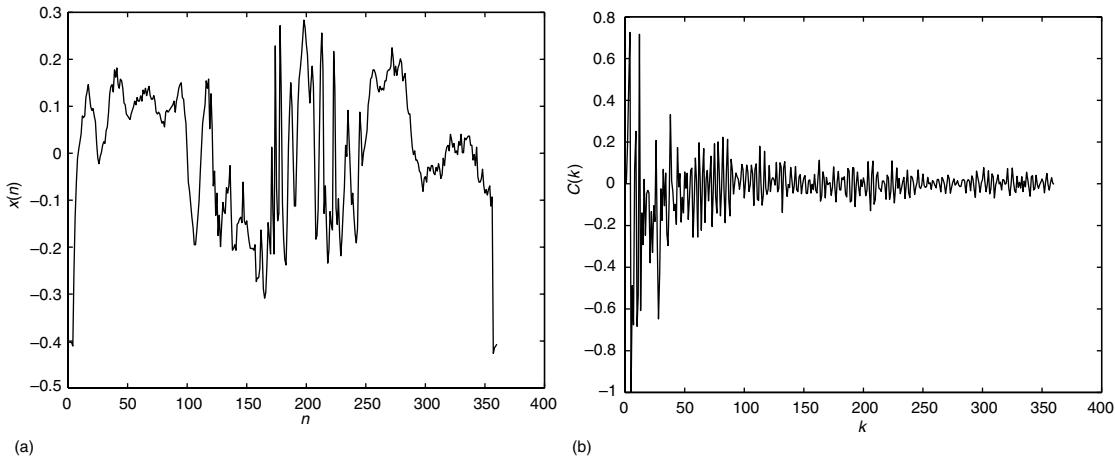


Fig. 3.18. The DCT of a video signal: (a) discrete-time video signal $x(n)$; (b) DCT of $x(n)$.

signal (Bhaskaran & Konstantinides, 1997). In fact, the DCT is part of most of the digital video broadcasting systems in operation in the world (Whitaker, 1999).

Since the DCT is based on sinusoids and $\cos(x) = \frac{1}{2}(e^{jx} + e^{-jx})$, then in the worst case the DCT of $x(n)$ can be computed using one length- $2N$ DFT. This can be deduced by observing that

$$\begin{aligned} \cos\left[\frac{\pi(n + \frac{1}{2})k}{N}\right] &= \cos\left(\frac{2\pi}{2N}kn + \frac{\pi}{2N}k\right) \\ &= \frac{1}{2} \left\{ e^{j[(2\pi/2N)kn + (\pi/2N)k]} + e^{-j[(2\pi/2N)kn + (\pi/2N)k]} \right\} \\ &= \frac{1}{2} \left(W_{2N}^{k/2} W_{2N}^{kn} + W_{2N}^{-k/2} W_{2N}^{-kn} \right), \end{aligned} \quad (3.210)$$

which implies that

$$\begin{aligned} C(k) &= \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(n + \frac{1}{2})k}{N}\right] \\ &= \frac{1}{2} \left(\alpha(k) W_{2N}^{k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \alpha(k) W_{2N}^{-k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{-kn} \right) \\ &= \frac{1}{2} \alpha(k) \left(W_{2N}^{k/2} \text{DFT}_{2N}\{\hat{x}(n)\}(k) + W_{2N}^{-k/2} \text{DFT}_{2N}\{\hat{x}(n)\}(-k) \right) \end{aligned} \quad (3.211)$$

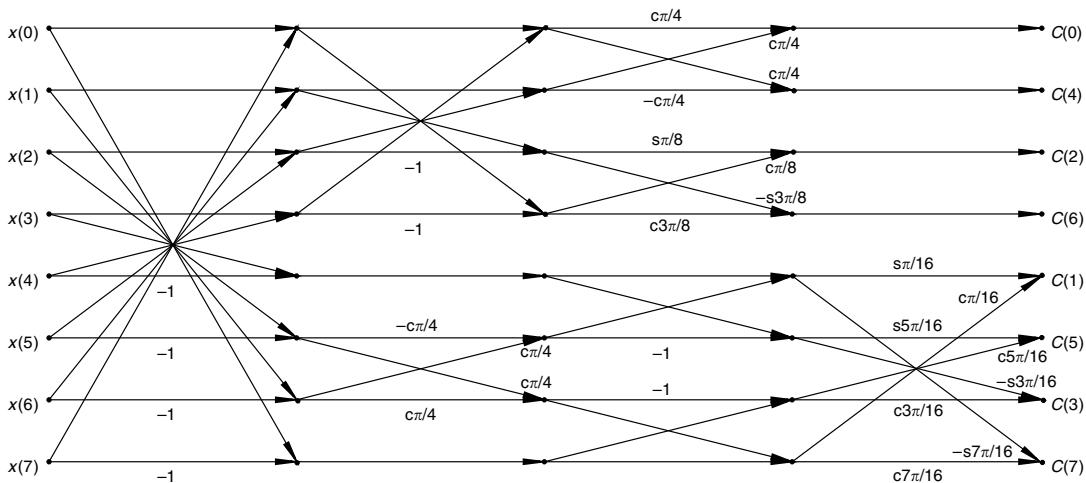


Fig. 3.19. A fast algorithm for the computation of a length-8 DCT. In this graph, $\cos x$ corresponds to $\cos x$ and $\sin x$ corresponds to $\sin x$.

for $0 \leq k \leq N - 1$, where $\hat{x}(n)$ is equal to $x(n)$ zero padded to length $2N$. Note that the second term in Equation (3.211) is equivalent to the first one computed at the index $-k$. Therefore, we actually need to compute only one length- $2N$ DFT.

One can see that the algorithm in Equation (3.211) has a complexity of one length- $2N$ DFT, plus the $2N$ multiplications by W_{2N}^k , which gives a total of $(2N + 2N \log_2 2N)$ complex multiplications.

However, there are other fast algorithms for the DCT which give a complexity of the order of $N \log_2 N$ real multiplications. One popular fast DCT algorithm is given in Chen *et al.* (1977). Its graph for $N = 8$ is shown in Figure 3.19.

This graph corresponds to the following factorization of \mathbf{C}_8 :

$$\mathbf{C}_8 = \mathbf{P}_8 \mathbf{A}_4 \mathbf{A}_3 \mathbf{A}_2 \mathbf{A}_1, \quad (3.212)$$

where

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\cos \frac{\pi}{4} & \cos \frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos \frac{\pi}{4} & \cos \frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.214)$$

$$\mathbf{A}_3 = \begin{bmatrix} \cos \frac{\pi}{4} & \cos \frac{\pi}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \cos \frac{\pi}{4} & -\cos \frac{\pi}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin \frac{\pi}{8} & \cos \frac{\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\sin \frac{3\pi}{8} & \cos \frac{3\pi}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad (3.215)$$

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin \frac{\pi}{16} & 0 & 0 & \cos \frac{\pi}{16} \\ 0 & 0 & 0 & 0 & 0 & \sin \frac{5\pi}{16} & \cos \frac{5\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sin \frac{3\pi}{16} & \cos \frac{3\pi}{16} & 0 \\ 0 & 0 & 0 & 0 & -\sin \frac{7\pi}{16} & 0 & 0 & \cos \frac{7\pi}{16} \end{bmatrix}, \quad (3.216)$$

and \mathbf{P}_8 is given by Equation (3.151), corresponding to the placement in normal ordering of the bit-reversed indexes of the output vector \mathbf{c} . Note that the operation corresponding to \mathbf{P}_8 is not shown in Figure 3.19.

After discounting the trivial multiplications, we have that, for a generic $N = 2^l$, the numbers of real multiplications $\mathcal{M}(N)$ and real additions $\mathcal{A}(N)$ in this fast implementation of the DCT are (Chen *et al.*, 1977)

$$\mathcal{M}(N) = N \log_2 N - \frac{3N}{2} + 4 \quad (3.217)$$

$$\mathcal{A}(N) = \frac{3N}{2}(\log_2 N - 1) + 2. \quad (3.218)$$

3.6.4 A family of sine and cosine transforms

The DCT is a particular case of a more general class of transforms, whose transform matrix has its rows composed of sines or cosines of increasing frequency. Such transforms have a

Table 3.1.

Definition of the even cosine and sine transforms.

	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
\mathbf{C}_{kn}^I	-1	1	1	0	0
\mathbf{C}_{kn}^{II}	0	1	0	0	$\frac{1}{2}$
\mathbf{C}_{kn}^{III}	0	0	1	$\frac{1}{2}$	0
\mathbf{C}_{kn}^{IV}	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
\mathbf{S}_{kn}^I	1	0	0	0	0
\mathbf{S}_{kn}^{II}	0	1	0	0	$-\frac{1}{2}$
\mathbf{S}_{kn}^{III}	0	0	1	$-\frac{1}{2}$	0
\mathbf{S}_{kn}^{IV}	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$

number of applications, including the design of filter banks (Section 9.9). In what follows, we present a brief description of the so-called even forms of the cosine and sine transforms.

There are four types of even cosine transform and four types of even sine transform. Their transform matrices are referred to as \mathbf{C}_N^{I-IV} for the cosine transforms and \mathbf{S}_N^{I-IV} for the sine transforms. Their (k, n) elements are given by

$$\{\mathbf{C}_N^x\}_{kn} = \sqrt{\frac{2}{N + \epsilon_1}} (\alpha_{N+\epsilon_1}(k))^{\epsilon_2} (\alpha_{N+\epsilon_1}(n))^{\epsilon_3} \cos\left[\frac{\pi(k + \epsilon_4)(n + \epsilon_5)}{N + \epsilon_1}\right] \quad (3.219)$$

$$\{\mathbf{S}_N^x\}_{kn} = \sqrt{\frac{2}{N + \epsilon_1}} (\alpha_{N+\epsilon_1}(k))^{\epsilon_2} (\alpha_{N+\epsilon_1}(n))^{\epsilon_3} \sin\left[\frac{\pi(k + \epsilon_4)(n + \epsilon_5)}{N + \epsilon_1}\right], \quad (3.220)$$

where

$$\alpha_\gamma(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k = 0 \text{ or } k = \gamma \\ 1, & \text{for } 1 \leq k \leq \gamma - 1. \end{cases} \quad (3.221)$$

The set $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5)$ defines the transform. Table 3.1 gives those values for all even cosine and sine transforms. One should note that all these transforms have fast algorithms and are unitary; that is, $\mathbf{A}^{-1} = \mathbf{A}^{*\top}$. For instance, the DCT defined in Section 3.6.3 is the same as \mathbf{C}_N^{II} .

3.6.5 Discrete Hartley transform

The discrete Hartley transform (DHT) can be viewed as a real counterpart of the DFT when it is applied to real signals. The definition of a length- N direct DHT is (Bracewell, 1994; Olejniczak & Heydt, 1994)

$$H(k) = \sum_{n=0}^{N-1} x(n) \operatorname{cas}\left(\frac{2\pi}{N}kn\right), \quad \text{for } 0 \leq k \leq N-1, \quad (3.222)$$

where $\operatorname{cas} x = \cos x + \sin x$. Similarly, the inverse DHT is determined by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) \operatorname{cas}\left(\frac{2\pi}{N}kn\right), \quad \text{for } 0 \leq k \leq N-1. \quad (3.223)$$

The Hartley transform is attractive due to the following properties:

- If the input signal is real, the DHT is real.
- The DFT can be easily derived from the DHT and vice versa.
- It has efficient fast algorithms (Bracewell, 1984).
- It has a convolution-multiplication property.

The first property follows trivially from the definition of the DHT. The other properties can be derived from the relations between the DFT $X(k)$ and the DHT $H(k)$ of a real sequence $x(n)$, which are

$$H(k) = \operatorname{Re}\{X(k)\} - \operatorname{Im}\{X(k)\} \quad (3.224)$$

$$X(k) = \mathcal{E}\{H(k)\} - j\mathcal{O}\{H(k)\}, \quad (3.225)$$

where the operators $\mathcal{E}\{\cdot\}$ and $\mathcal{O}\{\cdot\}$ correspond to the even and odd parts respectively; that is:

$$\mathcal{E}\{H(k)\} = \frac{H(k) + H(-k)}{2} \quad (3.226)$$

$$\mathcal{O}\{H(k)\} = \frac{H(k) - H(-k)}{2}. \quad (3.227)$$

The convolution property (the fourth property above) can be stated more precisely as: given two arbitrary length- N sequences $x_1(n)$ and $x_2(n)$, the DHT of their length- N circular convolution $y(n)$ is given by (from Equations (3.224) and (3.62))

$$Y(k) = H_1(k)\mathcal{E}\{H_2(k)\} + H_1(-k)\mathcal{O}\{H_2(k)\}, \quad (3.228)$$

where $H_1(k)$ is the DHT of $x_1(n)$ and $H_2(k)$ is the DHT of $x_2(n)$. This result is especially useful when the sequence $x_2(n)$ is even, which leads to

$$Y(k) = H_1(k)H_2(k). \quad (3.229)$$

This equation only involves real functions, whereas Equation (3.62) relates complex functions to determine $Y(k)$. Therefore, in the case $x_2(n)$ is even, it is advantageous to use Equation (3.229) instead of Equation (3.62) for performing convolutions.

To conclude, we can say that the DHT, as a signal representation, is not as widely used in practice as the DFT or the DCT. However, it has been used in a large range of applications as a tool to perform fast convolutions (Bracewell, 1984, 1994), as well as a tool to compute FFTs and fast DCTs (Malvar, 1986, 1987).

3.6.6 Hadamard transform

The Hadamard transform, also known as the Walsh–Hadamard transform, is a transform that is not based on sinusoidal functions, unlike the other transforms seen so far. Instead, the elements of its transform matrix are either 1 or -1 . When $N = 2^n$, its transform matrix is defined by the following recursion (Harmuth, 1970; Elliott & Rao, 1982):

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{H}_n = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{bmatrix}. \quad (3.230)$$

From the above equations, it is easy to see that the Hadamard transform is unitary. For example, a length-8 Hadamard transform matrix is

$$\mathbf{H}_3 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \quad (3.231)$$

An important aspect of the Hadamard transform is that, since the elements of its transform matrix are only either 1 or -1 , the Hadamard transform needs no multiplications for its computation, leading to simple hardware implementations. Because of this fact, the Hadamard transform has been used in digital video schemes in the past, although its compression performance is not as high as that of the DCT (Jain, 1989). Nowadays, with the advent of specialized hardware to compute the DCT, the Hadamard transform is used in digital video only in specific cases. However, one important area of application of the Hadamard transform today is in code-division multiple access (CDMA) systems for mobile communications, where it is employed as channelization code in synchronous communications systems (Stüber, 1996).

The Hadamard transform also has a fast algorithm. For example, the matrix \mathbf{H}_3 can be factored as (Jain, 1989)

$$\mathbf{H}_3 = \mathbf{A}_8^3, \quad (3.232)$$

where

$$\mathbf{A}_8 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}. \quad (3.233)$$

Therefore, the number of additions of a fast Hadamard transform algorithm is of the order of $N \log_2 N$.

3.6.7 Other important transforms

Wavelet transforms

Wavelet transforms constitute a different class of transforms which have become very popular in recent years. They come from the area of functional analysis, and in digital signal processing are usually studied under the discipline of multirate signal processing. Wavelet transforms are studied in Chapter 10 of this book.

Karhunen–Loève transform

As seen in Section 3.6.3, the DCT is widely used in image and video compression because it has the ability to concentrate the energy of a signal in a few transform coefficients. A question that naturally arises is which is the transform that maximizes this energy concentration. Given a statistical distribution of an ensemble of signals, the optimum transform in terms of this energy compaction capability is the Karhunen–Loève transform (KLT) (Jain, 1989). It is defined as the transform that diagonalizes the autocorrelation matrix, as defined in Equation (1.234), of a discrete random process.

From the above, we see that there is a different KLT for each of the different signal statistics. However, it can be shown that the DCT approximates the KLT when the signals can be modeled as Gauss–Markov processes with correlation coefficients near to 1 (Jain, 1989). This is a reasonably good model for several useful signals; probably the best example of this is given by video signals. For them, the DCT is indeed approximately optimum in terms of energy compaction capability, which explains its widespread use.

3.7 Signal representations

In Chapters 1–3 we have dealt with several forms of signal representations, both continuous and discrete. We now summarize the main characteristics of those representations. We classify them in terms of both the time and frequency variables as continuous or discrete, as well as real, imaginary, or complex. Also, we classify the representations as being periodic or nonperiodic. The time–frequency relationship for each transform is shown in Figures 3.20–3.25.

3.7.1 Laplace transform

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st} dt \quad \longleftrightarrow \quad x(t) = \frac{1}{2\pi} e^{\sigma t} \int_{-\infty}^{\infty} X(\sigma + j\omega) e^{j\omega t} d\omega. \quad (3.234)$$

- Time domain: nonperiodic function of a continuous and real-time variable.
- Frequency domain: nonperiodic function of a continuous and complex frequency variable.

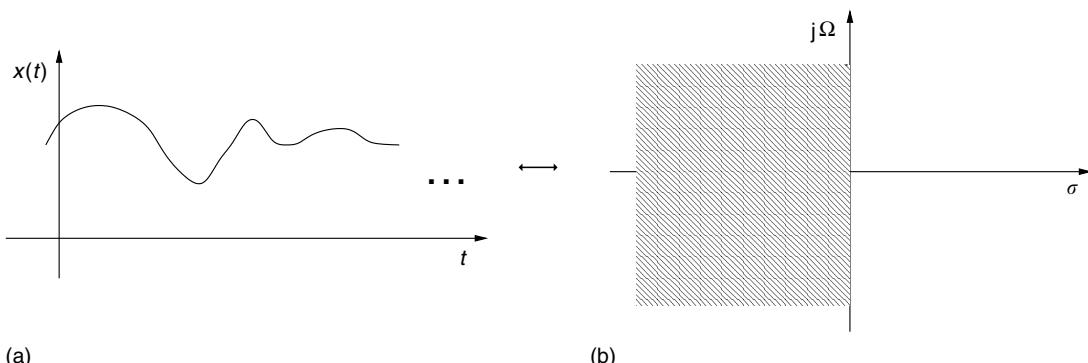


Fig. 3.20. Laplace transform: (a) continuous-time signal; (b) domain of the corresponding Laplace transform.

3.7.2 The z transform

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad \longleftrightarrow \quad x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1} dz. \quad (3.235)$$

- Time domain: nonperiodic function of a discrete and integer time variable.
- Frequency domain: nonperiodic function of a continuous and complex frequency variable.

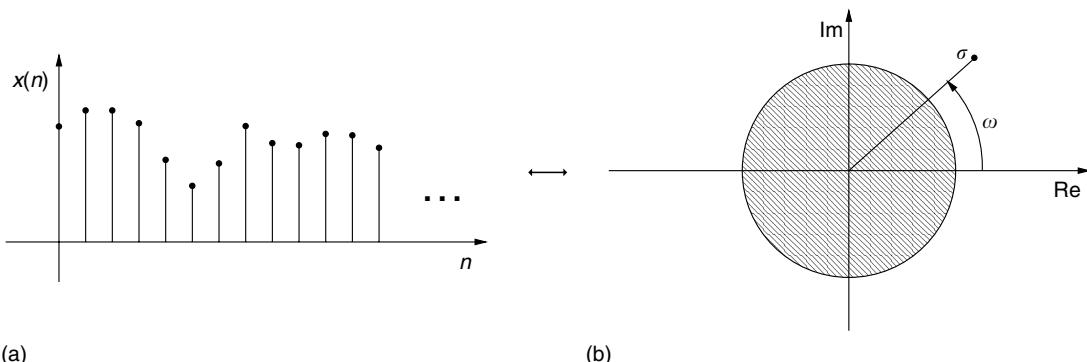


Fig. 3.21. The z transform: (a) discrete-time signal; (b) domain of the corresponding z transform.

3.7.3 Fourier transform (continuous time)

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt \quad \longleftrightarrow \quad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} d\Omega. \quad (3.236)$$

- Time domain: nonperiodic function of a continuous and real-time variable.
- Frequency domain: nonperiodic function of a continuous and imaginary frequency variable.

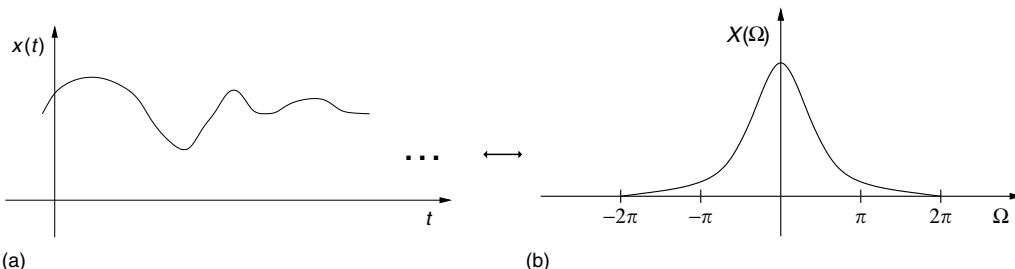


Fig. 3.22. Fourier transform: (a) continuous-time signal; (b) corresponding Fourier transform.

3.7.4 Fourier transform (discrete time)

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \quad \longleftrightarrow \quad x(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(e^{j\omega}) e^{j\omega n} d\omega. \quad (3.237)$$

- Time domain: nonperiodic function of a discrete and integer time variable.
- Frequency domain: periodic function of a continuous frequency variable.

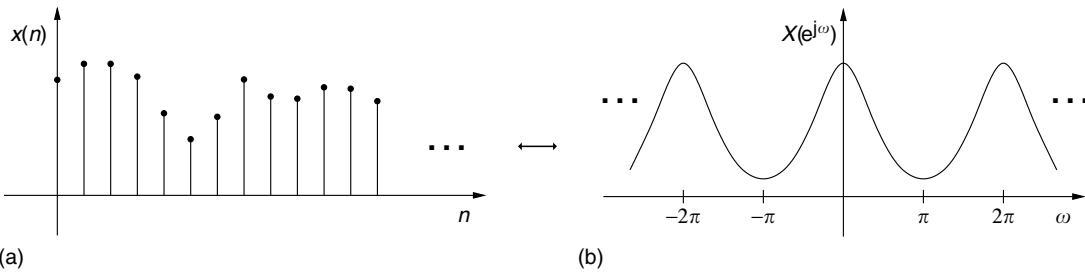


Fig. 3.23. Discrete-time Fourier transform: (a) discrete-time signal; (b) corresponding discrete-time Fourier transform.

3.7.5 Fourier series

$$X(k) = \frac{1}{T} \int_0^T x(t) e^{-j(2\pi/T)kt} dt \quad \longleftrightarrow \quad x(t) = \sum_{k=-\infty}^{\infty} X(k) e^{j(2\pi/T)kt}. \quad (3.238)$$

- Time domain: periodic function of a continuous and real-time variable.
 - Frequency domain: nonperiodic function of a discrete and integer frequency variable.

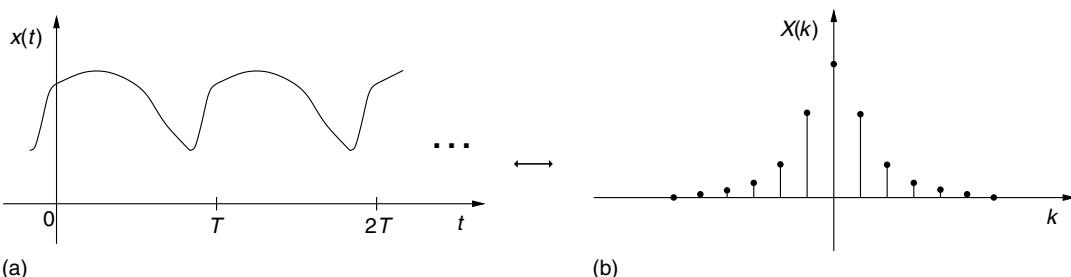


Fig. 3.24. Fourier series: (a) continuous-time periodic signal; (b) corresponding Fourier series.

3.7.6 Discrete Fourier transform

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn} \quad \longleftrightarrow \quad x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j(2\pi/N)kn}. \quad (3.239)$$

- Time domain: periodic function of a discrete and integer time variable.
 - Frequency domain: periodic function of a discrete and integer frequency variable.

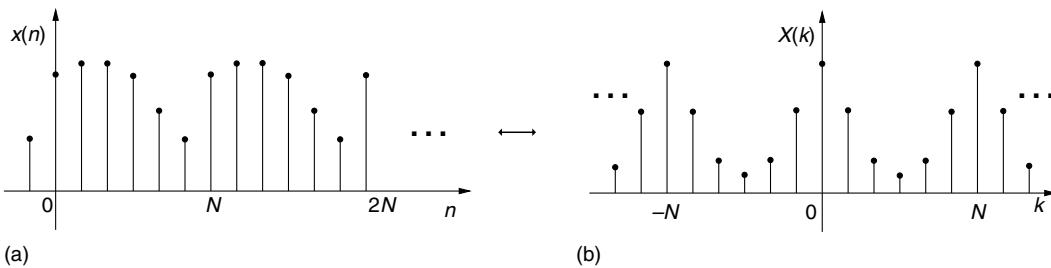


Fig. 3.25. DFT: (a) discrete-time signal; (b) corresponding discrete Fourier transform.

3.8 Do-it-yourself: discrete transforms

We now perform some numerical experiments on discrete transforms using MATLAB.

Experiment 3.1

We have seen that the output signal $y(n)$ of a linear, time-invariant, causal filter can be determined by the linear convolution between the input signal $x(n)$ and the system impulse response $h(n)$.

In this experiment, we investigate several ways to perform such an operation using MATLAB. We work here with short-length sequences $x(n)$ and $h(n)$, such as

```
x = ones(1,10);  
h = [1 2 3 4 5 6 7 8 9 10];
```

so that the reader can obtain the desired output algebraically in advance. Later, we compare the performance of each method seen below for longer sequences.

Given $x(n)$ and $h(n)$, perhaps the easiest way, but not necessarily the most numerically efficient one, is to employ the command `conv`:

```
y1 = conv(x,h);
```

whose input arguments can be swapped, since the convolution operation is symmetric.

For a general digital filter with transfer function

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \quad (3.240)$$

with $a_0 \neq 0$, one can determine the output $y(n)$ to the input $x(n)$ using the command `filter`. For this command, the input arguments are two vectors, containing the numerator and denominator coefficients of $H(z)$, and the input signal. In this experiment, where the impulse response $h(n)$ is given, we may use the command filter assuming that $A(z) = 1$ and associating $h(n)$ with the numerator-coefficient vector, such as:

```
v2 = filter(h,1,x);
```

It is interesting to note that the `filter` command forces the length of the output vector to be the same as that of the input. Therefore, if one wants to determine all nonzero samples of $y(n)$, we must force $x(n)$ to have the desired output length by padding the original input with the proper number of zeros:

```
xaux = [x zeros(1,length(h)-1)];
y3 = filter(h,1,xaux);
```

As mentioned in Section 3.4, one can implement the digital filtering operation via the frequency domain using the FFT. To avoid the circular convolution, one must first pad the $x(n)$ and $h(n)$ vectors with the proper numbers of zeros. The easiest way to determine these number of zeros is to remember that the length of the desired output signal should be the length of the linear convolution of $x(n)$ and $h(n)$; that is:

$$\text{length}(y) = \text{length}(x) + \text{length}(h) - 1. \quad (3.241)$$

Hence, we must guarantee that the FFTs of $x(n)$ and $h(n)$ are determined with this length, as performed by the following script:

```
length_y = length(x) + length(h) - 1;
X = fft(x,length_y);
H = fft(h,length_y);
Y4 = X.*H;
y4 = ifft(Y4);
```

In old MATLAB versions, numerical errors tended to accumulate throughout the filtering process, generating a complex output sequence, even when $x(n)$ and $h(n)$ were real signals. In these cases, one was forced to use the `real` command to store only the real part of the result. Current versions of MATLAB get rid of the spurious imaginary part of $y4$ automatically.

As mentioned before, this whole experiment was based on short $x(n)$ and $h(n)$ signals to allow the reader to follow closely all computations performed in MATLAB. In practice, if the lengths of these signals are sufficiently short (both of them around 100 coefficients), the simplest and fastest way to perform the digital filtering is with the `conv` command. If, however, both signals become too lengthy, then the frequency domain becomes quite advantageous in terms of numerical computations. In the case where only one of the signals has a long duration, the overlap-and-add method described in Section 3.4.2 can be implemented as

```
y5 = fftfilt(h,xaux);
```

where the FFT size and the segmentation of $x(n)$ are automatically chosen to guarantee efficient execution.

The reader is encouraged to experiment with all forms above to perform digital filtering with distinct input signals and impulse responses. In particular, you may increase the length of these signals, to the order of thousands of samples, in order to verify how the frequency domain becomes an important tool in several practical situations.

Experiment 3.2

Let us now employ the frequency domain to analyze the contents of a given signal $x(n)$ composed of a 10 Hz sinusoid corrupted by noise, with $F_s = 200$ samples/s for an interval of 1 s, as given by

```
fs = 200; f = 10;
time = 0:1/fs:(1-1/fs);
k = 0;
x = sin(2*pi*f.*time) + k*randn(1,fs);
figure(1);
plot(time,x);
```

where the parameter k controls the amount of noise present in $x(n)$.

Figure 3.26 shows examples of $x(n)$ and Figure 3.27 depicts the absolute value of the corresponding FFT for distinct values of k . Figure 3.26 indicates that the sinusoidal component

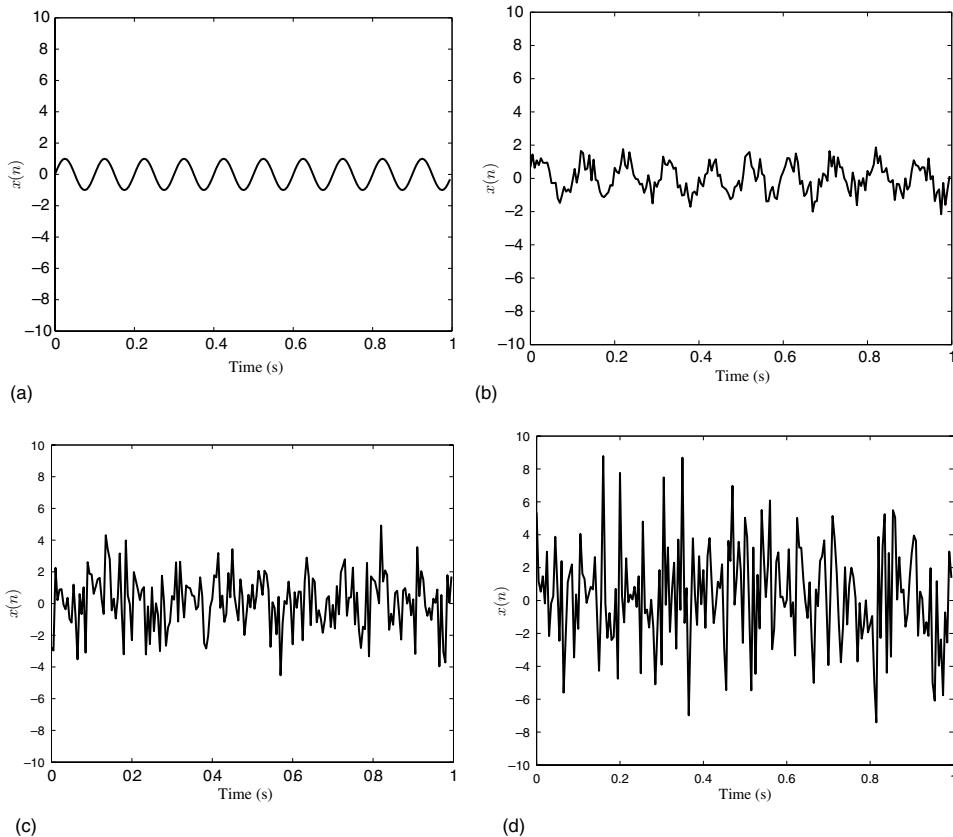


Fig. 3.26.

Sinusoidal signal corrupted with different levels of noise: (a) $k = 0$; (b) $k = 0.5$; (c) $k = 1.5$; (d) $k = 3$.

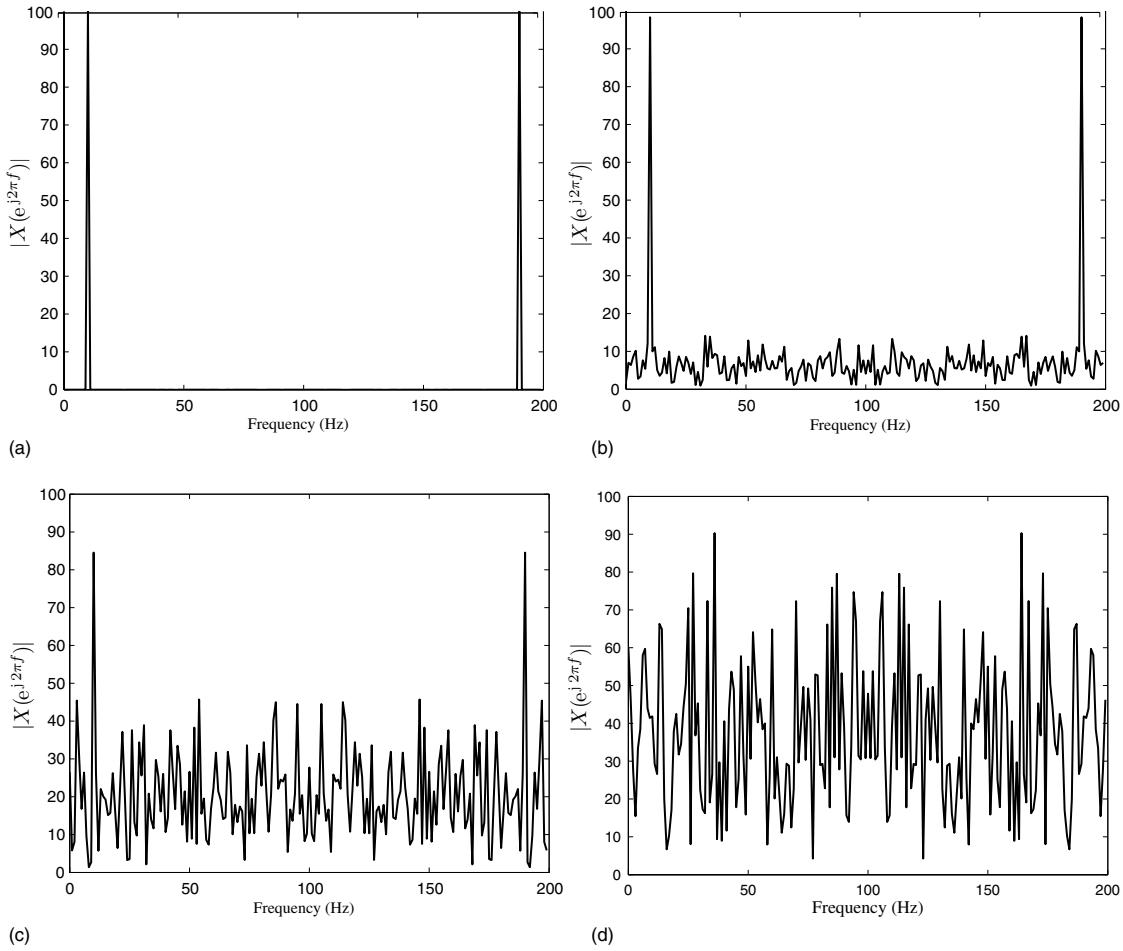


Fig. 3.27. Absolute value of FFT of sinusoidal signal corrupted with different levels of noise: (a) $k = 0$; (b) $k = 0.5$; (c) $k = 1.5$; (d) $k = 3$.

is clearly observed in the time domain for small amounts of noise, such as when $k \leq 0.5$. For larger amounts of noise, such as when $k = 1.5$, the frequency domain can then be employed to detect the sinusoidal component, as well as to estimate its frequency value, from the position of the dominating peaks in Figure 3.27. However, when k is too large, the sinusoid becomes masked by the noisy component even in the frequency domain, as observed in Figure 3.27d.

One way to deal with the large-noise case is to estimate the spectrum for several different time segments of $x(n)$ and average the results. By doing so, the 10 Hz sinusoidal peaks are present in all FFTs, whereas the noise peaks are randomly located in the different FFTs. Therefore, the averaging operation tends to preserve the FFT peaks corresponding to the 10 Hz sinusoid while attenuating the peaks due to the noise component. This approach is left as an exercise for the reader.

3.9 Discrete transforms with MATLAB

The functions described below are part of the MATLAB Signal Processing Toolbox.

- **fft:** Computes the DFT of a vector.

Input parameter: the input vector x .

Output parameter: the DFT of the input vector x .

Example:

```
t=0:0.001:0.25; x=sin(2*pi*50*t)+sin(2*pi*120*t);
y=fft(x); plot(abs(y));
```

- **ifft:** Computes the inverse DFT of a vector.

Input parameter: the complex input vector y having the DFT coefficients.

Output parameter: the inverse DFT of the input vector y .

Example:

```
w=[1:256]; y=zeros(size(w));
y(1:10)=1; y(248:256)=1;
x=ifft(y); plot(real(x));
```

- **fftshift:** Swaps the left and right halves of a vector. When applied to a DFT, shows it from $-(N/2) + 1$ to $N/2$.

Input parameter: the DFT vector y .

Output parameter: the swapped DFT vector z .

Example:

```
t=0:0.001:0.25; x=sin(2*pi*50*t)+sin(2*pi*120*t);
y=fft(x); z=fftshift(y); plot(abs(z));
```

- **dftmtx:** Generates a DFT matrix.

Input parameter: the size of the DFT n .

Output parameter: the $n \times n$ DFT matrix.

Example:

```
t=0:0.001:0.25; x=sin(2*pi*50*t)+sin(2*pi*120*t);
F=dftmtx(251); y=F*x'; z=fftshift(y); plot(abs(z));
```

- **fftfilt:** Performs linear filtering using the overlap-and-add method.

Input parameters:

- The vector h containing the filter coefficients.

- The vector x containing the input signal.

- The length n of the nonoverlapping blocks into which x is divided. If n is not provided, then the function uses its own optimized n (recommended).

Output parameter: the filtered signal y .

Example:

```
h=[1 2 3 4 4 3 2 1]; t=0:0.05:1;
x=sin(2*pi*3*t);
y=fftfilt(h,x); plot(y)
```

- **dct**: Computes the DCT of a vector.

Input parameter: the input vector x .

Output parameter: the DCT of the input vector x .

Example:

```
t=0:0.05:1; x=sin(2*pi*3*t);
y=dct(x); plot(y);
```

- **idct**: Computes the inverse DCT of a vector of coefficients.

Input parameter: the coefficient vector y .

Output parameter: the inverse DCT of the coefficient vector y .

Example:

```
y=1:32; x=idct(y); plot(x);
```

- **dctmtx**: Generates a DCT matrix.

Input parameter: the size of the DCT n .

Output parameter: the $n \times n$ DCT matrix.

Example (plots the first and sixth basis functions of a length-8 DCT):

```
C=dctmtx(8);
subplot(2,1,1), stem(C(1,:));
subplot(2,1,2), stem(C(6,:));
```

3.10 Summary

In this chapter we have thoroughly studied discrete transforms. We began with the DFT, which is a discrete-frequency representation of a finite discrete-time signal. We have shown that the DFT is a powerful tool for the computation of discrete-time convolutions. Several algorithms for the efficient implementation of the DFT (FFT algorithms) have been studied. In particular, we emphasized the radix-2 algorithms, because they are the most widely used. We have also dealt with discrete transforms other than the DFT, such as the discrete cosine transform, widely used in signal compression, and the Hartley transform, with many applications in fast convolutions and computation of the FFT. We then presented an overview of signal representations, where we compared the transforms studied in Chapters 1–3. A Do-it-yourself section guides the reader through MATLAB experiments on convolution with and without the DFT and the use of DFT in signal analysis. The chapter also included a brief description of MATLAB basic functions related to all the discrete transforms previously discussed.

3.11 Exercises

3.1 One wishes to measure the frequency content of a fast pulse $x(t)$ using a digital computer. In order to do so, a fast data acquisition system detects the beginning of the pulse and digitizes it. Knowing that:

- (a) the pulse duration is of approximately 1 ns,
 - (b) it has no significant frequency components beyond 5 GHz,
 - (c) one needs to discriminate frequency components spaced of 10 MHz,
- one asks:
- (a) Determine the smallest sampling rate at the A/D converter of the data acquisition system that makes the desired measurement possible.
 - (b) Describe the measurement procedure, providing the relevant parameter values for the minimum sampling frequency case.

3.2 Show that

$$\sum_{n=0}^{N-1} W_N^{nk} = \begin{cases} N, & \text{for } k = 0, \pm N, \pm 2N, \dots \\ 0, & \text{otherwise} \end{cases}.$$

3.3 Show, by substituting Equation (3.16) into Equation (3.15), and using the result given in Exercise 3.2, that Equation (3.16) gives the IDFT of a sequence $x(n)$.

3.4 Prove the following properties of the DFT:

- (a) Circular frequency-shift theorem – Equations (3.58), (3.60), and (3.61);
- (b) Correlation – Equation (3.68);
- (c) Parseval's theorem – Equation (3.89);
- (d) DFT of real and imaginary sequences – Equations (3.71) and (3.72).

3.5 Given the DFT coefficients represented by the vector

$$\mathbf{X} = [9 \quad 1 \quad 1 \quad 9 \quad 1 \quad 1 \quad 1 \quad 1]^T,$$

- (a) determine its length-8 IDFT;
- (b) determine the sequence whose length-8 DFT is given by $Y(k) = W_8^{-4k} X(k)$.

3.6 Suppose the DFT $X(k)$ of a sequence as represented by the vector

$$\mathbf{X} = [4 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 4]^T.$$

- (a) Compute the corresponding IDFT using the adequate DFT properties.
- (b) Compute the sequence whose length-8 DFT is given by $Y(k) = W_8^{3k} X(k)$.

3.7 Consider the sequence

$$x(n) = \delta(n) + 2\delta(n - 1) - \delta(n - 2) + \delta(n - 3)$$

- (a) Compute its length-4 DFT.
- (b) Compute the finite-length sequence $y(n)$ of length-6 whose DFT is equal to the imaginary part of the DFT of $x(n)$.
- (c) Compute the finite-length sequence $w(n)$ of length-4 whose DFT is equal to the imaginary part of the DFT of $x(n)$.
- 3.8 Show how, using a single DFT of length N , one can compute the DFT of four sequences: two even and real sequences and two odd and real sequences, all with length N .
- 3.9 Show how to compute the DFT of two even complex length- N sequences performing only one length- N transform calculation. Follow the steps below:
- Build the auxiliary sequence $y(n) = W_N^n x_1(n) + x_2(n)$.
 - Show that $Y(k) = X_1(k+1) + X_2(k)$.
 - Using properties of symmetric sequences, show that $Y(-k-1) = X_1(k) + X_2(k+1)$.
 - Use the results of (ii) and (iii) to create a recursion to compute $X_1(k)$ and $X_2(k)$. Note that $X(0) = \sum_{n=0}^{N-1} x(n)$.
- 3.10 Repeat Exercise 3.9 for the case of two complex antisymmetric sequences.
- 3.11 Show how to compute the DFT of four real, even, length- N sequences using only one length- N transform, using the results of Exercise 3.9.
- 3.12 Compute the coefficients of the Fourier series of the periodic sequences below using the DFT.
- $x'(n) = \sin\left(2\pi \frac{n}{N}\right)$, for $N = 20$.
 - $x'(n) = \begin{cases} 1, & \text{for } n \text{ even} \\ -1, & \text{for } n \text{ odd} \end{cases}$
- 3.13 Compute and plot the magnitude and phase of the DFT of the following finite-length sequences:
- $x(n) = 2 \cos\left(\pi \frac{n}{N}\right) + \sin^2\left(\pi \frac{n}{N}\right)$, for $0 \leq n \leq 10$ and $N = 11$.
 - $x(n) = e^{-2n}$, for $0 \leq n \leq 20$.
 - $x(n) = \delta(n-1)$, for $0 \leq n \leq 2$.
 - $x(n) = n$, for $0 \leq n \leq 5$.
- 3.14 Compute the linear convolution of the sequences in Figure 3.28 using the DFT.

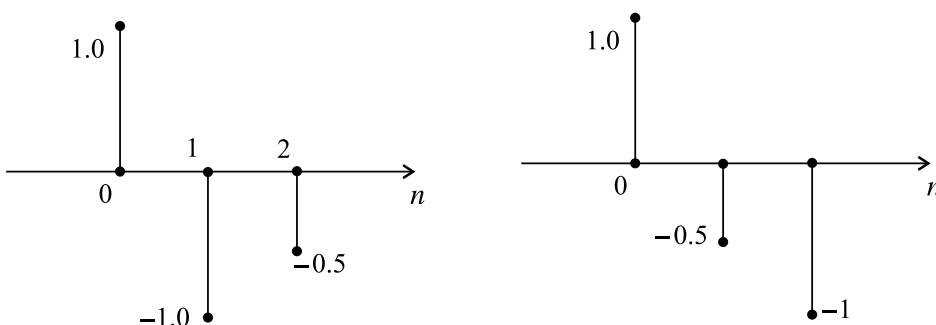


Fig. 3.28. Sequences from Exercise 3.14.

- 3.15 Compute the linear convolution of the sequences in Figure 3.29 using DFTs with the minimum possible lengths. Justify the solution, indicating the DFT properties employed.

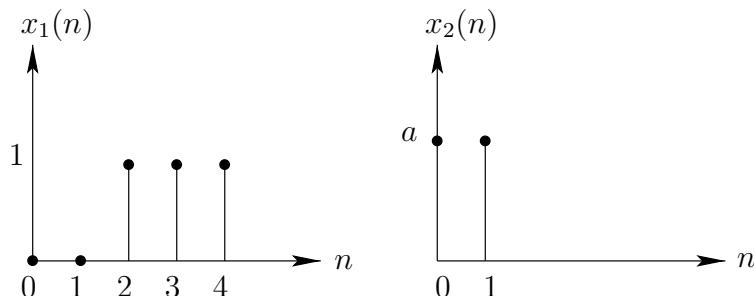


Fig. 3.29. Sequences from Exercise 3.15.

- 3.16 Given the sequences

$$\left. \begin{aligned} \mathbf{x} &= \left[1 \quad a \quad \frac{a^2}{2} \right]^T \\ \mathbf{h} &= \left[1 \quad -a \quad \frac{a^2}{2} \right]^T \end{aligned} \right\}$$

represented in matrix notation:

- (a) Calculate the linear convolution between the sequences employing the z transform.
- (b) For $a = 1$, compute the linear convolution using the overlap-and-add method when the second sequence is split into blocks of length 2, showing the partial results.
- 3.17 We want to compute the linear convolution of a long sequence $x(n)$, of length L , with a short sequence $h(n)$, of length K . If we use the overlap-and-save method to compute the convolution, determine the block length that minimizes the number of arithmetic operations involved in the convolution.
- 3.18 Repeat Exercise 3.17 for the case when the overlap-and-add method is used.
- 3.19 Express the algorithm described in the graph for the decimation-in-time FFT in Figure 3.11 in matrix form.
- 3.20 Express the algorithm described in the graph of the decimation-in-frequency FFT in Figure 3.13 in matrix form.
- 3.21 Determine the graph of a decimation-in-time length-6 DFT, and express its algorithm in matrix form.
- 3.22 Determine the basic cell of a radix-5 algorithm. Analyze the possible simplifications in the graph of the cell.
- 3.23 Repeat Exercise 3.22 for the radix-8 case, determining the complexity of a generic radix-8 algorithm.
- 3.24 Show that the number of complex multiplications of the FFT algorithm for generic N is given by Equation (3.176).

3.25 Compute, using an FFT algorithm, the linear convolution of the sequences (a) and (b) and then (b) and (c) in Exercise 3.13.

3.26 Show that:

- (a) The DCT of a length- N sequence $x(n)$ corresponds to the Fourier transform of the length- $2N$ sequence $\tilde{x}(n)$ consisting of $x(n)$ extended symmetrically; that is:

$$\tilde{x}(n) = \begin{cases} x(n), & \text{for } 0 \leq n \leq N - 1 \\ x(2N - n - 1), & \text{for } N \leq n \leq 2N - 1. \end{cases}$$

- (b) The DCT of $x(n)$ can be computed from the DFT of $\tilde{x}(n)$.

3.27 Show that the discrete cosine transform of a length- N sequence $x(n)$ can be computed from the length- N DFT of a sequence $\hat{x}(n)$ consisting of the following reordering of the even and odd elements of $x(n)$:

$$\hat{x}(n) = x(2n) \quad \left. \begin{array}{l} \\ x(N - 1 - n) = x(2n + 1) \end{array} \right\} \text{ for } 0 \leq n \leq \frac{N}{2} - 1.$$

3.28 Prove the relations between the DHT and the DFT given in Equations (3.224) and (3.225).

3.29 Prove the convolution-multiplication property of the DHT in Equation (3.228) and derive Equation (3.229) for the case $x_2(n)$ is even.

3.30 Show that the Hadamard transform matrix is unitary, using Equation (3.230).

3.31 The spectrum of a signal is given by its Fourier transform. In order to compute it, we need all the samples of the signal. Therefore, the spectrum is a characteristic of the whole signal. However, in many applications (e.g., in speech processing) one needs to find the spectrum of a short section of the signal. In order to do this, we define a length- N sliding window $x_i(n)$ of the signal $x(n)$ as

$$x_i(n) = x(n + i - N + 1), \text{ for } -\infty < i < \infty.$$

That is, we take N samples of $x(n)$ starting from position i backwards. We then define the short-time spectrum of $x(n)$ at the position i as the DFT of $x_i(n)$; that is:

$$X(k, i) = \sum_{n=0}^{N-1} x_i(n) W_N^{kn}, \text{ for } 0 \leq k \leq N - 1, \text{ for } -\infty < i < \infty.$$

Therefore, we have, for each frequency k , a signal $X(k, i)$, $-\infty < i < \infty$, which is the short-time spectrum of $x(n)$ at frequency k :

- (a) Show that the short-time spectrum can be efficiently computed with a bank of N IIR filters having the following transfer function for frequency k :

$$H_k(z) = \frac{1 - z^{-N}}{W_N^k - z^{-1}}, \quad 0 \leq k \leq N - 1.$$

- (b) Compare, in terms of the number of complex additions and multiplications, the complexity of the computations of the short-time spectrum using the above formula for $H_k(z)$ with that using the FFT algorithm.

- (c) Discuss whether it is advantageous or not, in terms of arithmetic operations, to use the above formula for $H_k(z)$ to compute a linear convolution using the overlap-and-add method. Repeat this item for the overlap-and-save method.

A good coverage of recursive computations of sinusoidal transforms can be found in Liu *et al.* (1994).

3.32 Linear convolution using `fft` in MATLAB.

- Use the `fft` command to determine the linear convolution between two given signals $x(n)$ and $h(n)$.
- Compare the function you have created in (a) with the `conv` and `filter` commands with respect to the output signal and to the total number of flops required to convolve the two signals of orders N and K , respectively.
- Verify your results experimentally (using the command `flops`) for general values of N and K .
- Repeat (c) considering solely the values of N and K such that $(N + K - 1)$ is a power of two.

3.33 Given the signal

$$x(n) = \sin\left(\frac{\omega_s}{10} n\right) + \sin\left[\left(\frac{\omega_s}{10} + \frac{\omega_s}{l}\right) n\right],$$

then:

- For $l = 100$, compute the DFT of $x(n)$ using 64 samples. Can you observe the presence of both sinusoids?
 - Increase the length of the DFT to 128 samples by padding 64 zeros to the original samples of $x(n)$. Comment on the results.
 - Compute the DFT of $x(n)$ using 128 samples. Can you now observe the presence of both sinusoids?
 - Increase the length of the DFT of $x(n)$ by 128 samples by padding 128 zeros to the samples of $x(n)$. Repeat for a padding of 384 zeros. Comment on the results.
- 3.34 Repeat Experiment 3.2 for a large amount $k \geq 3$ of noise component. Average the absolute value of FFT results for M repetitions of the experiment and identify the sinusoidal component on the resulting spectrum. Determine a good value of M for different values of k .

4.1 Introduction

In the previous chapters we studied different ways of describing discrete-time systems that are linear and time invariant. It was verified that the z transform greatly simplifies the analysis of discrete-time systems, especially those initially described by a difference equation.

In this chapter we study several structures used to realize a given transfer function associated with a specific difference equation through the use of the z transform. The transfer functions considered here will be of the polynomial form (nonrecursive filters) and of the rational-polynomial form (recursive filters). In the nonrecursive case we emphasize the existence of the important subclass of linear-phase filters. Then we introduce some tools to calculate the digital network transfer function, as well as to analyze its internal behavior. We also discuss some properties of generic digital filter structures associated with practical discrete-time systems. The chapter also introduces a number of useful building blocks often utilized in practical applications. A Do-it-yourself section is included in order to enlighten the reader on how to start from the concepts and generate some possible realizations for a given transfer function.

4.2 Basic structures of nonrecursive digital filters

Nonrecursive filters are characterized by a difference equation in the form

$$y(n) = \sum_{l=0}^M b_l x(n-l), \quad (4.1)$$

where the b_l coefficients are directly related to the system impulse response; that is, $b_l = h(l)$. Owing to the finite length of their impulse responses, nonrecursive filters are also referred to as finite-duration impulse response (FIR) filters. We can rewrite Equation (4.1) as

$$y(n) = \sum_{l=0}^M h(l)x(n-l). \quad (4.2)$$

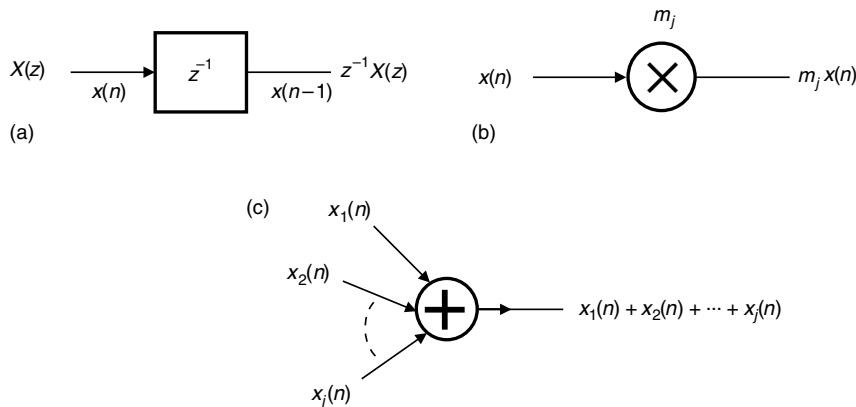


Fig. 4.1.

Classic representation of basic elements of digital filters: (a) delay; (b) multiplier; (c) adder.

Applying the z transform to Equation (4.2), we end up with the following input–output relationship:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{l=0}^M b_l z^{-l} = \sum_{l=0}^M h(l) z^{-l}. \quad (4.3)$$

In practical terms, Equation (4.3) can be implemented in several distinct forms, using as basic elements the delay, the multiplier, and the adder blocks. These basic elements of digital filters and their corresponding standard symbols are depicted in Figure 4.1. An alternative way of representing such elements is the so-called signal flowgraph shown in Figure 4.2. These two sets of symbolisms representing the delay, multiplier, and adder elements, are used throughout this book interchangeably.

4.2.1 Direct form

The simplest realization of an FIR digital filter is derived from Equation (4.3). The resulting structure, seen in Figure 4.3, is called the direct-form realization, as the multiplier coefficients are obtained directly from the filter transfer function. Such a structure is also referred to as the canonic direct form, where we understand canonic form to mean any structure that realizes a given transfer function with the minimum number of delays, multipliers, and adders. More specifically, a structure that utilizes the minimum number of delays is said to be canonic with respect to the delay element, and so on.

An alternative canonic direct form for Equation (4.3) can be derived by expressing $H(z)$ as

$$H(z) = \sum_{l=0}^M h(l) z^{-l} = h(0) + z^{-1}(h(1) + z^{-1}(h(2) + \dots + z^{-1}(h(M-1) + z^{-1}h(M)) \dots)). \quad (4.4)$$

The implementation of this form is shown in Figure 4.4.

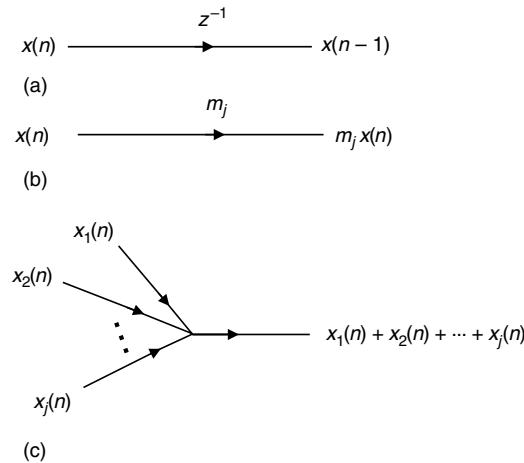


Fig. 4.2. Signal-flowgraph representation of basic elements of digital filters: (a) delay; (b) multiplier; (c) adder.

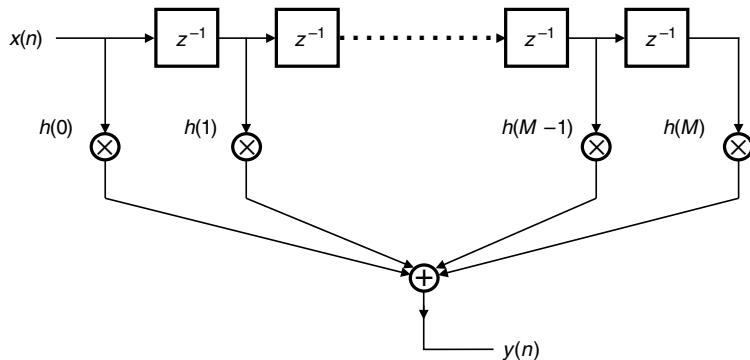


Fig. 4.3. Direct form for FIR digital filters.

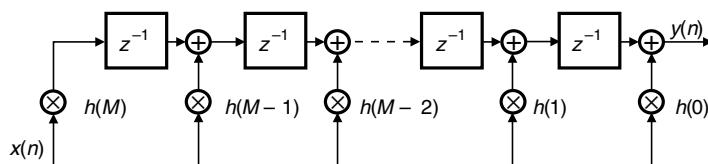


Fig. 4.4. Alternative direct form for FIR digital filters.

4.2.2 Cascade form

Equation (4.3) can be realized through a series of equivalent structures. However, the coefficients of such distinct realizations may not be explicitly the filter impulse response or the corresponding transfer function. An important example of such a realization is the

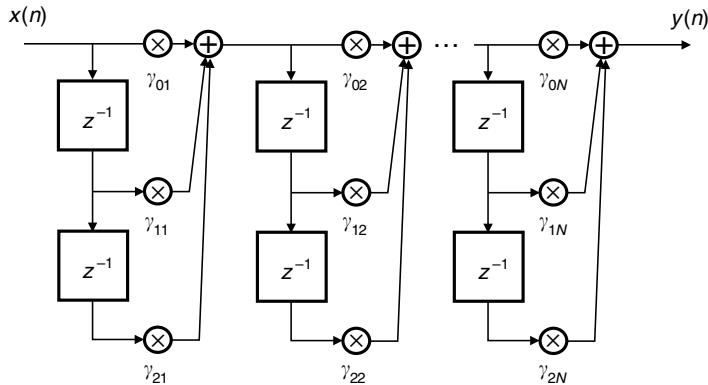


Fig. 4.5. Cascade form for FIR digital filters.

so-called cascade form, which consists of a series of second-order FIR filters connected in cascade, thus the name of the resulting structure, as seen in Figure 4.5.

The transfer function associated with such a realization is of the form

$$H(z) = \prod_{k=1}^N (\gamma_{0k} + \gamma_{1k}z^{-1} + \gamma_{2k}z^{-2}), \quad (4.5)$$

where if M is the filter order, then $N = M/2$ when M is even and $N = (M+1)/2$ when M is odd. In the latter case, one of the γ_{2k} becomes zero.

4.2.3 Linear-phase forms

An important subclass of FIR digital filters is the one that includes linear-phase filters. Such filters are characterized by a constant group delay τ ; therefore, they must present a frequency response of the following form:

$$H(e^{j\omega}) = B(\omega)e^{-j\omega\tau+j\phi}, \quad (4.6)$$

where $B(\omega)$ is real and τ and ϕ are constant. Hence, the impulse response $h(n)$ of linear-phase filters satisfies

$$\begin{aligned} h(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} B(\omega) e^{-j\omega\tau+j\phi} e^{j\omega n} d\omega \\ &= \frac{e^{j\phi}}{2\pi} \int_{-\pi}^{\pi} B(\omega) e^{j\omega(n-\tau)} d\omega. \end{aligned} \quad (4.7)$$

We are considering filters here where the group delay is a multiple of half a sample; that is:

$$\tau = \frac{k}{2}, \quad k \in \mathbb{Z}. \quad (4.8)$$

Thus, for such cases when 2τ is an integer, Equation (4.8) implies that

$$h(2\tau - n) = \frac{e^{j\phi}}{2\pi} \int_{-\pi}^{\pi} B(\omega) e^{j\omega(2\tau-n-\tau)} d\omega = \frac{e^{j\phi}}{2\pi} \int_{-\pi}^{\pi} B(\omega) e^{j\omega(\tau-n)} d\omega. \quad (4.9)$$

Since $B(\omega)$ is real, we have

$$h^*(2\tau - n) = \frac{e^{-j\phi}}{2\pi} \int_{-\pi}^{\pi} B^*(\omega) e^{-j\omega(\tau-n)} d\omega = \frac{e^{-j\phi}}{2\pi} \int_{-\pi}^{\pi} B(\omega) e^{j\omega(n-\tau)} d\omega. \quad (4.10)$$

Then, from Equations (4.7) and (4.10), in order for a filter to have linear phase with a constant group delay τ , its impulse response must satisfy

$$h(n) = e^{2j\phi} h^*(2\tau - n). \quad (4.11)$$

We now proceed to show that linear-phase FIR filters present impulse responses of very particular forms. In fact, Equation (4.11) implies that $h(0) = e^{2j\phi} h^*(2\tau)$. Hence, if $h(n)$ is causal and of finite duration, for $0 \leq n \leq M$, we must necessarily have that

$$\tau = \frac{M}{2} \quad (4.12)$$

and then Equation (4.11) becomes

$$h(n) = e^{2j\phi} h^*(M - n). \quad (4.13)$$

This is the general equation that the coefficients of a linear-phase FIR filter must satisfy.

In the common case where all the filter coefficients are real, then $h(n) = h^*(n)$ and Equation (4.13) implies that $e^{2j\phi}$ must be real. Thus:

$$\phi = \frac{k\pi}{2}, \quad k \in \mathbb{Z} \quad (4.14)$$

and Equation (4.13) becomes

$$h(n) = (-1)^k h(M - n), \quad k \in \mathbb{Z}. \quad (4.15)$$

That is, the filter impulse response must be either symmetric or antisymmetric.

From Equation (4.6), the frequency response of linear-phase FIR filters with real coefficients becomes

$$H(e^{j\omega}) = B(\omega) e^{-j\omega(M/2) + j(k\pi/2)} \quad (4.16)$$

For all practical purposes, we only need to consider the cases when $k = 0, 1, 2, 3$, as all other values of k will be equivalent to one of these four cases. Furthermore, as $B(\omega)$ can be either positive or negative, the cases $k = 2$ and $k = 3$ are obtained from cases $k = 0$ and $k = 1$ respectively by making $B(\omega) \leftarrow -B(\omega)$.

Therefore, we consider solely the four distinct cases described by Equations (4.13) and (4.16). They are referred to as follows:

- Type I: $k = 0$ and M even.
- Type II: $k = 0$ and M odd.
- Type III: $k = 1$ and M even.
- Type IV: $k = 1$ and M odd.

We now proceed to demonstrate that $h(n) = (-1)^k h(M - n)$ is a sufficient condition for an FIR filter with real coefficients to have a linear phase. The four types above are considered separately.

- Type I: $k = 0$ implies that the filter has symmetric impulse response; that is, $h(M - n) = h(n)$. Since the filter order M is even, Equation (4.3) may be rewritten as

$$\begin{aligned} H(z) &= \sum_{n=0}^{(M/2)-1} h(n)z^{-n} + h\left(\frac{M}{2}\right)z^{-M/2} + \sum_{n=(M/2)+1}^M h(n)z^{-n} \\ &= \sum_{n=0}^{(M/2)-1} h(n)\left[z^{-n} + z^{-(M-n)}\right] + h\left(\frac{M}{2}\right)z^{-M/2}. \end{aligned} \quad (4.17)$$

Evaluating this equation over the unit circle (that is, using the variable transformation $z \rightarrow e^{j\omega}$), one obtains

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^{(M/2)-1} h(n)\left(e^{-j\omega n} + e^{-j\omega M+j\omega n}\right) + h\left(\frac{M}{2}\right)e^{-j\omega M/2} \\ &= e^{-j\omega M/2} \left\{ h\left(\frac{M}{2}\right) + \sum_{n=0}^{(M/2)-1} 2h(n)\cos\left[\omega\left(n - \frac{M}{2}\right)\right] \right\}. \end{aligned} \quad (4.18)$$

Substituting n by $(M/2) - m$, we get

$$\begin{aligned} H(e^{j\omega}) &= e^{-j\omega M/2} \left[h\left(\frac{M}{2}\right) + \sum_{m=1}^{M/2} 2h\left(\frac{M}{2} - m\right)\cos(\omega m) \right] \\ &= e^{-j\omega M/2} \sum_{m=0}^{M/2} a(m)\cos(\omega m), \end{aligned} \quad (4.19)$$

with $a(0) = h(M/2)$ and $a(m) = 2h[(M/2) - m]$, for $m = 1, 2, \dots, M/2$.

Since this equation is in the form of Equation (4.16), this completes the sufficiency proof for Type I filters.

- Type II: $k = 0$ implies that the filter has a symmetric impulse response; that is, $h(M-n) = h(n)$. Since the filter order M is odd, Equation (4.3) may be rewritten as

$$\begin{aligned} H(z) &= \sum_{n=0}^{(M-1)/2} h(n)z^{-n} + \sum_{n=(M+1)/2}^M h(n)z^{-n} \\ &= \sum_{n=0}^{(M-1)/2} h(n)[z^{-n} + z^{-(M-n)}]. \end{aligned} \quad (4.20)$$

Evaluating this equation over the unit circle, one obtains

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^{(M-1)/2} h(n) \left(e^{-j\omega n} + e^{-j\omega M + j\omega n} \right) \\ &= e^{-j\omega M/2} \sum_{n=0}^{(M-1)/2} h(n) \left\{ e^{-j\omega[n-(M/2)]} + e^{j\omega[n-(M/2)]} \right\} \\ &= e^{-j\omega M/2} \sum_{n=0}^{(M-1)/2} 2h(n) \cos \left[\omega \left(n - \frac{M}{2} \right) \right]. \end{aligned} \quad (4.21)$$

Substituting n with $[(M+1)/2] - m$:

$$\begin{aligned} H(e^{j\omega}) &= e^{-j\omega M/2} \sum_{m=1}^{(M+1)/2} 2h\left(\frac{M+1}{2} - m\right) \cos \left[\omega \left(m - \frac{1}{2} \right) \right] \\ &= e^{-j\omega M/2} \sum_{m=1}^{(M+1)/2} b(m) \cos \left[\omega \left(m - \frac{1}{2} \right) \right], \end{aligned} \quad (4.22)$$

with $b(m) = 2h[(M+1/2) - m]$, for $m = 1, 2, \dots, (M+1)/2$.

Since this equation is in the form of Equation (4.16), this completes the sufficiency proof for Type II filters.

Notice that, at $\omega = \pi$, $H(e^{j\omega}) = 0$, as it consists of a summation of cosine functions evaluated at $\pm\pi/2$, which are obviously null. Therefore, highpass and bandstop filters cannot be approximated as Type II filters.

- Type III: $k = 1$ implies that the filter has an antisymmetric impulse response; that is, $h(M-n) = -h(n)$. In this case, $h(M/2)$ is necessarily null. Since the filter order M is even, Equation (4.3) may be rewritten as

$$\begin{aligned} H(z) &= \sum_{n=0}^{(M/2)-1} h(n)z^{-n} + \sum_{n=(M/2)+1}^M h(n)z^{-n} \\ &= \sum_{n=0}^{(M/2)-1} h(n) \left[z^{-n} - z^{-(M-n)} \right], \end{aligned} \quad (4.23)$$

which, when evaluated over the unit circle, yields

$$\begin{aligned}
 H(e^{j\omega}) &= \sum_{n=0}^{(M/2)-1} h(n) \left(e^{-j\omega n} - e^{-j\omega M + j\omega n} \right) \\
 &= e^{-j\omega M/2} \sum_{n=0}^{(M/2)-1} h(n) \left\{ e^{-j\omega[n-(M/2)]} - e^{j\omega[n-(M/2)]} \right\} \\
 &= e^{-j\omega M/2} \sum_{n=0}^{(M/2)-1} -2jh(n) \sin \left[\omega \left(n - \frac{M}{2} \right) \right] . \\
 &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{n=0}^{(M/2)-1} -2h(n) \sin \left[\omega \left(n - \frac{M}{2} \right) \right] . \quad (4.24)
 \end{aligned}$$

Substituting n by $(M/2) - m$:

$$\begin{aligned}
 H(e^{j\omega}) &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{M/2} -2h \left(\frac{M}{2} - m \right) \sin[\omega(-m)] \\
 &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{M/2} c(m) \sin(\omega m), \quad (4.25)
 \end{aligned}$$

with $c(m) = 2h[(M/2) - m]$, for $m = 1, 2, \dots, M/2$.

Since this equation is in the form of Equation (4.16), this completes the sufficiency proof for Type III filters.

Notice, in this case, that the frequency response becomes null at $\omega = 0$ and at $\omega = \pi$. That makes this type of realization suitable for bandpass filters, differentiators, and Hilbert transformers, these last two due to the phase shift of $\pi/2$, as will be seen in Chapter 5.

- Type IV: $k = 1$ implies that the filter has an antisymmetric impulse response; that is, $h(M - n) = -h(n)$. Since the filter order M is odd, Equation (4.3) may be rewritten as

$$\begin{aligned}
 H(z) &= \sum_{n=0}^{(M-1)/2} h(n) z^{-n} + \sum_{n=(M+1)/2}^M h(n) z^{-n} \\
 &= \sum_{n=0}^{(M-1)/2} h(n) \left[z^{-n} - z^{-(M-n)} \right]. \quad (4.26)
 \end{aligned}$$

Evaluating this equation over the unit circle:

$$\begin{aligned}
 H(e^{j\omega}) &= \sum_{n=0}^{(M-1)/2} h(n) \left(e^{-j\omega n} - e^{-j\omega M + j\omega n} \right) \\
 &= e^{-j\omega M/2} \sum_{n=0}^{(M-1)/2} h(n) \left\{ e^{-j\omega[n-(M/2)]} - e^{j\omega[n-(M/2)]} \right\}
 \end{aligned}$$

$$\begin{aligned}
 &= e^{-j\omega M/2} \sum_{n=0}^{(M-1)/2} -2jh(n) \sin\left[\omega\left(n - \frac{M}{2}\right)\right] \\
 &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{n=0}^{(M-1)/2} -2h(n) \sin\left[\omega\left(n - \frac{M}{2}\right)\right]. \quad (4.27)
 \end{aligned}$$

Substituting n by $[(M + 1)/2] - m$:

$$\begin{aligned}
 H(e^{j\omega}) &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{(M+1)/2} -2h\left(\frac{M+1}{2} - m\right) \sin\left[\omega\left(\frac{1}{2} - m\right)\right] \\
 &= e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{(M+1)/2} d(m) \sin\left[\omega\left(m - \frac{1}{2}\right)\right], \quad (4.28)
 \end{aligned}$$

with $d(m) = 2h\{[(M + 1)/2] - m\}$, for $m = 1, 2, \dots, (M + 1)/2$.

Since this equation is in the form of Equation (4.16), this completes the sufficiency proof for Type IV filters, thus finishing the whole proof.

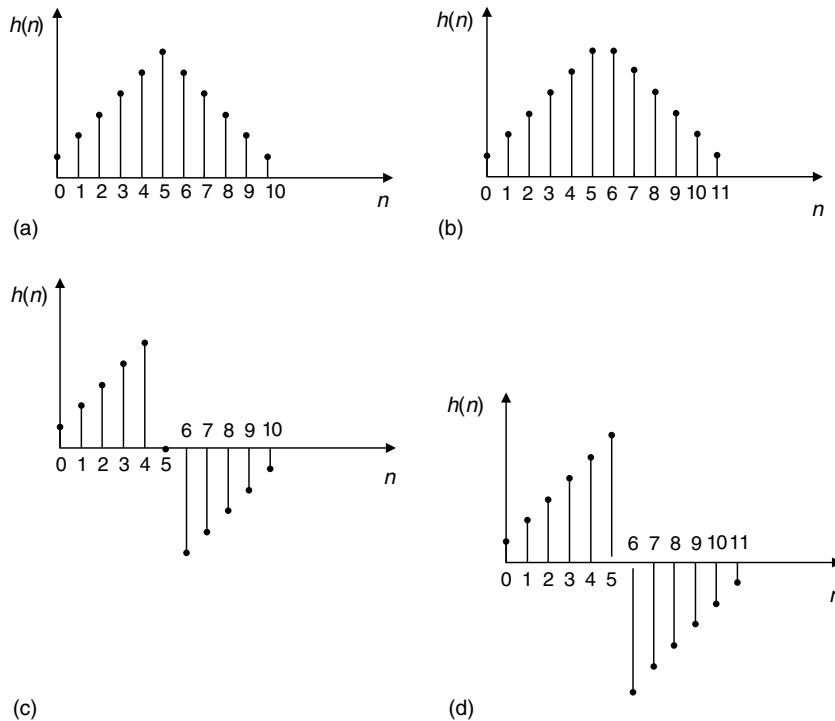


Fig. 4.6.

Example of impulse responses of linear-phase FIR digital filters: (a) Type I; (b) Type II; (c) Type III; (d) Type IV.

Table 4.1.

Main characteristics of linear-phase FIR filters: order, impulse response, frequency response, phase response, and group delay.

Type	M	$h(n)$	$H(e^{j\omega})$	$\Theta(\omega)$	τ
I	Even	Symmetric	$e^{-j\omega M/2} \sum_{m=0}^{M/2} a(m) \cos(\omega m)$ $a(0) = h(M/2); a(m) = 2h[(M/2) - m]$	$-\omega \frac{M}{2}$	$\frac{M}{2}$
II	Odd	Symmetric	$e^{-j\omega M/2} \sum_{m=1}^{(M+1)/2} b(m) \cos[\omega(m - \frac{1}{2})]$ $b(m) = 2h[((M+1)/2) - m]$	$-\omega \frac{M}{2}$	$\frac{M}{2}$
III	Even	Antisymmetric	$e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{M/2} c(m) \sin(\omega m)$ $c(m) = 2h[(M/2) - m]$	$-\omega \frac{M}{2} + \frac{\pi}{2}$	$\frac{M}{2}$
IV	Odd	Antisymmetric	$e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{(M+1)/2} d(m) \sin[\omega(m - \frac{1}{2})]$ $d(m) = 2h[((M+1)/2) - m]$	$-\omega \frac{M}{2} + \frac{\pi}{2}$	$\frac{M}{2}$

Notice that $H(e^{j\omega}) = 0$, at $\omega = 0$. Hence, lowpass filters cannot be approximated as Type IV filters, although this filter type is still suitable for differentiators and Hilbert transformers, like the Type III form.

Typical impulse responses of the four cases of linear-phase FIR digital filters are depicted in Figure 4.6. The properties of all four cases are summarized in Table 4.1.

One can derive important properties of linear-phase FIR filters by representing Equations (4.17), (4.20), (4.23), and (4.26) in a single framework as

$$H(z) = z^{-M/2} \sum_{n=0}^K h(n) \left\{ z^{(M/2)-n} \pm z^{-[(M/2)-n]} \right\} \quad (4.29)$$

where $K = M/2$ if M is even, or $K = (M-1)/2$ if M is odd. From Equation (4.29), it is easy to observe that if z_γ is a zero of $H(z)$, then so is z_γ^{-1} . This implies that all zeros of $H(z)$ occur in reciprocal pairs. Considering that if the coefficients $h(n)$ are real, all complex zeros occur in conjugate pairs, and then one can infer that the zeros of $H(z)$ must satisfy the following relationships:

- All complex zeros which are not on the unit circle occur in conjugate and reciprocal quadruples. In other words, if z_γ is complex, then z_γ^{-1} , z_γ^* , and $(z_\gamma^{-1})^*$ are also zeros of $H(z)$.

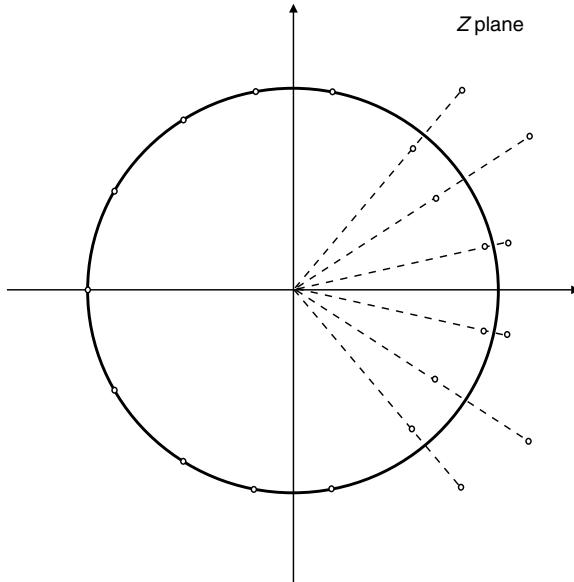


Fig. 4.7. Typical zero plot of a linear-phase FIR digital filter.

- There can be any given number of zeros over the unit circle, in conjugate pairs, since in this case we automatically have that $z_\gamma^{-1} = z_\gamma^*$.
- All real zeros outside the unit circle occur in reciprocal pairs.
- There can be any given number of zeros at $z = z_\gamma = \pm 1$, since in this case we necessarily have that $z_\gamma^{-1} = \pm 1$.

A typical zero plot for a linear-phase lowpass FIR filter is shown in Figure 4.7.

An interesting property of linear-phase FIR digital filters is that they can be realized with efficient structures that exploit their symmetric or antisymmetric impulse-response characteristics. In fact, when M is even, these efficient structures require $(M/2)+1$ multiplications, while when M is odd, only $(M+1)/2$ multiplications are necessary. Figure 4.8 depicts two of these efficient structures for linear-phase FIR filters when the impulse response is symmetric.

4.3 Basic structures of recursive digital filters

4.3.1 Direct forms

The transfer function of a recursive filter is given by

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^M b_i z^{-i}}{1 + \sum_{i=1}^N a_i z^{-i}}. \quad (4.30)$$

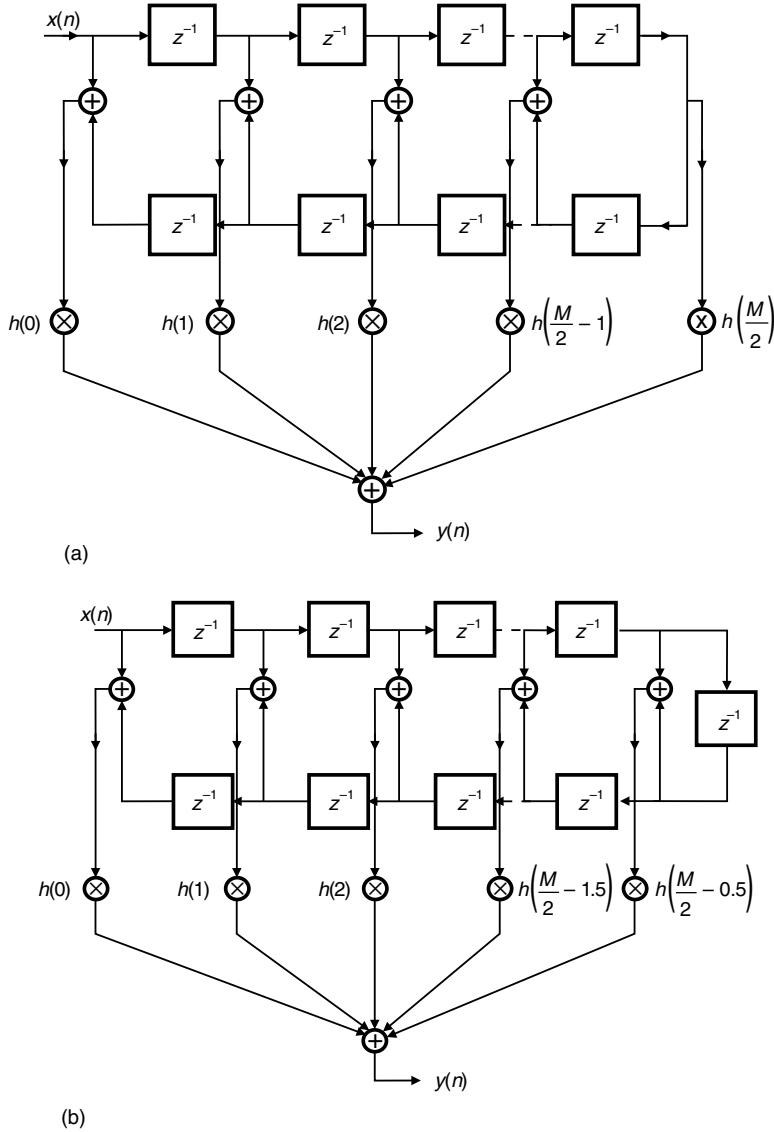


Fig. 4.8.

Realizations of linear-phase filters with symmetric impulse response: (a) even order; (b) odd order.

Since, in most cases, such transfer functions give rise to filters with impulse responses having infinite durations, recursive filters are also referred to as infinite-duration impulse response (IIR) filters.¹

We can consider that $H(z)$ as above results from the cascading of two separate filters of transfer functions $N(z)$ and $1/D(z)$. The $N(z)$ polynomial can be realized with the FIR direct form, as shown in the previous section. The realization of $1/D(z)$ can be performed

¹ It is important to note that in the cases where $D(z)$ divides $N(z)$, the filter $H(z)$ turns out to have a finite-duration impulse response and is actually an FIR filter.

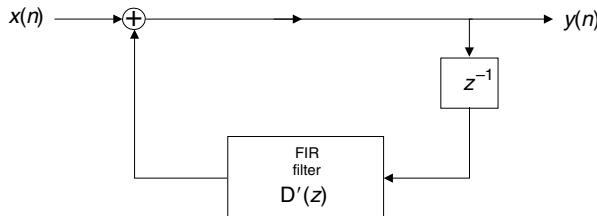


Fig. 4.9. Block diagram realization of $1/D(z)$.

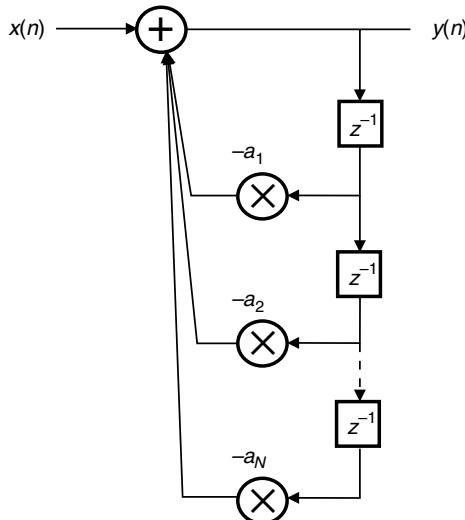


Fig. 4.10. Detailed realization of $1/D(z)$.

as depicted in Figure 4.9, where the FIR filter shown will be an $(N - 1)$ th-order filter with transfer function

$$D'(z) = z(1 - D(z)) = -z \sum_{i=1}^N a_i z^{-i}, \quad (4.31)$$

which can be realized as in Figure 4.3. The direct form for realizing $1/D(z)$ is then shown in Figure 4.10.

The complete realization of $H(z)$, as a cascade of $N(z)$ and $1/D(z)$, is shown in Figure 4.11. Such a structure is not canonic with respect to the delays, since for an (M, N) th-order filter this realization requires $(N + M)$ delays.

Clearly, in the general case we can change the order in which we cascade the two separate filters; that is, $H(z)$ can be realized as $N(z) \times 1/D(z)$ or $(1/D(z)) \times N(z)$. In the second option, all delays employed start from the same node, which allows us to eliminate the consequent redundant delays. In that manner, the resulting structure, usually referred to as the Type 1 canonic direct form, is the one depicted in Figure 4.12, for the special case when $N = M$.

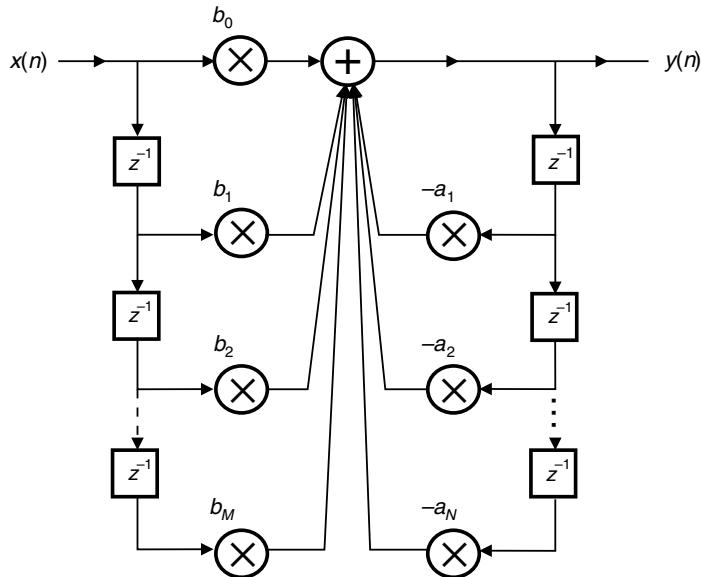


Fig. 4.11. Noncanonic IIR direct-form realization.

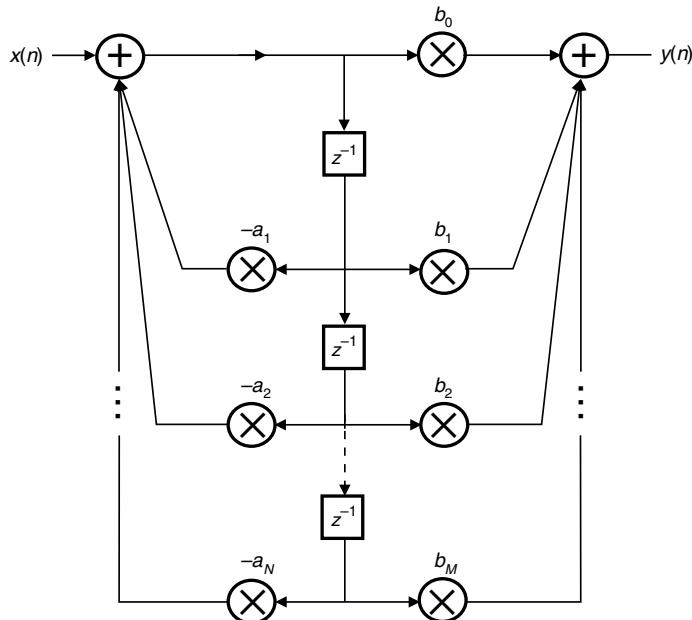


Fig. 4.12. Type 1 canonic direct form for IIR filters.

An alternative structure, the so-called Type 2 canonic direct form, is shown in Figure 4.13. Such a realization is generated from the nonrecursive form in Figure 4.4.

The majority of IIR filter transfer functions used in practice present a numerator degree M smaller than or equal to the denominator degree N . In general, one can consider, without

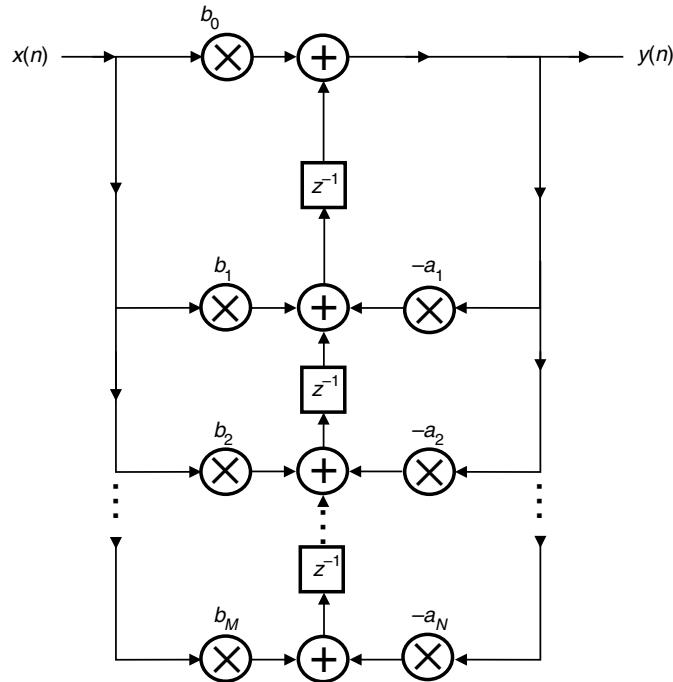


Fig. 4.13. Type 2 canonic direct form for IIR filters.

much loss of generality, that $M = N$. In the case where $M < N$, we just make the coefficients $b_{M+1}, b_{M+2}, \dots, b_N$ in Figures 4.12 and 4.13 equal to zero.

4.3.2 Cascade form

In the same way as their FIR counterparts, the IIR digital filters present a large variety of possible alternative realizations. An important one, referred to as the cascade realization, is depicted in Figure 4.14a, where the basic blocks represent simple transfer functions of orders 2 or 1. In fact, the cascade form, based on second-order blocks, is associated with the following transfer function decomposition:

$$\begin{aligned}
 H(z) &= \prod_{k=1}^m \frac{\gamma_{0k} + \gamma_{1k}z^{-1} + \gamma_{2k}z^{-2}}{1 + m_{1k}z^{-1} + m_{2k}z^{-2}} \\
 &= \prod_{k=1}^m \frac{\gamma_{0k}z^2 + \gamma_{1k}z + \gamma_{2k}}{z^2 + m_{1k}z + m_{2k}} \\
 &= H_0 \prod_{k=1}^m \frac{z^2 + \gamma'_{1k}z + \gamma'_{2k}}{z^2 + m_{1k}z + m_{2k}}.
 \end{aligned} \tag{4.32}$$

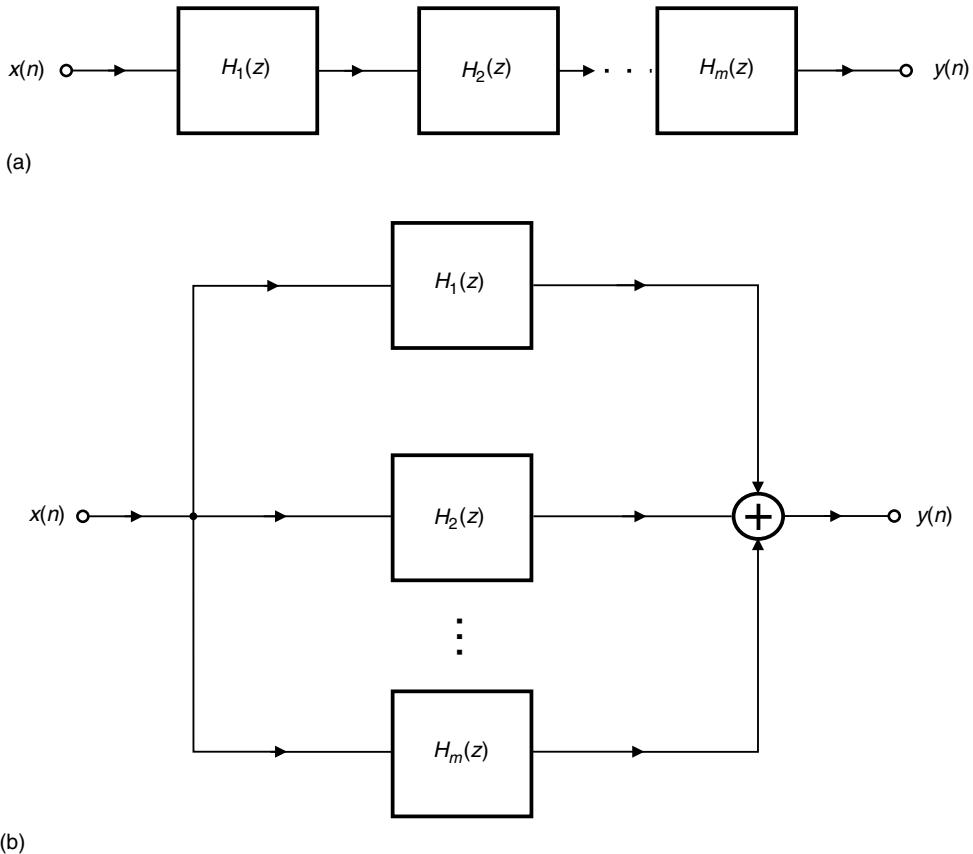


Fig. 4.14. Block diagrams of: (a) cascade form; (b) parallel form.

4.3.3 Parallel form

Another important realization for recursive digital filters is the parallel form represented in Figure 4.14b. Using second-order blocks, which are the most commonly used in practice, the parallel realization corresponds to the following transfer function decomposition:

$$\begin{aligned}
 H(z) &= \sum_{k=1}^m \frac{\gamma_{0k}^p z^2 + \gamma_{1k}^p z + \gamma_{2k}^p}{z^2 + m_{1k} z + m_{2k}} \\
 &= h_0 + \sum_{k=1}^m \frac{\gamma_{1k}^{p'} z + \gamma_{2k}^{p'}}{z^2 + m_{1k} z + m_{2k}} \\
 &= h'_0 + \sum_{k=1}^m \frac{\gamma_{0k}^{p''} z^2 + \gamma_{1k}^{p''} z}{z^2 + m_{1k} z + m_{2k}}
 \end{aligned} \tag{4.33}$$

also known as the partial-fraction decomposition. This equation indicates three alternative forms of the parallel realization, where the last two are canonic with respect to the number of multiplier elements.

It should be mentioned that each second-order block in the cascade and parallel forms can be realized by any of the existing distinct structures, as, for instance, one of the direct forms shown in Figure 4.15.

As will be seen in future chapters, all these digital filter realizations present different properties when one considers practical finite-precision implementations; that is, the quantization of the coefficients and the finite precision of the arithmetic operations, such as additions and multiplications (Jackson, 1969, 1996; Oppenheim & Schafer, 1975; Antoniou, 1993). In fact, the analysis of the finite-precision effects in the distinct realizations is

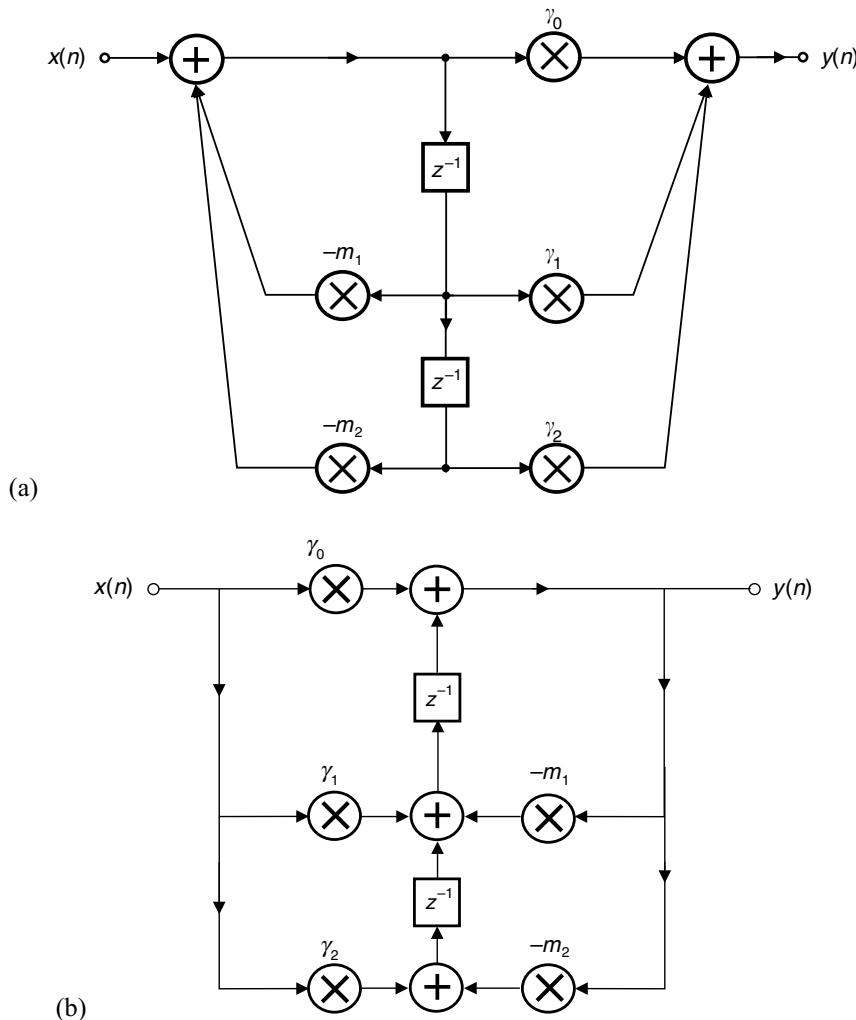


Fig. 4.15.

Realizations of second-order blocks: (a) Type 1 direct form; (b) Type 2 direct form.

a fundamental step in the overall process of designing any digital filter, as will be discussed in detail in Chapter 11.

Example 4.1. Describe the digital filter implementation of the transfer function

$$H(z) = \frac{16z^2(z+1)}{(4z^2 - 2z + 1)(4z + 3)} \quad (4.34)$$

using:

- (a) A cascade realization.
- (b) A parallel realization.

Solution

- (a) One cascade realization is obtained by describing the original transfer function as a product of second- and first-order building blocks as follows:

$$H(z) = \left(\frac{1}{1 - \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2}} \right) \left(\frac{1 + z^{-1}}{1 + \frac{3}{4}z^{-1}} \right). \quad (4.35)$$

Each section of the decomposed transfer function is implemented using the Type 1 canonic direct-form structure as illustrated in Figure 4.16.

- (b) Let us start by writing the original transfer function in a more convenient form as follows:

$$\begin{aligned} H(z) &= \frac{z^2(z+1)}{(z^2 - \frac{1}{2}z + \frac{1}{4})(z + \frac{3}{4})} \\ &= \frac{z^2(z+1)}{\left(z - \frac{1}{4} - j\frac{\sqrt{3}}{4}\right)\left(z - \frac{1}{4} + j\frac{\sqrt{3}}{4}\right)(z + \frac{3}{4})}. \end{aligned} \quad (4.36)$$

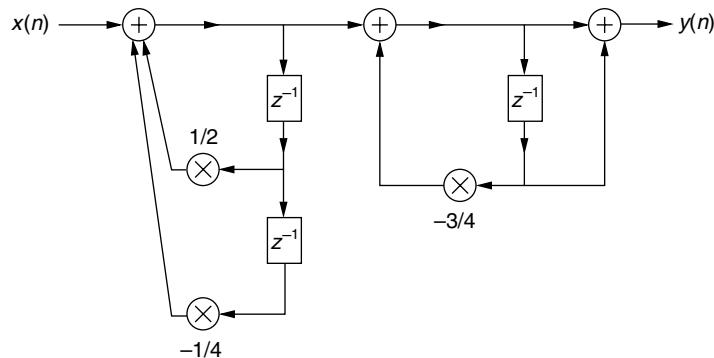


Fig. 4.16. Cascade implementation of $H(z)$ as given in Equation (4.35).

Next, we decompose $H(z)$ as a summation of first-order complex sections as

$$H(z) = r_1 + \frac{r_2}{z - p_2} + \frac{r_2^*}{z - p_2^*} + \frac{r_3}{z + p_3}, \quad (4.37)$$

where r_1 is the value of $H(z)$ at $z \rightarrow \infty$ and r_i is the residue associated with the pole p_i , for $i = 2, 3$, such that

$$\left. \begin{aligned} p_2 &= \frac{1}{4} + j\sqrt{3}/4 \\ p_3 &= \frac{3}{4} \\ r_1 &= 1 \\ r_2 &= \frac{6}{19} + j5/(3\sqrt{19}) \\ r_3 &= \frac{9}{76} \end{aligned} \right\}. \quad (4.38)$$

Given these values, the complex first-order sections are properly grouped to form second-order sections with real coefficients, and the constant r_1 is grouped with the real coefficient first-order section, resulting in the following decomposition for $H(z)$:

$$H(z) = \frac{1 + \frac{66}{76}z^{-1}}{1 + \frac{3}{4}z^{-1}} + \frac{\frac{12}{19}z^{-1} - \frac{11}{38}z^{-2}}{1 - \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2}}. \quad (4.39)$$

We can then implement each section using the Type 1 canonic direct-form structure leading to the realization shown in Figure 4.17. \triangle

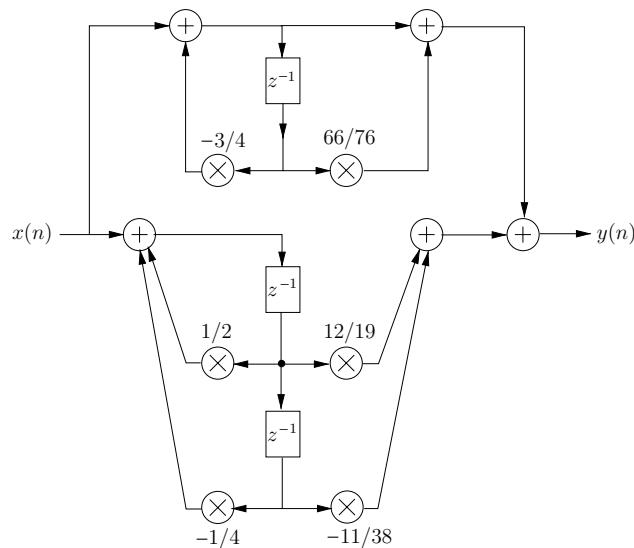


Fig. 4.17. Parallel implementation of $H(z)$ as given in Equation (4.39).

4.4 Digital network analysis

The signal-flowgraph representation greatly simplifies the analysis of digital networks composed of delays, multipliers, and adder elements (Cochiere & Oppenheim, 1975). In practice, the analysis of such devices is implemented by first numbering all nodes of the graph of interest. Then, one determines the relationship between the output signal of each node with respect to the output signals of all other nodes. The connections between two nodes, referred to as branches, consist of combinations of delays and/or multipliers. Branches that inject external signals into the graph are called source branches. Such branches have a transmission coefficient equal to 1. Following this framework, we can describe the output signal of each node as a combination of the signals of all other nodes and possibly an external signal; that is:

$$Y_j(z) = X_j(z) + \sum_{k=1}^N (a_{kj} Y_k(z) + z^{-1} b_{kj} Y_k(z)), \quad (4.40)$$

for $j = 1, 2, \dots, N$, where N is the number of nodes, a_{kj} and $z^{-1} b_{kj}$ are the transmission coefficients of the branch connecting node k to node j , $Y_j(z)$ is the z transform of the output signal of node j , and $X_j(z)$ is the z transform of the external signal injected in node j . We can express Equation (4.40) in a more compact form as

$$\mathbf{y}(z) = \mathbf{x}(z) + \mathbf{A}^T \mathbf{y}(z) + \mathbf{B}^T \mathbf{y}(z) z^{-1}, \quad (4.41)$$

where $\mathbf{y}(z)$ is the output signal $N \times 1$ vector and $\mathbf{x}(z)$ is the external input signal $N \times 1$ vector for all nodes in the given graph. Also, \mathbf{A}^T is an $N \times N$ matrix formed by the multiplier coefficients of the delayless branches of the circuit, while \mathbf{B}^T is the $N \times N$ matrix of multiplier coefficients of the branches with a delay.

Example 4.2. Describe the digital filter seen in Figure 4.18 using the compact representation given in Equation (4.41).

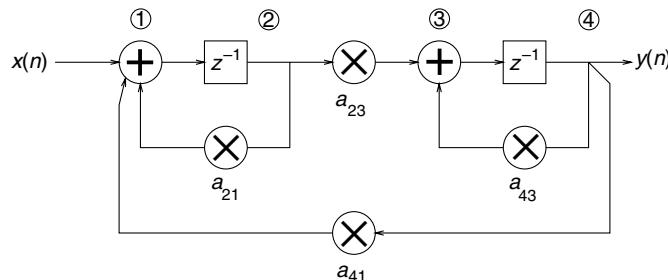


Fig. 4.18. Second-order digital filter.

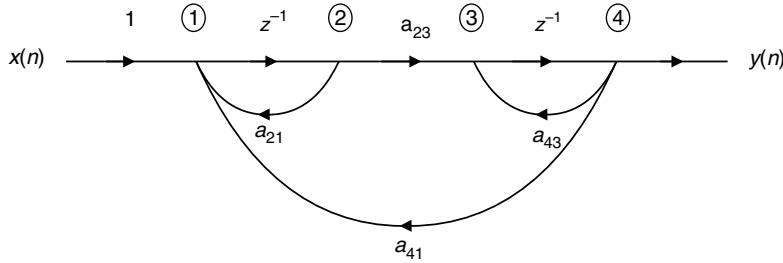


Fig. 4.19. Signal-flowgraph representation of a digital filter.

Solution

In order to carry out the description of the filter as in Equations (4.40) and (4.41), it is more convenient to represent it in the signal-flowgraph form, as in Figure 4.19.

Following the procedure described above, one can easily write that

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix} = \begin{bmatrix} X_1(z) \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{21} & 0 & a_{41} \\ 0 & 0 & 0 & 0 \\ 0 & a_{23} & 0 & a_{43} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix} + z^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix}. \quad (4.42)$$

△

The z -domain signals $Y_j(z)$ associated with the network nodes can be determined as

$$\mathbf{y}(z) = \mathbf{T}^T(z)\mathbf{x}(z), \quad (4.43)$$

with

$$\mathbf{T}^T(z) = \left(\mathbf{I} - \mathbf{A}^T - \mathbf{B}^T z^{-1} \right)^{-1}, \quad (4.44)$$

where \mathbf{I} is the N th-order identity matrix. In the above equation, $\mathbf{T}(z)$ is the so-called transfer matrix, whose entry $T_{ij}(z)$ describes the transfer function from node i to node j , namely

$$T_{ij}(z) = \left. \frac{Y_j(z)}{X_i(z)} \right|_{X_k(z)=0, k=1,2,\dots,N, k \neq j}, \quad (4.45)$$

and then $T_{ij}(z)$ gives the response at node j when the only nonzero input is applied to node i . If one is interested in the output signal of a particular node when several signals are injected into the network, then Equation (4.43) can be used to give

$$Y_j(z) = \sum_{i=1}^N T_{ij}(z) X_i(z). \quad (4.46)$$

Equation (4.41) can be expressed in the time domain as

$$\mathbf{y}(n) = \mathbf{x}(n) + \mathbf{A}^T \mathbf{y}(n) + \mathbf{B}^T \mathbf{y}(n-1). \quad (4.47)$$

If the above equation is used as a recurrence relation to determine the signal at a particular node given the input signal and initial conditions, then there is no guarantee that the signal at a particular node does not depend on the signal at a different node whose output is yet to be determined. This undesirable situation can be avoided by the use of special node orderings, such as the one provided by the algorithm below:

- (i) Enumerate the nodes only connected to either source branches or branches with a delay element. Note that, for computing the outputs of these nodes, we need only the current values of the external input signals or values of the internal signals at instant $(n-1)$.
- (ii) Enumerate the nodes only connected to source branches, branches with a delay element, or branches connected to the nodes whose outputs were already computed in step (i). For this new group of nodes, their corresponding outputs depend on external signals, signals at instant $(n-1)$, or signals previously determined in step (i).
- (iii) Repeat the procedure above until the outputs of all nodes have been enumerated. The only case in which this is not achievable (note that, at each step, at least the output of one new node should be enumerated) occurs when the given network presents a delayless loop, which is of no practical use.

Example 4.3. Analyze the network given in Example 4.2 using the algorithm described above.

Solution

In the given example, the first group consists of nodes 2 and 4, and the second group consists of nodes 1 and 3. If we reorder the nodes 2, 4, 1, and 3 as 1, 2, 3, and 4 respectively, then we end up with the network shown in Figure 4.20, which corresponds to

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ X_3(z) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a_{21} & a_{41} & 0 & 0 \\ a_{23} & a_{43} & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix} + z^{-1} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \end{bmatrix}. \quad (4.48)$$

△

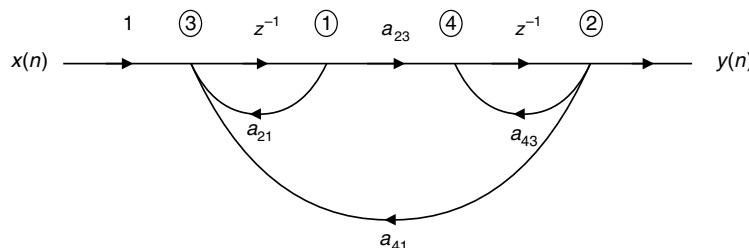


Fig. 4.20.

Reordering the nodes in the signal flowgraph.

In general, after reordering, \mathbf{A}^T can be put in the following form:

$$\mathbf{A}^T = \begin{bmatrix} 0 & \dots & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ a_{1j} & \dots & a_{kj} & \dots & \vdots & \vdots \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ a_{1N} & \dots & a_{kN} & \dots & 0 & 0 \end{bmatrix}. \quad (4.49)$$

This means that \mathbf{A}^T and \mathbf{B}^T tend to be sparse matrices. Therefore, efficient algorithms can be employed to solve the time- or z -domain analyses described in this section.

4.5 State-space description

An alternative form of representing digital filters is to use what is called the state-space representation. In such a description, the outputs of the memory elements (delays) are considered the system states. Once all the values of the external and state signals are known, we can determine the future values of the system states (the delay inputs) and the system output signals as follows:

$$\left. \begin{aligned} \mathbf{x}(n+1) &= \mathbf{Ax}(n) + \mathbf{Bu}(n) \\ \mathbf{y}(n) &= \mathbf{C}^T \mathbf{x}(n) + \mathbf{Du}(n) \end{aligned} \right\}, \quad (4.50)$$

where $\mathbf{x}(n)$ is the $N \times 1$ vector of the state variables. If M is the number of system inputs and M' is the number of system outputs, then we have that \mathbf{A} is $N \times N$, \mathbf{B} is $N \times M$, \mathbf{C} is $N \times M'$, and \mathbf{D} is $M' \times M$. In general, we work with single-input and single-output systems. In such cases, \mathbf{B} is $N \times 1$, \mathbf{C} is $N \times 1$, and \mathbf{D} is 1×1 ; that is, $\mathbf{D} = d$ is a scalar.

Note that this representation is essentially different from the one given in Equation (4.47), because in that equation the variables are the outputs of each node, whereas in the state-space approach the variables are just the outputs of the delays.

The impulse response for a system as described in Equation (4.50) is given by

$$h(n) = \begin{cases} d, & \text{for } n = 0 \\ \mathbf{C}^T \mathbf{A}^{n-1} \mathbf{B}, & \text{for } n > 0. \end{cases} \quad (4.51)$$

To determine the corresponding transfer function, we first apply the z transform to Equation (4.50), obtaining (note that in this case both $\mathbf{y}(n)$ and $\mathbf{u}(n)$ are scalars)

$$\left. \begin{aligned} z\mathbf{X}(z) &= \mathbf{AX}(z) + \mathbf{BU}(z) \\ Y(z) &= \mathbf{C}^T \mathbf{X}(z) + dU(z) \end{aligned} \right\} \quad (4.52)$$

and then

$$H(z) = \frac{Y(z)}{U(z)} = \mathbf{C}^T (z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} + d. \quad (4.53)$$

From Equation (4.53), it should be noted that the poles of $H(z)$ are the eigenvalues of \mathbf{A} , as the denominator of $H(z)$ will be given by the determinant of $(z\mathbf{I} - \mathbf{A})$.

By applying a linear transformation \mathbf{T} to the state vector such that

$$\mathbf{x}(n) = \mathbf{T}\mathbf{x}'(n), \quad (4.54)$$

where \mathbf{T} is any $N \times N$ nonsingular matrix, we end up with a system characterized by

$$\mathbf{A}' = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}; \quad \mathbf{B}' = \mathbf{T}^{-1}\mathbf{B}; \quad \mathbf{C}' = \mathbf{T}^T\mathbf{C}; \quad d' = d. \quad (4.55)$$

Such a system will present the same transfer function as the original system, and, consequently, the same poles and zeros. The proof of this fact is left to the interested reader as an exercise.

It is worth mentioning that the state-space representation is a compact form to describe a network, since it collects the essential relations among the key signals, namely the input, output, and memory (state) signals. In fact, the state-space representation leads to the minimum number of equations describing all the internal behavior and the input–output relationship associated with a given network, except for possible cases of insufficient controllability and observability characteristics, as concisely discussed by Vaidyanathan (1993).

Example 4.4. Determine the state-space equations of the filters given in Figures 4.15a and 4.18.

Solution

Associating a state-space variable $x_i(n)$ with each delay output, the corresponding delay input is represented by $x_i(n+1)$. In Figure 4.15a, let us use $i = 1$ in the upper delay and $i = 2$ in the lower one. The state-space description can be determined as follows.

- (i) The elements of the state-space transition matrix \mathbf{A} can be obtained by inspection as follows: for each delay input $x_i(n+1)$, locate the direct paths from the states $x_j(n)$, for all j , without crossing any other delay input. In Figure 4.15a, the coefficients $-m_1$ and $-m_2$ form the only direct paths from the states $x_1(n)$ and $x_2(n)$ respectively to $x_1(n+1)$. In addition, the relationship $x_1(n) = x_2(n+1)$ defines the only direct path from all states to $x_2(n+1)$.
- (ii) The elements of the input vector \mathbf{B} represent the direct path between the input signal to each delay input without crossing any other delay input. In Figure 4.15a, only the delay input $x_1(n+1)$ is directly connected to the input signal with coefficient value of 1.
- (iii) The elements of the output vector \mathbf{C} account for the direct connection between each state and the output node without crossing any delay input. In Figure 4.15a, the first state $x_1(n)$ has two direct connections with the output signal: one through the multiplier γ_1 and the other across the multipliers $-m_1$ and γ_0 . Similarly, the second state $x_2(n)$

has direct connections to the output node through the multiplier γ_2 and through the cascade of $-m_2$ and γ_0 .

- (iv) The feedforward coefficient d accounts for the direct connections between the input signal and the output node without crossing any state. In Figure 4.15a, there is a single direct connection through the multiplier with coefficient γ_0 .

Following the procedure described above, for the filter shown in Figure 4.15a, we have that

$$\left. \begin{aligned} \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} &= \begin{bmatrix} -m_1 & -m_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(n) \\ y(n) &= [(\gamma_1 - m_1\gamma_0) \quad (\gamma_2 - m_2\gamma_0)] \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \gamma_0 u(n) \end{aligned} \right\}. \quad (4.56)$$

By employing the same procedure described above for the filter in Figure 4.18, we can write that

$$\left. \begin{aligned} x_1(n+1) &= a_{21}x_1(n) + a_{41}x_2(n) + u(n) \\ x_2(n+1) &= a_{23}x_1(n) + a_{43}x_2(n) \\ y(n) &= x_2(n) \end{aligned} \right\}, \quad (4.57)$$

leading to the following state-space description:

$$\left. \begin{aligned} \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} &= \begin{bmatrix} a_{21} & a_{41} \\ a_{23} & a_{43} \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(n) \\ y(n) &= [0 \quad 1] \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + 0u(n) \end{aligned} \right\}. \quad (4.58)$$

Note that, for this realization, each element of the state-space representation is represented by a single coefficient, since there is at most one direct path among the states, input node, and output node. \triangle

4.6 Basic properties of digital networks

In this section we introduce some network properties that are very useful for designing and analyzing digital filters. The material covered in this section is mainly based on Fettweis (1971b).

4.6.1 Tellegen's theorem

Consider a digital network represented by the corresponding signal flowgraph, in which the signal at node j is y_j and the signal that reaches node j from node i is denoted by x_{ij} .

We can also use this notation to represent a branch that leaves from and arrives at the same node. Such a branch is called a loop. Among other things, loops are used to represent source branches entering a node. In fact, every source branch will be represented as a loop having the value of its source. In this case, x_{ii} includes the external signal and any other loop connecting node i to itself. Following this framework, for each node of a given graph, we can write that

$$y_j = \sum_{i=1}^N x_{ij}, \quad (4.59)$$

where N in this case is the total number of nodes. Consider now the following result.

Theorem 4.1 (Tellegen's Theorem). *All corresponding signals, (x_{ij}, y_j) and (x'_{ij}, y'_j) , of two distinct networks with equal signal-flowgraph representations satisfy*

$$\sum_{i=1}^N \sum_{j=1}^N (y_j x'_{ij} - y'_i x_{ji}) = 0, \quad (4.60)$$

where both sums include all nodes of both networks. \diamond

Proof Equation (4.60) can be rewritten as

$$\sum_{j=1}^N \left(y_j \sum_{i=1}^N x'_{ij} \right) - \sum_{i=1}^N \left(y'_i \sum_{j=1}^N x_{ji} \right) = \sum_{j=1}^N y_j y'_j - \sum_{i=1}^N y'_i y_i = 0, \quad (4.61)$$

which completes the proof. \square

Tellegen's theorem can be generalized for the frequency domain, since

$$Y_j = \sum_{i=1}^N X_{ij} \quad (4.62)$$

and then

$$\sum_{i=1}^N \sum_{j=1}^N (Y_j X'_{ij} - Y'_i X_{ji}) = 0. \quad (4.63)$$

Notice that, in Tellegen's theorem, x_{ij} is actually the sum of all signals departing from node i and arriving at node j . Therefore, in the most general case where the two graphs have different topologies, Tellegen's theorem can still be applied, making the two topologies equal by adding as many nodes and branches with null transmission values as necessary.

4.6.2 Reciprocity

Consider a particular network in which M of its nodes each have two branches connecting them to the outside world, as depicted in Figure 4.21. The first branch in each of these nodes is a source branch through which an external signal is injected into the node. The second branch makes the signal of each node available as an output signal. Naturally, in nodes where there are neither external input nor output signals, one must consider the corresponding branch to have a null transmission value. For generality, if a node does not have either external or output signals, then both corresponding branches are considered to have a null transmission value.

Suppose that we apply a set of signals X_i to this network and collect as output the signals Y_i . Alternatively, we could apply the signals X'_i and observe the Y'_i signals. The particular network is said to be reciprocal if

$$\sum_{i=1}^M (X_i Y'_i - X'_i Y_i) = 0. \quad (4.64)$$

In such cases, if the M -port network is described by

$$Y_i = \sum_{j=1}^M T_{ji} X_j, \quad (4.65)$$

where T_{ji} is the transfer function from port j to port i , Equation (4.64) is then equivalent to

$$T_{ij} = T_{ji}. \quad (4.66)$$

The proof for this statement is based on the substitution of Equation (4.65) into Equation (4.64), which yields

$$\sum_{i=1}^M \left(X_i \sum_{j=1}^M T_{ji} X'_j - X'_i \sum_{j=1}^M T_{ji} X_j \right) = \sum_{i=1}^M \sum_{j=1}^M (X_i T_{ji} X'_j) - \sum_{i=1}^M \sum_{j=1}^M (X'_i T_{ji} X_j)$$

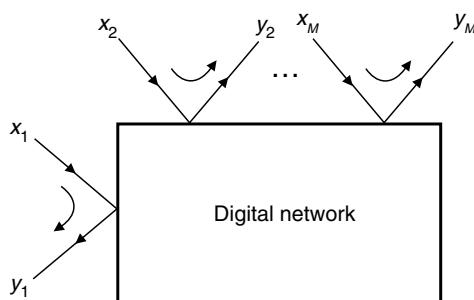


Fig. 4.21. General digital network with M ports.

$$\begin{aligned}
&= \sum_{i=1}^M \sum_{j=1}^M (X_i T_{ji} X'_j) - \sum_{i=1}^M \sum_{j=1}^M (X'_j T_{ij} X_i) \\
&= \sum_{i=1}^M \sum_{j=1}^M (T_{ji} - T_{ij}) (X_i X'_j) \\
&= 0
\end{aligned} \tag{4.67}$$

and thus

$$T_{ij} = T_{ji}. \tag{4.68}$$

□

4.6.3 Interreciprocity

The vast majority of the digital networks associated with digital filters are not reciprocal. However, such a concept is crucial in some cases. Fortunately, there is another related property, called interreciprocity, between two networks which is very common and useful. Consider two networks with the same number of nodes, and also consider that X_i and Y_i are respectively input and output signals of the first network. Correspondingly, X'_i and Y'_i represent input and output signals of the second network. Such networks are considered interreciprocal if Equation (4.64) holds for (X_i, Y_i) and (X'_i, Y'_i) , $i = 1, 2, \dots, M$.

If two networks are described by

$$Y_i = \sum_{j=1}^M T_{ji} X_j \tag{4.69}$$

and

$$Y'_i = \sum_{j=1}^M T'_{ji} X'_j, \tag{4.70}$$

then it can be easily shown that these two networks are interreciprocal if

$$T_{ji} = T'_{ij}. \tag{4.71}$$

Once again, the proof is left as an exercise for the interested reader.

4.6.4 Transposition

Given any signal-flowgraph representation of a digital network, we can generate another network by reversing the directions of all branches. In such a procedure, all addition nodes turn into distribution nodes, and vice versa. Also, if in the original network the branch from

node i to node j is F_{ij} (that is, $X_{ij} = F_{ij}Y_j$), then the transpose network will have a branch from node j to node i with transmission F'_{ji} such that

$$F_{ij} = F'_{ji}. \quad (4.72)$$

Using Tellegen's theorem, one can easily show that the original network and its corresponding transpose network are interreciprocal. If we represent X_i as X_{ii} and number the M input-output nodes in Figure 4.21 as 1 to M , leaving indexes ($M+1$) to N to represent the internal nodes, then by applying Tellegen's theorem to all signals of both networks one obtains

$$\begin{aligned} & \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i, \text{ if } i < M+1}}^N (Y_j X'_{ij} - Y'_i X_{ji}) + \sum_{i=1}^M (Y_i X'_{ii} - Y'_i X_{ii}) \\ &= \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i, \text{ if } i < M+1}}^N (Y_j F'_{ij} Y'_i - Y'_i F_{ji} Y_j) + \sum_{i=1}^M (Y_i X'_{ii} - Y'_i X_{ii}) \\ &= 0 + \sum_{i=1}^M (Y_i X'_{ii} - Y'_i X_{ii}) \\ &= 0, \end{aligned} \quad (4.73)$$

where all external signals are considered to be injected at the first M nodes. Naturally, $\sum_{i=1}^M (Y_i X'_{ii} - Y'_i X_{ii}) = 0$ is equivalent to interreciprocity (see Equation (4.64) applied to the interreciprocity case), which implies that (Equation (4.71))

$$T_{ij} = T'_{ji}. \quad (4.74)$$

This is a very important result, because it indicates that a network and its transpose must have the same transfer function. For instance, the equivalence of the networks in Figures 4.3 and 4.4 can be deduced from the fact that one is the transpose of the other. The same can be said about the networks in Figures 4.12 and 4.13.

4.6.5 Sensitivity

Sensitivity is a measure of the degree of variation of a network's overall transfer function with respect to small fluctuations in the value of one of its elements. In the specific case of digital filters, one is often interested in the sensitivity with respect to variations of the multiplier coefficients; that is:

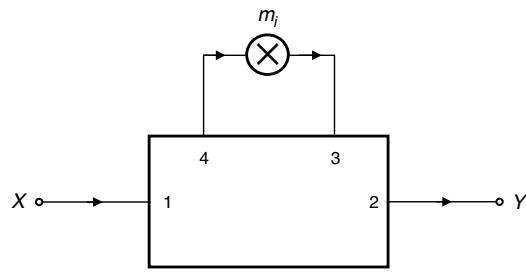
$$S_{m_i}^{H(z)} = \frac{\partial H(z)}{\partial m_i} \quad (4.75)$$

for $i = 1, 2, \dots, L$, where L is the total number of multipliers in the particular network.

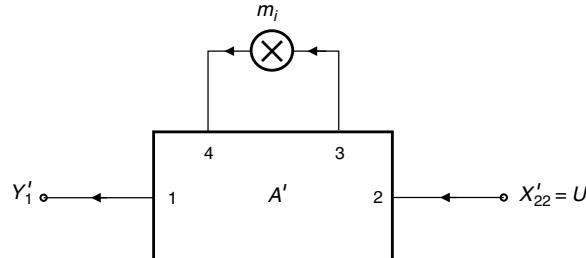
Using the concept of transposition, we can determine the sensitivity of $H(z)$ with respect to a given coefficient m_i in a very efficient way. To understand how, consider a network, its transpose, and also the original network with a specific coefficient slightly modified, as depicted in Figure 4.22.

Using Tellegen's theorem on the networks shown in Figures 4.22b and 4.22c, one obtains

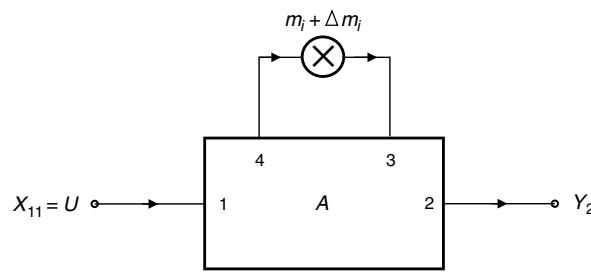
$$\begin{aligned} \sum_{i=1}^N \sum_{j=1}^N (Y_j X'_{ij} - Y'_i X_{ji}) &= \underbrace{\sum_{j=1}^N (Y_j X'_{1j} - Y'_1 X_{j1})}_{A_1} + \underbrace{\sum_{j=1}^N (Y_j X'_{2j} - Y'_2 X_{j2})}_{A_2} \\ &\quad + \underbrace{\sum_{j=1}^N (Y_j X'_{3j} - Y'_3 X_{j3})}_{A_3} + \underbrace{\sum_{i=4}^N \sum_{j=1}^N (Y_j X'_{ij} - Y'_i X_{ji})}_{A_4} \end{aligned}$$



(a)



(b)



(c)

Fig. 4.22. Digital networks: (a) original; (b) transpose; (c) original with modified coefficient.

$$\begin{aligned}
&= A_1 + A_2 + A_3 + A_4 \\
&= 0,
\end{aligned} \tag{4.76}$$

where A_1, A_2, A_3 , and A_4 are separately determined below.

$$\begin{aligned}
A_1 &= Y_1 X'_{11} - Y'_1 X_{11} + \sum_{j=2}^N (Y_j X'_{1j} - Y'_1 X_{j1}) \\
&= -UY'_1 + \sum_{j=2}^N (Y_j F'_{1j} Y'_1 - Y'_1 F_{j1} Y_j) \\
&= -UY'_1,
\end{aligned} \tag{4.77}$$

since $F'_{1j} = F_{j1}$, for all j . In addition:

$$\begin{aligned}
A_2 &= Y_2 X'_{22} - Y'_2 X_{22} + \sum_{\substack{j=1 \\ j \neq 2}}^N (Y_j X'_{2j} - Y'_2 X_{j2}) \\
&= UY_2 + \sum_{\substack{j=1 \\ j \neq 2}}^N (Y_j F'_{2j} Y'_2 - Y'_2 F_{j2} Y_j) \\
&= UY_2,
\end{aligned} \tag{4.78}$$

$$\begin{aligned}
A_3 &= Y_4 X'_{34} - Y'_3 X_{43} + \sum_{\substack{j=1 \\ j \neq 4}}^N (Y_j F'_{3j} Y'_3 - Y'_3 F_{j3} Y_j) \\
&= Y_4 m_i Y'_3 - Y'_3 (m_i + \Delta m_i) Y_4 \\
&= -\Delta m_i Y_4 Y'_3,
\end{aligned} \tag{4.79}$$

and

$$\begin{aligned}
A_4 &= \sum_{i=4}^N \sum_{j=1}^N (Y_j X'_{ij} - Y'_i X_{ji}) \\
&= \sum_{i=4}^N \sum_{j=1}^N (Y_j F'_{ij} Y'_i - Y'_i F_{ji} Y_j) \\
&= 0.
\end{aligned} \tag{4.80}$$

Hence, one has that

$$-UY'_1 + UY_2 - \Delta m_i Y_4 Y'_3 = 0. \tag{4.81}$$

Thus:

$$U(Y_2 - Y'_1) = \Delta m_i Y_4 Y'_3. \quad (4.82)$$

Defining

$$\Delta H_{12} = (H_{12} - H'_{21}) = \left(\frac{Y_2}{U} - \frac{Y'_1}{U} \right) \quad (4.83)$$

one gets, from Equation (4.82), that

$$U^2 (H_{12} - H'_{21}) = U^2 \Delta H_{12} = \Delta m_i Y_4 Y'_3. \quad (4.84)$$

If we now let Δm_i converge to zero, then H_{12} tends to H'_{21} , and consequently

$$\frac{\partial H_{12}}{\partial m_i} = \frac{Y_4 Y'_3}{U^2} = H'_{23} H_{14} = H_{32} H_{14}. \quad (4.85)$$

This equation indicates that the sensitivity of the transfer function of the original network, H_{12} , with respect to variations of one of its coefficients, can be determined based on transfer functions between the system input and the node before the multiplier, H_{14} , and between the multiplier output node and the system output, H_{32} .

Example 4.5. Determine the sensitivity of $H(z)$ with respect to the coefficients a_{11} , a_{22} , a_{12} , and a_{21} in the network of Figure 4.23.

Solution

The state-space description of the network in Figure 4.23 can be determined by using the procedure described in Example 4.4 for the network in Figure 4.15a. In the present case, entries of the state-space matrices correspond exactly to the multiplier coefficients of the network (owing to this fact, the network in Figure 4.23 is called the state-space structure):

$$\begin{aligned} \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(n) \\ y(n) &= [c_1 \ c_2] \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + d u(n) \end{aligned} \quad (4.86)$$

The transfer function of the state-space structure is given by

$$\begin{aligned} H(z) &= \mathbf{C}^T (z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} + d \\ &= \frac{(b_1 c_1 + b_2 c_2)z + b_1 c_2 a_{21} + b_2 c_1 a_{12} - b_1 c_1 a_{22} - b_2 c_2 a_{11}}{D(z)} + d, \end{aligned} \quad (4.87)$$

with

$$D(z) = z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21}). \quad (4.88)$$

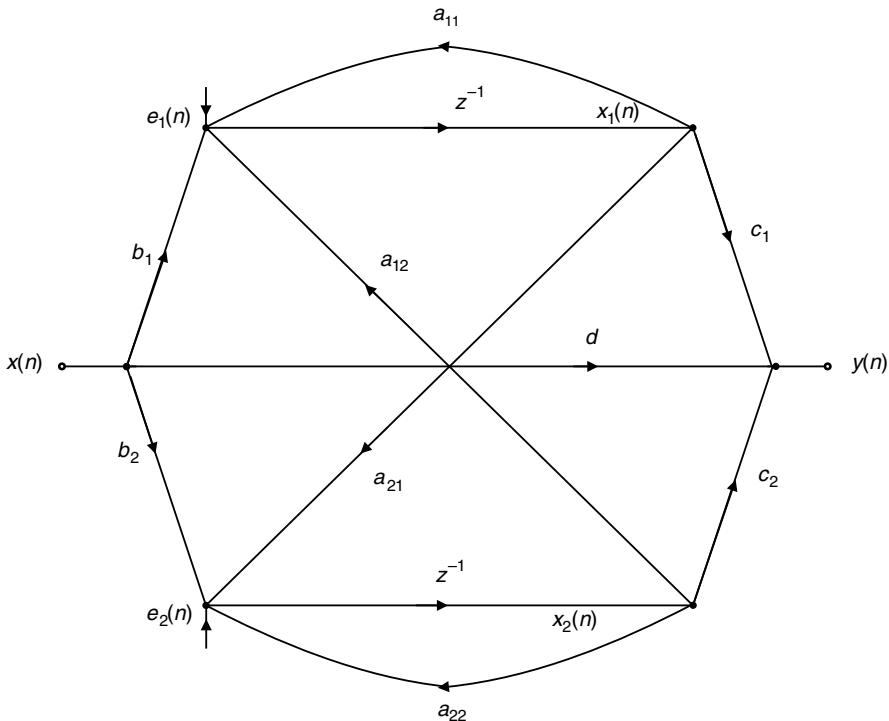


Fig. 4.23. State variable network.

The transfer functions required for computing the desired sensitivity functions can be obtained as special cases of the general transfer function $H(z)$. For instance, the transfer function from the filter input to the state $x_1(n)$ is obtained by setting $c_1 = 1$, $c_2 = 0$, and $d = 0$ in Equation (4.87), leading to

$$F_1(z) = \frac{X_1(z)}{X(z)} = \frac{b_1 z + (b_2 a_{12} - b_1 a_{22})}{D(z)}. \quad (4.89)$$

The transfer function from the filter input to state $x_2(n)$ is obtained by setting $c_1 = 0$, $c_2 = 1$, and $d = 0$ in Equation (4.87), resulting in

$$F_2(z) = \frac{X_2(z)}{X(z)} = \frac{b_2 z + (b_1 a_{21} - b_2 a_{11})}{D(z)}. \quad (4.90)$$

Using $b_1 = 1$, $b_2 = 0$, and $d = 0$ in Equation (4.87), one determines the transfer function from state $x_1(n)$ to the filter output, such that

$$G_1(z) = \frac{Y(z)}{E_1(z)} = \frac{c_1 z + (c_2 a_{21} - c_1 a_{22})}{D(z)}. \quad (4.91)$$

Finally, the transfer function from state $x_2(n)$ to the filter output is

$$G_2(z) = \frac{Y(z)}{E_2(z)} = \frac{c_2 z + (c_1 a_{12} - c_2 a_{11})}{D(z)}, \quad (4.92)$$

which is determined by setting $b_1 = 0$, $b_2 = 1$, and $d = 0$ in Equation (4.87). The required sensitivities are then

$$S_{a_{11}}^{H(z)} = F_1(z)G_1(z) \quad (4.93)$$

$$S_{a_{22}}^{H(z)} = F_2(z)G_2(z) \quad (4.94)$$

$$S_{a_{12}}^{H(z)} = F_2(z)G_1(z) \quad (4.95)$$

$$S_{a_{21}}^{H(z)} = F_1(z)G_2(z). \quad (4.96)$$

△

Example 4.6. Given the digital filter structure of Figure 4.24:

- (a) Generate its transposed realization.
- (b) Derive the structure to compute the sensitivity of the transfer function with respect to multiplier coefficient λ_1 .

Solution

- (a) Starting from the structure of Figure 4.24, we can obtain the transposed lattice structure by changing the branches' directions, turning the summations' nodes into distribution nodes and vice versa, and redrawing the network so that the input is placed at the right-hand side, as given in Figure 4.25.

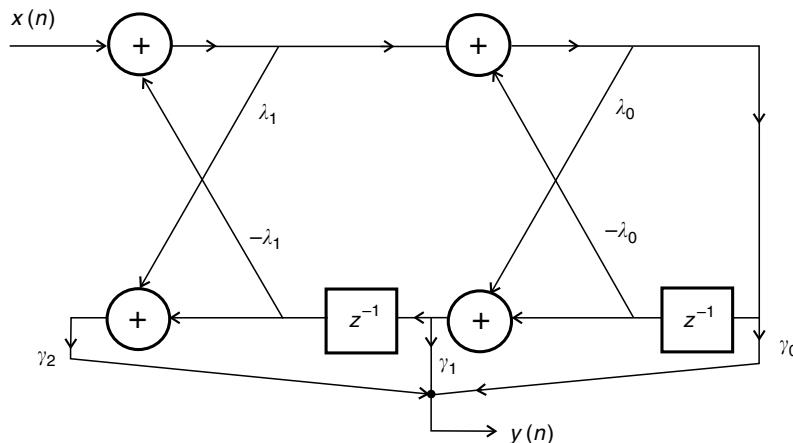


Fig. 4.24. Second-order lattice structure in Example 4.6.

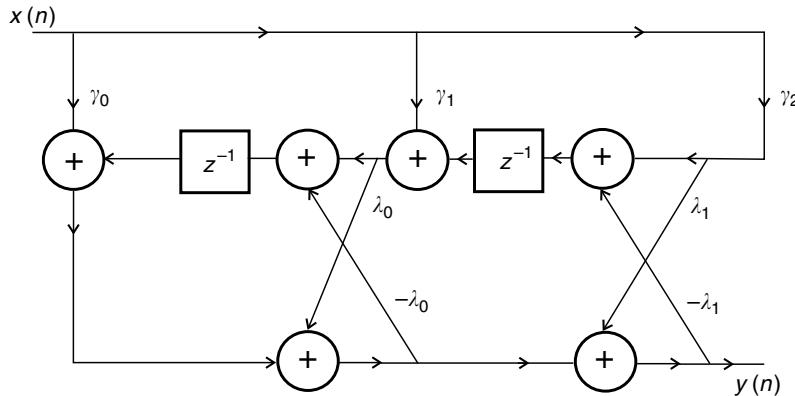


Fig. 4.25. Transposed lattice structure in Example 4.6.

- (b) For the sensitivity calculation, we must first note that the multiplier λ_1 appears in the network also as $-\lambda_1$. Actually, if they were two different multipliers λ_1 and $\bar{\lambda}_1$ respectively, then we would have that

$$\Delta H(z) = \frac{\partial H(z)}{\partial \lambda_1} \Delta \lambda_1 + \frac{\partial H(z)}{\partial \bar{\lambda}_1} \Delta \bar{\lambda}_1. \quad (4.97)$$

Since $\bar{\lambda}_1 = -\lambda_1$, we have that $\Delta \bar{\lambda}_1 = -\Delta \lambda_1$, and thus

$$\frac{\Delta H(z)}{\Delta \lambda_1} = \frac{\partial H(z)}{\partial \lambda_1} - \frac{\partial H(z)}{\partial \bar{\lambda}_1}. \quad (4.98)$$

As indicated in Figure 4.22, from Equation (4.85), if multiplier λ_1 goes from node 4 to 3 and multiplier $-\lambda_1$ goes from node 4' to 3', then the above equation becomes

$$\frac{\Delta H(z)}{\Delta \lambda_1} = H_{14}(z)H_{32}(z) - H_{14'}(z)H_{3'2}(z). \quad (4.99)$$

We now build a network to compute the above equation. The upper subnetwork of Figure 4.26 computes two outputs: one equal to $H_{14}(z)X(z)$ and another equal to $H_{14'}(z)X(z)$. We then use the lower subnetwork of Figure 4.26 to compute $H_{32}(z)$ and $H_{3'2}(z)$. The output $H_{14}(z)X(z)$ is input to node 3 and the output $H_{14'}(z)X(z)$ is multiplied by -1 and input to node 3' (note the -1 multiplier in Figure 4.26). The output of the network in Figure 4.26 is then $H_{14}(z)H_{32}(z)X(z) - H_{14'}(z)H_{3'2}(z)X(z)$. From Equation (4.99), this implies that its transfer function is equal to $\partial H(z)/\partial \lambda_1$. \triangle

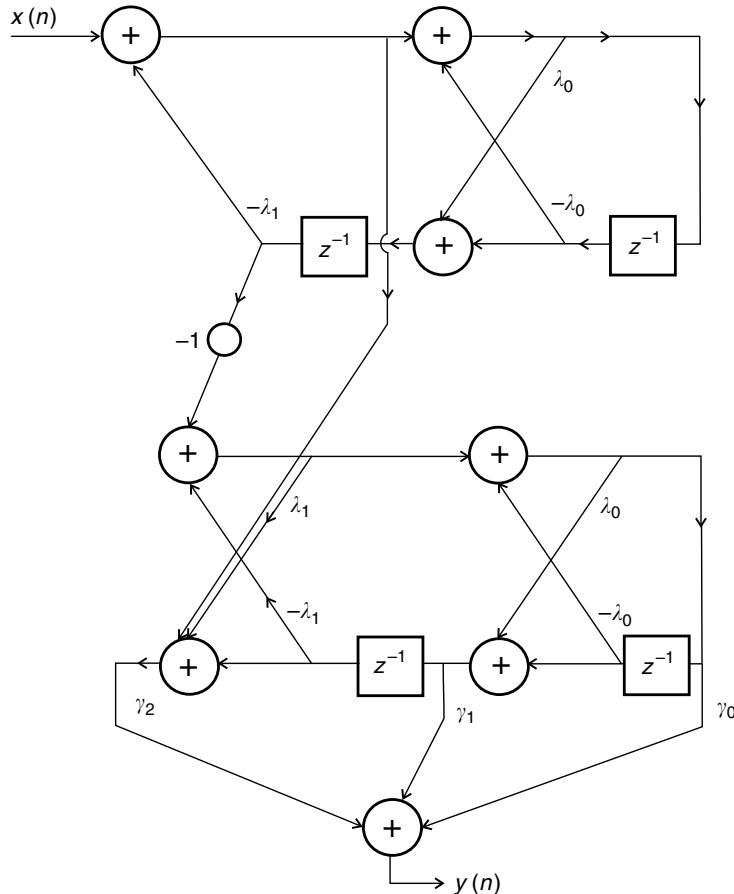


Fig. 4.26. Derivative structure in Example 4.6.

4.7 Useful building blocks

In this section, several building blocks with particularly attractive features are presented and briefly analyzed.

4.7.1 Second-order building blocks

The typical second-order transfer functions resulting from classical approximation methods are lowpass, bandpass, highpass, lowpass notch, highpass notch, and allpass. The transfer functions discussed below are special cases where the numerator polynomial is constrained to have its zeros either on the unit circle, where the zeros are more effective in shaping the magnitude response, or are reciprocals of the poles, as in the allpass case.

- *Lowpass*

$$H(z) = \frac{(z+1)^2}{z^2 + m_1 z + m_2}. \quad (4.100)$$

The zeros are placed at $z = -1$, leading to trivial coefficients in the numerator. Typically, the magnitude response will be increasing close to $z = 1$, it will reach a maximum value at the frequency corresponding directly to the poles' angles, and then it will decrease to reach a zero value at $z = -1$, as illustrated in Figure 4.27a.

- *Bandpass*

$$H(z) = \frac{z^2 - 1}{z^2 + m_1 z + m_2} = \frac{(z-1)(z+1)}{z^2 + m_1 z + m_2} \quad (4.101)$$

In this case, the zeros are placed at $z = \pm 1$, also leading to trivial coefficients in the numerator. Typically, the magnitude response will be zero at $z = \pm 1$ and will reach a maximum value at a frequency directly related to the poles' angles, as depicted in Figure 4.27b.

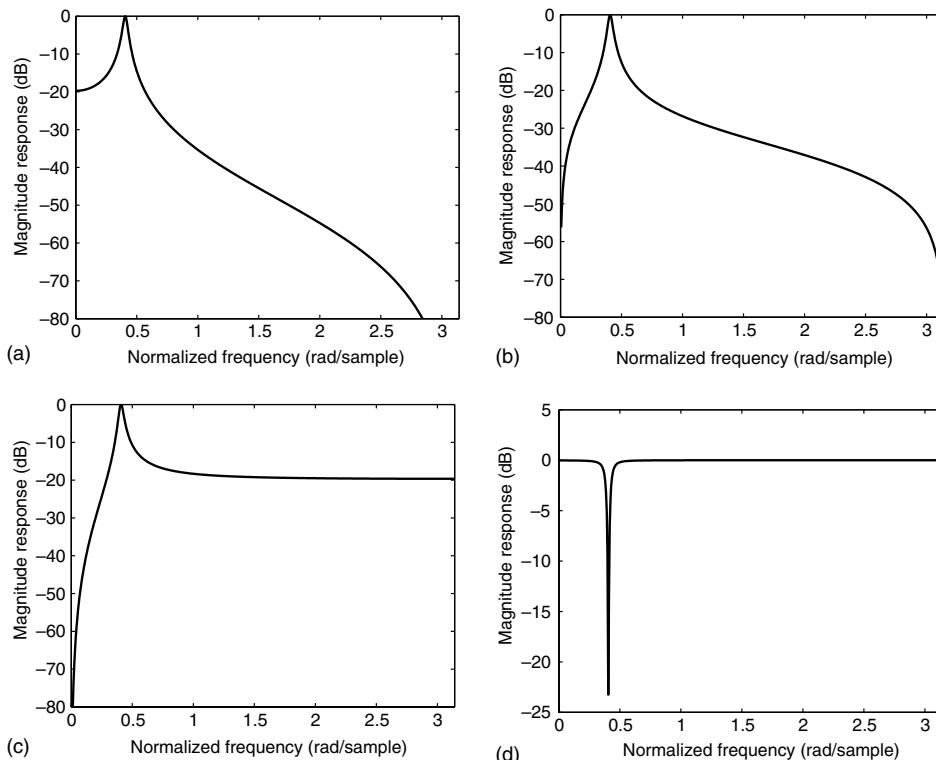


Fig. 4.27.

Magnitude responses of normalized, standard, second-order blocks with $m_1 = -1.8$ and $m_2 = 0.96$: (a) lowpass; (b) bandpass; (c) highpass; (d) notch.

- *Highpass*

$$H(z) = \frac{(z - 1)^2}{z^2 + m_1 z + m_2}. \quad (4.102)$$

As can be observed in Figure 4.27c, the zeros are placed at $z = 1$, so that all numerator coefficients are simple to implement. The magnitude response will be decreasing close to $z = -1$, it will reach a maximum value at a frequency directly related to the poles' angles, and then it will decrease to reach a zero value at $z = 1$.

- *Notch*

$$H(z) = \frac{z^2 + (m_1/\sqrt{m_2})z + 1}{z^2 + m_1 z + m_2}. \quad (4.103)$$

The zeros are placed on the unit circle with angles coinciding with the poles' angles, whereas the poles are obviously placed inside the unit circle, as in all other building blocks discussed here (this requires that $m_2 < 1$). An example is shown in Figure 4.27d.

- *Lowpass/highpass notch*

$$H(z) = \frac{z^2 + m_3 z + 1}{z^2 + m_1 z + m_2}. \quad (4.104)$$

The zero at positive frequency is placed on the unit circle with smaller positive angle than the pole positive angle in the highpass case and with larger positive angle than the pole positive angle in the lowpass case. Figures 4.28a and 4.28b show typical magnitude responses of lowpass and highpass notch filters respectively.

- *Allpass*

$$H(z) = \frac{m_2 z^2 + m_1 z + 1}{z^2 + m_1 z + m_2}. \quad (4.105)$$

For allpass filters the zeros are reciprocals of the poles; that is, if p_1 and p_1^* are the stable filter poles, then $z_1^* = 1/p_1$ and $z_1 = 1/p_1^*$ are the zeros. Note that from Equation (4.105)

$$H(z) = z^2 \frac{m_2 + m_1 z^{-1} + z^{-2}}{z^2 + m_1 z + m_2} = z^2 \frac{A(z^{-1})}{A(z)} \quad (4.106)$$

and the magnitude response is

$$|H(e^{j\omega})| = \frac{|A(e^{-j\omega})|}{|A(e^{j\omega})|} = \frac{|A^*(e^{j\omega})|}{|A(e^{j\omega})|} = 1, \quad (4.107)$$

since m_1 and m_2 are real. The magnitude and phase responses of an allpass filter are shown in Figures 4.28c and 4.28d respectively. Such blocks are usually employed in delay equalizers, since they modify the phase without changing the magnitude.

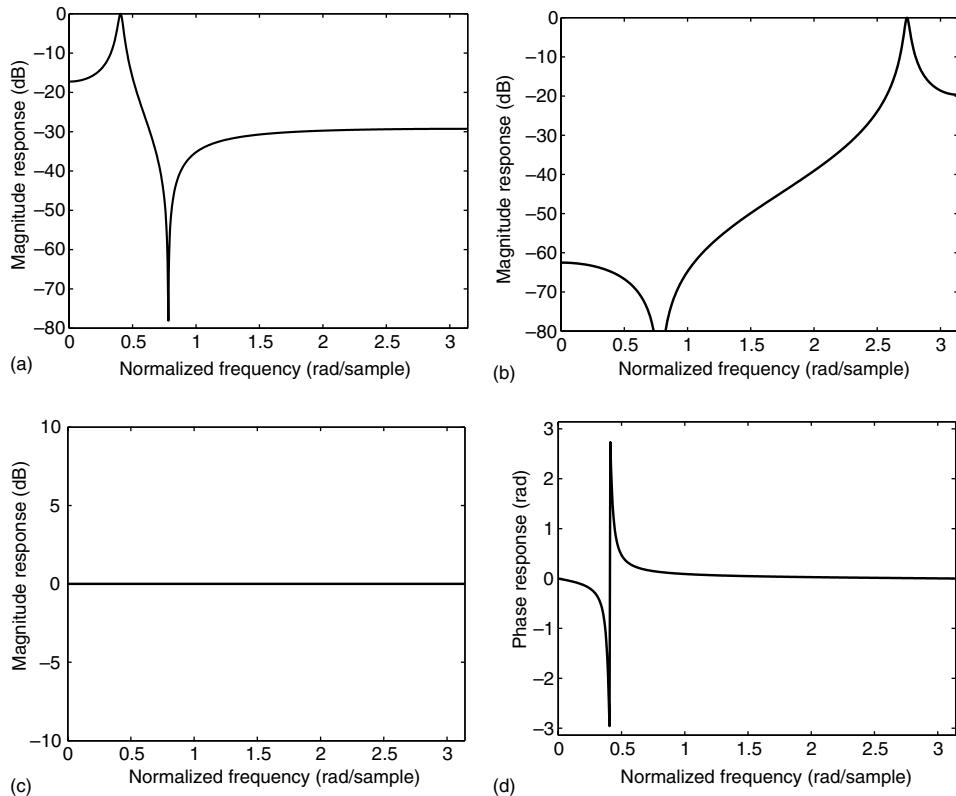


Fig. 4.28. Magnitude (and phase) responses for normalized, standard, second-order blocks: (a) lowpass notch with $m_1 = -1.8$, $m_2 = 0.96$, and $m_3 = -1.42$; (b) highpass notch with $m_1 = 1.8$, $m_2 = 0.96$, and $m_3 = -1.42$; (c) allpass (magnitude) with $m_1 = -1.8$ and $m_2 = 0.96$; (d) allpass (phase) with $m_1 = -1.8$ and $m_2 = 0.96$.

4.7.2 Digital oscillators

A realization of a digital oscillator has the transfer function

$$H(z) = \frac{z \sin(\omega_0)}{z^2 - 2 \cos(\omega_0)z + 1}, \quad (4.108)$$

where the poles are placed exactly on the unit circle. According to Table 2.1, the impulse response for this system is an oscillation of the form $\sin(\omega_0 n)u(n)$, as illustrated in Figure 4.29 for $\omega_0 = 7\pi/10$. Note that in this case the self-sustained oscillation does not look like a simple sinusoid, because the sampling frequency is not a multiple of the oscillation frequency.

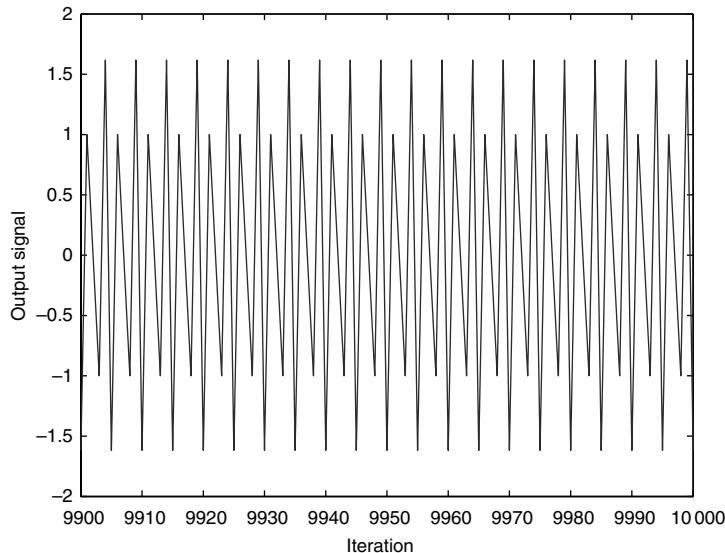


Fig. 4.29. Example of a digital oscillator output.

4.7.3 Comb filter

The comb filter is characterized by a magnitude response with multiple identical passbands. This device is a very useful building block for digital signal processing, finding applications in instrument synthesis in audio and harmonics removal (including DC), among others. The main task of a comb filter is to place equally spaced zeros on the unit circle, as illustrated in the following example.

Example 4.7. For the first-order network seen in Figure 4.30:

- Determine the corresponding transfer function.
- Replace z^{-1} by z^{-L} and show its transposed version.
- For the network obtained in item (b), determine the pole–zero constellation when $L = 8$ and $a = 0.5$, plotting the resulting frequency response.

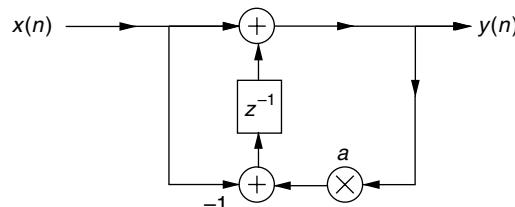


Fig. 4.30. Comb filter structure in Example 4.7.

Solution

(a) The transfer function of the first-order comb filter is

$$H(z) = \frac{1 - z^{-1}}{1 - az^{-1}}, \quad (4.109)$$

which has a zero at $z = 1$ and a real pole at $z = a$.

(b) The transposed realization is depicted in Figure 4.31, with z^{-1} replaced by z^{-L} . The corresponding transfer function is given by

$$H(z) = \frac{1 - z^{-L}}{1 - az^{-L}}. \quad (4.110)$$

(c) The pole–zero constellation associated with Equation (4.110) consists of L equally spaced zeros on the unit circle placed at $z = e^{j2\pi/L}$ with L poles placed at the same angles but on a circle with radius $a^{1/L}$. For $L = 8$ the pole–zero constellation of the comb filter is depicted in Figure 4.32.

Figures 4.33a and 4.33b show the magnitude and phase responses respectively of the comb filter, where the effects of the equally spaced zeros can be observed. The

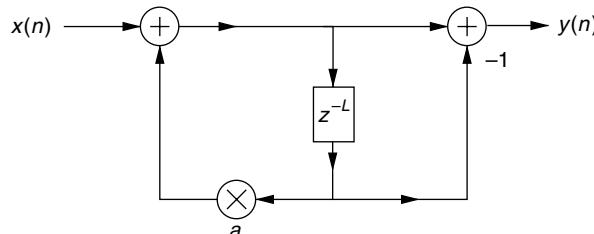


Fig. 4.31. Transposed comb filter in Example 4.7.

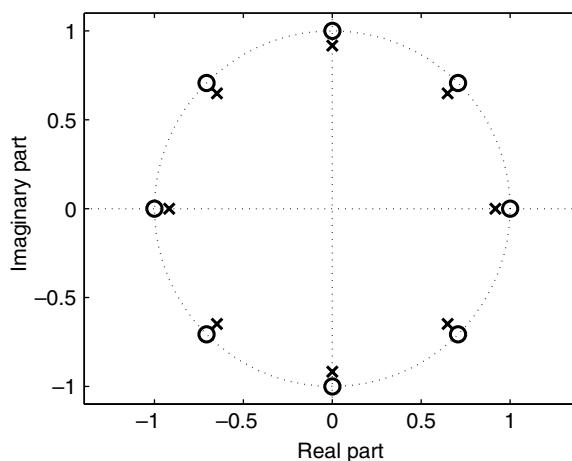


Fig. 4.32. Pole-zero constellation of comb filter with $L = 8$ in Example 4.7.

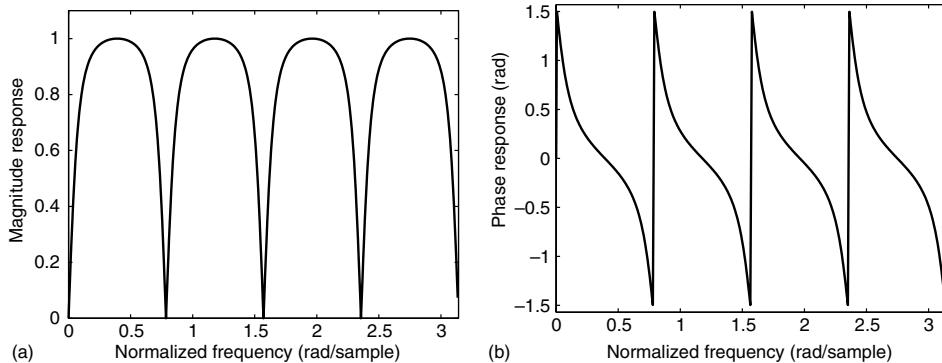


Fig. 4.33. Frequency response of the normalized comb filter in Example 4.7: (a) magnitude response; (b) phase response.

transitions between the peaks and valleys of the magnitude response are related to the value of a . A sharper magnitude response and a more nonlinear phase result the closer the value of a is to one. In particular, for $a = 0$, the comb filter becomes a linear-phase FIR filter. This effect is further explored in Exercise 4.25. \triangle

4.8 Do-it-yourself: digital filters

Experiment 4.1

Consider the direct-form transfer function

$$H(z) = \frac{z^6 + z^5 + z^4 + z^3 + z^2 + z + 1}{z^6 + 3z^5 + \frac{121}{30}z^4 + \frac{92}{30}z^3 + \frac{41}{30}z^2 + \frac{1}{3}z + \frac{1}{30}}. \quad (4.111)$$

One can easily find the poles and zeros of such a function in MATLAB using the command lines

```
num = [1 1 1 1 1 1];
den = [1 3 121/30 92/30 41/30 1/3 1/30];
[zc, pc, kc] = tf2zp(num, den);
```

leading to the results shown in Table 4.2.

In general, the `tf2zp` command groups the complex poles and zeros in conjugate pairs. However, as one can see in the pole column of Table 4.2, these complex-conjugate pairs must be sorted out from the real roots to compose the second-order blocks of a cascade realization with real coefficients. This is automatically done by the `zp2sos` command, whose usage is exemplified below:

```
Hcascade = zp2sos(zc, pc, kc)
```

Table 4.2.Zeros and poles of transfer function $H(z)$ in Experiment 4.1.

Zeros	Poles
$0.6235 + 0.7818j$	$-0.5000 + 0.5000j$
$0.6235 - 0.7818j$	$-0.5000 - 0.5000j$
$-0.9010 + 0.4339j$	-0.7236
$-0.9010 - 0.4339j$	$-0.5000 + 0.2887j$
$-0.2225 + 0.9749j$	$-0.5000 - 0.2887j$
$-0.2225 - 0.9749j$	-0.2764

yielding

```
Hcascade =
1.0000  -1.2470  1.0000  1.0000  1.0000  0.2000
1.0000   0.4450  1.0000  1.0000  1.0000  0.3333
1.0000   1.8019  1.0000  1.0000  1.0000  0.5000
```

which corresponds to the realization

$$H(z) = \frac{z^2 - 1.2470z + 1}{z^2 + z + \frac{1}{3}} \frac{z^2 + 0.4450z + 1}{z^2 + z + \frac{1}{3}} \frac{z^2 + 1.8019z + 1}{z^2 + z + \frac{1}{2}}. \quad (4.112)$$

The parallel realization of a given transfer function can be determined with the aid of the `residue` command that expresses $H(z)$ as the sum

$$H(z) = \frac{r_1}{z - p_1} + \frac{r_2}{z - p_2} + \cdots + \frac{r_N}{z - p_N} + k, \quad (4.113)$$

where N is the number of poles and the parameters r_i , p_i , and k are the outputs of

```
[rp, pp, kp] = residue(num, den);
```

Once again, one needs to sort out the pairs of complex-conjugate poles from the real ones in `pp` to determine the second-order parallel blocks with strictly real coefficients. This time, however, we cannot rely on the `zp2sos` command, which is suitable only for the cascade decomposition. The solution is to employ the `cplxpairs` command, which places the real roots after all complex pairs, and rearrange the residue vector `rp` accordingly, to allow the proper combination of residues and poles to form the second-order terms, as in the script below:

```
N = length(pp);
pp2 = cplxpairs(pp);
rp2 = zeros(N,1);
for i = 1:N,
    rp2(find(pp2 == pp(i)),1) = rp(i);
end;
num_blocks = ceil(N/2);
```

```

Hparallel = zeros(num_blocks, 6);
for count_p = 1:num_blocks,
    if length(pp2) ~= 1,
        Hparallel(count_p, 2) = rp2(1)+rp2(2);
        Hparallel(count_p, 3) = -rp2(1)*pp2(2)-rp2(2)*pp2(1);
        Hparallel(count_p, 5) = -pp2(1)-pp2(2);
        Hparallel(count_p, 6) = pp2(1)*pp2(2);
        rp2(1:2) = []; pp2(1:2) = [];
    else,
        Hparallel(count_p, 2) = rp2(1);
        Hparallel(count_p, 5) = -pp2(1);
    end;
    Hparallel(count_p, 4) = 1;
end;
Hparallel = real(Hparallel);

```

yielding, for this experiment:

```

Hparallel =
0   10   17.5000   1   1   0.5000
0  -20  -38.3333   1   1   0.3333
0     8   21.8000   1   1   0.2000

```

which, taking $k_p = 1$ also into consideration, corresponds to the parallel decomposition

$$H(z) = \frac{10z + 17.5}{z^2 + z + \frac{1}{2}} - \frac{20z + 38.3333}{z^2 + z + \frac{1}{3}} + \frac{8z + 21.8}{z^2 + z + \frac{1}{5}} + 1. \quad (4.114)$$

A state-space description corresponding to a given transfer function is easily determined in MATLAB with the command

```
[A, B, C, D] = tf2ss(num, den);
```

which, for $H(z)$ as given in Equation (4.111), results in

$$A = \begin{bmatrix} -3 & -\frac{121}{30} & -\frac{92}{30} & -\frac{41}{30} & -\frac{1}{3} & -\frac{1}{30} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad (4.115)$$

$$B = [1 \ 0 \ 0 \ 0 \ 0 \ 0]^T, \quad (4.116)$$

$$C = \begin{bmatrix} -2 & -\frac{91}{30} & -\frac{62}{30} & -\frac{11}{30} & \frac{2}{3} & \frac{29}{30} \end{bmatrix}, \quad (4.117)$$

$$D = 1. \quad (4.118)$$

4.9 Digital filter forms with MATLAB

As has been seen in this chapter, there are various different forms for representing a given transfer function. These include the direct form, the cascade form, the parallel form, and the state-space form. The MATLAB Signal Processing Toolbox has a series of commands that are suitable for transforming a given representation into another one of interest. Such commands are summarized in Table 4.3 and explained in detail below.

- `tf2zp`: Converts the direct form into the zero-pole-gain form, inverting the operation of `zp2tf`. The zero-pole-gain form is described by

$$H(z) = k \frac{[z - Z(1)][z - Z(2)] \cdots [z - Z(M)]}{[z - P(1)][z - P(2)] \cdots [z - P(N)]}, \quad (4.119)$$

where k is a gain factor, $Z(1), Z(2), \dots, Z(M)$ is the set of filter zeros, and $P(1), P(2), \dots, P(N)$ is the set of filter poles.

Input parameters: vectors with the numerator b and denominator coefficients a .

Output parameters: column vectors of filter zeros z and poles p , and the filter gain factor k .

Example:

```
b=[1 0.6 -0.16]; a=[1 0.7 0.12];
[z,p,k]=tf2zp(b,a);
```

- `zp2tf`: Converts the zero-pole-gain form (see `tf2zp` command) into the direct form, inverting the operation of `tf2zp`.

Input parameters: column vectors of filter zeros z and poles p , and the filter gain factor k .

Output parameters: vectors with the numerator b and denominator coefficients a .

Example:

```
z=[-0.8 0.2]'; p=[-0.4 -0.3]'; k=1;
[num,den]=zp2tf(z,p,k);
```

Table 4.3.

List of MATLAB commands for transforming digital filter representations.

	Direct	Zero-pole	Cascade	Parallel	State-space
Direct		<code>tf2zp</code> <code>roots</code>		<code>residuez</code>	<code>tf2ss</code>
Zero-pole	<code>zp2tf</code> <code>poly</code>		<code>zp2sos</code>		<code>zp2ss</code>
Cascade	<code>sos2tf</code>	<code>sos2zp</code>			<code>sos2ss</code>
Parallel	<code>residuez</code>				
State-space	<code>ss2tf</code>	<code>ss2zp</code>	<code>ss2sos</code>		

- **roots**: Determines the roots of a polynomial. This command also can be used to decompose a given transfer function into the zero–pole–gain format.

Input parameter: a vector of polynomial coefficients.

Output parameter: a vector of roots.

Example:

```
r=roots([1 0.6 -0.16]);
```

- **poly**: Inverts the operation of **roots**; that is, given a set of roots, this command determines the monic polynomial associated with these roots.

Input parameter: a vector of roots.

Output parameter: a vector of polynomial coefficients.

Example:

```
pol=poly([-0.8 0.2]);
```

- **sos2tf**: Converts the cascade form into the direct form. Notice that there is no direct command that reverses this operation.

Input parameter: an $L \times 6$ matrix, **sos**, whose rows contain the coefficients of each second-order section of the form

$$H_k(z) = \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}} \quad (4.120)$$

such that

$$\text{sos} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}. \quad (4.121)$$

Output parameters: vectors **num** and **den** containing the numerator and denominator coefficients.

Example:

```
sos=[1 1 1 1 10 1; -2 3 1 1 0 -1];
[num,den]=sos2tf(sos);
```

- **residuez**: Performs the partial-fraction expansion in the z domain, if there are two input parameters. This command considers complex roots. To obtain a parallel expansion of a given transfer function, one should combine such roots in complex conjugate pairs with the **cplxpair** command to form second-order sections with solely real coefficients. The **residuez** command also converts the partial-fraction expansion back to original direct form, if there are three input parameters.

Input parameters: vectors of numerator coefficients **b** and denominator coefficients **a**.

Output parameters: vectors of residues **r** and poles **p**, and gain factor **k**.

Example:

```
b=[1 0.6 -0.16]; a=[1 0.7 0.12];
[r,p,k]=residuez(b,a);
```

Input parameters: vectors of residues r , poles p , and gain factor k .

Output parameters: vectors of numerator coefficients b and denominator coefficients a .

Example:

```
r=[14 -43/3]; p=[-0.4 -0.3]; k=4/3;
[b,a]=residuez(r,p,k);
```

- **cplxpair**: Rearranges the elements of a vector into complex conjugate pairs. The pairs are ordered by increasing real part. Real elements are placed after all complex pairs.

Input parameter: a vector of complex numbers.

Output parameter: a vector containing the ordered complex numbers.

Example:

```
Xord=cplxpair(roots([ 1 4 2 1 3 1 4]));
```

- **tf2ss**: Converts the direct form into the state-space form, inverting the operation of **ss2tf**.

Input parameters: as for command **tf2zp**.

Output parameters: the state-space matrix A , the input column vector B , the state row vector C , and the scalar D .

Example:

```
b=[1 0.6 -0.16]; a=[1 0.7 0.12];
[A,B,C,D]=tf2ss(b,a);
```

- **ss2tf**: Converts the state-space form into the direct form, inverting the operation of **tf2ss**.

Input parameters: the state-space matrix A , the input column vector B , the state row vector C , and the scalar D .

Output parameters: as for command **zp2tf**.

Example:

```
A=[-0.7 -0.12; 1 0]; b=[1 0]'; c=[-0.1 -0.28]; d=1;
[num,den]=ss2tf(A,b,c,d);
```

- **zp2sos**: Converts the zero-pole-gain form into the cascade form, inverting the operation of **sos2zp**. The **zp2sos** command can also order the resulting sections according to the pole positions with respect to the unit circle.

Input parameters: the same as for **zp2tf**, with the addition of a string, **up** (default) or **down**, that indicates the desired section ordering starts with the poles further away from or closer to the unit circle respectively.

Output parameter: as for the input parameter of the command **sos2tf**.

Example:

```
z=[1 1 j -j]; p=[0.9 0.8 0.7 0.6];
sos=zp2sos(z,p);
```

- **sos2zp:** Converts the cascade form into the zero–pole-gain form, inverting the operation of `zp2sos`.

Input parameter: as for command `sos2tf`.

Output parameter: as for command `tf2zp`.

Example:

```
sos=[1 1 1 1 10 1; -2 3 1 1 0 -1];
[z,p,k]=sos2zp(sos);
```

- **zp2ss:** Converts the zero–pole-gain form into the state-space form, inverting the operation of `ss2zp`.

Input parameters: as for command `zp2tf`, but with no restriction on the format of the zero and pole vectors.

Output parameters: as for command `tf2ss`.

Example:

```
z=[-0.8 0.2]; p=[-0.4 -0.3]; k=1;
[A,B,C,D]=zp2ss(z,p,k);
```

- **ss2zp:** Converts the state-space form into the zero–pole-gain form, inverting the operation of `zp2ss`.

Input parameters: as for command `ss2tf`.

Output parameters: as for command `tf2zp`.

Example:

```
A=[-0.7 -0.12; 1 0]; b=[1 0]'; c=[-0.1 -0.28]; d=1;
[z,p,k]=ss2zp(A,b,c,d);
```

- **sos2ss:** Converts the cascade form into the state-space form, inverting the operation of `ss2sos`.

Input parameter: as for command `sos2tf`.

Output parameters: as for command `tf2ss`.

Example:

```
sos=[1 1 1 1 10 1; -2 3 1 1 0 -1];
[A,B,C,D]=sos2ss(sos);
```

- **ss2sos:** Converts the state-space form into the cascade form, inverting the operation of `sos2ss`.

Input parameters: as for command `ss2tf`.

Output parameter: as for input parameter of command `sos2tf`.

Example:

```
A=[-0.7 -0.12; 1 0]; b=[1 0]'; c=[-0.1 -0.28]; d=1;
sos=ss2sos(A,b,c,d);
```

- **filter:** Performs signal filtering using the Type 2 canonic direct form for IIR filters (see Figure 4.13).

Input parameters: a vector `b` of numerator coefficients, a vector `a` of denominator coefficients, a vector `x` holding the signal to be filtered, and a vector `Zi` holding the initial conditions of the delays.

Output parameters: a vector y holding the filtered signal and a vector Zf holding the initial conditions of the delays.

Example:

```
a=[1 -0.22 -0.21 0.017 0.01];
b=[1 2 1]; Zi=[0 0 0 0];
x=randn(100,1); [y,Zf]=filter(b,a,x,Zi); plot(y);
```

4.10 Summary

In this chapter, some basic realizations for digital filters with FIRs or IIRs were presented. In particular, FIR filters with linear phase were introduced and their practical importance was highlighted. Other, more advanced, FIR and IIR structures will be discussed later in this book.

A procedure for analyzing digital networks in the time and frequency domains was devised. Following this, the state-space description was introduced.

The digital version of Tellegen's theorem was presented, as well as the reciprocity, inter-reciprocity, and transposition network properties. Then, with the help of Tellegen's theorem, a simple formulation for calculating the sensitivity of a given transfer function with respect to coefficient variations was provided.

This chapter also introduced some building blocks for FIR and IIR digital filters which are very often used in practical implementations. The Do-it-yourself section illustrated how to get alternative realizations for a given transfer function as a guided tour for the reader. Finally, the MATLAB functions related to basic digital filter representations were discussed.

4.11 Exercises

4.1 Give two distinct realizations for the transfer functions below:

$$(a) H(z) = 0.0034 + 0.0106z^{-2} + 0.0025z^{-4} + 0.0149z^{-6}.$$

$$(b) H(z) = \left(\frac{z^2-1.349z+1}{z^2-1.919z+0.923}\right) \left(\frac{z^2-1.889z+1}{z^2-1.937z+0.952}\right).$$

4.2 Write the equations that describe the networks in Figure 4.34, by numbering the nodes appropriately.

4.3 Show that:

(a) If $H(z)$ is a Type I filter, then $H(-z)$ is Type I.

(b) If $H(z)$ is a Type II filter, then $H(-z)$ is Type IV.

(c) If $H(z)$ is a Type III filter, then $H(-z)$ is Type III.

(d) If $H(z)$ is a Type IV filter, then $H(-z)$ is Type II.

4.4 Determine the transfer functions of the digital filters in Figure 4.34.

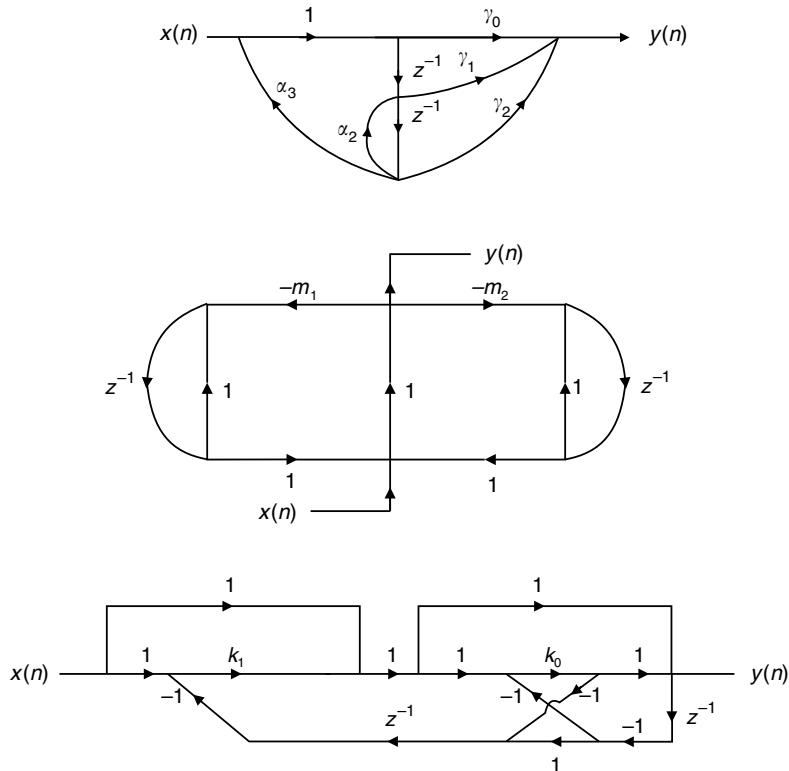


Fig. 4.34. Signal flowgraphs of three digital filters.

- 4.5 Show that the transfer function of a given digital filter is invariant with respect to a linear transformation of the state vector

$$\mathbf{x}(n) = \mathbf{T}\mathbf{x}'(n),$$

where \mathbf{T} is any $N \times N$ nonsingular matrix.

- 4.6 Describe the networks in Figure 4.34 using state variables.
 4.7 Implement the transfer function below using a parallel realization with the minimum number of multipliers:

$$H(z) = \frac{z^3 + 3z^2 + \frac{11}{4} + \frac{5}{4}}{(z^2 + \frac{1}{2}z + \frac{1}{2})(z + \frac{1}{2})}.$$

- 4.8 Given the realization depicted in Figure 4.35:
 (a) Show its state-space description.
 (b) Determine its transfer function.
 (c) Derive the expression for its magnitude response and interpret the result.

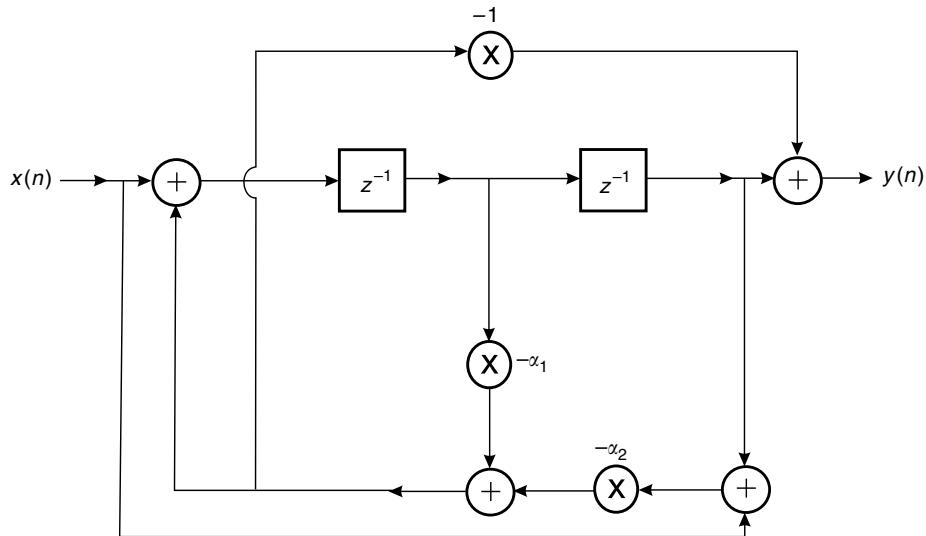


Fig. 4.35. Digital filter structure for Exercise 4.8.

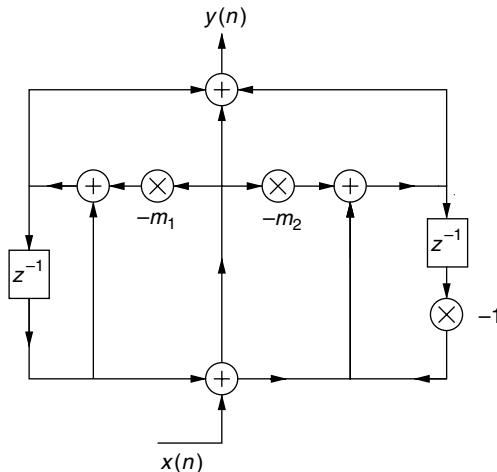


Fig. 4.36. Digital filter structure for Exercise 4.9.

- 4.9 Consider the digital filter of Figure 4.36:
- Determine its state-space description.
 - Compute its transfer function and plot the magnitude response.
- 4.10 Determine the transfer function of the digital filter shown in Figure 4.37 using the state-space formulation.
- 4.11 Given the digital filter structure of Figure 4.38:
- Determine its state-space description.
 - Compute its transfer function.

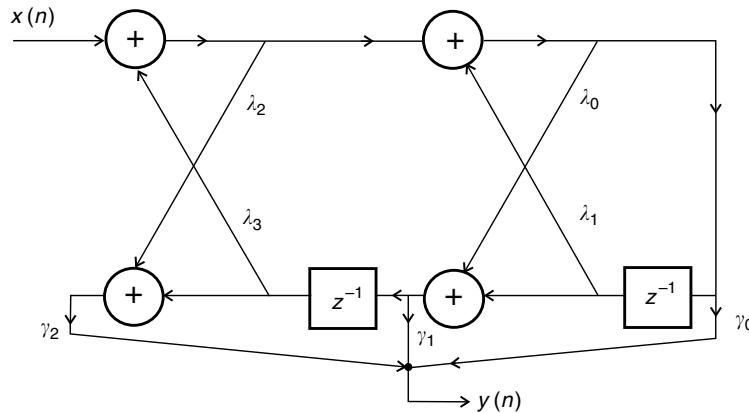


Fig. 4.37. Second-order lattice structure for Exercise 4.10.

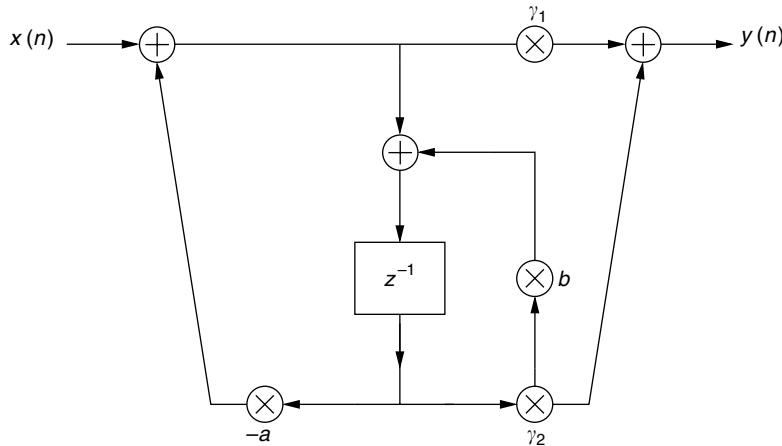


Fig. 4.38. Digital filter structure for Exercise 4.11.

- (c) Show its transposed circuit.
 - (d) Use this structure with $a = -b = \frac{1}{4}$ to design a filter with a unit DC gain to eliminate the frequency $\omega_s/2$, where ω_s represents the sampling frequency.
- 4.12 Given the digital filter structure of Figure 4.39:
- (a) Determine its transfer function.
 - (b) Generate its transposed realization.
- 4.13 Given the structure shown in Figure 4.40:
- (a) Determine the corresponding transfer function employing the state-space formulation.
 - (b) Generate its transposed realization.
 - (c) If $\gamma_0 = \gamma_1 = \gamma_2$ and $m_1 = m_2$, analyze the resulting transfer function.
- 4.14 Find the transpose for each network in Figure 4.34.

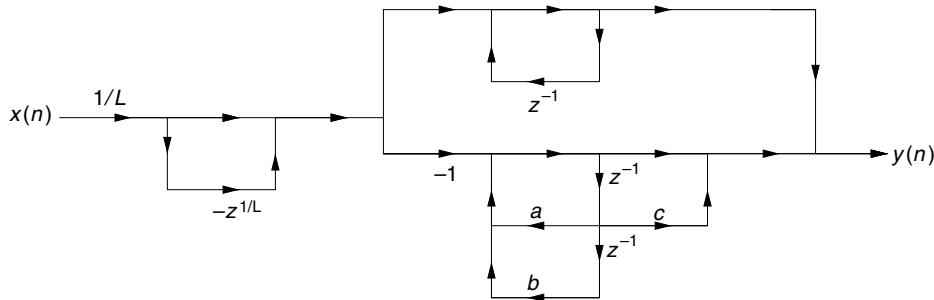


Fig. 4.39. Digital filter structure for Exercise 4.12.

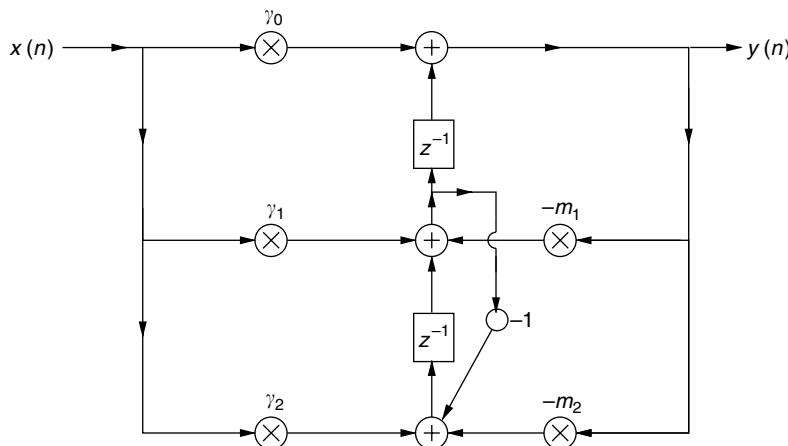


Fig. 4.40. Digital filter structure for Exercise 4.13.

- 4.15 Determine the sensitivity of the transfer functions determined in Exercise 4.4, for the filters given in Figure 4.34, with respect to each filter coefficient.
- 4.16 Determine the transfer function of the filter in Figure 4.41, considering the two possible positions for the switches.
- 4.17 Determine and plot the frequency response of the filter shown in Figure 4.41, considering the two possible positions for the switches.
- 4.18 Determine the impulse response of the filter shown in Figure 4.42.
- 4.19 Show that if two given networks are described by $Y_i = \sum_{j=1}^M T_{ij}X_j$ and $Y'_i = \sum_{j=1}^M T'_{ij}X'_j$, then these networks are interreciprocal if $T_{ji} = T'_{ij}$.
- 4.20 Some FIR filters present a rational transfer function:
 - (a) Show that the transfer function

$$H(z) = \frac{(r^{-1}z)^{-(M+1)} - 1}{re^{j2\pi/(M+1)}z^{-1} - 1}$$

corresponds to an FIR filter (Lyons, 2007).

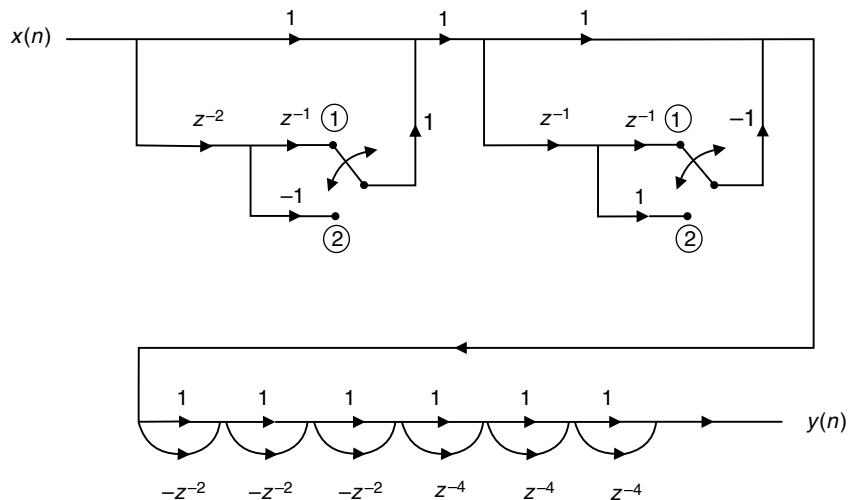


Fig. 4.41. Signal flowgraph of a digital filter.

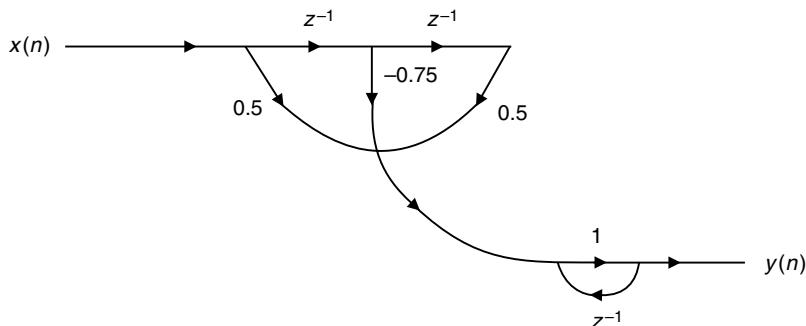


Fig. 4.42. Signal flowgraph of a digital filter.

- (b) Determine the operation performed by such a filter.
 (c) Discuss the general case when a rational transfer function corresponds to an FIR filter.
- 4.21 Plot the pole-zero constellation as well as the magnitude response of the transfer function of Exercise 4.20 for $M = 6, 7, 8$ and comment on the results.
- 4.22 Design second-order lowpass and highpass blocks, and combine them in cascade, to form a bandpass filter with passband $0.3 \leq \omega \leq 0.4$, where $\omega_s = 1$. Plot the resulting magnitude response.
- 4.23 Design second-order lowpass and highpass blocks, and combine them in parallel, to form a bandstop filter with stopband $0.25 \leq \omega \leq 0.35$, where $\omega_s = 1$. Plot the resulting magnitude response.
- 4.24 Design a second-order notch filter capable of eliminating a 10 Hz sinusoidal component when $\omega_s = 200$ rad/sample and show the resulting magnitude response.

- 4.25 For the comb filter of Figure 4.31, choose $L = 10$ and compute the magnitude and phase responses for the cases where $a = 0$, $a = 0.6$, and $a = 0.8$. Comment on the result.
- 4.26 For the comb filter of Equation (4.110), compute the value of the normalization factor that should be multiplied by the transfer function such that the maximum value of the magnitude response becomes 1.
- 4.27 An FIR filter with the transfer function

$$H(z) = (z^{-L} - 1)^N$$

with N and L integers is also a comb filter. Discuss the properties of this filter regarding its zero positions and sharpness.

- 4.28 Given the transfer function

$$H(z) = \frac{z[z - \cos(\omega_0)]}{z^2 - 2\cos(\omega_0)z + 1}.$$

- (a) Where are its poles located exactly on the unit circle?
- (b) Using $2\cos(\omega_0) = -2, 0, 1, 2$, and without any external excitation, place an initial value of 0.5 in one of the filter states and determine the output signal for a few hundred iterations. Comment on the observed results.
- 4.29 With a state-space structure, prove that, by choosing $a_{11} = a_{22} = \cos(\omega_0)$ and $a_{21} = -a_{12} = -\sin(\omega_0)$, the resulting oscillations in states $x_1(n)$ and $x_2(n)$ correspond to $\cos(\omega_0 n)$ and $\sin(\omega_0 n)$ respectively.
- 4.30 Let us revisit the state-space description of $H(z)$ as given in Experiment 4.1.
- (a) Use the command `eig` in MATLAB to determine the eigenvalues of the system matrix A and compare your results with the poles of $H(z)$ provided in Table 4.2.
- (b) Implement Equation (4.51) in MATLAB to determine the impulse response of the state-space description (4.115)–(4.118) and compare your result with the impulse response obtained with the `filter` command.
- 4.31 Create MATLAB commands to fill in the blanks of Table 4.3.

5.1 Introduction

In this chapter we will study the approximation schemes for digital filters with FIR and we will present the methods for determining the multiplier coefficients and the filter order, in such a way that the resulting frequency response satisfies a set of prescribed specifications.

In some cases, FIR filters are considered inefficient, in the sense that they require a high-order transfer function to satisfy the system requirements when compared with the order required by IIR digital filters. However, FIR digital filters do possess a few implementation advantages, such as a possible exact linear-phase characteristic and intrinsically stable implementations, when using nonrecursive realizations. In addition, the computational complexity of FIR digital filters can be reduced if they are implemented using fast numerical algorithms such as the FFT.

We start by discussing the ideal frequency response characteristics of commonly used FIR filters, as well as their corresponding impulse responses. We include lowpass, highpass, bandpass, and bandstop filters in the discussion, and also treat two other important filters, namely differentiators and Hilbert transformers.

We go on to discuss the frequency sampling and the window methods for approximating FIR digital filters, focusing on the rectangular, triangular, Bartlett, Hamming, Blackman, Kaiser, and Dolph–Chebyshev windows. In addition, the design of maximally flat FIR filters is addressed.

Following this, numerical methods for designing FIR filters are discussed. A unified framework for the general approximation problem is provided. The weighted least-squares (WLS) method is presented as a generalization of the rectangular window approach. We then introduce the Chebyshev (or minimax) approach as the most efficient form, with respect to the resulting filter order, to approximate FIR filters which minimize the maximum passband and stopband ripples. We also discuss the WLS–Chebyshev approach, which is able to combine the desired characteristics of high attenuation of the Chebyshev scheme with the low energy level of the WLS scheme in the filter stopband.

We conclude the chapter by discussing the use of MATLAB for designing FIR filters.

5.2 Ideal characteristics of standard filters

In this section we analyze the time and frequency response characteristics of commonly used FIR filters. First, we deal with lowpass, highpass, bandpass, and bandstop filters.

Then, two types of filter widely used in the field of digital signal processing, namely the differentiators and Hilbert transformers (Oppenheim & Schafer, 1975; Antoniou, 1993), are analyzed and their implementations as special cases of FIR digital filters are studied.

The behavior of a filter is usually best characterized by its frequency response $H(e^{j\omega})$. As seen in Chapter 4, a filter implementation is based on its transfer function $H(z)$ of the form

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n}. \quad (5.1)$$

The FIR filter design starts by calculating the coefficients $h(n)$ which will be used in one of the structures discussed in Section 4.2.

As seen in Section 2.8, the relationship between $H(e^{j\omega})$ and $h(n)$ is given by the following pair of equations:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-j\omega n} \quad (5.2)$$

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n} d\omega. \quad (5.3)$$

In what follows, we determine $H(e^{j\omega})$ and $h(n)$ related to ideal standard filters.

5.2.1 Lowpass, highpass, bandpass, and bandstop filters

The ideal magnitude responses of some standard digital filters are depicted in Figure 5.1.

For instance, the lowpass filter, as seen in Figure 5.1a, is described by

$$|H(e^{j\omega})| = \begin{cases} 1, & \text{for } |\omega| \leq \omega_c \\ 0, & \text{for } \omega_c < |\omega| \leq \pi \end{cases}. \quad (5.4)$$

Using Equation (5.3), the impulse response for the ideal lowpass filter is

$$h(n) = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega = \begin{cases} \frac{\omega_c}{\pi}, & \text{for } n = 0 \\ \frac{\sin(\omega_c n)}{\pi n}, & \text{for } n \neq 0 \end{cases}. \quad (5.5)$$

One should note that in the above inverse transform calculations we have supposed that the phase of the filter is zero. From Equation (4.12) in Section 4.2.3, we have that the phase of an FIR filter must be of the form $e^{-j\omega M/2}$, where M is an integer. Therefore, for M even, it suffices to shift the above impulse response by $M/2$ samples. However, for M odd, $M/2$

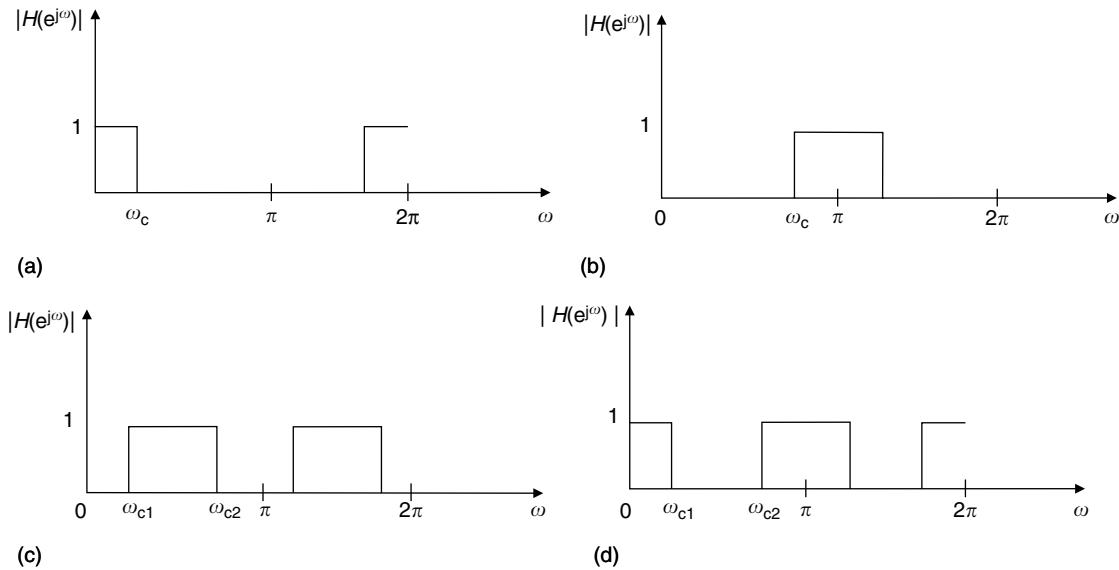


Fig. 5.1. Ideal magnitude responses: (a) lowpass; (b) highpass; (c) bandpass; (d) bandstop filters.

is not an integer, and the impulse response must be computed as

$$\begin{aligned}
 h(n) &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{-j\omega M/2} e^{j\omega n} d\omega \\
 &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega[n-(M/2)]} d\omega \\
 &= \frac{\sin\{\omega_c[n - (M/2)]\}}{\pi [n - (M/2)]} \tag{5.6}
 \end{aligned}$$

Likewise, for bandstop filters, the ideal magnitude response, depicted in Figure 5.1d, is given by

$$|H(e^{j\omega})| = \begin{cases} 1, & \text{for } 0 \leq |\omega| \leq \omega_{c_1} \\ 0, & \text{for } \omega_{c_1} < |\omega| < \omega_{c_2} \\ 1, & \text{for } \omega_{c_2} \leq |\omega| \leq \pi \end{cases} \quad (5.7)$$

Then, using Equation (5.3), the impulse response for such an ideal filter is

$$h(n) = \frac{1}{2\pi} \left[\int_{-\omega_{c_1}}^{\omega_{c_1}} e^{j\omega n} d\omega + \int_{\omega_{c_2}}^{\pi} e^{j\omega n} d\omega + \int_{-\pi}^{-\omega_{c_2}} e^{j\omega n} d\omega \right]$$

$$= \begin{cases} 1 + \frac{\omega_{c_1} - \omega_{c_2}}{\pi}, & \text{for } n = 0 \\ \frac{1}{\pi n} [\sin(\omega_{c_1}n) - \sin(\omega_{c_2}n)], & \text{for } n \neq 0. \end{cases} \quad (5.8)$$

Again, the above impulse response is valid only for zero phase. For nonzero linear phase, the discussion following Equation (5.5) applies (see Exercise 5.1).

Following an analogous reasoning, one can easily find the magnitude responses of the ideal highpass and bandpass filters, depicted in Figures 5.1b and 5.1c respectively. Table 5.1 (see Section 5.2.4) includes the ideal magnitude responses and their respective impulse responses for the ideal zero-phase lowpass, highpass, bandpass, and bandstop filters. The nonzero-phase case is considered in Exercise 5.1.

5.2.2 Differentiators

An ideal discrete-time differentiator is a linear system in which, when samples of a band-limited continuous signal are used as input, the output samples represent the derivative of the continuous signal. More precisely, given a continuous-time signal $x_a(t)$ band-limited to $[-\pi/T, \pi/T]$, when its corresponding sampled version $x(n) = x_a(nT)$ is input to an ideal differentiator, it produces the output signal $y(n)$ such that

$$y(n) = \left. \frac{dx_a(t)}{dt} \right|_{t=nT}. \quad (5.9)$$

If the Fourier transform of the continuous-time signal is denoted by $X_a(j\Omega)$, then we have that the Fourier transform of its derivative is $j\Omega X_a(j\Omega)$, as can be deduced from Equation (2.206). Therefore, an ideal discrete-time differentiator is characterized by a frequency response, up to a multiplicative constant, of the form

$$H(e^{j\omega}) = j\omega, \text{ for } -\pi \leq \omega < \pi. \quad (5.10)$$

The magnitude and phase responses of a differentiator are depicted in Figure 5.2.

Using Equation (5.3), the corresponding impulse response is given by

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} j\omega e^{j\omega n} d\omega = \begin{cases} 0, & \text{for } n = 0 \\ \frac{1}{2\pi} \left[e^{j\omega n} \left(\frac{\omega}{n} - \frac{1}{jn^2} \right) \right] \Big|_{-\pi}^{\pi} = \frac{(-1)^n}{n}, & \text{for } n \neq 0 \end{cases} \quad (5.11)$$

One should note that, comparing Equation (5.10) with Equation (4.16), if a differentiator is to be approximated by a linear-phase FIR filter, then one should necessarily use either a Type III or a Type IV form. In fact, using an argument similar to the one following Equation (5.5), we can see that Equation (5.11) can be used only in the case of Type III filters. For Type IV filters, we must perform a derivation similar to the one in Equation (5.6) (see Exercise 5.1).

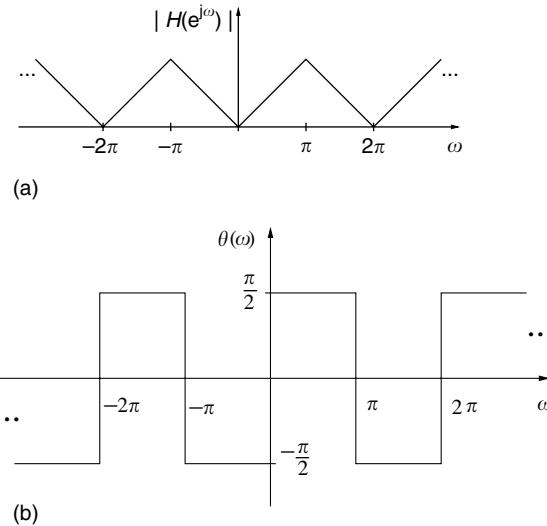


Fig. 5.2. Characteristics of an ideal discrete-time differentiator: (a) magnitude response; (b) phase response.

5.2.3 Hilbert transformers

The Hilbert transformer is a system that, when fed with the real part of a complex signal whose Fourier transform is null for $-\pi \leq \omega < 0$, produces at its output the imaginary part of the complex signal. In other words, let $x(n)$ be the inverse Fourier transform of $X(e^{j\omega})$ such that $X(e^{j\omega}) = 0$, $-\pi \leq \omega < 0$. The real and imaginary parts of $x(n)$, $x_R(n)$, and $x_I(n)$, are defined as

$$\left. \begin{aligned} \operatorname{Re}\{x(n)\} &= \frac{x(n) + x^*(n)}{2} \\ \operatorname{Im}\{x(n)\} &= \frac{x(n) - x^*(n)}{2j} \end{aligned} \right\}. \quad (5.12)$$

Hence, their Fourier transforms, $X_R(e^{j\omega}) = \mathcal{F}\{\operatorname{Re}\{x(n)\}\}$ and $X_I(e^{j\omega}) = \mathcal{F}\{\operatorname{Im}\{x(n)\}\}$, are

$$\left. \begin{aligned} X_R(e^{j\omega}) &= \frac{X(e^{j\omega}) + X^*(e^{-j\omega})}{2} \\ X_I(e^{j\omega}) &= \frac{X(e^{j\omega}) - X^*(e^{-j\omega})}{2j} \end{aligned} \right\}. \quad (5.13)$$

For $-\pi \leq \omega < 0$, since $X(e^{j\omega}) = 0$, we have that

$$\left. \begin{aligned} X_R(e^{j\omega}) &= \frac{X^*(e^{-j\omega})}{2} \\ X_I(e^{j\omega}) &= j \frac{X^*(e^{-j\omega})}{2} \end{aligned} \right\} \quad (5.14)$$

and, for $0 \leq \omega < \pi$, since $X^*(e^{-j\omega}) = 0$, we also have that

$$\left. \begin{aligned} X_R(e^{j\omega}) &= \frac{X(e^{j\omega})}{2} \\ X_I(e^{j\omega}) &= -j \frac{X(e^{j\omega})}{2} \end{aligned} \right\}. \quad (5.15)$$

From Equations (5.14) and (5.15), we can easily conclude that

$$\left. \begin{aligned} X_I(e^{j\omega}) &= -jX_R(e^{j\omega}), & \text{for } 0 \leq \omega < \pi \\ X_I(e^{j\omega}) &= jX_R(e^{j\omega}), & \text{for } -\pi \leq \omega < 0 \end{aligned} \right\}. \quad (5.16)$$

These equations provide a relation between the Fourier transforms of the real and imaginary parts of a signal whose Fourier transform is null for $-\pi \leq \omega < 0$. Thus, this implies that the ideal Hilbert transformer has the following transfer function:

$$H(e^{j\omega}) = \begin{cases} -j, & \text{for } 0 \leq \omega < \pi \\ j, & \text{for } -\pi \leq \omega < 0 \end{cases}. \quad (5.17)$$

The magnitude and phase components of such a frequency response are depicted in Figure 5.3.

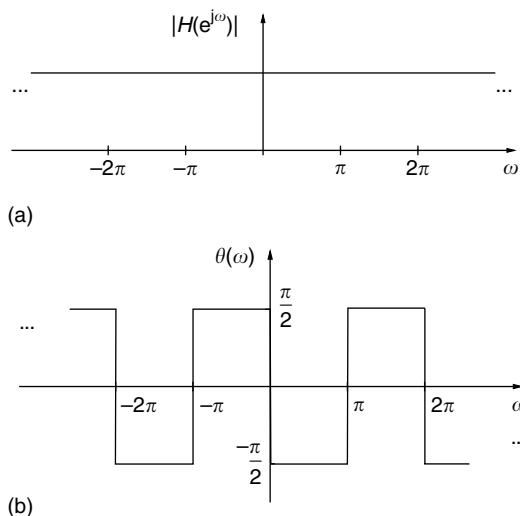


Fig. 5.3. Characteristics of an ideal Hilbert transformer: (a) magnitude response; (b) phase response.

Using Equation (5.3), the corresponding impulse response for the ideal Hilbert transformer is given by

$$h(n) = \frac{1}{2\pi} \left[\int_0^\pi -j e^{j\omega n} d\omega + \int_{-\pi}^0 j e^{j\omega n} d\omega \right] = \begin{cases} 0, & \text{for } n = 0 \\ \frac{1}{\pi n} [1 - (-1)^n], & \text{for } n \neq 0 \end{cases}. \quad (5.18)$$

By examining Equation (5.17) and comparing it with Equation (4.16), we conclude, as in the case of the differentiator, that a Hilbert transformer must be approximated, when using a linear-phase FIR filter, by either a Type III or Type IV structure (see the discussion following Equations (5.5), (5.8), and (5.11), as well as Exercise 5.1).

An interesting interpretation of Hilbert transformers comes from the observation that Equation (5.17) implies that every positive-frequency sinusoid $e^{j\omega_0}$ input to a Hilbert transformer has its phase shifted by $-\pi/2$ at the output, whereas every negative-frequency sinusoid $e^{-j\omega_0}$ has its phase shifted by $+\pi/2$ at the output, as seen in Figure 5.3b. This is equivalent to shifting the phase of every sine or cosine function by $-\pi/2$. Therefore, an ideal Hilbert transformer transforms every “cosine” component of a signal into a “sine” and every “sine” component into a “cosine.”

5.2.4 Summary

Table 5.1 summarizes the ideal frequency responses and corresponding impulse responses for the basic lowpass, highpass, bandpass, and bandstop filters, as well as for differentiators and Hilbert transformers. By examining this table, we note that the impulse responses corresponding to all these ideal filters are not directly realizable, since they have infinite duration and are noncausal. In the remainder of this chapter, we deal with the problem of approximating ideal frequency responses, as the ones seen in this section, by a digital filter with a finite-duration impulse response.

5.3 FIR filter approximation by frequency sampling

In general, the problem of FIR filter design is to find a finite-length impulse response $h(n)$ whose Fourier transform $H(e^{j\omega})$ approximates a given frequency response well enough. As seen in Section 3.2, one way of achieving such a goal is by noting that the DFT of a length- N sequence $h(n)$ corresponds to samples of its Fourier transform at the frequencies $\omega = 2\pi k/N$; that is:

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n} \quad (5.19)$$

Table 5.1.

Ideal frequency characteristics and corresponding impulse responses for lowpass, highpass, bandpass, and bandstop filters, as well as for differentiators and Hilbert transformers.

Filter type	Magnitude response $ H(e^{j\omega}) $	Impulse response $h(n)$
Lowpass	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_c \\ 0, & \text{for } \omega_c < \omega \leq \pi \end{cases}$	$\begin{cases} \frac{\omega_c}{\pi}, & \text{for } n = 0 \\ \frac{1}{\pi n} \sin(\omega_c n), & \text{for } n \neq 0 \end{cases}$
Highpass	$\begin{cases} 0, & \text{for } 0 \leq \omega < \omega_c \\ 1, & \text{for } \omega_c \leq \omega \leq \pi \end{cases}$	$\begin{cases} 1 - \frac{\omega_c}{\pi}, & \text{for } n = 0 \\ -\frac{1}{\pi n} \sin(\omega_c n), & \text{for } n \neq 0 \end{cases}$
Bandpass	$\begin{cases} 0, & \text{for } 0 \leq \omega < \omega_{c1} \\ 1, & \text{for } \omega_{c1} \leq \omega \leq \omega_{c2} \\ 0, & \text{for } \omega_{c2} < \omega \leq \pi \end{cases}$	$\begin{cases} \frac{\omega_{c2} - \omega_{c1}}{\pi}, & \text{for } n = 0 \\ \frac{1}{\pi n} [\sin(\omega_{c2} n) - \sin(\omega_{c1} n)], & \text{for } n \neq 0 \end{cases}$
Bandstop	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_{c1} \\ 0, & \text{for } \omega_{c1} < \omega < \omega_{c2} \\ 1, & \text{for } \omega_{c2} \leq \omega \leq \pi \end{cases}$	$\begin{cases} 1 - \frac{\omega_{c2} - \omega_{c1}}{\pi}, & \text{for } n = 0 \\ \frac{1}{\pi n} [\sin(\omega_{c1} n) - \sin(\omega_{c2} n)], & \text{for } n \neq 0 \end{cases}$
Filter type	Frequency response $H(e^{j\omega})$	Impulse response $h(n)$
Differentiator	$j\omega, \quad \text{for } -\pi \leq \omega < \pi$	$\begin{cases} 0, & \text{for } n = 0 \\ \frac{(-1)^n}{n}, & \text{for } n \neq 0 \end{cases}$
Hilbert transformer	$\begin{cases} -j, & \text{for } 0 \leq \omega < \pi \\ j, & \text{for } -\pi \leq \omega < 0 \end{cases}$	$\begin{cases} 0, & \text{for } n = 0 \\ \frac{1}{\pi n} [1 - (-1)^n], & \text{for } n \neq 0 \end{cases}$

and then

$$H(e^{j2\pi k/N}) = \sum_{n=0}^{N-1} h(n)e^{-j2\pi kn/N}, \quad \text{for } k = 0, 1, \dots, (N-1). \quad (5.20)$$

It is then natural to consider designing a length- N FIR filter by finding an $h(n)$ whose DFT corresponds exactly to samples of the desired frequency response. In other words, $h(n)$ can be determined by sampling the desired frequency response at the N points $e^{j(2\pi/N)k}$ and finding its inverse DFT, given in Equation (3.16). This method is generally referred to as the frequency sampling approach (Gold & Jordan, 1969; Rabiner *et al.*, 1970; Rabiner & Gold, 1975).

More precisely, if the desired frequency response is given by $D(\omega)$, then one must first find

$$A(k)e^{j\theta(k)} = D\left(\frac{\omega_s k}{N}\right), \text{ for } k = 0, 1, \dots, (N-1), \quad (5.21)$$

where $A(k)$ and $\theta(k)$ are samples of the desired amplitude and phase responses, respectively. If we want the resulting filter to have linear phase, then $h(n)$ must be of one of the forms given in Section 4.2.3. For each form, the functions $A(k)$ and $\theta(k)$ present particular properties. We then summarize the results given in Antoniou (1993) for these four cases separately.¹

- Type I: even order M and symmetrical impulse response. In this case, the phase and amplitude responses must satisfy

$$\theta(k) = -\frac{\pi k M}{M+1}, \text{ for } 0 \leq k \leq M \quad (5.22)$$

$$A(k) = A(M-k+1), \text{ for } 1 \leq k \leq \frac{M}{2}, \quad (5.23)$$

and then the impulse response is given by

$$h(n) = \frac{1}{M+1} \left[A(0) + 2 \sum_{k=1}^{M/2} (-1)^k A(k) \cos \frac{\pi k(1+2n)}{M+1} \right] \quad (5.24)$$

for $n = 0, 1, \dots, M$.

- Type II: odd order M and symmetrical impulse response. The phase and amplitude responses, in this case, become

$$\theta(k) = \begin{cases} -\frac{\pi k M}{M+1}, & \text{for } 0 \leq k \leq (M-1)/2 \\ \pi - \frac{\pi k M}{M+1}, & \text{for } (M+3)/2 \leq k \leq M \end{cases} \quad (5.25)$$

$$A(k) = A(M-k+1), \text{ for } 1 \leq k \leq (M+1)/2 \quad (5.26)$$

$$A\left(\frac{M+1}{2}\right) = 0 \quad (5.27)$$

and the impulse response is

$$h(n) = \frac{1}{M+1} \left[A(0) + 2 \sum_{k=1}^{(M-1)/2} (-1)^k A(k) \cos \frac{\pi k(1+2n)}{M+1} \right] \quad (5.28)$$

for $n = 0, 1, \dots, M$.

¹ To maintain consistency with the notation in Section 4.2.3, in the following discussion we will use the filter order $M = N - 1$ instead of the filter length N .

- Type III: even order M and antisymmetric impulse response. The phase and amplitude responses are such that

$$\theta(k) = \frac{(1+2r)\pi}{2} - \frac{\pi k M}{M+1}, \text{ for } r \in \mathbb{Z} \text{ and } 0 \leq k \leq M \quad (5.29)$$

$$A(k) = A(M-k+1), \text{ for } 1 \leq k \leq M/2 \quad (5.30)$$

$$A(0) = 0 \quad (5.31)$$

and the impulse response is given by

$$h(n) = \frac{2}{M+1} \sum_{k=1}^{M/2} (-1)^{k+1} A(k) \sin \frac{\pi k(1+2n)}{M+1} \quad (5.32)$$

for $n = 0, 1, \dots, M$.

- Type IV: odd order M and antisymmetric impulse response. In this case, the phase and amplitude responses are of the form

$$\theta(k) = \begin{cases} \frac{\pi}{2} - \frac{\pi k M}{M+1}, & \text{for } 1 \leq k \leq (M-1)/2 \\ -\frac{\pi}{2} - \frac{\pi k M}{M+1}, & \text{for } (M+1)/2 \leq k \leq M \end{cases} \quad (5.33)$$

$$A(k) = A(M-k+1), \quad \text{for } 1 \leq k \leq M \quad (5.34)$$

$$A(0) = 0 \quad (5.35)$$

and the impulse response then becomes

$$h(n) = \frac{1}{M+1} \left\{ (-1)^{[(M+1)/2]+n} A((M+1)/2) + 2 \sum_{k=1}^{(M-1)/2} (-1)^k A(k) \sin \frac{\pi k(1+2n)}{M+1} \right\} \quad (5.36)$$

for $n = 0, 1, \dots, M$.

The results given above are summarized in Table 5.2.

Example 5.1. Design a lowpass filter satisfying the specification below using the frequency sampling method:²

$$\left. \begin{array}{l} M = 52 \\ \Omega_p = 4.0 \text{ rad/s} \\ \Omega_r = 4.2 \text{ rad/s} \\ \Omega_s = 10.0 \text{ rad/s} \end{array} \right\}. \quad (5.37)$$

² Note that in this text, in general, the variable Ω represents an analog frequency and the variable ω a digital frequency.

Table 5.2.

Impulse responses for linear-phase FIR filters using the frequency sampling approach.

Filter type	Impulse response $h(n)$, for $n = 0, 1, \dots, M$	Condition
Type I	$\frac{1}{M+1} \left[A(0) + 2 \sum_{k=1}^{M/2} (-1)^k A(k) \cos \frac{\pi k(1+2n)}{M+1} \right]$	
Type II	$\frac{1}{M+1} \left[A(0) + 2 \sum_{k=1}^{(M-1)/2} (-1)^k A(k) \cos \frac{\pi k(1+2n)}{M+1} \right]$	$A\left(\frac{M+1}{2}\right) = 0$
Type III	$\frac{2}{M+1} \sum_{k=1}^{M/2} (-1)^{k+1} A(k) \sin \frac{\pi k(1+2n)}{M+1}$	$A(0) = 0$
Type IV	$\frac{1}{M+1} \left[(-1)^{[(M+1)/2]+n} A\left(\frac{M+1}{2}\right) + 2 \sum_{k=1}^{(M-1)/2} (-1)^k A(k) \sin \frac{\pi k(1+2n)}{M+1} \right]$	$A(0) = 0$

Solution

We divide the $[0, \Omega_s]$ interval into $(M+1) = 53$ sub-intervals of the same length $\Omega_s/(M+1)$, each starting at $\Omega_k = [\Omega_s/(M+1)]k$, for $k = 0, 1, \dots, M$. According to the prescribed specifications, Ω_p and Ω_r lie close to the extremes

$$k_p = \left\lfloor (M+1) \times \frac{\Omega_p}{\Omega_s} \right\rfloor = \left\lfloor 53 \times \frac{4}{10} \right\rfloor = 21 \quad (5.38)$$

$$k_r = \left\lfloor (M+1) \times \frac{\Omega_r}{\Omega_s} \right\rfloor = \left\lfloor 53 \times \frac{4.2}{10} \right\rfloor = 22. \quad (5.39)$$

Thus, we assign

$$A(k) = \begin{cases} 1, & \text{for } 0 \leq k \leq k_p \\ 0, & \text{for } k_r \leq k \leq M/2 \end{cases} \quad (5.40)$$

and then, one can employ the following MATLAB script, implementing the first row in Table 5.2, to design a Type I lowpass filter using the frequency sampling method:

```
M = 52; N = M+1;
Omega_p = 4; Omega_r = 4.2; Omega_s = 10;
kp = floor(N*Omega_p/Omega_s);
kr = floor(N*Omega_r/Omega_s);
```

Table 5.3.

Coefficients $h(0)$ to $h(26)$ of the lowpass filter designed with the frequency sampling method.

$h(0) = -0.0055$	$h(7) = -0.0202$	$h(14) = -0.0213$	$h(21) = 0.0114$
$h(1) = 0.0147$	$h(8) = 0.0204$	$h(15) = 0.0073$	$h(22) = -0.0560$
$h(2) = -0.0190$	$h(9) = -0.0135$	$h(16) = 0.0118$	$h(23) = 0.1044$
$h(3) = 0.0169$	$h(10) = 0.0014$	$h(17) = -0.0301$	$h(24) = -0.1478$
$h(4) = -0.0089$	$h(11) = 0.0124$	$h(18) = 0.0413$	$h(25) = 0.1779$
$h(5) = -0.0024$	$h(12) = -0.0231$	$h(19) = -0.0396$	$h(26) = 0.8113$
$h(6) = 0.0133$	$h(13) = 0.0268$	$h(20) = 0.0218$	

```
A = [ones(1,kp+1) zeros(1,M/2-kr+1)];
k = 1:M/2;
for n=0:M,
    h(n+1) = A(1) + 2*sum((-1).^k.*A(k+1) .
        *cos(pi.*k*(1+2*n)/N));
end;
h = h./N;
```

Using this script, one ends up with the set of coefficients shown in Table 5.3, where only half of the filter coefficients are given, as the other half can be obtained from symmetry as $h(n) = h(52 - n)$.

The corresponding magnitude response is shown in Figure 5.4.

△

By examining the magnitude response shown in Figure 5.4, one notices that there is a great deal of ripple both at the passband and at the stopband. This is the main reason why this method has not found widespread use in filter design. This is not a surprising result, because the equations derived in this section guarantee only that the Fourier transform of $h(n)$ and the desired frequency response $D(\omega)$ (expressed as a function of the digital frequencies; that is, $\omega = 2\pi\Omega/\Omega_s = \Omega T$) coincide at the $M + 1$ distinct frequencies $2\pi k/(M + 1)$, for $k = 0, 1, \dots, M$, where M is the filter order. At the other frequencies, as is illustrated in Figure 5.5, there is no constraint on the magnitude response, and, as a consequence, no control over the ripple δ .

An interesting explanation of this fact comes from the expression of the inverse Fourier transform of the desired frequency response $D(\omega)$, which, from Equation (5.3), is given by

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} D(\omega) e^{j\omega n} d\omega = \frac{1}{2\pi} \int_0^{2\pi} D(\omega) e^{j\omega n} d\omega. \quad (5.41)$$

If we try to approximate the above integral as a summation over the discrete frequencies $2\pi k/N$, by substituting $\omega \rightarrow 2\pi k/N$ and $d\omega \rightarrow 2\pi/N$, we end up with an approximation,

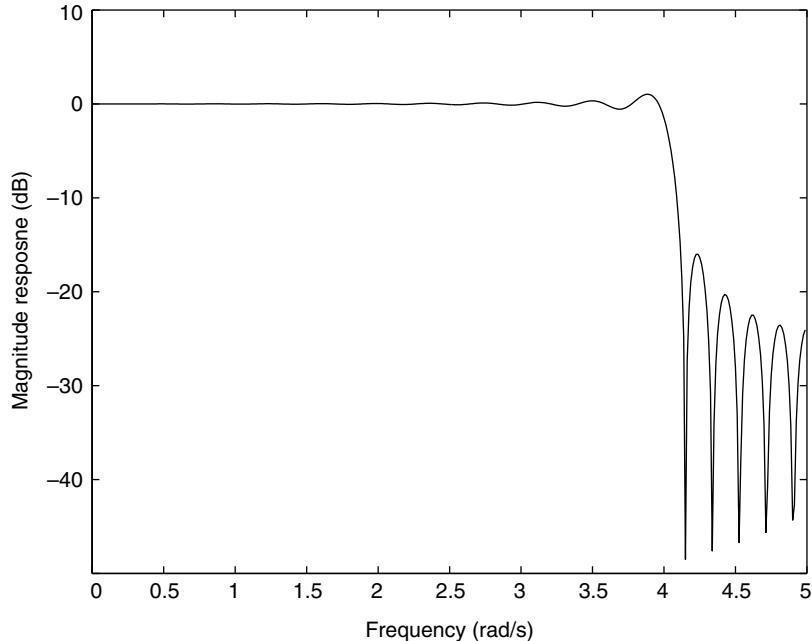


Fig. 5.4. Magnitude response of the lowpass filter designed with the frequency sampling method.

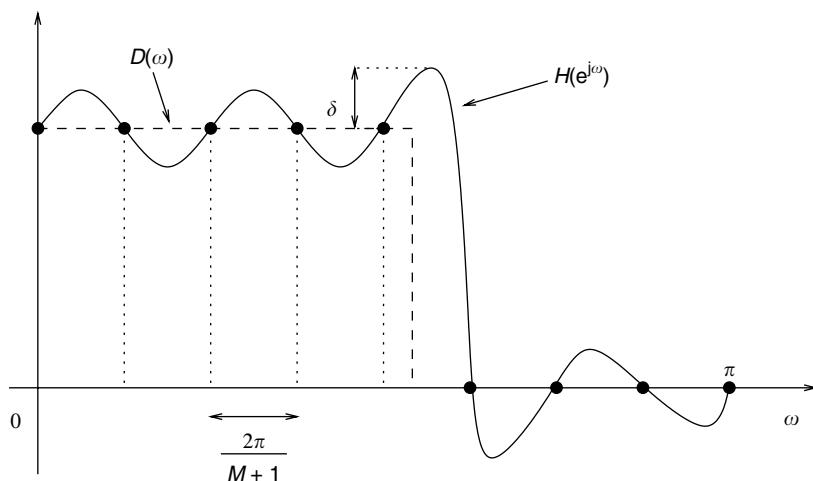


Fig. 5.5. The desired magnitude response and the Fourier transform of $h(n)$ coincide only at the frequencies $2\pi k/(M + 1)$, when using the frequency sampling approximation method.

$d(n)$, of $h(n)$ given by

$$d(n) = \frac{1}{2\pi} \sum_{n=0}^{N-1} D\left(\frac{2\pi k}{N}\right) e^{-j2\pi kn/N} \frac{2\pi}{N} = \frac{1}{N} \sum_{n=0}^{N-1} D\left(\frac{2\pi k}{N}\right) e^{-j2\pi kn/N}. \quad (5.42)$$

From Equation (3.16), we see that $d(n)$ represents the IDFT of the sequence $D(2\pi k/N)$, for $k = 0, 1, \dots, (N - 1)$. However, considering that the argument of the integral in Equation (5.41) is $D(\omega)e^{j\omega n}$, the resolution of a good sampling grid for approximating it would have to be of the order of 10% of the period of the sinusoid $e^{j\omega n}$. This would require a sampling grid with a resolution of the order of $2\pi/(10N)$. In Equation (5.42) we are approximating the integral using a sampling grid with resolution $2\pi/N$ and such approximation would only be valid for values of $n \leq N/10$. This is clearly not sufficient in most practical cases, which explains the large values of ripple depicted in Figure 5.4.

One important situation, however, in which the frequency sampling method gives exact results is when the desired frequency response $D(\omega)$ is composed of a sum of sinusoids equally spaced in frequency. Such a result is formally stated in the following theorem.

Theorem 5.1. *If the desired frequency response $D(\omega)$ is a finite sum of complex sinusoids equally spaced in frequency, that is*

$$D(\omega) = \sum_{n=N_0}^{N_1} a(n)e^{-j\omega n}, \quad (5.43)$$

then the frequency sampling method yields exact results, except for a constant group-delay term, provided that the length of the impulse response, N , satisfies $N \geq N_1 - N_0 + 1$.

◇

Proof The theorem essentially states that the Fourier transform of the impulse response, $h(n)$, given by the frequency sampling method is identical to the desired frequency response $D(\omega)$, except for a constant group-delay term. That is, the above theorem is equivalent to

$$\mathcal{F}\left\{\text{IDFT}\left[D\left(\frac{2\pi k}{N}\right)\right]\right\} = D(\omega), \quad (5.44)$$

where $\mathcal{F}\{\cdot\}$ is the Fourier transform. The proof becomes simpler if we rewrite Equation (5.44) as

$$D\left(\frac{2\pi k}{N}\right) = \text{DFT}\left\{\mathcal{F}^{-1}[D(\omega)]\right\} \quad (5.45)$$

for $k = 0, 1, \dots, (N - 1)$.

For a desired frequency response in the form of Equation (5.43), the corresponding inverse Fourier transform is given by

$$d(n) = \begin{cases} 0, & \text{for } n < N_0 \\ a(n), & \text{for } n = N_0, (N_0 + 1), \dots, N_1 \\ 0, & \text{for } n > N_1 \end{cases}. \quad (5.46)$$

The length- N DFT $H(k)$ of the length- N signal composed of the nonzero samples of $d(n)$, is then equal to the length- N DFT of $a(n)$, adequately shifted in time to the interval $n \in$

$[0, N - 1]$. Therefore, if $N \geq N_1 - N_0 + 1$, we get that

$$\begin{aligned}
H(k) &= \text{DFT} [a(n' + N_0)] \\
&= \sum_{n'=0}^{N-1} a(n' + N_0) e^{-j(2\pi k/N)n'} \\
&= \sum_{n'=0}^{N-1} d(n' + N_0) e^{-j(2\pi k/N)n'} \\
&= \sum_{n=N_0}^{N+N_0-1} d(n) e^{-j(2\pi k/N)(n-N_0)} \\
&= e^{j(2\pi k/N)N_0} \sum_{n=N_0}^{N_1} d(n) e^{-j(2\pi k/N)n} \\
&= e^{j(2\pi k/N)N_0} D\left(\frac{2\pi k}{N}\right)
\end{aligned} \tag{5.47}$$

for $k = 0, 1, \dots, (N - 1)$, and this completes the proof. \square

This result is very useful whenever the desired frequency response $D(\omega)$ is of the form described by Equation (5.43), as is the case in the approximation methods discussed in Sections 5.5 and 5.6.2.

5.4 FIR filter approximation with window functions

For all ideal filters analyzed in Section 5.2, the impulse responses obtained from Equation (5.3) have infinite duration, which leads to nonrealizable FIR filters. A straightforward way to overcome this limitation is to define a finite-length auxiliary sequence $h'(n)$, yielding a filter of order M , as

$$h'(n) = \begin{cases} h(n), & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}, \tag{5.48}$$

assuming that M is even. The resulting transfer function is written as

$$H'(z) = h(0) + \sum_{n=1}^{M/2} (h(-n)z^n + h(n)z^{-n}). \tag{5.49}$$

This is still a noncausal function which we can make causal by multiplying it by $z^{-M/2}$, without either distorting the filter magnitude response or destroying the linear-phase property. The example below highlights some of the impacts that the truncation of the impulse response in Equations (5.48) and (5.49) has on the filter frequency response.

Table 5.4.Bandstop filter coefficients $h(0)$ to $h(25)$.

$h(0) = -0.0037$	$h(7) = 0.0177$	$h(14) = 0.0494$	$h(21) = 0.0000$
$h(1) = 0.0000$	$h(8) = -0.0055$	$h(15) = 0.0318$	$h(22) = 0.1811$
$h(2) = 0.0041$	$h(9) = 0.0000$	$h(16) = -0.0104$	$h(23) = 0.1592$
$h(3) = -0.0145$	$h(10) = 0.0062$	$h(17) = 0.0000$	$h(24) = -0.0932$
$h(4) = -0.0259$	$h(11) = -0.0227$	$h(18) = 0.0133$	$h(25) = 0.7500$
$h(5) = 0.0000$	$h(12) = -0.0418$	$h(19) = -0.0531$	
$h(6) = 0.0286$	$h(13) = 0.0000$	$h(20) = -0.1087$	

Example 5.2. Design a bandstop filter satisfying the following specification:

$$\left. \begin{aligned} M &= 50 \\ \Omega_{c_1} &= \pi/4 \text{ rad/s} \\ \Omega_{c_2} &= \pi/2 \text{ rad/s} \\ \Omega_s &= 2\pi \text{ rad/s} \end{aligned} \right\}. \quad (5.50)$$

Solution

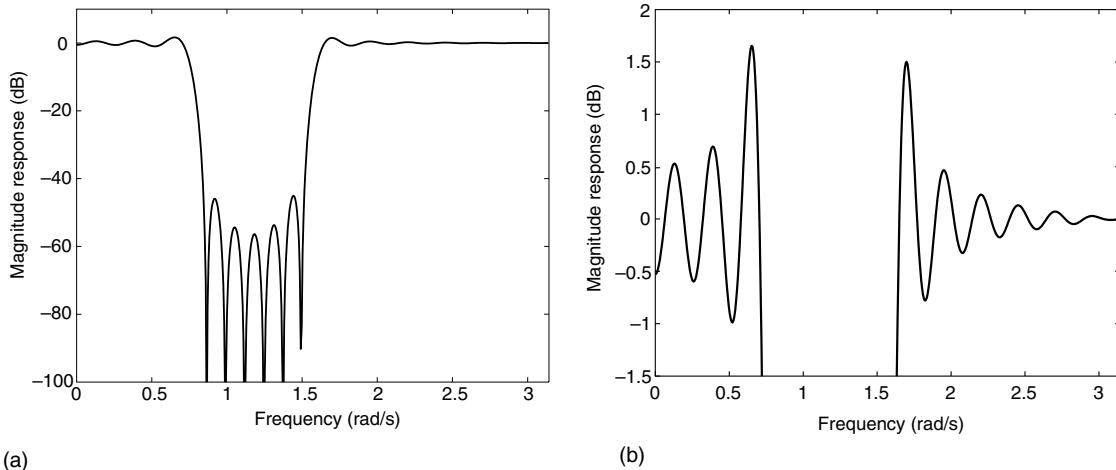
Applying Equations (5.48) and (5.49) to the corresponding bandstop equations in Table 5.1, one may use the script

```
M = 50;
wc1 = pi/4; wc2 = pi/2; ws = 2*pi;
n = 1:M/2;
h0 = 1 - (wc2 - wc1)/pi;
haux = (sin(wc1.*n) - sin(wc2.*n))./(pi.*n);
h = [fliplr(haux) h0 haux];
```

to obtain the filter coefficients listed in Table 5.4 (only half of them are listed, as the others can be found using $h(n) = h(50 - n)$). The resulting magnitude response is depicted in Figure 5.6.

△

The ripple seen in Figure 5.6 close to the band edges is due to the slow convergence of the Fourier series $h(n)$ when approximating functions presenting discontinuities, such as the ideal responses seen in Figure 5.1. This implies that large-amplitude ripples in the magnitude response appear close to the edges whenever an infinite-length $h(n)$ is truncated to generate a finite-length filter. These ripples are commonly referred to as Gibbs' oscillations. It can be shown that Gibbs' oscillations possess the property that their amplitudes do not decrease even when the filter order M is increased dramatically (Kreider *et al.*, 1966; Oppenheim *et al.*, 1983). This severely limits the practical usefulness of Equations (5.48) and (5.49) in FIR design, because the maximum deviation from the ideal magnitude response cannot be minimized by increasing the filter length.



(a)

(b)

Fig. 5.6. Bandstop filter: (a) magnitude response; (b) passband detail.

Although we cannot remove the ripples introduced by the poor convergence of the Fourier series, we can still attempt to control their amplitude by multiplying the impulse response $h(n)$ by a window function $w(n)$. The window $w(n)$ must be designed such that it introduces minimum deviation from the ideal frequency response. The coefficients of the resulting impulse response $h'(n)$ become

$$h'(n) = h(n)w(n). \quad (5.51)$$

In the frequency domain, such a multiplication corresponds to a periodic convolution operation between the frequency responses of the ideal filter $H(e^{j\omega})$ and of the window function $W(e^{j\omega})$; that is:

$$H'(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega'}) W(e^{j(\omega-\omega')}) d\omega'. \quad (5.52)$$

We can then infer that a good window is a finite-length sequence whose frequency response, when convolved with an ideal frequency response, produces the minimum distortion possible. This minimum distortion would occur when the frequency response of the window has an impulse-like shape, concentrated around $\omega = 0$, as depicted in Figure 5.7a. However, band-limited signals in frequency are not limited in time; therefore, such a window sequence would have to be infinite, which contradicts our main requirement. This means that we must find a finite-length window which has a frequency response that has most of its energy concentrated around $\omega = 0$. Also, in order to avoid the oscillations in the filter magnitude response, the sidelobes of the window magnitude response should quickly decay as $|\omega|$ is increased (Kaiser, 1974; Rabiner & Gold, 1975; Rabiner *et al.*, 1975; Antoniou, 1993).

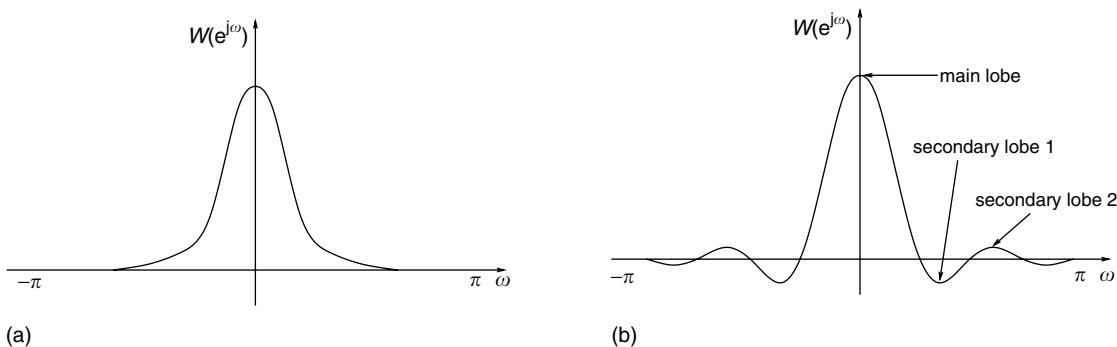


Fig. 5.7. Magnitude responses of a window function: (a) ideal case; (b) practical case.

A practical window function is in general as shown in Figure 5.7b. The effect of the secondary lobe is to introduce the largest ripple close to the band edges. From Equation (5.52), we see that the main-lobe width determines the transition bandwidth of the resulting filter. Based on these facts, a practical window function must present a magnitude response characterized by:

- The ratio of the main-lobe amplitude to the secondary-lobe amplitude must be as large as possible.
- The energy must decay rapidly when $|\omega|$ increases from 0 to π .

We can now proceed to perform a thorough study of the more widely used window functions in FIR filter design.

5.4.1 Rectangular window

The simple truncation of the impulse response as described in Equation (5.48) can be interpreted as the product between the ideal $h(n)$ and a rectangular window defined by

$$w_R(n) = \begin{cases} 1, & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}. \quad (5.53)$$

Note that if we want to truncate the impulse responses in Table 5.1 using the above equation and still keep the linear-phase property, then the resulting truncated sequences would have to be either symmetric or antisymmetric around $n = 0$. This implies that, for those cases, M would have to be even (Type I and Type III filters, as seen in Section 4.2.3). For the case of M odd (Type II and Type IV filters, see Exercise 5.1), the solution would be to shift $h(n)$ so that it is causal and apply a window different from zero from $n = 0$ to $n = M - 1$. This solution, however, is not commonly used in practice.

From Equation (5.53), the frequency response of the rectangular window is given by

$$\begin{aligned}
 W_r(e^{j\omega}) &= \sum_{n=-M/2}^{M/2} e^{-j\omega n} \\
 &= \frac{e^{j\omega M/2} - e^{-j\omega M/2} e^{-j\omega}}{1 - e^{-j\omega}} \\
 &= e^{-j\omega/2} \frac{e^{j\omega(M+1)/2} - e^{-j\omega(M+1)/2}}{1 - e^{-j\omega}} \\
 &= \frac{\sin [\omega(M + 1)/2]}{\sin (\omega/2)}. \tag{5.54}
 \end{aligned}$$

5.4.2 Triangular windows

The main problem associated with the rectangular window is the presence of ripples near the band edges of the resulting filter, which are caused by the existence of sidelobes in the frequency response of the window. Such a problem is due to the inherent discontinuity of the rectangular window in the time domain. One way to reduce such a discontinuity is to employ a triangular-shaped window, which will present only small discontinuities near its edges. The standard triangular window is defined as

$$w_t(n) = \begin{cases} -\frac{2|n|}{M+2} + 1, & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}. \tag{5.55}$$

A small variation of such a window is called the Bartlett window and is defined by

$$w_{tB}(n) = \begin{cases} -\frac{2|n|}{M} + 1, & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}. \tag{5.56}$$

Clearly, these two triangular-type window functions are closely related. Their main difference lies in the fact that the Bartlett window presents one null element at each of its extremities. In that manner, an M th-order Bartlett window can be obtained by juxtaposing one zero at each extremity of the $(M - 2)$ th-order standard triangular window.

In some cases, an even greater reduction of the sidelobes is necessary, and then more complex window functions should be used, such as the ones described in the following subsections.

5.4.3 Hamming and Hann windows

The generalized Hamming window is defined as

$$w_H(n) = \begin{cases} \alpha + (1 - \alpha) \cos\left(\frac{2\pi n}{M}\right), & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases} \quad (5.57)$$

with $0 \leq \alpha \leq 1$. This generalized window is referred to as the Hamming window when $\alpha = 0.54$, and for $\alpha = 0.5$ it is known as the Hann or Hanning window.

The frequency response for the general Hamming window can be expressed based on the frequency response of the rectangular window. We first write Equation (5.57) as

$$w_H(n) = w_r(n) \left[\alpha + (1 - \alpha) \cos\left(\frac{2\pi n}{M}\right) \right]. \quad (5.58)$$

By transforming the above equation to the frequency domain, clearly the frequency response of the generalized Hamming window results from the periodic convolution between $W_r(e^{j\omega})$ and three impulse functions as

$$W_H(e^{j\omega}) = W_r(e^{j\omega}) * \left[\alpha \delta(\omega) + \left(\frac{1-\alpha}{2} \right) \delta\left(\omega - \frac{2\pi}{M}\right) + \left(\frac{1-\alpha}{2} \right) \delta\left(\omega + \frac{2\pi}{M}\right) \right] \quad (5.59)$$

and then

$$W_H(e^{j\omega}) = \alpha W_r(e^{j\omega}) + \left(\frac{1-\alpha}{2} \right) W_r\left\{ e^{j[\omega - (2\pi/M)]} \right\} + \left(\frac{1-\alpha}{2} \right) W_r\left\{ e^{j[\omega + (2\pi/M)]} \right\} \quad (5.60)$$

From this equation, one clearly sees that $W_H(e^{j\omega})$ is composed of three versions of the rectangular spectrum $W_r(e^{j\omega})$: the main component, $\alpha W_r(e^{j\omega})$, centered at $\omega = 0$, and two additional ones with smaller amplitudes, centered at $\omega = \pm 2\pi/M$, that reduce the secondary lobe of the main component. This is illustrated in Figure 5.8.

The main characteristics of the generalized Hamming window are:

- All three $W_r(e^{j\omega})$ components have zeros close to $\omega = \pm 4\pi/(M+1)$. Hence, the main-lobe total width is $8\pi/(M+1)$.
- When $\alpha = 0.54$, the main-lobe total energy is approximately 99.96% of the window total energy.
- The transition band of the Hamming window is larger than the transition band of the rectangular window, due to its wider main lobe.
- The ratio between the amplitudes of the main and secondary lobes of the Hamming window is much larger than for the rectangular window.
- The stopband attenuation for the Hamming window is larger than the attenuation for the rectangular window.

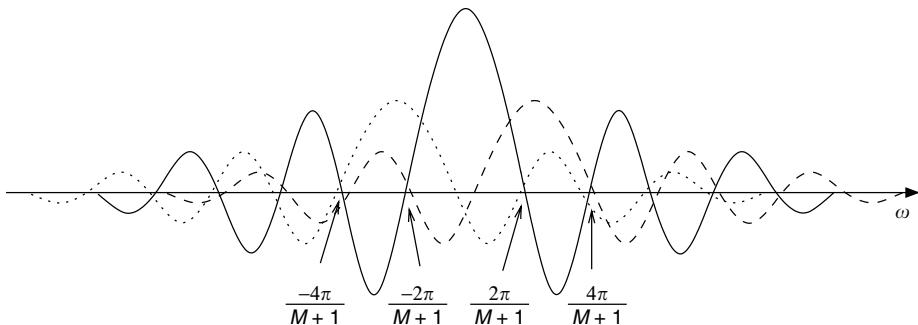


Fig. 5.8.

The three components of the generalized Hamming window combine to reduce the resulting secondary lobes. Solid line: $\alpha W_r(e^{j\omega})$; dashed line: $[(1 - \alpha)/2]W_r(e^{j[(\omega - \pi)/M]})$; dotted line: $1 - \alpha/2 W_r(e^{j[\omega + (\pi/M)]})$

5.4.4 Blackman window

The Blackman window is defined as

$$w_B(n) = \begin{cases} 0.42 + 0.5 \cos\left(\frac{2\pi n}{M}\right) + 0.08 \cos\left(\frac{4\pi n}{M}\right), & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}. \quad (5.61)$$

Compared with the Hamming window function, the Blackman window introduces a second cosine term to further reduce the effects of the secondary lobes of $W_r(e^{j\omega})$. The Blackman window is characterized by the following issues:

- The main-lobe width is approximately $12\pi/(M + 1)$, which is wider than that for the previous windows.
- The passband ripples are smaller than in the previous windows.
- The stopband attenuation is larger than in the previous windows.

Example 5.3. Design a bandstop filter satisfying the specification below using the rectangular, Hamming, Hann, and Blackman windows:

$$\left. \begin{array}{l} M = 80 \\ \Omega_{c_1} = 2000 \text{ rad/s} \\ \Omega_{c_2} = 4000 \text{ rad/s} \\ \Omega_s = 10000 \text{ rad/s} \end{array} \right\}. \quad (5.62)$$

Solution

This time, filter specifications are given in the analog frequency domain. Hence, one must first normalize Ω_{c_1} and Ω_{c_2} before employing a script similar to the one given in Example 5.2 to obtain the impulse response using the rectangular window:

```
M = 80;
Omega_c1 = 2000; Omega_c2 = 4000; Omega_s = 10000;
```

Table 5.5.

Filter coefficients $h(0)$ to $h(40)$ using the rectangular window.

$h(0) = 0.0000$	$h(11) = -0.0040$	$h(22) = -0.0272$	$h(33) = 0.0700$
$h(1) = -0.0030$	$h(12) = -0.0175$	$h(23) = 0.0288$	$h(34) = 0.0193$
$h(2) = -0.0129$	$h(13) = 0.0181$	$h(24) = 0.0072$	$h(35) = 0.0000$
$h(3) = 0.0132$	$h(14) = 0.0044$	$h(25) = 0.0000$	$h(36) = -0.0289$
$h(4) = 0.0032$	$h(15) = 0.0000$	$h(26) = -0.0083$	$h(37) = -0.1633$
$h(5) = 0.0000$	$h(16) = -0.0048$	$h(27) = -0.0377$	$h(38) = 0.2449$
$h(6) = -0.0034$	$h(17) = -0.0213$	$h(28) = 0.0408$	$h(39) = 0.1156$
$h(7) = -0.0148$	$h(18) = 0.0223$	$h(29) = 0.0105$	$h(40) = 0.6000$
$h(8) = 0.0153$	$h(19) = 0.0055$	$h(30) = 0.0000$	
$h(9) = 0.0037$	$h(20) = 0.0000$	$h(31) = -0.0128$	
$h(10) = 0.0000$	$h(21) = -0.0061$	$h(32) = -0.0612$	

Table 5.6.

Filter coefficients $h(0)$ to $h(40)$ using the Hamming window.

$h(0) = 0.0000$	$h(11) = -0.0010$	$h(22) = -0.0167$	$h(33) = 0.0652$
$h(1) = -0.0002$	$h(12) = -0.0047$	$h(23) = 0.0187$	$h(34) = 0.0183$
$h(2) = -0.0011$	$h(13) = 0.0054$	$h(24) = 0.0049$	$h(35) = 0.0000$
$h(3) = 0.0012$	$h(14) = 0.0015$	$h(25) = 0.0000$	$h(36) = -0.0283$
$h(4) = 0.0003$	$h(15) = 0.0000$	$h(26) = -0.0062$	$h(37) = -0.1612$
$h(5) = 0.0000$	$h(16) = -0.0019$	$h(27) = -0.0294$	$h(38) = 0.2435$
$h(6) = -0.0004$	$h(17) = -0.0092$	$h(28) = 0.0331$	$h(39) = 0.1155$
$h(7) = -0.0022$	$h(18) = 0.0104$	$h(29) = 0.0088$	$h(40) = 0.6000$
$h(8) = 0.0026$	$h(19) = 0.0028$	$h(30) = 0.0000$	
$h(9) = 0.0007$	$h(20) = 0.0000$	$h(31) = -0.0114$	
$h(10) = 0.0000$	$h(21) = -0.0035$	$h(32) = -0.0558$	

```
wc1 = Omega_c1*2*pi/Omega_s; wc2 = Omega_c2*2*pi/Omega_s;
n = 1:M/2;
h0 = 1 - (wc2 - wc1)/pi;
haux = (sin(wc1.*n) - sin(wc2.*n))./(pi.*n);
h = [fliplr(haux) h0 haux];
```

For the other three windows, one must multiply sample-by-sample $h(n)$ above by the corresponding window obtained with the MATLAB commands `hamming(M+1)`, `hanning(M+1)`, and `blackman(M+1)`. The resulting impulse responses are shown in Tables 5.5–5.8, where only the filter coefficients for $0 \leq n \leq 40$ are given, since the remaining coefficients can be obtained as $h(n) = h(80 - n)$.

The magnitude responses associated with the four impulse responses listed in Tables 5.5–5.8 are depicted in Figure 5.9. The reader should note the compromise between the transition

Table 5.7.

Filter coefficients $h(0)$ to $h(40)$ using the Hann window.

$h(0) = 0.0000$	$h(11) = -0.0008$	$h(22) = -0.0162$	$h(33) = 0.0651$
$h(1) = -0.0000$	$h(12) = -0.0040$	$h(23) = 0.0182$	$h(34) = 0.0183$
$h(2) = -0.0002$	$h(13) = 0.0047$	$h(24) = 0.0048$	$h(35) = 0.0000$
$h(3) = 0.0003$	$h(14) = 0.0013$	$h(25) = 0.0000$	$h(36) = -0.0282$
$h(4) = 0.0001$	$h(15) = 0.0000$	$h(26) = -0.0061$	$h(37) = -0.1611$
$h(5) = 0.0000$	$h(16) = -0.0018$	$h(27) = -0.0291$	$h(38) = 0.2435$
$h(6) = -0.0002$	$h(17) = -0.0086$	$h(28) = 0.0328$	$h(39) = 0.1155$
$h(7) = -0.0014$	$h(18) = 0.0099$	$h(29) = 0.0088$	$h(40) = 0.6000$
$h(8) = 0.0017$	$h(19) = 0.0026$	$h(30) = 0.0000$	
$h(9) = 0.0005$	$h(20) = 0.0000$	$h(31) = -0.0114$	
$h(10) = 0.0000$	$h(21) = -0.0034$	$h(32) = -0.0557$	

Table 5.8.

Filter coefficients $h(0)$ to $h(40)$ using the Blackman window.

$h(0) = 0.0000$	$h(11) = -0.0003$	$h(22) = -0.0115$	$h(33) = 0.0618$
$h(1) = -0.0000$	$h(12) = -0.0018$	$h(23) = 0.0134$	$h(34) = 0.0176$
$h(2) = -0.0000$	$h(13) = 0.0022$	$h(24) = 0.0037$	$h(35) = 0.0000$
$h(3) = 0.0001$	$h(14) = 0.0006$	$h(25) = 0.0000$	$h(36) = -0.0278$
$h(4) = 0.0000$	$h(15) = 0.0000$	$h(26) = -0.0050$	$h(37) = -0.1596$
$h(5) = 0.0000$	$h(16) = -0.0010$	$h(27) = -0.0243$	$h(38) = 0.2424$
$h(6) = -0.0001$	$h(17) = -0.0049$	$h(28) = 0.0281$	$h(39) = 0.1153$
$h(7) = -0.0004$	$h(18) = 0.0059$	$h(29) = 0.0077$	$h(40) = 0.6000$
$h(8) = 0.0006$	$h(19) = 0.0017$	$h(30) = 0.0000$	
$h(9) = 0.0002$	$h(20) = 0.0000$	$h(31) = -0.0104$	
$h(10) = 0.0000$	$h(21) = -0.0023$	$h(32) = -0.0520$	

bandwidth and the ripple in the passband and stopband when going from the rectangular to the Blackman window; that is, as the ripple decreases, the width of the transition band increases accordingly.



5.4.5 Kaiser window

All the window functions seen so far allow us to control the transition band through a proper choice of the filter order M . However, no control can be achieved over the passband and stopband ripples, which makes these windows of little use when designing filters with

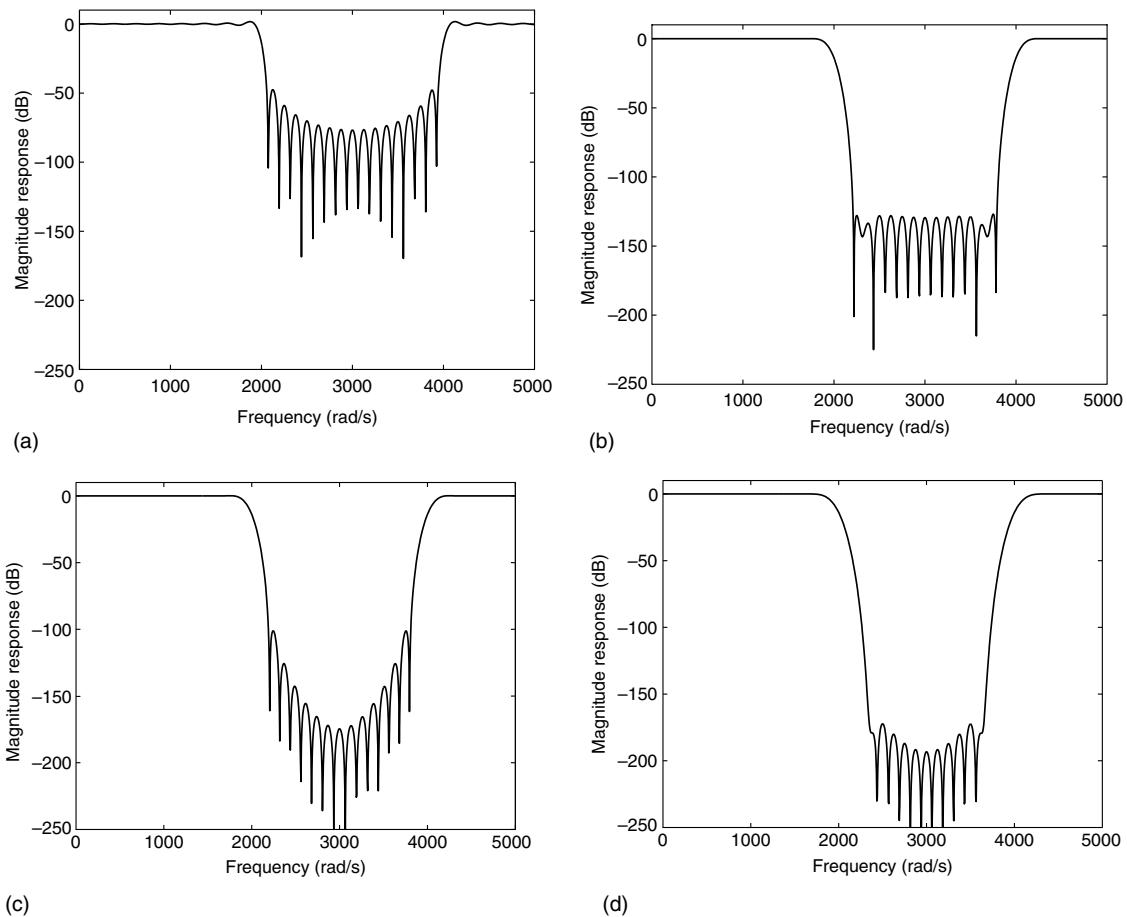


Fig. 5.9. Magnitude responses when using: (a) rectangular; (b) Hamming; (c) Hann; (d) Blackman windows.

prescribed frequency specifications, such as that depicted in Figure 5.10. Such problems are overcome with the Kaiser and Dolph–Chebyshev windows, presented in this and in the next subsections.

As seen earlier in this section, the ideal window should be a finite-duration function such that most of its spectral energy is concentrated around $|\omega| = 0$, quickly decaying when $|\omega|$ increases. There is a family of continuous-time functions, called the prolate spheroidal functions (Kaiser, 1974), which are optimal for achieving these properties. Such functions, although very difficult to implement in practice, can be effectively approximated as

$$w(t) = \begin{cases} \frac{I_0 \left[\beta \sqrt{1 - (t/\tau)^2} \right]}{I_0(\beta)}, & \text{for } |t| \leq \tau, \\ 0, & \text{for } |t| > \tau \end{cases} \quad (5.63)$$

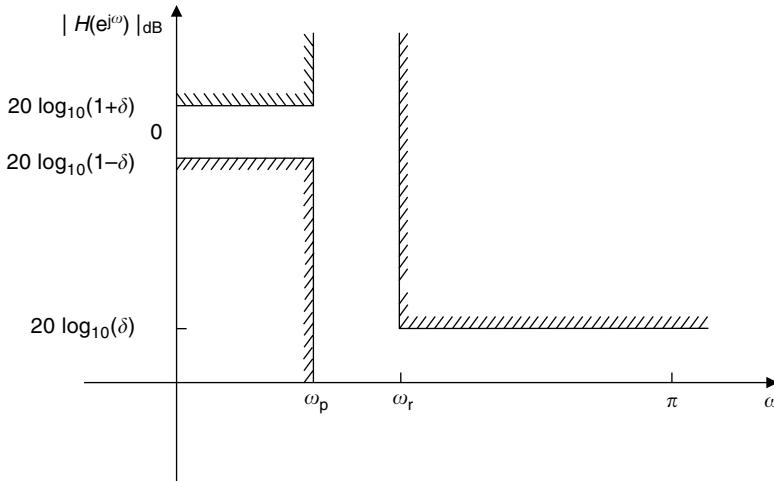


Fig. 5.10. Typical specification of a lowpass filter. The specifications are in terms of the digital frequency $\omega = 2\pi\Omega/\Omega_s = \Omega T$.

where β is a window parameter and $I_0(x)$ is the zeroth-order modified Bessel function of the first kind, which can be efficiently determined through its series expansion given by

$$I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{(x/2)^k}{k!} \right]^2. \quad (5.64)$$

The Fourier transform of $w(t)$ is given by

$$W(j\Omega) = \frac{2\tau \sin \left[\beta \sqrt{(\Omega/\Omega_a)^2 - 1} \right]}{\beta I_0(\beta) \sqrt{(\Omega/\Omega_a)^2 - 1}} \quad (5.65)$$

with $\Omega_a = \beta/\tau$. The Kaiser window is derived from Equation (5.63) by making the transformation to the discrete-time domain given by $\tau \rightarrow (M/2)T$ and $t \rightarrow nT$. The window is then described by

$$w_K(n) = \begin{cases} \frac{I_0 \left[\beta \sqrt{1 - (2n/M)^2} \right]}{I_0(\beta)}, & \text{for } |n| \leq M/2 \\ 0, & \text{for } |n| > M/2 \end{cases}. \quad (5.66)$$

Since the functions given by Equation (5.65) tend to be highly concentrated around $|\Omega| = 0$, we can assume that $W(j\Omega) \approx 0$, for $|\Omega| \geq \Omega_s/2$. Therefore, from Equation (2.219), we can approximate the frequency response for the Kaiser window by

$$W_K(e^{j\omega}) \approx \frac{1}{T} W \left(j \frac{\omega}{T} \right), \quad (5.67)$$

where $W(j\Omega)$ is given by Equation (5.65) when τ is replaced by $(M/2)T$. This yields

$$W_K(e^{j\omega}) \approx \frac{M \sin[\beta \sqrt{(\omega/\omega_a)^2 - 1}]}{\beta I_0(\beta) \sqrt{(\omega/\omega_a)^2 - 1}}, \quad (5.68)$$

where $\omega_a = \Omega_a T$ and $\beta = \Omega_a \tau = (\omega_a/T)(M/2)T = \omega_a M/2$.

The main advantage of the Kaiser window appears in the design of FIR digital filters with prescribed specifications, such as that depicted in Figure 5.10. In such an application, the parameter β is used to control both the main-lobe width and the ratio of the main to the secondary lobes.

The overall procedure for designing FIR filters using the Kaiser window is as follows:

- (i) From the ideal frequency response that the filter is supposed to approximate, determine the impulse response $h(n)$ using Table 5.1. If the filter is either lowpass or highpass, then one should make $\Omega_c = (\Omega_p + \Omega_r)/2$. The case of bandpass and bandstop filters is dealt with later in this subsection.
- (ii) Given the maximum passband ripple in decibels, A_p , and the minimum stopband attenuation in decibels, A_r , determine the corresponding ripples

$$\delta_p = \frac{10^{0.05A_p} - 1}{10^{0.05A_p} + 1} \quad (5.69)$$

$$\delta_r = 10^{-0.05A_r}. \quad (5.70)$$

- (iii) As with all other window functions, the Kaiser window can only be used to design filters that present the same passband and stopband ripples. Therefore, in order to satisfy the prescribed specifications, one should use $\delta = \min\{\delta_p, \delta_r\}$.
- (iv) Compute the resulting passband ripple and stopband attenuation in decibels using

$$A_p = 20 \log \frac{1 + \delta}{1 - \delta} \quad (5.71)$$

$$A_r = -20 \log \delta \quad (5.72)$$

- (v) Given the passband and stopband edges Ω_p and Ω_r , respectively, compute the transition bandwidth $T_r = (\Omega_r - \Omega_p)$.
- (vi) Compute β using

$$\beta = \begin{cases} 0, & \text{for } A_r \leq 21 \\ 0.5842(A_r - 21)^{0.4} + 0.07886(A_r - 21), & \text{for } 21 < A_r \leq 50 \\ 0.1102(A_r - 8.7), & \text{for } 50 < A_r \end{cases} \quad (5.73)$$

This empirical formula was devised by Kaiser (1974) based on the behavior of the function $W(j\Omega)$ in Equation (5.65).

- (vii) Defining the normalized window length D as

$$D = \frac{T_r M}{\Omega_s}, \quad (5.74)$$

where Ω_s is the sampling frequency, we have that D is related to A_r by the following empirical formula:

$$D = \begin{cases} 0.9222, & \text{for } A_r \leq 21 \\ \frac{A_r - 7.95}{14.36}, & \text{for } 21 < A_r \end{cases}. \quad (5.75)$$

- (viii) Having computed D using Equation (5.75), we can use Equation (5.74) to determine the filter order M as the smallest even number that satisfies

$$M \geq \frac{\Omega_s D}{T_r} \quad (5.76)$$

One should remember that T_r must be in the same units as Ω_s .

- (ix) With M and β determined, we compute the window $w_K(n)$ using Equation (5.66). We are now ready to form the sequence $h'(n) = w_K(n)h(n)$, where $h(n)$ is the ideal filter impulse response computed in step (i).
- (x) The desired transfer function is then given by

$$H(z) = z^{-M/2} \mathcal{Z}\{h'(n)\}. \quad (5.77)$$

The above procedure applies to lowpass filters (see Figure 5.10) as well as highpass filters. If the filter is either bandpass or bandstop, then we must include the following reasoning in step (i) above:

1. Compute the narrower transition band

$$T_r = \pm \min\{|\Omega_{r_1} - \Omega_{p_1}|, |\Omega_{p_2} - \Omega_{r_2}|\}. \quad (5.78)$$

Notice that T_r is negative for bandpass filters and positive for bandstop filters.

2. Determine the two central frequencies as

$$\Omega_{c_1} = \Omega_{p_1} + \frac{T_r}{2} \quad (5.79)$$

$$\Omega_{c_2} = \Omega_{p_2} - \frac{T_r}{2} \quad (5.80)$$

A typical magnitude specification for a bandstop filter is depicted in Figure 5.11.

Example 5.4. Design a bandstop filter satisfying the specification below using the Kaiser window:

$$\left. \begin{array}{l} A_p = 1.0 \text{ dB} \\ A_r = 45 \text{ dB} \\ \Omega_{p_1} = 800 \text{ Hz} \\ \Omega_{r_1} = 950 \text{ Hz} \\ \Omega_{r_2} = 1050 \text{ Hz} \\ \Omega_{p_2} = 1200 \text{ Hz} \\ \Omega_s = 6000 \text{ Hz} \end{array} \right\}. \quad (5.81)$$

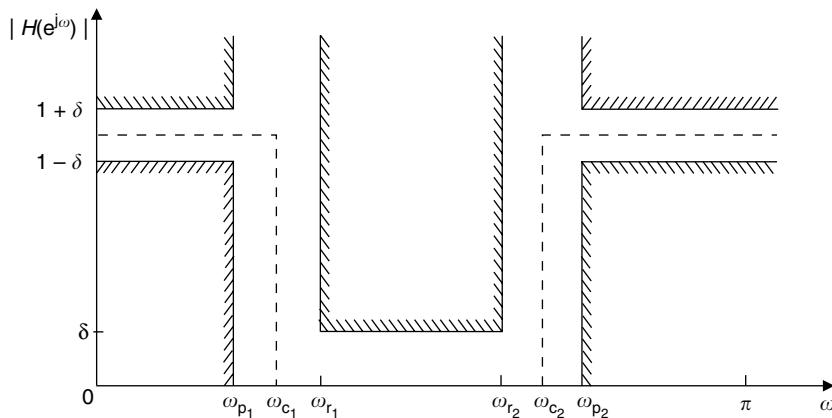


Fig. 5.11. Typical specification of a bandstop filter.

Solution

Following the procedure described above, the resulting filter is obtained as follows (note that, in the FIR design procedure described above, the parameters of the Kaiser window depend only on the ratio of the analog frequencies in the filter specification to the sampling frequency; therefore, the frequencies can be entered in the formula in hertz, as long as the sampling frequency Ω_s is also in hertz):

(i) From Equations (5.78)–(5.80), we have that

$$T_r = +\min\{(950 - 800), (1200 - 1050)\} = 150 \text{ Hz} \quad (5.82)$$

$$\Omega_{c1} = 800 + 75 = 875 \text{ Hz} \quad (5.83)$$

$$\Omega_{c2} = 1200 - 75 = 1125 \text{ Hz}. \quad (5.84)$$

(ii) From Equations (5.69) and (5.70):

$$\delta_p = \frac{10^{0.05} - 1}{10^{0.05} + 1} = 0.0575 \quad (5.85)$$

$$\delta_r = 10^{-0.05 \times 45} = 0.00562. \quad (5.86)$$

$$(5.87)$$

$$(iii) \delta = \min\{0.0575, 0.00562\} = 0.00562$$

(iv) From Equations (5.71) and (5.72):

$$A_p = 20 \log \frac{1 + 0.00562}{1 - 0.00562} = 0.0977 \text{ dB} \quad (5.88)$$

$$A_r = -20 \log 0.00562 = 45 \text{ dB}. \quad (5.89)$$

- (v) T_r has already been computed as 150 Hz in step (i).
 (vi) From Equation (5.73), since $A_r = 45$ dB, then

$$\beta = 0.5842(45 - 21)^{0.4} + 0.07886(45 - 21) = 3.9754327. \quad (5.90)$$

- (vii) From Equation (5.75), since $A_r = 45$ dB, then

$$D = \frac{(45 - 7.95)}{14.36} = 2.5800835. \quad (5.91)$$

- (viii) Since the sampling period is $T = 1/6000$ s, from Equation (5.76), we have

$$M \geq \frac{6000 \times 2.5800835}{150} = 103.20334 \Rightarrow M = 104. \quad (5.92)$$

This whole procedure is implemented by a simple MATLAB script:

```
Ap = 1; Ar = 45;
Omega_p1 = 800; Omega_r1 = 950;
Omega_r2 = 1050; Omega_p2 = 1200;
Omega_s = 6000;
delta_p = (10^(0.05*Ap) - 1)/(10^(0.05*Ap) + 1);
delta_r = 10^(-0.05*Ar);
F = [Omega_p1 Omega_r1 Omega_r2 Omega_p2];
A = [1 0 1];
ripples = [delta_p delta_r delta_p];
[M,Wn,beta,FILTYPE] = kaiserord(F,A,ripples,Omega_s);
```

This yields as outputs $\text{beta} = 3.9754$ and $M = 104$, as determined above. In this short script, the auxiliary vectors A and $ripples$ specify the desired gain and allowed ripple respectively in each filter band.

The Kaiser window coefficients are determined by

```
kaiser_win = kaiser(M+1,beta);
```

and are shown in Figure 5.12 along with the associated magnitude response.

The desired filter is obtained using the `fir1` command, as exemplified by

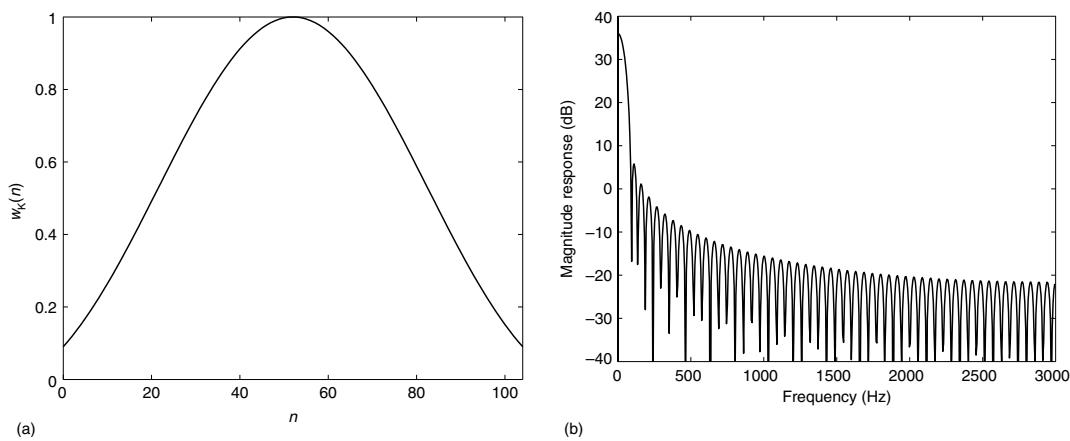
```
h = fir1(M,Wn,FILTYPE,kaiser_win,'noscale');
```

where the `noscale` flag avoids the unitary gain at the first passband center imposed by MATLAB. The designed filter characteristics are summarized in Table 5.9.

Table 5.9.

Characteristics of the designed filter.

Ω_{c_1}	875 Hz
Ω_{c_2}	1125 Hz
Ω_{p_1}	800 Hz
Ω_{r_1}	950 Hz
Ω_{r_2}	1050 Hz
Ω_{p_2}	1200 Hz
δ_p	0.0575
δ_r	0.00562
T_r	150 Hz
D	2.580 0835
β	3.975 4327
M	104

**Fig. 5.12.** Kaiser window: (a) window function; (b) magnitude response.

The filter coefficients $h(n)$ are given in Table 5.10. Once again, owing to the symmetry inherent in the Kaiser window function, Table 5.10 only shows half of the filter coefficients, as the remaining coefficients are obtained as $h(n) = h(M - n)$.

The filter impulse response and associated magnitude response are shown in Figure 5.13.

△

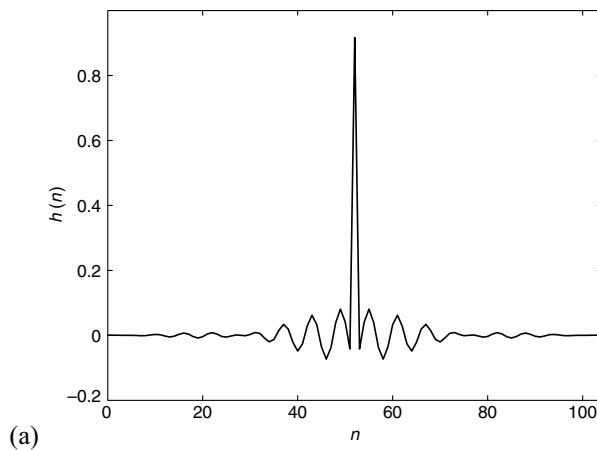
5.4.6 Dolph–Chebyshev window

Based on the M th-order Chebyshev polynomial given by

$$C_M(x) = \begin{cases} \cos[M \cos^{-1}(x)], & \text{for } |x| \leq 1 \\ \cosh[M \cosh^{-1}(x)], & \text{for } |x| > 1 \end{cases}, \quad (5.93)$$

Table 5.10.Filter coefficients $h(0)$ to $h(52)$ using the Kaiser window.

$h(0) = 0.0003$	$h(14) = -0.0028$	$h(28) = 0.0000$	$h(42) = 0.0288$
$h(1) = 0.0005$	$h(15) = 0.0032$	$h(29) = -0.0013$	$h(43) = 0.0621$
$h(2) = 0.0002$	$h(16) = 0.0070$	$h(30) = 0.0027$	$h(44) = 0.0331$
$h(3) = -0.0001$	$h(17) = 0.0038$	$h(31) = 0.0087$	$h(45) = -0.0350$
$h(4) = -0.0000$	$h(18) = -0.0040$	$h(32) = 0.0061$	$h(46) = -0.0733$
$h(5) = 0.0001$	$h(19) = -0.0083$	$h(33) = -0.0081$	$h(47) = -0.0381$
$h(6) = -0.0003$	$h(20) = -0.0042$	$h(34) = -0.0203$	$h(48) = 0.0394$
$h(7) = -0.0011$	$h(21) = 0.0042$	$h(35) = -0.0123$	$h(49) = 0.0807$
$h(8) = -0.0008$	$h(22) = 0.0081$	$h(36) = 0.0146$	$h(50) = 0.0411$
$h(9) = 0.0011$	$h(23) = 0.0038$	$h(37) = 0.0339$	$h(51) = -0.0415$
$h(10) = 0.0028$	$h(24) = -0.0033$	$h(38) = 0.0194$	$h(52) = 0.9167$
$h(11) = 0.0018$	$h(25) = -0.0055$	$h(39) = -0.0218$	
$h(12) = -0.0021$	$h(26) = -0.0020$	$h(40) = -0.0484$	
$h(13) = -0.0050$	$h(27) = 0.0011$	$h(41) = -0.0266$	



(a)

Fig. 5.13. Resulting bandstop filter: (a) impulse response; (b) magnitude response; (c) passband detail.

the Dolph–Chebyshev window is defined as

$$w_{DC}(n) = \begin{cases} \frac{1}{M+1} \left\{ \frac{1}{r} + 2 \sum_{i=1}^{M/2} C_M \left[x_0 \cos\left(\frac{i\pi}{M+1}\right) \right] \cos\left(\frac{2ni\pi}{M+1}\right) \right\}, & \text{for } |n| \leq M/2, \\ 0, & \text{for } |n| > M/2 \end{cases} \quad (5.94)$$

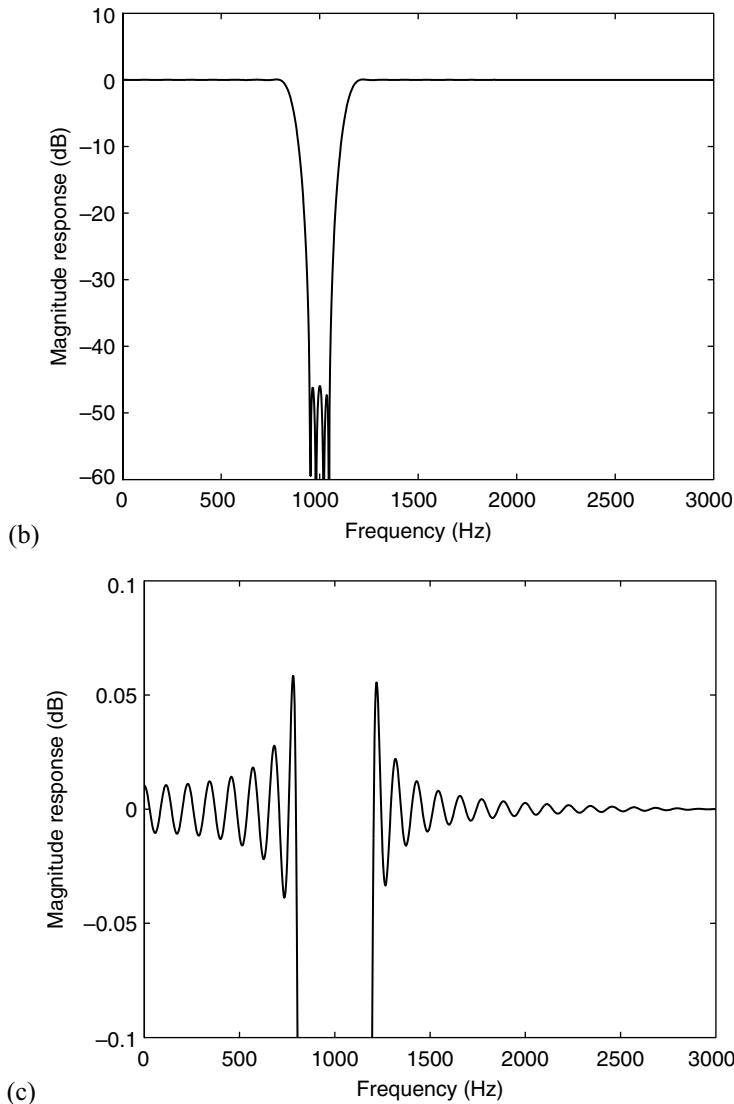


Fig. 5.13. (continued)

where r is the ripple ratio, defined as

$$r = \frac{\delta_r}{\delta_p} \quad (5.95)$$

and x_0 is given by

$$x_0 = \cosh \left[\frac{1}{M} \cosh^{-1} \left(\frac{1}{r} \right) \right]. \quad (5.96)$$

The procedure for designing FIR filters using the Dolph–Chebyshev window is very similar to the one for the Kaiser window:

- (i) Perform steps (i) and (ii) of the Kaiser procedure.
- (ii) Determine r from Equation (5.95).
- (iii) Perform steps (iii)–(v) and (vii)–(viii) of the Kaiser procedure, to determine the filter order M . In step (vii), however, as the stopband attenuation achieved with the Dolph–Chebyshev window is typically 1 to 4 dB higher than that obtained using the Kaiser window, one should compute D for the Dolph–Chebyshev window using Equation (5.74) with A_r replaced by $A_r + 2.5$ (Saramäki, 1993). With this approximation, the resulting order value M may not be precise, and small corrections may be necessary at the end of the design routine to completely satisfy the prescribed specifications.
- (iv) With r and M determined, compute x_0 from Equation (5.96), and then compute the window coefficients from Equation (5.94).
- (v) We are now ready to form the sequence $h'(n) = w_{DC}(n)h(n)$, where $h(n)$ is the ideal filter impulse response computed in step (i) of the Kaiser procedure in Section 5.4.5.
- (vi) Perform step (x) of the Kaiser procedure to determine the resulting FIR filter.

Overall, the Dolph–Chebyshev window is characterized by:

- The main-lobe width, and consequently the resulting filter transition band, can be controlled by varying M .
- The ripple ratio is controlled through an independent parameter r .
- All secondary lobes have the same amplitude. Therefore, the stopband of the resulting filter is equiripple.

5.5 Maximally flat FIR filter approximation

Maximally flat approximations should be employed when a signal must be preserved with minimal error around the zero frequency or when a monotone frequency response is necessary. FIR filters with a maximally flat frequency response at $\omega = 0$ and $\omega = \pi$ were introduced in Herrmann (1971). We consider here, following the standard literature on the subject (Herrmann, 1971; Vaidyanathan, 1984, 1985), the lowpass Type I FIR filter of even order M and symmetric impulse response.

In this case, the frequency response of a maximally flat FIR filter is determined in such a way that $H(e^{j\omega}) - 1$ has $2L$ zeros at $\omega = 0$, and $H(e^{j\omega})$ has $2K$ zeros at $\omega = \pi$. To achieve a maximally flat response, the filter order M must satisfy $M = (2K + 2L - 2)$. Thus, the first $2L - 1$ derivatives of $H(e^{j\omega})$ are zero at $\omega = 0$, and the first $2K - 1$ derivatives of $H(e^{j\omega})$ are zero at $\omega = \pi$. If the above two conditions are satisfied, then $H(e^{j\omega})$ can be

written (Herrmann, 1971) either as

$$\begin{aligned} H(e^{j\omega}) &= \left(\cos \frac{\omega}{2}\right)^{2K} \sum_{n=0}^{L-1} d(n) \left(\sin \frac{\omega}{2}\right)^{2n} \\ &= \left(\frac{1 + \cos \omega}{2}\right)^K \sum_{n=0}^{L-1} d(n) \left(\frac{1 - \cos \omega}{2}\right)^n \end{aligned} \quad (5.97)$$

or as

$$\begin{aligned} H(e^{j\omega}) &= 1 - \left(\sin \frac{\omega}{2}\right)^{2L} \sum_{n=0}^{K-1} \hat{d}(n) \left(\cos \frac{\omega}{2}\right)^{2n} \\ &= 1 - \left(\frac{1 - \cos \omega}{2}\right)^L \sum_{n=0}^{K-1} \hat{d}(n) \left(\frac{1 + \cos \omega}{2}\right)^n, \end{aligned} \quad (5.98)$$

where the coefficients $d(n)$ or $\hat{d}(n)$ are respectively given by

$$d(n) = \frac{(K-1+n)!}{(K-1)!n!} \quad (5.99)$$

$$\hat{d}(n) = \frac{(L-1+n)!}{(L-1)!n!}. \quad (5.100)$$

Note that, from Equations (5.97) and (5.99), $H(e^{j\omega})$ is real and positive. Therefore, for the maximally flat filters, $|H(e^{j\omega})| = H(e^{j\omega})$. From the above equations it is also easy to see that $H(e^{j\omega})$ can be expressed as a sum of complex exponentials in ω having frequencies ranging from $(-K-L+1)$ to $(K+L-1)$, with an increment of 1. Therefore, from Theorem 5.1, one can see that $h(n)$ can be exactly recovered by sampling $H(e^{j\omega})$ at $(2K+2L-1) = M+1$ equally spaced points located at frequencies $\omega = 2\pi n/(M+1)$, for $n = 0, 1, \dots, M$, and taking the IDFT, as in the frequency sampling approach.

A more efficient implementation, however, results from writing the transfer function as (Vaidyanathan, 1984)

$$H(z) = \left(\frac{1+z^{-1}}{2}\right)^{2K} \sum_{n=0}^{L-1} (-1)^n d(n) z^{-(L-1-n)} \left(\frac{1-z^{-1}}{2}\right)^{2n} \quad (5.101)$$

or

$$H(z) = z^{-M/2} - (-1)^L \left(\frac{1-z^{-1}}{2}\right)^{2L} \sum_{n=0}^{K-1} \hat{d}(n) z^{-(K-1-n)} \left(\frac{1+z^{-1}}{2}\right)^{2n}. \quad (5.102)$$

These equations require significantly fewer multipliers than the direct-form implementation. The drawback with these designs is the large dynamic range necessary to represent the sequences $d(n)$ and $\hat{d}(n)$. This can be avoided by an efficient cascade implementation of

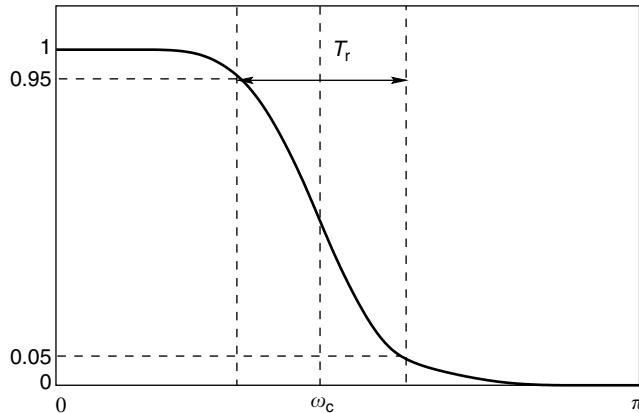


Fig. 5.14. Typical specification of a maximally flat lowpass FIR filter.

these coefficients, as discussed in Vaidyanathan (1984), utilizing the following relationships:

$$d(n+1) = d(n) \frac{K+n}{n+1} \quad (5.103)$$

$$\hat{d}(n+1) = \hat{d}(n) \frac{L+n}{n+1}. \quad (5.104)$$

In the procedure described above, the sole design parameters for the maximally flat FIR filters are the values of K and L . Given a desired magnitude response, as depicted in Figure 5.14, the transition band is defined as the region where the magnitude response varies from 0.95 to 0.05, and the normalized center frequency is the center of this band; that is, $\omega_c = (\omega_p + \omega_t)/2$. If the transition band in rad/sample is T_r , we need to compute the following parameters:

$$M_1 = \left(\frac{\pi}{T_r} \right)^2 \quad (5.105)$$

$$\rho = \frac{1 + \cos \omega_c}{2} \quad (5.106)$$

Then, for all integer values of M_p in the range $M_1 \leq M_p \leq 2M_1$, we compute K_p as the nearest integer to ρM_p . We then choose K_p^* and M_p^* as the values of K_p and M_p for which the ratio K_p/M_p is closest to ρ . The desired values of K , L , and M are given by

$$K = K_p^* \quad (5.107)$$

$$L = M_p^* - K_p^* \quad (5.108)$$

and

$$M = 2K + 2L - 2 = 2M_p^* - 2. \quad (5.109)$$

Table 5.11.

Coefficients $d(0)$ to $d(5)$ of the lowpass filter designed with the maximally flat method.

$d(0) = 1$
$d(1) = 27$
$d(2) = 378$
$d(3) = 3654$
$d(4) = 27405$
$d(5) = 169911$
$d(6) = 906192$

Example 5.5. Design a maximally flat lowpass filter satisfying the specification below:

$$\left. \begin{array}{l} \Omega_c = 0.3\pi \text{ rad/s} \\ T_r = 0.2\pi \text{ rad/s} \\ \Omega_s = 2\pi \text{ rad/s} \end{array} \right\}. \quad (5.110)$$

Solution

One may use the script

```
Omega_c = 0.3*pi; Tr = 0.2*pi; Omega_s = 2*pi;
M1 = (pi/Tr)^2;
rho = (1 + cos(Omega_c))/2;
Mp = ceil(M1):floor(2*M1);
Kp = round(rho*Mp);
rho_p = Kp./Mp;
[value,index] = min(abs(rho_p - rho));
K = Kp(index); L = Mp(index)-Kp(index); M = 2*Mp(index)-2;
```

to obtain the values of $K = 27$, $L = 7$, and $M = 66$. The resulting coefficients $d(n)$, as given in Equation (5.103), may be determined as

```
d(1) = 1;
for i = 0:L-2,
    d(i+2) = d(i+1)*(K+i)/(i+1);
end;
```

and are seen in Table 5.11. Notice that in this case, even though the filter order is $M = 66$, there are only $L = 7$ nonzero coefficients $d(n)$.

The corresponding magnitude response can be determined as

```
omega = 2*pi.* (0:M) / (M+1);
i = (0:L-1)';
```

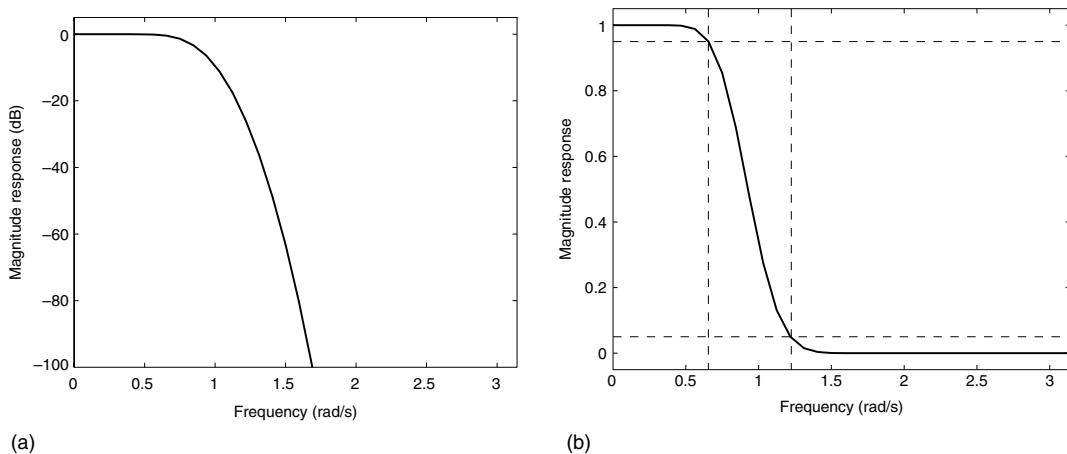


Fig. 5.15. Magnitude response of maximally flat lowpass FIR filter: (a) linear scale; (b) decibel scale.

```

for k = 0:M,
    H(k+1) = d*sin(omega(k+1)/2).^(2*i);
end;
H = H.*cos(omega./2).^(2*K);

```

the result of which is depicted in Figure 5.15. \triangle

5.6 FIR filter approximation by optimization

The window method seen in Section 5.4 has a very straightforward design procedure for approximating the desired magnitude response. However, the window method is not efficient for designing, for example, FIR filters with different ripples in the passband and stopband, or nonsymmetric bandpass or bandstop filters. To fill this gap, in this section we present several numerical algorithms for designing more general FIR digital filters.

In many signal processing systems, filters with linear or zero phase are required. Unfortunately, filters designed to have zero phase are not causal; this can be a problem in applications where very little processing delay is permissible. Also, nonlinear phase causes distortion in the processed signal, which can be very perceptible in applications like data transmission, image processing, and so on. One of the major advantages of using an FIR system instead of a causal IIR system is that FIR systems can be designed with exact linear phase. As seen in Section 4.2.3, there are four distinct cases where an FIR filter presents linear phase. To present general algorithms for designing linear-phase FIR filters, a unified representation of these four cases is necessary. We define an auxiliary function $P(\omega)$ as

$$P(\omega) = \sum_{l=0}^L p(l) \cos(\omega l), \quad (5.111)$$

where $L + 1$ is the number of cosine functions in the expression of $H(e^{j\omega})$. Based on this function, we can express the frequency response of the four types of linear-phase FIR filters as (McClellan & Parks, 1973):

- Type I: even order M and symmetric impulse response. From Table 4.1, we can write that

$$\begin{aligned} H(e^{j\omega}) &= e^{-j\omega M/2} \sum_{m=0}^{M/2} a(m) \cos(\omega m) \\ &= e^{-j\omega M/2} \sum_{l=0}^{M/2} p(l) \cos(\omega l) \\ &= e^{-j\omega M/2} P(\omega) \end{aligned} \quad (5.112)$$

with

$$a(m) = p(m), \text{ for } m = 0, 1, \dots, L, \quad (5.113)$$

where $L = M/2$.

- Type II: odd order M and symmetric impulse response. In this case, from Table 4.1, we have

$$H(e^{j\omega}) = e^{-j\omega M/2} \sum_{m=1}^{(M+1)/2} b(m) \cos\left[\omega\left(m - \frac{1}{2}\right)\right]. \quad (5.114)$$

Using

$$b(m) = \begin{cases} p(0) + \frac{1}{2}p(1), & \text{for } m = 1 \\ \frac{1}{2}(p(m-1) + p(m)), & \text{for } m = 2, 3, \dots, L \\ \frac{1}{2}p(L), & \text{for } m = L + 1 \end{cases} \quad (5.115)$$

with $L = (M - 1)/2$, then $H(e^{j\omega})$ can be written in the form

$$H(e^{j\omega}) = e^{-j\omega M/2} \cos\left(\frac{\omega}{2}\right) P(\omega) \quad (5.116)$$

using the trigonometric identity

$$2 \cos\left(\frac{\omega}{2}\right) \cos(\omega m) = \cos\left[\omega\left(m + \frac{1}{2}\right)\right] + \cos\left[\omega\left(m - \frac{1}{2}\right)\right]. \quad (5.117)$$

The complete algebraic development is left as an exercise to the interested reader.

- Type III: even order M and antisymmetric impulse response. In this case, from Table 4.1, we have

$$H(e^{j\omega}) = e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{M/2} c(m) \sin(\omega m) \quad (5.118)$$

and then, by substituting

$$c(m) = \begin{cases} p(0) - \frac{1}{2}p(2), & \text{for } m = 1 \\ \frac{1}{2}(p(m-1) - p(m+1)), & \text{for } m = 2, 3, \dots, (L-1) \\ \frac{1}{2}p(m-1), & \text{for } m = L, L+1 \end{cases} \quad (5.119)$$

with $L = (M/2) - 1$, Equation (5.118) can be written as

$$H(e^{j\omega}) = e^{-j[\omega(M/2) - (\pi/2)]} \sin(\omega)P(\omega). \quad (5.120)$$

using, in this case, the identity

$$2 \sin(\omega) \cos(\omega m) = \sin[\omega(m+1)] - \sin[\omega(m-1)]. \quad (5.121)$$

Once again, the algebraic proof is left as an exercise at the end of this chapter.

- Type IV: odd order M and antisymmetric impulse response. We have, from Table 4.1, that

$$H(e^{j\omega}) = e^{-j[\omega(M/2) - (\pi/2)]} \sum_{m=1}^{(M+1)/2} d(m) \sin\left[\omega\left(m - \frac{1}{2}\right)\right]. \quad (5.122)$$

By substituting

$$d(m) = \begin{cases} p(0) - \frac{1}{2}p(1), & \text{for } m = 1 \\ \frac{1}{2}(p(m-1) - p(m)), & \text{for } m = 2, 3, \dots, L \\ \frac{1}{2}p(L), & \text{for } m = L+1 \end{cases} \quad (5.123)$$

with $L = (M-1)/2$, then $H(e^{j\omega})$ can be written as

$$H(e^{j\omega}) = e^{-j[\omega(M/2) - (\pi/2)]} \sin\left(\frac{\omega}{2}\right) P(\omega) \quad (5.124)$$

using the identity

$$2 \sin\left(\frac{\omega}{2}\right) \cos(\omega m) = \sin\left[\omega\left(m + \frac{1}{2}\right)\right] - \sin\left[\omega\left(m - \frac{1}{2}\right)\right]. \quad (5.125)$$

Once more, the entire algebraic development is left as an exercise to the reader.

Equations (5.112), (5.116), (5.120), and (5.124) indicate that we can write the frequency response for any linear-phase FIR filter as

$$H(e^{j\omega}) = e^{-j(\alpha\omega - \beta)} Q(\omega) P(\omega) = e^{-j(\alpha\omega - \beta)} A(\omega), \quad (5.126)$$

where $A(\omega) = Q(\omega)P(\omega)$, $\alpha = M/2$, and for each case we have that:

- Type I: $\beta = 0$ and $Q(\omega) = 1$.
- Type II: $\beta = 0$ and $Q(\omega) = \cos(\omega/2)$.
- Type III: $\beta = \pi/2$ and $Q(\omega) = \sin(\omega)$.
- Type IV: $\beta = \pi/2$ and $Q(\omega) = \sin(\omega/2)$.

Let $D(\omega)$ be the desired amplitude response. We define the weighted error function as

$$E(\omega) = W(\omega)(D(\omega) - A(\omega)). \quad (5.127)$$

We can then write $E(\omega)$ as

$$\begin{aligned} E(\omega) &= W(\omega)(D(\omega) - Q(\omega)P(\omega)) \\ &= W(\omega)Q(\omega) \left(\frac{D(\omega)}{Q(\omega)} - P(\omega) \right) \end{aligned} \quad (5.128)$$

for all $0 \leq \omega \leq \pi$, as $Q(\omega)$ is independent of the coefficients for each ω . Defining

$$W_q(\omega) = W(\omega)Q(\omega) \quad (5.129)$$

$$D_q(\omega) = \frac{D(\omega)}{Q(\omega)}, \quad (5.130)$$

the error function can be rewritten as

$$E(\omega) = W_q(\omega)(D_q(\omega) - P(\omega)) \quad (5.131)$$

and one can formulate the optimization problem for approximating linear-phase FIR filters as: *determine the set of coefficients $p(l)$ that minimizes some objective function of the weighted error $E(\omega)$ over a set of prescribed frequency bands.*

To solve such a problem numerically, we evaluate the weighted error function on a dense frequency grid with $0 \leq \omega_i \leq \pi$, for $i = 1, 2, \dots, KM$, where M is the filter order, obtaining a good discrete approximation of $E(\omega)$. For most practical purposes, using $8 \leq K \leq 16$ is recommended. Points associated with the transition bands can be disregarded, and the remaining frequencies should be linearly redistributed in the passbands and stopbands to include their corresponding edges. Thus, the following equation results:

$$\mathbf{e} = \mathbf{W}_q (\mathbf{d}_q - \mathbf{U}\mathbf{p}), \quad (5.132)$$

where

$$\mathbf{e} = [E(\omega_1) \ E(\omega_2) \ \dots \ E(\omega_{KM})]^T \quad (5.133)$$

$$\mathbf{W}_q = \text{diag} [W_q(\omega_1) \ W_q(\omega_2) \ \dots \ W_q(\omega_{KM})] \quad (5.134)$$

$$\mathbf{d}_q = [D_q(\omega_1) \ D_q(\omega_2) \ \dots \ D_q(\omega_{KM})]^T \quad (5.135)$$

$$\mathbf{U} = \begin{bmatrix} 1 & \cos(\omega_1) & \cos(2\omega_1) & \dots & \cos(L\omega_1) \\ 1 & \cos(\omega_2) & \cos(2\omega_2) & \dots & \cos(L\omega_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos(\omega_{\overline{KM}}) & \cos(2\omega_{\overline{KM}}) & \dots & \cos(L\omega_{\overline{KM}}) \end{bmatrix} \quad (5.136)$$

$$\mathbf{p} = [p(0) \ p(1) \ \dots \ p(L)]^T, \quad (5.137)$$

with $\overline{KM} \leq KM$, as the original frequencies in the transition band were discarded.

For the four standard types of filter, namely lowpass, highpass, bandpass, and bandstop, as well as differentiators and Hilbert transformers, the definitions of $W(\omega)$ and $D(\omega)$ are summarized in Table 5.12.

It is important to remember all design constraints, due to the magnitude and phase characteristics of the four linear-phase filter types. Such constraints are summarized in Table 5.13, where a “Yes” entry indicates that the corresponding filter structure is suitable to implement the given filter type.

5.6.1 Weighted least-squares method

In the weighted least-squares (WLS) approach, the idea is to minimize the square of the energy of the error function $E(\omega)$; that is:

$$\min_{\mathbf{p}} \left\{ \|E(\omega)\|_2^2 \right\} = \min_{\mathbf{p}} \left\{ \int_0^\pi |E(\omega)|^2 d\omega \right\}. \quad (5.138)$$

For a discrete set of frequencies, this objective function is approximated by (see Equations (5.132)–(5.137))

$$\|E(\omega)\|_2^2 \approx \frac{1}{KM} \sum_{k=1}^{\overline{KM}} |E(\omega_k)|^2 = \frac{1}{KM} \mathbf{e}^T \mathbf{e}, \quad (5.139)$$

since in these equations \mathbf{e} is a real vector. Using Equation (5.132), and noting that \mathbf{W}_q is diagonal, we can write that

$$\begin{aligned} \mathbf{e}^T \mathbf{e} &= (\mathbf{d}_q^T - \mathbf{p}^T \mathbf{U}^T) \mathbf{W}_q^T \mathbf{W}_q (\mathbf{d}_q - \mathbf{U} \mathbf{p}) \\ &= (\mathbf{d}_q^T - \mathbf{p}^T \mathbf{U}^T) \mathbf{W}_q^2 (\mathbf{d}_q - \mathbf{U} \mathbf{p}) \\ &= \mathbf{d}_q^T \mathbf{W}_q^2 \mathbf{d}_q - \mathbf{d}_q^T \mathbf{W}_q^2 \mathbf{U} \mathbf{p} - \mathbf{p}^T \mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q + \mathbf{p}^T \mathbf{U}^T \mathbf{W}_q^2 \mathbf{U} \mathbf{p} \\ &= \mathbf{d}_q^T \mathbf{W}_q^2 \mathbf{d}_q - 2\mathbf{p}^T \mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q + \mathbf{p}^T \mathbf{U}^T \mathbf{W}_q^2 \mathbf{U} \mathbf{p} \end{aligned} \quad (5.140)$$

because $\mathbf{d}_q^T \mathbf{W}_q^2 \mathbf{U} \mathbf{p} = \mathbf{p}^T \mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q$, since these two terms are scalar. The minimization of such a functional is achieved by calculating its gradient vector with respect to the coefficient

Table 5.12.

Weight functions and ideal magnitude responses for basic lowpass, highpass, bandpass, and bandstop filters, as well as differentiators and Hilbert transformers.

Filter type	Weight function $W(\omega)$	Ideal amplitude response $D(\omega)$
Lowpass	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_p \\ \frac{\delta_p}{\delta_r}, & \text{for } \omega_r \leq \omega \leq \pi \end{cases}$	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_p \\ 0, & \text{for } \omega_r \leq \omega \leq \pi \end{cases}$
Highpass	$\begin{cases} \frac{\delta_p}{\delta_r}, & \text{for } 0 \leq \omega \leq \omega_r \\ 1, & \text{for } \omega_p \leq \omega \leq \pi \end{cases}$	$\begin{cases} 0, & \text{for } 0 \leq \omega \leq \omega_r \\ 1, & \text{for } \omega_p \leq \omega \leq \pi \end{cases}$
Bandpass	$\begin{cases} \frac{\delta_p}{\delta_r}, & \text{for } 0 \leq \omega \leq \omega_{r1} \\ 1, & \text{for } \omega_{p1} \leq \omega \leq \omega_{p2} \\ \frac{\delta_p}{\delta_r}, & \text{for } \omega_{r2} \leq \omega \leq \pi \end{cases}$	$\begin{cases} 0, & \text{for } 0 \leq \omega \leq \omega_{r1} \\ 1, & \text{for } \omega_{p1} \leq \omega \leq \omega_{p2} \\ 0, & \text{for } \omega_{r2} \leq \omega \leq \pi \end{cases}$
Bandstop	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_{p1} \\ \frac{\delta_p}{\delta_r}, & \text{for } \omega_{r1} \leq \omega \leq \omega_{r2} \\ 1, & \text{for } \omega_{p2} \leq \omega \leq \pi \end{cases}$	$\begin{cases} 1, & \text{for } 0 \leq \omega \leq \omega_{p1} \\ 0, & \text{for } \omega_{r1} \leq \omega \leq \omega_{r2} \\ 1, & \text{for } \omega_{p2} \leq \omega \leq \pi \end{cases}$
Differentiator	$\begin{cases} \frac{1}{\omega}, & \text{for } 0 < \omega \leq \omega_p \\ 0, & \text{for } \omega_p < \omega \leq \pi \end{cases}$	$\omega, \quad \text{for } 0 \leq \omega \leq \pi$
Hilbert transformer	$\begin{cases} 0, & \text{for } 0 \leq \omega < \omega_{p1} \\ 1, & \text{for } \omega_{p1} \leq \omega \leq \omega_{p2} \\ 0, & \text{for } \omega_{p2} < \omega \leq \pi \end{cases}$	$1, \quad \text{for } 0 \leq \omega \leq \pi$

Table 5.13.

Suitability of linear-phase FIR structures for basic lowpass, highpass, bandpass, and bandstop filters, as well as differentiators and Hilbert transformers.

Filter type	Type I	Type II	Type III	Type IV
Lowpass	Yes	Yes	No	No
Highpass	Yes	No	No	Yes
Bandpass	Yes	Yes	Yes	Yes
Bandstop	Yes	No	No	No
Differentiator	No	No	Yes	Yes
Hilbert transformer	No	No	Yes	Yes

vector and equating it to zero. Since

$$\nabla_{\mathbf{x}} \{\mathbf{A}\mathbf{x}\} = \mathbf{A}^T \quad (5.141)$$

$$\nabla_{\mathbf{x}} \{\mathbf{x}^T \mathbf{A}\mathbf{x}\} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}, \quad (5.142)$$

this yields

$$\nabla_{\mathbf{p}} \left\{ \mathbf{e}^T \mathbf{e} \right\} = -2\mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q + 2\mathbf{U}^T \mathbf{W}_q^2 \mathbf{U} \mathbf{p}^* = \mathbf{0}, \quad (5.143)$$

which implies that

$$\mathbf{p}^* = \left(\mathbf{U}^T \mathbf{W}_q^2 \mathbf{U} \right)^{-1} \mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q. \quad (5.144)$$

It can be shown that, when the weight function $W(\omega)$ is made constant, the WLS approach is equivalent to the rectangular window presented in the previous section, and so suffers from the same problem of Gibbs' oscillations near the band edges. When $W(\omega)$ is not constant, the oscillations still occur but their energies will vary from band to band.

Amongst the several extensions and generalizations of the WLS approach, we refer here to the constrained-WLS and eigenfilter methods. The constrained-WLS method was presented in Selesnick *et al.* (1996, 1998). In this method, the designer specifies the maximum and minimum values allowed for the desired magnitude response in each band. In the proposed algorithm, the transition bands are not completely specified, as only their central frequencies need to be provided. The transition bands are then automatically adjusted to satisfy the constraints. The overall method consists of an iterative procedure where in each step a modified WLS design is performed, using Lagrange multipliers, and the constraints are subsequently tested and updated. Such a procedure involves verification of the Kuhn–Tucker conditions (Winston, 1991) – that is, it checks if all the resulting multipliers are nonnegative – followed by a search routine that finds the positions of all local extremals in each band and tests if all the constraints are satisfied. For the eigenfilter method presented by Vaidyanathan (1987) and used in Nguyen *et al.* (1994), the objective function in Equation (5.131) is rewritten in a distinct form and the results from linear algebra are used to find the optimal filter for the resulting equation. With such a procedure, the eigenfilter method enables linear-phase FIR filters with different characteristics to be designed, and the WLS scheme appears as a special case of the eigenfilter approach.

Example 5.6. Design a Hilbert transformer of order $M = 5$ using the WLS approach by choosing an appropriate grid of only three frequencies. Obtain \mathbf{p}^* and the filter transfer function.

Solution

For the odd order $M = 5$, the FIR Hilbert transformer should be of Type IV and the number of coefficients of \mathbf{p} is $(L + 1)$, where $L = (M - 1)/2 = 2$.

According to Equations (5.131) and (5.132), the response error is

$$\mathbf{e} = \begin{bmatrix} \sin(\omega_1/2) & 0 & 0 \\ 0 & \sin(\omega_2/2) & 0 \\ 0 & 0 & \sin(\omega_3/2) \end{bmatrix} \left\{ \begin{bmatrix} \frac{1}{\sin(\omega_1/2)} \\ \frac{1}{\sin(\omega_2/2)} \\ \frac{1}{\sin(\omega_3/2)} \end{bmatrix} - \begin{bmatrix} 1 & \cos \omega_1 & \cos 2\omega_1 \\ 1 & \cos \omega_2 & \cos 2\omega_2 \\ 1 & \cos \omega_3 & \cos 2\omega_3 \end{bmatrix} \begin{bmatrix} p(0) \\ p(1) \\ p(2) \end{bmatrix} \right\}. \quad (5.145)$$

We then form the frequency grid within the range defined in Table 5.12, such as

$$\left. \begin{array}{l} \omega_1 = \frac{\pi}{3} \\ \omega_2 = \frac{\pi}{2} \\ \omega_3 = \frac{2\pi}{3} \end{array} \right\}, \quad (5.146)$$

in such a way that the error vector becomes

$$\mathbf{e} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 1 & 0 & -1 \\ 1 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} p(0) \\ p(1) \\ p(2) \end{bmatrix} \right\}. \quad (5.147)$$

The WLS solution requires the following matrix:

$$\begin{aligned} \mathbf{U}^T \mathbf{W}_q^2 \mathbf{U} &= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & -1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 1 & 0 & -1 \\ 1 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \\ &= \frac{1}{4} \begin{bmatrix} 6 & -1 & -4 \\ -1 & 1 & \frac{1}{2} \\ -4 & \frac{1}{2} & 3 \end{bmatrix}, \end{aligned} \quad (5.148)$$

whose inverse is

$$(\mathbf{U}^T \mathbf{W}_q^2 \mathbf{U})^{-1} = \frac{1}{3} \begin{bmatrix} 22 & 8 & 28 \\ 8 & 16 & 8 \\ 28 & 8 & 40 \end{bmatrix}. \quad (5.149)$$

Then, the vector \mathbf{p}^* is computed as follows:

$$\begin{aligned}
 \mathbf{p}^* &= (\mathbf{U}^T \mathbf{W}_q^2 \mathbf{U})^{-1} \mathbf{U}^T \mathbf{W}_q^2 \mathbf{d}_q \\
 &= \frac{1}{3} \begin{bmatrix} 22 & 8 & 28 \\ 8 & 16 & 8 \\ 28 & 8 & 40 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & -1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} \end{bmatrix}^2 \begin{bmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1.7405 \\ 0.8453 \\ 0.3263 \end{bmatrix}.
 \end{aligned} \tag{5.150}$$

According to Equations (5.123) and (4.28), we then have

$$\left. \begin{array}{l} d(1) = p(0) - \frac{1}{2}p(1) = 2h(2) = 1.31785 \\ d(2) = \frac{1}{2}(p(1) - p(2)) = 2h(1) = 0.2595 \\ d(3) = \frac{1}{2}p(2) = 2h(0) = 0.16315 \end{array} \right\} \tag{5.151}$$

and the overall transfer function is given by

$$H(z) = 0.0816 + 0.1298z^{-1} + 0.6589z^{-2} - 0.6589z^{-3} - 0.1298z^{-4} - 0.0816z^{-5}. \tag{5.152}$$

If a Hilbert filter of the same order is designed with the MATLAB `firls` command, which uses a uniform sampling to determine the frequency grid, then the resulting transfer function is

$$\bar{H}(z) = -0.0828 - 0.1853z^{-1} - 0.6277z^{-2} + 0.6277z^{-3} + 0.1853z^{-4} + 0.0828z^{-5}. \tag{5.153}$$

As can be observed in Figure 5.16, $H(z)$ and $\bar{H}(z)$ have very similar magnitude responses, with the differences arising from the nonuniform frequency grid employed in the $H(z)$ design for didactic purposes.

△

5.6.2 Chebyshev method

In the Chebyshev optimization design approach, the idea is to minimize the maximum absolute value of the error function $E(\omega)$. Mathematically, such a scheme is described by

$$\min_{\mathbf{p}} \{ \|E(\omega)\|_{\infty}\} = \min_{\mathbf{p}} \left\{ \max_{\omega \in F} \{|E(\omega)|\} \right\}, \tag{5.154}$$

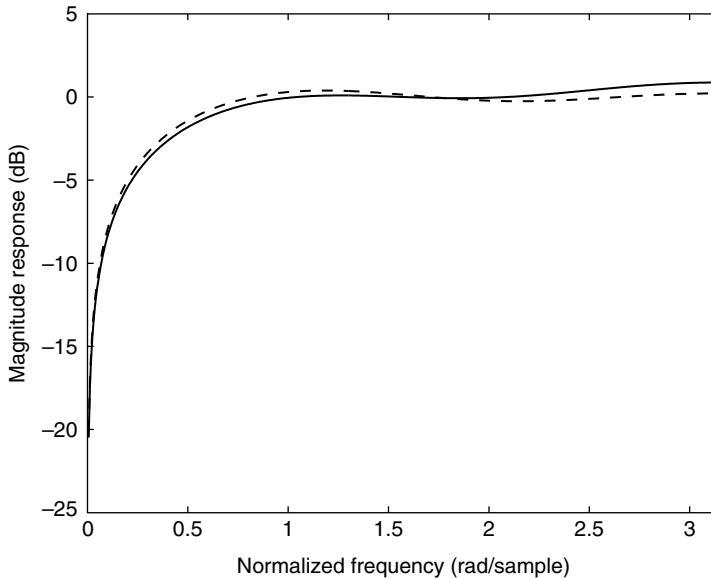


Fig. 5.16. Magnitude responses of Hilbert transformers in Example 5.6: step-by-step $H(z)$ design (solid line) and Matlab $\bar{H}(z)$ design (dashed line).

where F is the set of prescribed frequency bands. This problem can be solved with the help of the following important theorem.

Theorem 5.2 (Alternation Theorem). *If $P(\omega)$ is a linear combination of $(L + 1)$ cosine functions, that is*

$$P(\omega) = \sum_{l=0}^L p(l) \cos(\omega l), \quad (5.155)$$

then the necessary and sufficient condition for $P(\omega)$ to be the Chebyshev approximation of a continuous function $D(\omega)$ in F , a compact subset of $[0, \pi]$, is that the error function $E(\omega)$ must present at least $(L + 2)$ extreme frequencies in F . That is, there must be at least $(L + 2)$ points ω_k in F , where $\omega_0 < \omega_1 < \dots < \omega_{L+1}$, such that

$$E(\omega_k) = -E(\omega_{k+1}), \text{ for } k = 0, 1, \dots, L \quad (5.156)$$

and

$$|E(\omega_k)| = \max_{\omega \in F} \{|E(\omega)|\}, \text{ for } k = 0, 1, \dots, (L + 1). \quad (5.157)$$

◇

A proof of this theorem can be found in Cheney (1966).

The extremes of $E(\omega)$ are related to the extremes of $A(\omega)$ defined in Equation (5.127). The values of ω for which $\partial A(\omega)/\partial \omega = 0$ allow us to determine that the number N_k of extremes of $A(\omega)$ is such that:

- Type I: $N_k \leq M + 2/2$.
- Type II: $N_k \leq M + 1/2$.
- Type III: $N_k \leq M/2$.
- Type IV: $N_k \leq M + 1/2$.

In general, the extremes of $A(\omega)$ are also extremes of $E(\omega)$. However, $E(\omega)$ presents more extremes than $A(\omega)$, as $E(\omega)$ may present extremes at the band edges, which are, in general, not extremes of $A(\omega)$. The only exception to this rule occurs at the band edges $\omega = 0$ or $\omega = \pi$, where $A(\omega)$ also presents an extreme. For instance, for a Type I bandstop filter, as depicted in Figure 5.17, $E(\omega)$ will have up to $(M/2) + 5$ extremes, where $(M/2) + 1$ are extremes of $A(\omega)$, and the other four are band edges.

To solve the Chebyshev approximation problem, we briefly describe the Remez exchange algorithm that searches for the extreme frequencies of $E(\omega)$ through the following steps:

- (i) Initialize an estimate for the extreme frequencies $\omega_0, \omega_1, \dots, \omega_{L+1}$ by selecting $(L+2)$ equally spaced frequencies at the bands specified for the desired filter.

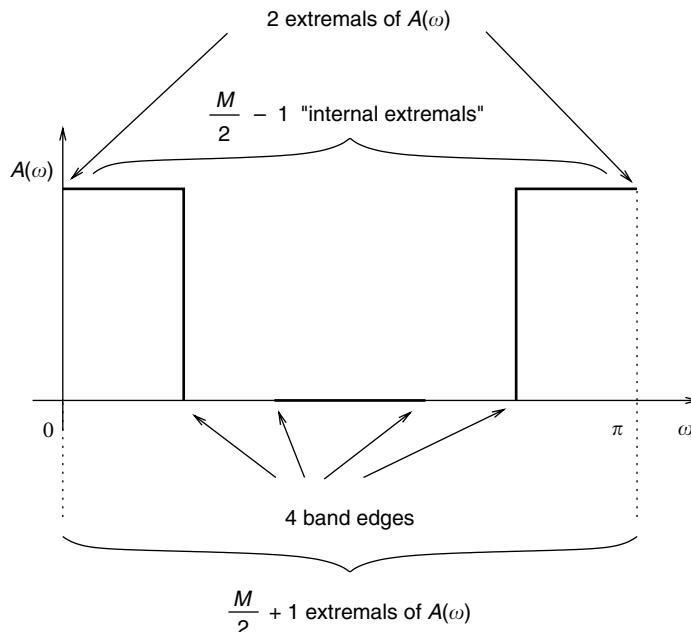


Fig. 5.17. Extremes of $A(\omega)$ for a bandstop filter.

- (ii) Find $P(\omega_k)$ and δ such that

$$W_q(\omega_k)(D_q(\omega_k) - P(\omega_k)) = (-1)^k \delta, \text{ for } k = 0, 1, \dots, (L+1). \quad (5.158)$$

This equation can be written in a matrix form and have its solution analytically calculated. Such a procedure, however, is computationally intensive (Rabiner *et al.*, 1975). An alternative and more efficient approach computes δ by

$$\delta = \frac{a_0 D_q(\omega_0) + a_1 D_q(\omega_1) + \dots + a_{L+1} D_q(\omega_{L+1})}{\frac{a_0}{W_q(\omega_0)} - \frac{a_1}{W_q(\omega_1)} + \dots + \frac{(-1)^{L+1} a_{L+1}}{W_q(\omega_{L+1})}} \quad (5.159)$$

where

$$a_k = \prod_{i=0, i \neq k}^{L+1} \frac{1}{\cos \omega_k - \cos \omega_i}. \quad (5.160)$$

- (iii) Use the barycentric-form Lagrange interpolator for $P(\omega)$; that is:

$$P(\omega) = \begin{cases} c_k, & \text{for } \omega = \omega_k \in \{\omega_0, \omega_1, \dots, \omega_L\} \\ \sum_{i=0}^L \frac{\beta_i}{\cos \omega - \cos \omega_i} c_i, & \text{for } \omega \notin \{\omega_0, \omega_1, \dots, \omega_L\} \\ \sum_{i=0}^L \frac{\beta_i}{\cos \omega - \cos \omega_i} \end{cases}, \quad (5.161)$$

where

$$c_k = D_q(\omega_k) - (-1)^k \frac{\delta}{W_q(\omega_k)} \quad (5.162)$$

$$\beta_k = \prod_{i=0, i \neq k}^L \frac{1}{\cos \omega_k - \cos \omega_i} = a_k (\cos \omega_k - \cos \omega_{L+1}) \quad (5.163)$$

for $k = 0, 1, \dots, (L+1)$.

- (iv) Evaluate $|E(\omega)|$ in a dense set of frequencies. If $|E(\omega)| \leq |\delta|$ for all frequencies in the set, the optimal solution has been found, go to the next step. If $|E(\omega)| > |\delta|$ for some frequencies, a new set of candidate extremes must be chosen as the peaks of $|E(\omega)|$. In that manner, we force δ to grow and to converge to its upper limit. If there are more than $(L+2)$ peaks in $|E(\omega)|$, keep the locations of the $(L+2)$ largest values of the peaks of $|E(\omega)|$, making sure that the band edges are always kept, and return to step (ii).
- (v) Since $P(\omega)$ is a sum of $(L+1)$ cosines, with frequencies from zero to L , then it is also a sum of $(2L+1)$ complex exponentials, with frequencies from $-L$ to L . Then, from Theorem 4.1, $p(l)$ can be exactly recovered by sampling $P(\omega)$ at $(2L+1)$ equally

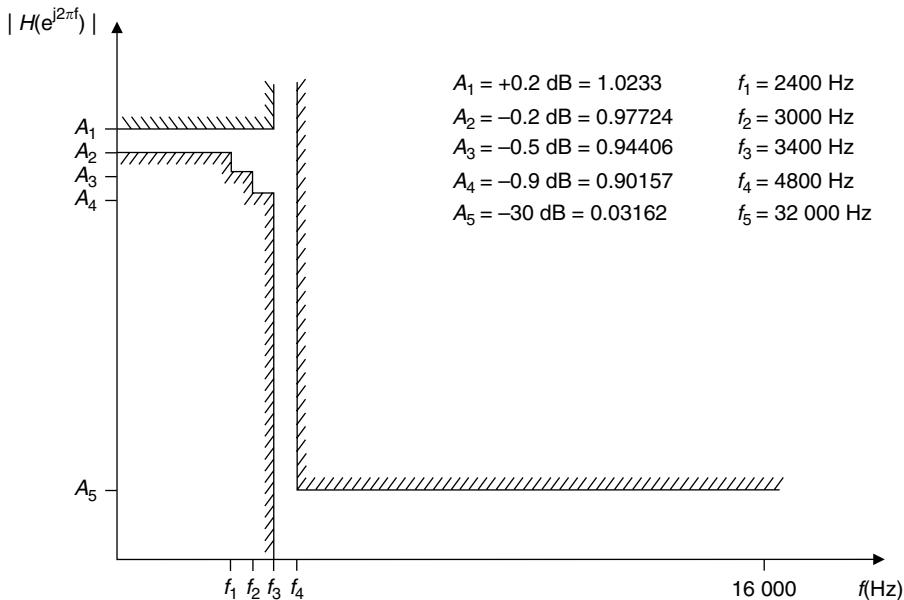


Fig. 5.18. Lowpass PCM filter specifications.

spaced frequencies $\omega = 2\pi n/(2L + 1)$, for $n = 0, 1, \dots, 2L$ and taking the IDFT. The resulting impulse response follows from Equations (5.113), (5.115), (5.119), or (5.123), depending on the filter type.

Such an algorithm was coded in Fortran and published in McClellan *et al.* (1973). A few improvements to speed up the overall convergence routine are given in Antoniou (1982, 1983). The corresponding MATLAB function is `firpm` (see Section 5.8).

Example 5.7. Design the pulse-coded modulation (PCM) filter specified as in Figure 5.18.

Solution

Two approaches are employed:

- In the first approach, we simplify the specifications and consider a single passband with constant weight and ideal magnitude response. In this case, the specifications employed correspond to the following description:

$$\begin{aligned} f3 &= 3400; \quad f4 = 4800; \quad f5 = 32000; \\ Ap &= 0.4; \quad Ar = 30; \end{aligned}$$

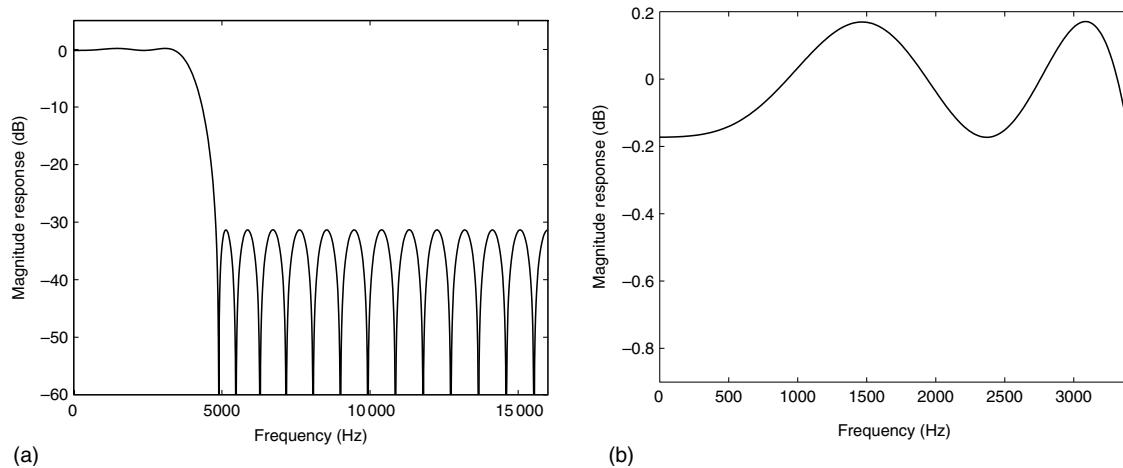
One may estimate the required filter order using the command lines

```
F = [f3 f4];
A = [1 0];
delta_p = (10^(0.05*Ap) - 1)/(10^(0.05*Ap) + 1);
delta_r = 10^(-0.05*Ar);
```

Table 5.14.

Coefficients $h(0)$ to $h(17)$ for optimal PCM filter obtained with approach 1.

$h(0) = 0.0153$	$h(5) = -0.0062$	$h(10) = -0.0237$	$h(15) = 0.1569$
$h(1) = 0.0006$	$h(6) = 0.0093$	$h(11) = -0.0482$	$h(16) = 0.2294$
$h(2) = -0.0066$	$h(7) = 0.0226$	$h(12) = -0.0474$	$h(17) = 0.2572$
$h(3) = -0.0139$	$h(8) = 0.0231$	$h(13) = -0.0078$	
$h(4) = -0.0149$	$h(9) = 0.0057$	$h(14) = 0.0674$	

**Fig. 5.19.** Approach 1: (a) magnitude response; (b) passband detail.

```
ripples = [delta_p delta_r];
M = firpmord(F,A,ripples,f5);
```

which yields $M = 32$. Such a value, however, is not able to satisfy the filter requirements, forcing one to use $M = 34$. The desired filter can then be designed as

```
wp = f3*2/f5; wr = f4*2/f5;
F1 = [0 f3 f4 f5/2]*2/f5;
A1 = [1 1 0 0];
W1 = [1 delta_p/delta_r];
h = firpm(M,F1,A1,W1);
```

which yields the filter coefficients provided in Table 5.14 for $0 \leq n \leq 17$, with $h(n) = h(34 - n)$, and the magnitude response shown in Figure 5.19.

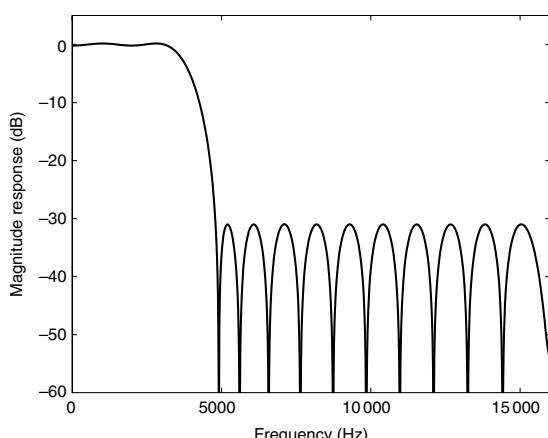
- In a second approach, we explore the loosened specifications along the passband and characterize an additional band for the `firpm` command:

```
f1 = 2400; f2 = 3000;
F2 = [0 f1 f2 f3 f4 f5/2]*2/f5;
a1 = 10^(0.05*0.2);
```

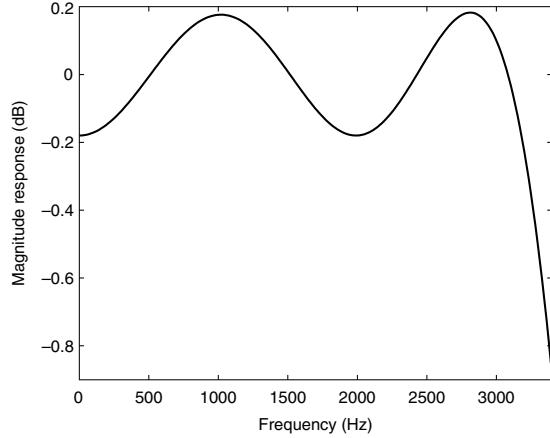
Table 5.15.

Coefficients $h(0)$ to $h(14)$ for optimal PCM filter obtained with approach 2.

$h(0) = -0.0186$	$h(4) = 0.0230$	$h(8) = -0.0456$	$h(12) = 0.1579$
$h(1) = -0.0099$	$h(5) = 0.0178$	$h(9) = -0.0426$	$h(13) = 0.2239$
$h(2) = 0.0010$	$h(6) = 0.0017$	$h(10) = 0.0002$	$h(14) = 0.2517$
$h(3) = 0.0114$	$h(7) = -0.0274$	$h(11) = 0.0712$	



(a)



(b)

Fig. 5.20. Approach 2: (a) magnitude response; (b) passband detail.

```
a4 = 10^(0.05*(-0.9));
gain = (a1+a4)/2-0.005;
A2 = [1 1 gain gain 0 0];
delta_p2 = (a1-a4)/2;
W2 = [1 delta_p/delta_p2 delta_p/delta_r];
h2 = firpm(M,F2,A2,W2);
```

In this case, the simplest filter obtained was of order $M = 28$. The filter magnitude response is shown in Figure 5.20, and its coefficients are listed in Table 5.15, where only half of the coefficients are shown due to filter symmetry. The remaining coefficients are determined by $h(n) = h(28 - n)$.



5.6.3 WLS–Chebyshev method

In the standard literature, the design of FIR filters is dominated by the Chebyshev and the WLS approaches. Some applications that use narrowband filters, like frequency-division multiplexing for communications, do require both the minimum stopband attenuation and

the total stopband energy to be considered simultaneously. For these cases, it can be shown that both the Chebyshev and WLS approaches are unsuitable, as they completely disregard one of these two measurements in their objective function (Adams, 1991a,b). A solution to this problem is to combine the positive aspects of the WLS and Chebyshev methods to obtain a design procedure with good characteristics with respect to both the minimum attenuation and the total energy in the stopband.

Lawson (1968) derived a scheme that performs Chebyshev approximation as a limit of a special sequence of weighted least- p (L_p) approximations, with p fixed. The particular case with $p = 2$ thus relates the Chebyshev approximation to the WLS method. The L_2 Lawson algorithm is implemented by a series of WLS approximations using a varying weight matrix \mathbf{W}_k , the elements of which are calculated by (Rice & Usow, 1968)

$$W_{k+1}^2(\omega) = W_k^2(\omega)B_k(\omega), \quad (5.164)$$

where

$$B_k(\omega) = |E_k(\omega)|. \quad (5.165)$$

Convergence of the Lawson algorithm is slow; in practice, usually 10 to 15 WLS iterations are required to approximate the Chebyshev solution. An efficiently accelerated version of the Lawson algorithm was presented by Lim *et al.* (1992). The Lim–Lee–Chen–Yang (LLCY) approach is characterized by the weight matrix \mathbf{W}_k recurrently updated by

$$W_{k+1}^2(\omega) = W_k^2(\omega)B_k^e(\omega), \quad (5.166)$$

where $B_k^e(\omega)$ is the envelope function of $B_k(\omega)$ composed by a set of piecewise linear segments that start and end at consecutive extremes of $B_k(\omega)$. Band edges are considered extreme frequencies, and edges from different bands are not connected. In that manner, labeling the extreme frequencies at a particular iteration k as ω_J^* , for $J \in \mathbb{N}$, the envelope function is formed as (Lim *et al.*, 1992)

$$B_k^e(\omega) = \frac{(\omega - \omega_J^*)B_k(\omega_{J+1}^*) + (\omega_{J+1}^* - \omega)B_k(\omega_J^*)}{(\omega_{J+1}^* - \omega_J^*)}; \quad \omega_J^* \leq \omega \leq \omega_{J+1}^*. \quad (5.167)$$

Figure 5.21 depicts a typical format of the absolute value of the error function (dash-dotted curve), at any particular iteration, used by the Lawson algorithm to update the weighting function, and its corresponding envelope (solid curve) used by the LLCY algorithm.

Comparing the adjustments used by the Lawson and LLCY algorithms, described in Equations (5.164)–(5.167), and seen in Figure 5.21, with the piecewise-constant weight function used by the WLS method, one can devise a very simple approach for designing digital filters that compromises the minimax and WLS constraints. The approach consists of a modification of the weight-function updating procedure so that it becomes constant after a particular extreme of the stopband of $B_k(\omega)$; that is (Diniz & Netto, 1999):

$$W_{k+1}^2(\omega) = W_k^2(\omega)\beta_k(\omega) \quad (5.168)$$

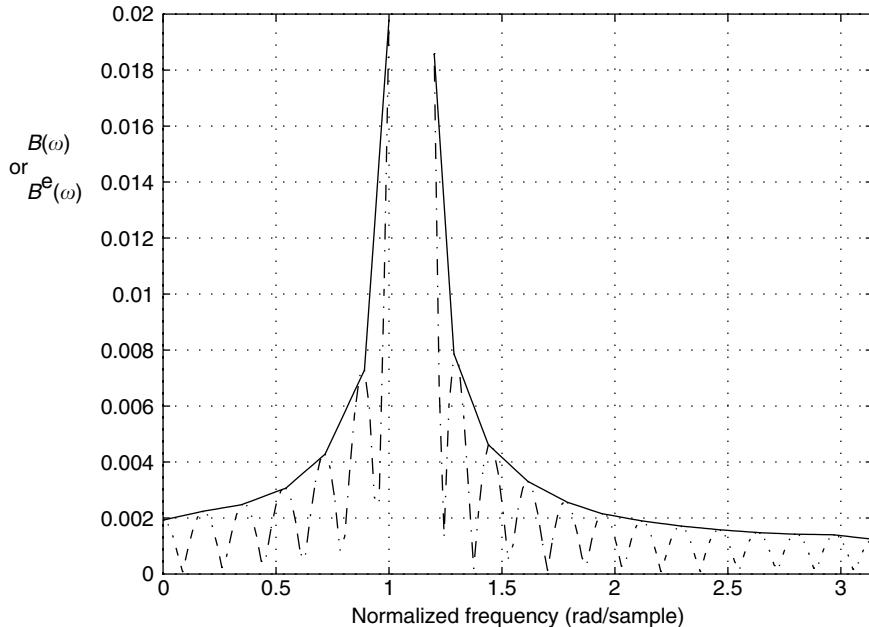


Fig. 5.21. Typical absolute error function $B(\omega)$ (dash-dotted line) and corresponding envelope $B^e(\omega)$ (solid curve).

where, for the modified Lawson algorithm, $\beta_k(\omega)$ is defined as

$$\beta_k(\omega) \equiv \tilde{B}_k(\omega) = \begin{cases} B_k(\omega), & 0 \leq \omega \leq \omega_J^* \\ B_k(\omega_J^*), & \omega_J^* < \omega \leq \pi \end{cases} \quad (5.169)$$

and where, for the modified LLCY algorithm, $\beta_k(\omega)$ is given by

$$\beta_k(\omega) \equiv \tilde{B}_k^e(\omega) = \begin{cases} B_k^e(\omega), & \text{for } 0 \leq \omega \leq \omega_J^* \\ B_k^e(\omega_J^*), & \text{for } \omega_J^* < \omega \leq \pi \end{cases}, \quad (5.170)$$

where ω_J^* is the J th extreme value of the stopband of $B(\omega) = |E(\omega)|$. The passband values of $B(\omega)$ and $B^e(\omega)$ are left unchanged in Equations (5.169) and (5.170) to preserve the equiripple property of the minimax method. The parameter J is the single design parameter for the WLS–Chebyshev scheme. Choosing $J = 1$ makes the new scheme similar to an equiripple-passband WLS design. On the other hand, choosing J as large as possible (that is, making $\omega_J^* = \pi$), turns the design method into the Lawson or the LLCY schemes.

An example of the new approach being applied to the generic functions seen in Figure 5.21 is depicted in Figure 5.22, where ω_J^* was chosen as the fifth extreme in the filter stopband.

The computational complexity of WLS-based algorithms, like the algorithms described here, is of the order of N^3 , where N is the length of the filter. This burden, however, can be greatly reduced by taking advantage of the Toeplitz-plus-Hankel internal structure of the matrix $(\mathbf{U}^T \mathbf{W}^2 \mathbf{U})$, as discussed in Merchant and Parks (1982), and by using an efficient grid scheme to minimize the number of frequency values, as described in Yang

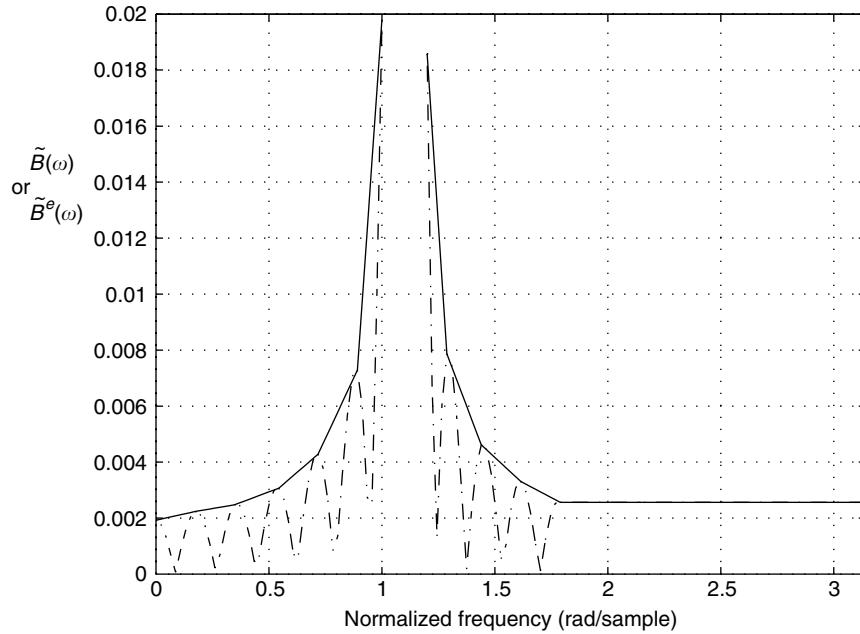


Fig. 5.22. WLS-Chebyshev approach applied to the functions in Figure 5.21. Modified Lawson algorithm $\tilde{B}(\omega)$ (dash-dotted curve) and modified LLCY algorithm $\tilde{B}^e(\omega)$ (solid curve). The curves coincide for $\omega \geq \omega_5^*$.

and Lim (1991, 1993, 1996). These simplifications make the computational complexity of WLS-based algorithms comparable to that for the minimax approach. The WLS-based methods, however, do have the additional advantage of being easily coded into computer routines.

The overall implementation of the WLS–Chebyshev algorithm is as follows:

- (i) Estimate the order M , select $8 \leq K \leq 16$, the maximum number of iterations k_{\max} , the value of J , and some small error tolerance $\epsilon > 0$.
- (ii) Create a frequency grid of KM points within $[0, \pi]$. For linear-phase filters, points in the transition band should be discarded and the remaining points redistributed in the interval $\omega \in [0, \omega_p] \cup [\omega_r, \pi]$.
- (iii) Set $k = 0$ and form \mathbf{W}_q , \mathbf{d}_d , and \mathbf{U} , as defined in Equations (5.134)–(5.136), based on Table 5.12.
- (iv) Set $k = k + 1$, and determine $\mathbf{p}^*(k)$ from Equation (5.144).
- (v) Determine the error vector $\mathbf{e}(k)$ as given in Equation (5.132), always using \mathbf{W}_q corresponding to $k = 0$.
- (vi) Check if $k > k_{\max}$ or if convergence has been achieved via, for instance, the criterion $\| \mathbf{e}(k) \| - \| \mathbf{e}(k-1) \| \leq \epsilon$. If so, then go to step (x).
- (vii) Compute $\{\mathbf{B}_k\} = \{|\{\mathbf{e}(k)\}_j|\}$, for $j = 0, 1, \dots, KM$, and \mathbf{B}_k^e as the envelope of \mathbf{B}_k .
- (viii) Find the J th stopband extreme of \mathbf{B}_k^e . For a lowpass filter, consider the interval $\omega \in [\omega_r, \pi]$, starting at ω_r . For a highpass filter, search in the interval $\omega \in [0, \omega_r]$,

starting at ω_r . For bandpass filters, consider the intervals $\omega \in [0, \omega_{r_1}]$, starting at ω_{r_1} , and $\omega \in [\omega_{r_2}, \pi]$, starting at ω_{r_2} . For the bandstop filter, look for the extreme in the interval $\omega \in [\omega_{r_1}, \omega_{r_2}]$, starting at both ω_{r_1} and ω_{r_2} .

- (ix) Update \mathbf{W}_q^2 using either Equation (5.169) or Equation (5.170) and go back to step (iv).
- (x) Determine the set of coefficients $h(n)$ of the linear-phase filter and verify that the specifications are satisfied. If so, then decrease the filter order M and repeat the above procedure starting from step (ii). The best filter would be the one obtained at the iteration just before the specifications are not met. If the specifications are not satisfied at the first attempt, then increase the value of M and repeat the above procedure once again starting at step (ii). In this case, the best filter would be the one obtained when the specifications are first met.

Example 5.8. Design a bandpass filter satisfying the specification below using the WLS and Chebyshev methods and discuss the results obtained when using the WLS-Chebyshev approach.

$$\left. \begin{array}{l} M = 40 \\ A_p = 1.0 \text{ dB} \\ \Omega_{r_1} = \frac{\pi}{2} - 0.4 \text{ rad/s} \\ \Omega_{r_1} = \frac{\pi}{2} - 0.1 \text{ rad/s} \\ \Omega_{r_2} = \frac{\pi}{2} + 0.1 \text{ rad/s} \\ \Omega_{r_2} = \frac{\pi}{2} + 0.4 \text{ rad/s} \\ \Omega_s = 2\pi \text{ rad/s} \end{array} \right\}. \quad (5.171)$$

Solution

As detailed in Example 5.7, the Chebyshev filter can be designed using the `firpm` command:

```
Omega_r1 = pi/2 - 0.4; Omega_p1 = pi/2 - 0.1;
Omega_p2 = pi/2 + 0.1; Omega_r2 = pi/2 + 0.4;
wr1 = Omega_r1/pi; wp1 = Omega_p1/pi;
wp2 = Omega_p2/pi; wr2 = Omega_r2/pi;
Ap = 1; Ar = 40;
delta_p = (10^(0.05*2*Ap) - 1)/(10^(0.05*2*Ap) + 1);
delta_r = 10^(-0.05*Ar);
F1 = [0 wr1 wp1 wp2 wr2 1];
A1 = [0 0 1 1 0 0];
W1 = [delta_p/delta_r 1 delta_p/delta_r];
h_cheb = firpm(M,F1,A1,W1);
```

The WLS filter is designed using the `firls` command, whose syntax is entirely analogous to the `firpm` command:

```
h_wls = firls(M,F1,A1,W1);
```

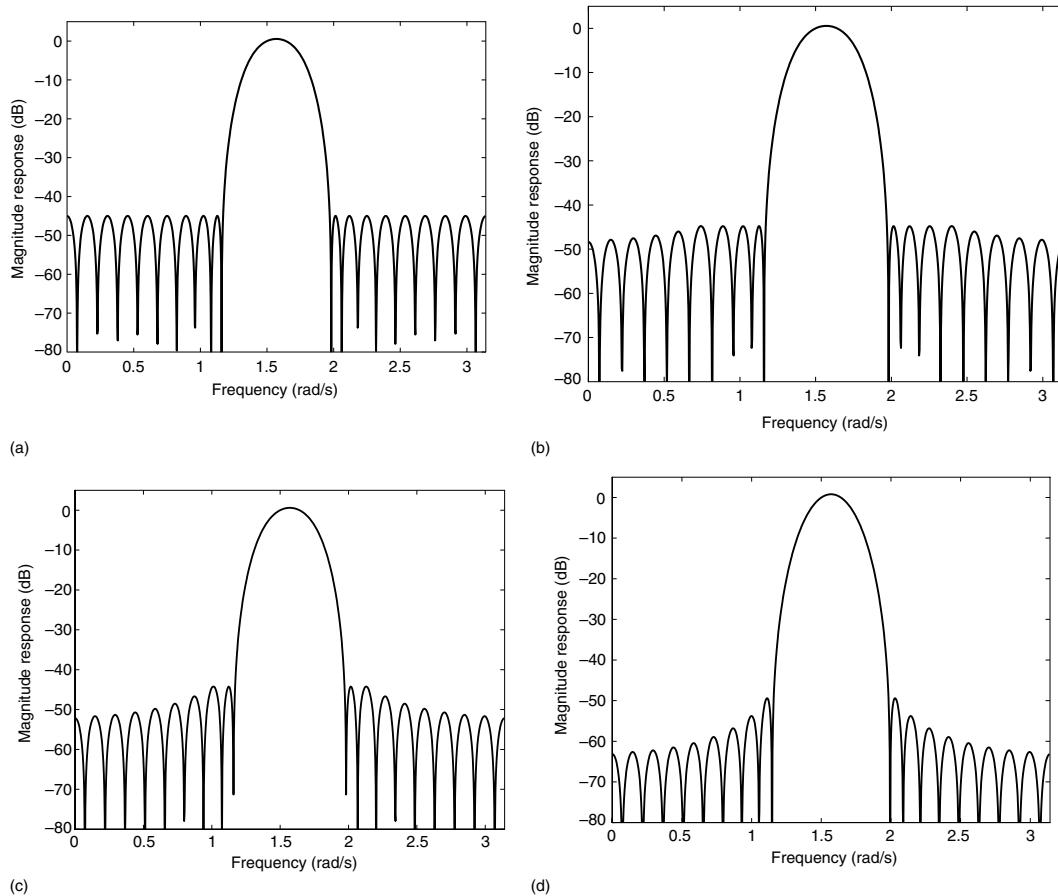


Fig. 5.23. Magnitude responses using WLS–Chebyshev method with: (a) $J = 10$; (b) $J = 5$; (c) $J = 3$; (d) $J = 1$.

The magnitude responses for the Chebyshev and WLS filters designed as above are seen in Figure 5.23a and d, which correspond to the $J = 10$ and $J = 1$ cases respectively. Other values of J , whose magnitude responses are also shown in Figure 5.23, require a specific MATLAB script implementing the WLS–Chebyshev approach, as detailed earlier in this subsection.

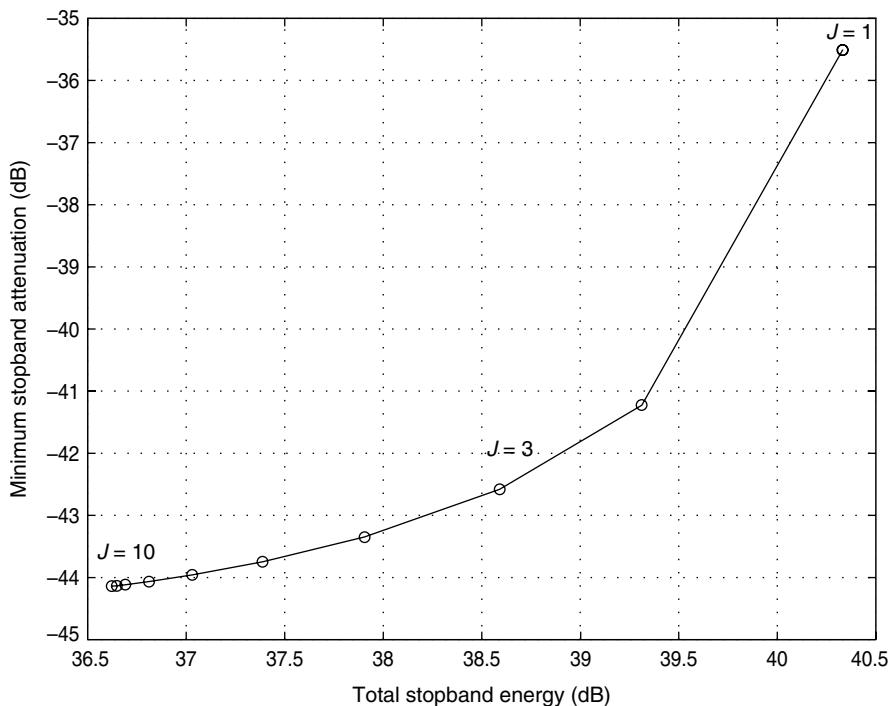
Table 5.16 shows half of the filter coefficients for the case when $J = 3$. The remaining coefficients are obtained as $h(n) = h(40 - n)$. Figure 5.24 shows the trade-off between the stopband minimum attenuation and total stopband energy when J varies from 1 to 10. Notice how the two extremes correspond to optimal values for the attenuation and energy figures of merit respectively. On the other hand, the same extremes are also the worst-case scenarios for the energy and the attenuation in the stopband. In this example, a good compromise between the two measures can be obtained when $J = 3$.



Table 5.16.

Coefficients $h(0)$ to $h(20)$ of the bandpass filter designed with the WLS–Chebyshev method with $J = 3$.

$h(0) = -0.0035$	$h(6) = -0.0052$	$h(12) = 0.0653$	$h(18) = -0.1349$
$h(1) = -0.0000$	$h(7) = -0.0000$	$h(13) = 0.0000$	$h(19) = -0.0000$
$h(2) = 0.0043$	$h(8) = 0.0190$	$h(14) = -0.0929$	$h(20) = 0.1410$
$h(3) = 0.0000$	$h(9) = 0.0000$	$h(15) = -0.0000$	
$h(4) = -0.0020$	$h(10) = -0.0396$	$h(16) = 0.1176$	
$h(5) = 0.0000$	$h(11) = -0.0000$	$h(17) = 0.0000$	

**Fig. 5.24.**

Trade-off between the minimum stopband attenuation and total stopband energy using the WLS–Chebyshev method.

5.7 Do-it-yourself: FIR filter approximations

Experiment 5.1

The output signal of a differentiator device to a complex sinusoid input is given by

$$y(t) = \frac{dx(t)}{dt} = \frac{de^{j\Omega t}}{dt} = j\Omega e^{j\Omega t}. \quad (5.172)$$

Hence, the ideal differentiator has a magnitude gain proportional to the input frequency Ω and a phase shift of $\pm\pi/2$ depending on the sign of Ω . For discrete-time systems, this analysis holds for the digital frequency within $\omega \in [-F_s/2, F_s/2]$, as depicted in Figure 5.2.

Consider a 1 interval of a cosine signal generated in MATLAB as

```
Fs = 1500; Ts = 1/Fs; t = 0:Ts:1-Ts;
fc = 200;
x = cos(2*pi*fc.*t);
```

whose output to a differentiator is

```
y1 = -2*pi*fc*sin(2*pi*fc.*t);
```

as discussed above.

The differentiation operation can be approximated by

$$y_2(t) \approx \frac{x(t + \Delta t) - x(t)}{\Delta t}, \quad (5.173)$$

leading, in the discrete-time domain, to the output

$$y_2(n) \approx \frac{\cos[2\pi f_c(n+1)T_s] - \cos(2\pi f_c n T_s)}{T_s}. \quad (5.174)$$

This approximation can be determined in MATLAB as

```
y2 = [0 diff(x)]/Ts;
```

which, in the z domain, corresponds to the transfer function

$$H_2(z) = (z - 1)/T_s, \quad (5.175)$$

whose magnitude response is shown as the dashed line in Figure 5.25a. In this plot, one can notice that this first-order approximation works very well for small values of f_c , but deviates from the desired response (indicated by the solid line) as f_c approximates $F_s/2$.

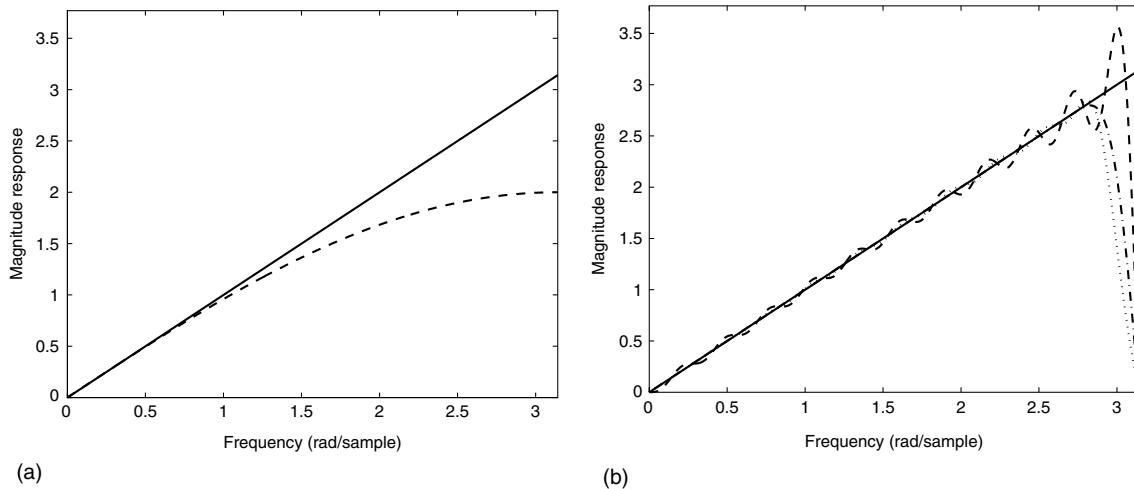
This fact motivates one to design better differentiators.

Using a rectangular window, the impulse response of a differentiator device is obtained directly from Table 5.1, and can be determined in MATLAB, for an odd length N , as

```
N = 45;
h3 = zeros(N, 1);
for n = -(N-1)/2:(N-1)/2,
    if n ~= 0,
        h3((N+1)/2+n) = ((-1)^n)/n;
    end;
end;
```

yielding the frequency response

```
[H3, W] = freqz(h3, 1);
```



(a)

(b)

Fig. 5.25. Magnitude responses of differentiators in Experiment 5.1: (a) ideal (solid line) and first-order approximation (dashed line); (b) ideal (solid line), rectangular window (dashed line), Blackman window (dash-dotted line), and Chebyshev algorithm (dotted line).

Using any other window function, as for instance the Blackman window, one may get

```
h4 = h3.*blackman(N);
H4 = freqz(h4, 1);
```

A differentiator may also be designed with Chebyshev algorithm using the `firpm` command. In this case, one must specify vectors $F = [0 \ f1 \ f2 \ 1]$ and $A = [0 \ pi*f1 \ pi*f2 \ 0]$, characterizing the desired response, which should vary from 0 to πf_1 within the differentiator passband $[0, f_1]$ and from πf_2 to 0 within the normalized interval $[f_2, 1]$. An example of such a design is given by

```
F = [0 0.9 0.91 1];
A = [0 0.9*pi 0.91*pi 0];
h5 = firpm(N-1,F,A,'differentiator');
H5 = freqz(h5, 1);
```

The magnitude responses of all differentiators designed above are shown in Figure 5.25b.

Experiment 5.2

Using the specifications

$$\left. \begin{array}{l} N = 20 \\ \omega_p = 0.1 \\ \omega_r = 0.2 \\ \Omega_s = 2000\pi \text{Hz} \end{array} \right\}, \quad (5.176)$$

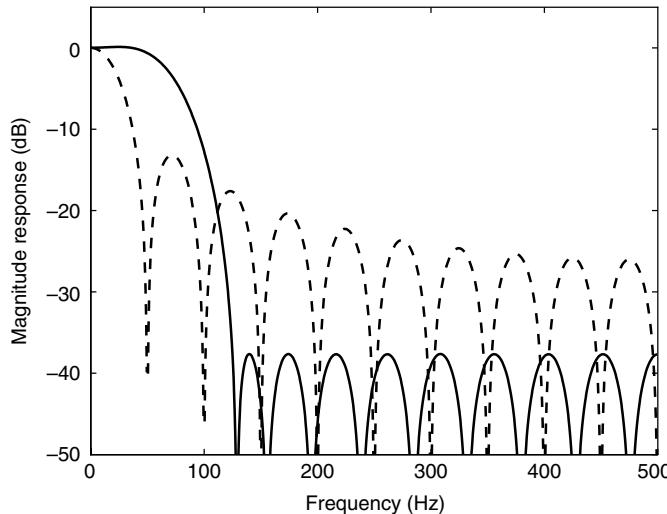


Fig. 5.26. Magnitude responses of lowpass filter with $N = 20$ in Experiment 5.2: `firpm` (solid line) and moving average (dashed line).

a nice lowpass FIR filter can be designed with, for instance, the MATLAB command `firpm` as given by

```
N = 20; Freq = [0 0.1 0.2 1]; Weight = [1 1 0 0];
h = firpm(N,Freq,Weight);
```

The magnitude response of the corresponding filter is depicted in Figure 5.26, along with the one of the moving average filter with $N = 20$ employed in Experiments 1.3 and 2.2. From this figure, one clearly notices how the `firpm` filter can sustain a flatter passband as desired, better preserving the two sinusoidal components in signal x from Experiment 1.3, while strongly attenuating the noise components within the specified stopband, as seen in Figure 5.27.

5.8 FIR filter approximation with MATLAB

MATLAB has the two specific functions described below to perform the discrete-time differentiation and the Hilbert transformation. However, if real-time processing is required, such operations must be implemented as a digital filter, whose approximation should be performed as described in this chapter.

- `diff`: Performs the difference between consecutive entries of a vector. It can be used to approximate the differentiation.
Input parameter: a data vector x .
Output parameter: a vector h with the differences.
Example:

```
x=sin(0:0.01:pi); h=diff(x)/0.01;
```

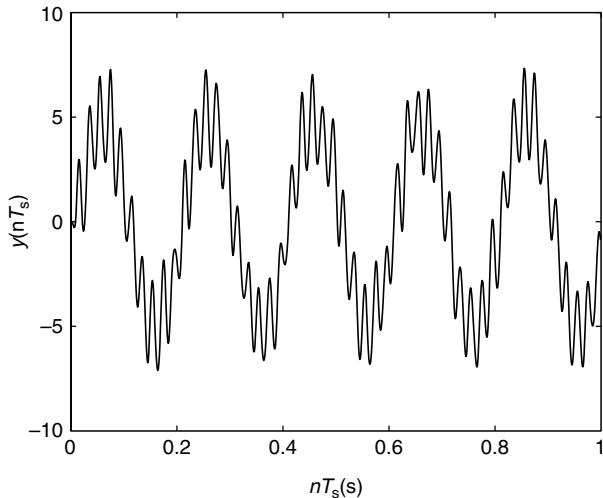


Fig. 5.27. Output signal from `firpm` filter in Experiment 5.2 for noisy sinusoidal components in `x` signal in Experiment 1.3.

- `hilbert`: Performs the Hilbert transform on the real part of a vector. The result is a complex vector, whose real part is the original data and whose imaginary part is the resulting transform.

Input parameter: a data vector `x`.

Output parameter: a vector `H` with the Hilbert transform.

Example:

```
x=rand(30:1); H=hilbert(x);
```

MATLAB also has a series of commands that are useful for solving the approximation problem of FIR digital filters. More specifically, the window-based, the WLS, and the Chebyshev methods are easily implemented in MATLAB with the help of the following commands.

MATLAB commands related to the window method

- `fir1`: Designs standard (lowpass, highpass, bandpass, and bandstop) FIR filters using the window method.

Input parameters:

- The filter order `M`. For highpass and bandstop filters this value must be even. In such cases, if an odd value is provided, MATLAB increments it by 1.
- A vector `f` of band edges. If `f` has one element, the filter is lowpass or highpass, and the given value becomes the passband edge. If `f` has two elements, the filter is either bandpass (where the two values become the passband edges) or bandstop (where the two values become the stopband edges). A larger `f` corresponds to multiband filters.
- A string specifying the standard filter type. The default value is associated with lowpass and bandpass filters, according to the size of `f`. ‘`high`’ indicates that the filter is

highpass; 'stop' indicates that the filter is bandstop; 'DC-1' indicates that the first band of a multiband filter is a passband; 'DC-0' indicates that the first band of a multiband filter is a stopband.

- The default window type used by `fir1` is the Hamming window. The user can change it, if desired, by using one of the window commands seen below.

Output parameter: a vector `h` containing the filter coefficients.

Example:

```
M=40; f=0.2;
h=fir1(M,f,'high',chebwin(M+1,30));
```

- `fir2`: Designs arbitrary-response FIR filters using the window method.

Input parameters:

- the filter order `M`;
- a frequency vector `f` specifying the filter bands;
- a magnitude response vector `m`, the same size as `f`;
- a string specifying the window type, as in the case of the `fir1` command.

Output parameter: a vector `h` containing the filter coefficients.

- `boxcar`: Determines the rectangular window function.

Input parameter: the window length `N=M+1`.

Output parameter: a vector `wr` containing the window.

Example:

```
N=11; wr=boxcar(N);
```

- `triang`: Determines the triangular window function.

Input parameter: the window length `N=M+1`. If this value is even, the direct relationship between the triangular and Bartlett windows disappears.

Output parameter: a vector `wt` containing the window.

Example:

```
N=20; wt=triang(N);
```

- `bartlett`: Determines the Bartlett window function.

Input parameter: the window length `N=M+1`.

Output parameter: a vector `wtB` containing the window.

Example:

```
N=10; wtB=bartlett(N);
```

- `hamming`: Determines the Hamming window function.

Input parameter: the window length `N=M+1`.

Output parameter: a vector `wH` containing the window.

Example:

```
N=31; wH=hamming(N);
```

- **hanning:** Determines the Hann window function.

Input parameter: the window length $N=M+1$.

Output parameter: a vector wHn containing the window.

Example:

```
N=18; wHn=hanning(N);
```

- **blackman:** Determines the Blackman window function.

Input parameter: the window length $N=M+1$.

Output parameter: a vector wB containing the window.

Example:

```
N=49; wB=blackman(N);
```

- **kaiser:** Determines the Kaiser window function.

Input parameters: the window length $N=M+1$ and the auxiliary parameter β , as determined in Equation (5.73).

Output parameter: a vector wK containing the window.

Example:

```
N=23; beta=4.1;
wK=kaiser(N,beta);
```

- **kaiserord:** Estimates the order of the filter designed with the Kaiser window (see Exercise 5.25). This command is suited for use with the `fir1` command with the Kaiser window.

Input parameters:

- a vector f of band edges;
- a vector a of desired amplitudes in the bands defined by f ;
- a vector, the same size as f and a , that specifies the maximum error in each band between the desired amplitude and the resulting amplitude of the designed filter;
- the sampling frequency F_s .

Output parameter: the order of the Kaiser window.

Example:

```
f=[100 200]; a=[1 0]; err=[0.1 0.01]; Fs=800;
M=kaiserord(f,a,err,Fs);
```

- **chebwin:** Determines the Dolph–Chebyshev window function.

Input parameters: the window length $N=M+1$ and the ripple ratio r in decibels, $20 \log(\delta_p/\delta_r)$.

Output parameter: a vector wDC containing the window.

Example:

```
N=51; r=20;
wDC=chebwin(N,r);
```

MATLAB commands related to the WLS

- **firls:** Designs linear-phase FIR filters using the WLS method.

Input parameters:

- the filter order M ;
- a vector f of pairs of normalized frequency points between 0 and 1;
- a vector a containing the desired amplitudes of the points specified in f ;
- a vector w , half of the size of f and a , of weights for each band specified in f ;
- a string specifying the filter type other than standard ones. The options 'differentiator' and 'hilbert' can be used to design a differentiator or a Hilbert transformer, respectively.

Output parameter: a vector h containing the filter coefficients.

Example:

```
M=40; f=[0 0.4 0.6 0.9]; a=[0 1 0.5 0.5]; w=[1 2];
h=firls(M,f,a,w);
```

- **fircls:** Designs multiband FIR filters with the constrained least-squares method.

Input parameters:

- The filter order M .
- A vector f of normalized band edges starting with 0 and ending with 1, necessarily.
- A vector a describing the piecewise constant desired amplitude response. The length of a is the number of prescribed bands; that is, $\text{length}(f) - 1$.
- Two vectors, up and lo , with the same length as a , defining the upper and lower bounds for the magnitude response in each band.

Output parameter: a vector h containing the filter coefficients.

Example:

```
M=40; f=[0 0.4 0.6 1]; a=[1 0 1];
up=[1.02 0.02 1.02]; lo=[0.98 -0.02 0.98];
h=fircls(M,f,a,up,lo);
```

- **fircls1:** Designs lowpass and highpass linear-phase FIR filters with the constrained least-squares method.

Input parameters:

- the filter order M ;
- the passband normalized frequency wp ;
- the passband ripple dp ;
- the stopband ripple dr ;
- the stopband normalized frequency wr ;
- a string 'high' to indicate that the filter is highpass, if this is the case.

Output parameter: a vector h containing the filter coefficients.

Example:

```
M=30; wp=0.4; dp=0.1; dr=0.01; wr=0.5;
h=fircls1(M,wp,dp,dr,wr);
```

MATLAB commands related to the Chebyshev optimal method

- **firpm:** Designs a linear-phase FIR filter using the Parks–McClellan algorithm (McClellan *et al.*, 1973). Old versions of MATLAB have the command `remez` instead.

Input parameters: as for the command `firls`.

Output parameter: a vector `h` containing the filter coefficients.

Example:

```
M=40; f=[0 0.4 0.6 0.9]; a=[0 1 0.5 0.5]; w=[1 2];
h=firpm(M,f,a,w);
```

- **cfirpm:** Generalizes the `firpm` command for complex and nonlinear-phase FIR filters. Old versions of MATLAB have the command `cremez` instead.

Input parameters: there are several possibilities for using the `cfirpm` command. One of them is similar to the one for the `firpm` and `firls` commands (see example below), where `f`, for the `cfirpm` command, should be specified in the range -1 to 1 .

Output parameter: a vector `h` containing the filter coefficients.

Example:

```
M=30; f=[-1 -0.4 0.6 0.9]; a=[0 1 0.5 0.5]; w=[2 1];
h=cfirpm(M,f,a,w);
```

- **firpmord:** Estimates the order of the filter designed with the Chebyshev method (see Exercise 5.26). Old versions of MATLAB have the command `remezord` instead. This command is perfectly suited to the `firpm` command.

Input parameters:

- A vector `f` of band edges.
- A vector `a` of desired amplitudes in the bands defined by `f`. The length of `f` in this case is twice the length of `a`, minus 2.
- A vector, the same size as `a`, that specifies the maximum error in each band between the desired amplitude and the resulting amplitude of the designed filter.
- The sampling frequency `Fs`.

Output parameter: the order `M` of the filter.

Example:

```
f=[400 500]; a=[1 0]; err=[0.1 0.01]; Fs=2000;
M=firpmord(f,a,err,Fs);
```

In versions 5.0 or higher, MATLAB provides a filter design tool along with the Signal Processing toolbox. Such a tool is invoked with the command `sptool`, presenting a very friendly graphic user interface for filter design. In that manner, defining the filter specifications turns into a very easy task and analyzing the resulting filter characteristics becomes straightforward.

5.9 Summary

The subject of FIR filter design is very extensive and could be, in its own right, the subject of a complete textbook. Among the methods studied in the present chapter for designing FIR filters, we focused on the frequency sampling, window-based, maximally flat, WLS, Chebyshev, and WLS–Chebyshev approaches.

The frequency sampling method consists of performing an IDFT over a set of samples of a desired frequency response. Its implementation is very simple, but the results tend to be poor, especially when very sharp transition bands are involved. Despite this drawback, the frequency sampling method is very useful when the desired frequency response is composed of a sum of complex sinusoids.

The window method consists of truncating the impulse response of a given desired magnitude response through the use of a so-called window function. Simple window functions, the rectangular, triangular, Blackman, and so on, do not allow designs satisfying prescribed specifications, in general. More sophisticated window functions, such as the Dolph–Chebyshev and Kaiser windows, are able to control the passband and stopband ripples simultaneously, thus enabling FIR filters to be designed that satisfy prescribed specifications. The subject of window functions is very rich. Other window functions, distinct from the ones seen here, can be found for instance in Nuttall (1981), Webster (1985), Ha and Pearce (1989), Adams (1991b), Yang and Ke (1992), Saramäki (1993) and Kay and Smith (1999).

The maximally flat filter design generates lowpass and highpass FIR filters with very flat passband and stopband. The method is extremely simple, although it is only suitable for low-order filters, because the filter coefficients tend to present a very large dynamic range as the order increases.

In addition, a unified framework for studying numerical methods for FIR filter design was given. In this context, we presented the WLS, the Chebyshev (or minimax), and the WLS–Chebyshev schemes. The first method minimizes the stopband total energy for a given passband energy level. The second method, which is commonly implemented with the Remez exchange algorithm, is able to minimize the maximum error between the designed and the desired responses. There are two interesting cases where the optimal solution for this problem presents a closed-form solution: when the passband is equiripple and the stopband is monotonically decreasing, or when the passband is monotonically decreasing and the stopband is equiripple. The interested reader may refer to Saramäki (1993) to investigate this subject further. The WLS–Chebyshev was introduced as a numerical method able to unite the good performances of the WLS and Chebyshev methods with respect to the total energy level and minimum attenuation level in the stopband, simultaneously.

Finally, the FIR filter design problem was analyzed using the MATLAB software tool. MATLAB was shown to present a series of commands for designing FIR filters. Although the WLS–Chebyshev method is not part of any MATLAB toolbox, we have supplied a description of it in pseudo-code (Section 5.6.3), so that it can be easily implemented using any version of MATLAB. It must be emphasized that a very powerful built-in graphic user interface exists in MATLAB that allows one to perform several filter designs with only a few clicks of the mouse.

5.10 Exercises

- 5.1 Write a table equivalent to Table 5.1 supposing that the ideal impulse responses have an added phase term of $-(M/2)\omega$, for M odd.
- 5.2 Assume that a periodic signal has four sinusoidal components at frequencies $\omega_0, 2\omega_0, 4\omega_0, 6\omega_0$. Design a nonrecursive filter, as simple as possible, that eliminates only the components $2\omega_0, 4\omega_0, 6\omega_0$.
- 5.3 Given a lowpass FIR filter with transfer function $H(z)$, describe what happens to the filter frequency response when:
- z is replaced by $-z$.
 - z is replaced by z^{-1} .
 - z is replaced by z^2 .
- 5.4 Complementary filters are such that their frequency responses add to a delay. Given an M th-order linear-phase FIR filter with transfer function $H(z)$, deduce the conditions on L and M such that the overall filter shown in Figure 5.28 is complementary to $H(z)$.

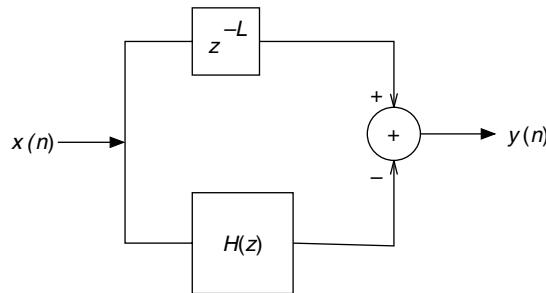


Fig. 5.28. Overall filter block diagram.

- 5.5 Determine the relationship between L , M , and N which means that the overall filter in Figure 5.29 has linear phase.

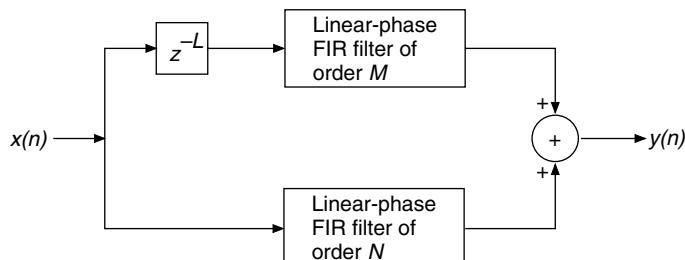


Fig. 5.29. Overall filter block diagram.

- 5.6 Design a highpass filter satisfying the specification below using the frequency sampling method:

$$\begin{aligned}M &= 40 \\ \Omega_r &= 1.0 \text{ rad/s} \\ \Omega_p &= 1.5 \text{ rad/s} \\ \Omega_s &= 5.0 \text{ rad/s.}\end{aligned}$$

- 5.7 Plot and compare the characteristics of the Hamming window and the corresponding magnitude response for $M = 5, 10, 15, 20$.
- 5.8 Plot and compare the rectangular, triangular, Bartlett, Hamming, Hann, and Blackman window functions and the corresponding magnitude responses for $M = 20$.
- 5.9 Determine the ideal impulse response associated with the magnitude response shown in Figure 5.30, and compute the corresponding practical filter of orders $M = 10, 20, 30$ using the Hamming window.

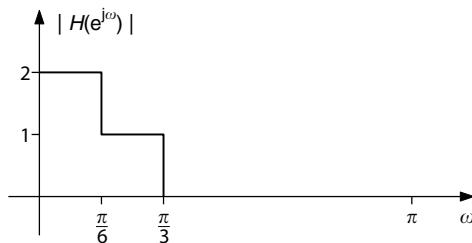


Fig. 5.30. Ideal magnitude response of Exercise 5.9.

- 5.10 For the magnitude response shown in Figure 5.31, where $\omega_s = 2\pi$ denotes the sampling frequency:
- Determine the ideal impulse response associated with it.
 - Design a fourth-order FIR filter using the triangular window with $\omega_c = \frac{\pi}{4}$.

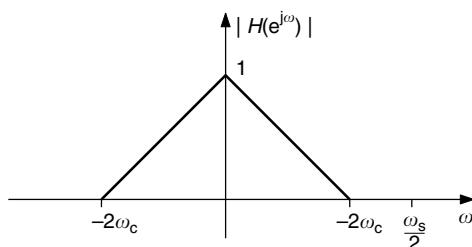


Fig. 5.31. Ideal magnitude response of Exercise 5.10.

- 5.11 For the magnitude response shown in Figure 5.32:
- Determine the ideal impulse response associated with it.
 - Design a fourth-order FIR filter using the Hann window.

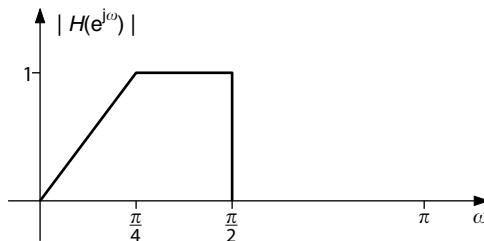


Fig. 5.32. Ideal magnitude response of Exercise 5.11.

5.12 Design a bandpass filter satisfying the specification below using the Hamming, Hann, and Blackman windows:

$$\begin{aligned}M &= 10 \\ \Omega_{c_1} &= 1.125 \text{ rad/s} \\ \Omega_{c_2} &= 2.5 \text{ rad/s} \\ \Omega_s &= 10 \text{ rad/s.}\end{aligned}$$

5.13 Plot and compare the characteristics of the Kaiser window function and corresponding magnitude response for $M = 20$ and different values of β .

5.14 Design the following filters using the Kaiser window:

(a) $A_p = 1.0 \text{ dB}$
 $A_r = 40 \text{ dB}$
 $\Omega_p = 1000 \text{ rad/s}$
 $\Omega_r = 1200 \text{ rad/s}$
 $\Omega_s = 5000 \text{ rad/s.}$

(b) $A_p = 1.0 \text{ dB}$
 $A_r = 40 \text{ dB}$
 $\Omega_r = 1000 \text{ rad/s}$
 $\Omega_p = 1200 \text{ rad/s}$
 $\Omega_s = 5000 \text{ rad/s.}$

(c) $A_p = 1.0 \text{ dB}$
 $A_r = 50 \text{ dB}$
 $\Omega_{r_1} = 800 \text{ rad/s}$
 $\Omega_{p_1} = 1000 \text{ rad/s}$
 $\Omega_{p_2} = 1100 \text{ rad/s}$
 $\Omega_{r_2} = 1400 \text{ rad/s}$
 $\Omega_s = 10000 \text{ rad/s.}$

5.15 Determine a complete procedure for designing differentiators using the Kaiser window.

5.16 Repeat Exercise 5.14(a) using the Dolph–Chebyshev window. Compare the transition bandwidths and the stopband attenuation levels for the two resulting filters.

- 5.17 Design a maximally flat lowpass filter satisfying the specification below:

$$\begin{aligned}\omega_c &= 0.4\pi \text{ rad/sample} \\ T_r &= 0.2\pi \text{ rad/sample},\end{aligned}$$

where T_r is the transition band as defined in Section 5.5. Find also the direct-form coefficients $h(n)$, for $n = 0, 1, \dots, M$, and the alternative coefficients $\hat{d}(n)$, for $n = 0, 1, \dots, (K - 1)$.

- 5.18 (a) Show that Type II linear-phase FIR filters can be put in the form of Equation (5.116), by substituting Equation (5.115) into Equation (5.114).
 (b) Repeat item (a) for Type III filters, showing that Equation (5.120) can be obtained from substituting Equation (5.119) into Equation (5.118).
 (c) Repeat item (a) for Type IV filters, showing that Equation (5.124) can be obtained from substituting Equation (5.123) into Equation (5.122).
- 5.19 Design three narrowband filters, centered on the frequencies 770 Hz, 852 Hz, and 941 Hz, satisfying the specification below, using the minimax approach:

$$M = 98$$

$$\Omega_s = 2\pi \times 5 \text{ kHz.}$$

- 5.20 Using the minimax approach, design a bandpass filter for tone detection with a center frequency of 700 Hz, given that the sampling frequency is 8000 Hz and the order is 95. Use the following band parameters:
- band 1
 - edges: 0 and 555.2 Hz
 - objective: 0
 - weight: 1;
 - band 2
 - edges: 699.5 Hz and 700.5 Hz
 - objective: 1
 - weight: 1;
 - band 3
 - edges: 844.8 Hz and 4000 Hz
 - objective: 0
 - weight: 1.
- 5.21 Design Hilbert transformers of orders $M = 38, 68$, and 98 using a Type IV structure and the Hamming window method.
- 5.22 Design a Hilbert transformer of order $M = 98$ using a Type IV structure and the triangular, Hann, and Blackman window methods.
- 5.23 Design a Hilbert transformer of order $M = 98$ using a Type IV structure and the Chebyshev method, and compare your results with those from Exercise 5.22.

- 5.24 Determine the output of the Hilbert transformer designed in Exercise 5.23 to the input signal x determined as

```
Fs = 1500; Ts = 1/Fs; t = 0:Ts:1-Ts;
fc1 = 200; fc2 = 300;
x = cos(2*pi*fc1.*t) + sin(2*pi*fc2.*t);
```

- 5.25 The following relationship estimates the order of a lowpass filter designed with the minimax approach (Rabiner *et al.*, 1975). Design a series of lowpass filters and verify the validity of this estimate (Ω_s is the sampling frequency):

$$M \approx \frac{D_\infty(\delta_p, \delta_r) - f(\delta_p, \delta_r)(\Delta F)^2}{\Delta F} + 1,$$

where

$$\begin{aligned} D_\infty(\delta_p, \delta_r) &= \{0.005\ 309[\log_{10}(\delta_p)]^2 + 0.071\ 140[\log_{10}(\delta_p)] - 0.4761\} \log_{10}(\delta_r) \\ &\quad - \{0.002\ 660[\log_{10}(\delta_p)]^2 + 0.594\ 100[\log_{10}(\delta_p)] + 0.4278\} \\ \Delta F &= \frac{\Omega_r - \Omega_p}{\Omega_s} \\ f(\delta_p, \delta_r) &= 11.012 + 0.512\ 44[\log_{10}(\delta_p) - \log_{10}(\delta_r)]. \end{aligned}$$

- 5.26 Repeat Exercise 5.25 with the following order estimate (Kaiser, 1974):

$$M \approx \frac{-20 \log_{10} (\sqrt{\delta_p \delta_r}) - 13}{2.3237 (\omega_r - \omega_p)} + 1,$$

where ω_p and ω_r are the digital passband and stopband edges. Which estimate tends to be more accurate?

- 5.27 Perform the algebraic design of a highpass FIR filter such that

$$\begin{aligned} \omega_p &= \frac{\omega_s}{8} \\ \delta_p &= 8\delta_r, \end{aligned}$$

using the WLS algorithm with a frequency grid of only two points.

- 5.28 Design a bandpass filter satisfying the specification below using the WLS and Chebyshев methods. Discuss the trade-off between the stopband minimum attenuation and total stopband energy when using the WLS–Chebyshev scheme.

$$\begin{aligned} M &= 50 \\ \Omega_{r_1} &= 100 \text{ rad/s} \\ \Omega_{p_1} &= 150 \text{ rad/s} \\ \Omega_{p_2} &= 200 \text{ rad/s} \\ \Omega_{r_2} &= 300 \text{ rad/s} \\ \Omega_s &= 1000 \text{ rad/s.} \end{aligned}$$

- 5.29 Design an order $M = 8$ notch filter with a zero at frequency $\omega_0 = \pi/5$ using the WLS approach.

Hint: Note that a zero in frequency ω_0 demands a pair of zeros at $e^{\pm j\omega_0}$, which implies that $P(\omega)$ in Equation (5.126) must have a factor of the form $\cos(\omega) - \cos(\omega_0)$. Embed this factor into $Q(\omega)$ and proceed to design a filter with flat amplitude response, avoiding the frequency ω_0 when defining the dense frequency grid that leads to Equations (5.132) to (5.137).

- 5.30 Use the MATLAB command `filter` to differentiate signal x , as defined in Experiment 5.1, with the systems designed for that purpose in the same experiment. Compare your results with the theoretical output $y_1(t)$. Do not forget to normalize your output signal by T_s , as seen in Equation (5.174), and to compensate for the group delay of $M/2$ samples introduced by the FIR structure.

- 5.31 Repeat Experiment 5.1 with an input signal defined as

```
Fs = 1500; Ts = 1/Fs; t = 0:Ts:1-Ts;
fc1 = 200; fc2 = 700;
x = cos(2*pi*fc1.*t) + cos(2*pi*fc2.*t);
```

and compare the results obtained with each differentiator system, by verifying what happens with each sinusoidal component in x .

- 5.32 Change the values of F_s , total time length, and f_c in Experiment 5.1, one parameter at a time, and verify their individual influences on the output signal yielded by a differentiator system. Validate your analyses by differentiating the signal x in Experiment 5.1 using the designed systems.
- 5.33 Change the filter specifications in Experiment 5.2, designing the filter accordingly, using the `firpm` command. Analyze the resulting magnitude response and the output signal to the input x as defined in Experiment 1.3.

6.1 Introduction

This chapter deals with the design methods in which a desired frequency response is approximated by a transfer function consisting of a ratio of polynomials. In general, this type of transfer function yields an impulse response of infinite duration. Therefore, the systems approximated in this chapter are commonly referred to as IIR filters.

In general, IIR filters are able to approximate a prescribed frequency response with fewer multiplications than FIR filters. For that matter, IIR filters can be more suitable for some practical applications, especially those involving real-time signal processing.

In Section 6.2 we study the classical methods of analog filter approximation, namely the Butterworth, Chebyshev, and elliptic approximations. These methods are the most widely used for approximations meeting prescribed magnitude specifications. They originated in the continuous-time domain and their use in the discrete-time domain requires an appropriate transformation.

We then address, in Section 6.3, two approaches that transform a continuous-time transfer function into a discrete-time transfer function, namely the impulse-invariance and bilinear transformation methods.

Section 6.4 deals with frequency transformation methods in the discrete-time domain. These methods allow the mapping of a given filter type to another; for example, the transformation of a given lowpass filter into a desired bandpass filter.

In applications where magnitude and phase specifications are imposed, we can approximate the desired magnitude specifications by one of the classical transfer functions and design a phase equalizer to meet the phase specifications. As an alternative, we can carry out the design entirely in the digital domain, by using optimization methods to design transfer functions satisfying the magnitude and phase specifications simultaneously. Section 6.5 covers a procedure to approximate a given frequency response iteratively, employing a nonlinear optimization algorithm.

In Section 6.6 we address the situations where an IIR digital filter must present an impulse response similar to a given discrete-time sequence. This problem is commonly known as time-domain approximation.

Finally, we present some hands-on experiments with IIR filters in the Do-it-yourself section, and in Section 6.8 the role of MATLAB in the approximation of IIR filters is briefly discussed.

6.2 Analog filter approximations

This section covers the classical approximations for normalized-lowpass analog filters.¹ The other types of filters, such as the denormalized-lowpass, highpass, bandstop, and bandpass filters, are obtained from the normalized-lowpass prototype through frequency transformations, which are also addressed in this section.

6.2.1 Analog filter specification

An important step in the design of an analog filter is the definition of the desired magnitude and/or phase specifications that should be satisfied by the filter frequency response. Usually, a classical analog filter is specified through a region of the $\Omega \times H(j\Omega)$ plane² where its frequency response must be contained. This is illustrated in Figure 6.1 for a lowpass filter. In this figure, Ω_p and Ω_r denote the passband and stopband edge frequencies respectively. The frequency region between Ω_p and Ω_r is the so-called transition band, where no specification is provided. In addition, the maximum ripples in the passband and the stopband are denoted by δ_p and δ_r respectively.

Alternatively, the specifications can be given in decibels, as shown in Figure 6.2a, in the case of gain specifications. Figure 6.2b shows the same filter specified in terms of attenuation instead of gain. The relationships between the parameters of these three representations are given in Table 6.1.

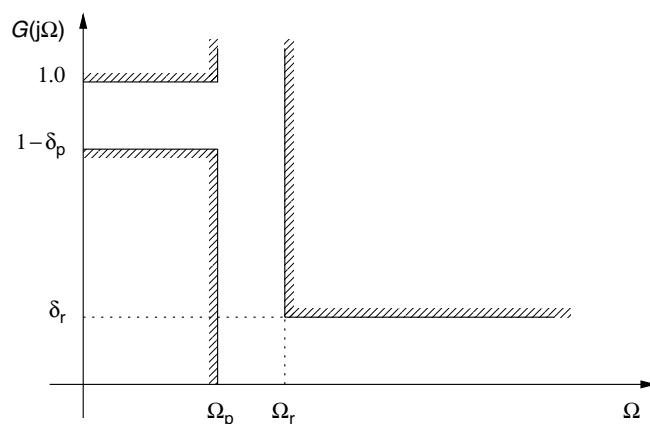


Fig. 6.1. Typical gain specifications of a lowpass filter.

¹ Normalized filters are derived from standard ones through a simple variable scaling. The original filter is then determined by reversing the frequency transformation previously applied. In this section, to avoid any source of confusion, a normalized analog frequency is always denoted by a primed variable such as Ω' .

² Note that once more Ω usually refers to analog frequency and ω to digital frequency.

Table 6.1.

Relationships among the parameters for the ripple, gain in decibels, and attenuation in decibels specification formats.

	Ripple	Gain (dB)	Attenuation (dB)
Passband	δ_p	$G_p = 20 \log_{10}(1 - \delta_p)$	$A_p = -G_p$
Stopband	δ_r	$G_r = 20 \log_{10} \delta_r$	$A_r = -G_r$

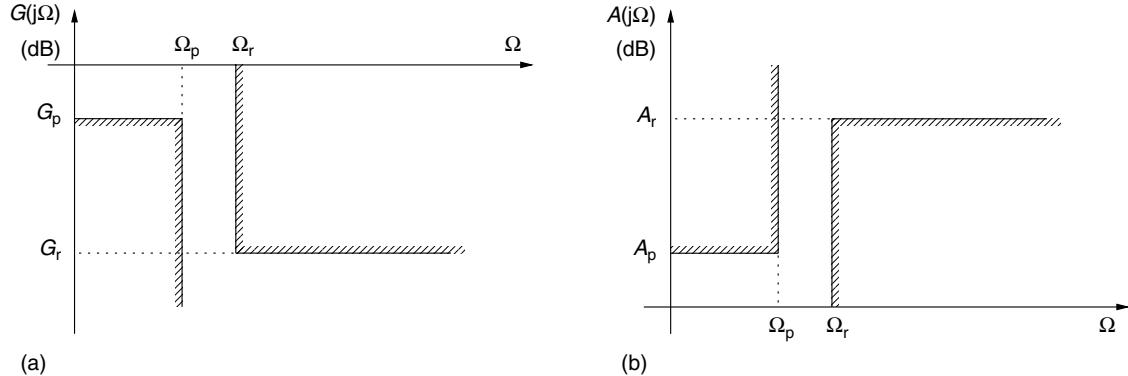


Fig. 6.2.

Typical specifications of a lowpass filter in decibels: (a) gain; (b) attenuation.

For historical reasons, in this chapter we work with the attenuation specifications in decibels. Using the relationships given in Table 6.1, readers should be able to transform any other format into the set of parameters that characterize the attenuation in decibels.

6.2.2 Butterworth approximation

Usually, the attenuation of an all-pole normalized-lowpass filter (that is, with $\Omega'_p = 1$), is expressed by an equation of the following type:

$$|A(j\Omega')|^2 = 1 + |E(j\Omega')|^2, \quad (6.1)$$

where $A(s')$ is the desired attenuation function and $E(s')$ is a polynomial which has low magnitude at low frequencies and large magnitude at high frequencies.

The Butterworth approximation is characterized by a maximally flat magnitude response at $\Omega' = 0$. In order to achieve this property, we choose $E(j\Omega')$ as

$$E(j\Omega') = \epsilon (j\Omega')^n, \quad (6.2)$$

where ϵ is a constant and n is the filter order. Equation (6.1) then becomes

$$|A(j\Omega')|^2 = 1 + \epsilon^2 (\Omega')^{2n}, \quad (6.3)$$

resulting in the fact that the first $(2n - 1)$ derivatives of the attenuation function at $\Omega' = 0$ are equal to zero, as desired in the Butterworth approximation.

The choice of the parameter ϵ depends on the maximum attenuation A_p allowed in the passband. In that manner, since

$$A_{\text{dB}}(\Omega') = 20 \log_{10} |A(j\Omega')| = 10 \log_{10} \left[1 + \epsilon^2 (\Omega')^{2n} \right] \quad (6.4)$$

at $\Omega' = \Omega'_p = 1$, we must have that

$$A_p = A_{\text{dB}}(1) = 10 \log_{10} \left(1 + \epsilon^2 \right) \quad (6.5)$$

and then

$$\epsilon = \sqrt{10^{0.1A_p} - 1}. \quad (6.6)$$

To determine the filter order required to meet the attenuation specification, A_r , in the stopband, at $\Omega' = \Omega'_r$ we must have that

$$A_r = A_{\text{dB}}(\Omega'_r) = 10 \log_{10} \left[1 + \epsilon^2 (\Omega'_r)^{2n} \right]. \quad (6.7)$$

Therefore, n should be the smallest integer such that

$$n \geq \frac{\log_{10} [(10^{0.1A_r} - 1)/\epsilon^2]}{2 \log_{10} \Omega'_r} \quad (6.8)$$

with ϵ as in Equation (6.6).

With n and ϵ available, one has to find the transfer function $A(s')$. We can factor $|A(j\Omega')|^2$ in Equation (6.3) as

$$|A(j\Omega')|^2 = A(-j\Omega')A(j\Omega') = 1 + \epsilon^2 \Omega'^{2n} = 1 + \epsilon^2 [-(j\Omega')^2]^n. \quad (6.9)$$

Using the analytical continuation for complex variables (Churchill, 1975) – that is, replacing $j\Omega'$ by s' – we have that

$$A(s')A(-s') = 1 + \epsilon^2 (-s'^2)^n. \quad (6.10)$$

In order to determine $A(s')$, we must find the roots of $[1 + \epsilon^2 (-s'^2)^n]$ and then choose which ones belong to $A(s')$ and which ones belong to $A(-s')$. The solutions of

$$1 + \epsilon^2 (-s'^2)^n = 0 \quad (6.11)$$

are

$$s_i = \epsilon^{-1/n} e^{j(\pi/2)[(2i+n+1)/n]} \quad (6.12)$$

with $i = 1, 2, \dots, 2n$. These $2n$ roots are located at equally spaced positions on the circumference of radius $\epsilon^{-1/n}$ centered at the origin of the s plane. In order to obtain a stable filter,

we choose the n roots p_i on the left-hand side of the s plane to belong to the polynomial $A(s')$. As a result, the normalized transfer function is obtained as

$$H'(s') = \frac{H'_0}{A(s')} = \frac{H'_0}{\prod_{i=1}^n (s' - p_i)}, \quad (6.13)$$

where H'_0 is chosen so that $|H'(\text{j}0)| = 1$, and thus

$$H'_0 = \prod_{i=1}^n (-p_i). \quad (6.14)$$

An important characteristic of the Butterworth approximation is that its attenuation increases monotonically with frequency. In addition, it increases very slowly in the passband and quickly in the stopband. In the Butterworth approximation, if one wants to increase the attenuation, then one has to increase the filter order. However, if one sacrifices the monotonicity of the attenuation, then a higher attenuation in the stopband can be obtained for the same filter order. A classic example of one such approximation is the Chebyshev approximation.

6.2.3 Chebyshev approximation

The attenuation function of a normalized-lowpass Chebyshev filter is characterized by

$$|A(\text{j}\Omega')|^2 = 1 + \epsilon^2 C_n^2(\Omega'), \quad (6.15)$$

where $C_n(\Omega')$ is a Chebyshev function of order n , which can be written in its trigonometric form as

$$C_n(\Omega') = \begin{cases} \cos(n \cos^{-1} \Omega'), & 0 \leq \Omega' \leq 1 \\ \cosh(n \cosh^{-1} \Omega'), & \Omega' > 1 \end{cases}. \quad (6.16)$$

These functions $C_n(\Omega')$ have the following properties:

$$\begin{cases} 0 \leq C_n^2(\Omega') \leq 1, & 0 \leq \Omega' \leq 1 \\ C_n^2(\Omega') > 1, & \Omega' > 1 \end{cases}. \quad (6.17)$$

As a consequence, for the attenuation function defined in Equation (6.15), the passband is placed in the frequency range $0 \leq \Omega' \leq \Omega'_p = 1$, the rejection band is in the range $\Omega' \geq \Omega'_r > 1$, as desired, and the parameter ϵ once again determines the maximum passband ripple.

The Chebyshev functions defined above can also be expressed in polynomial form as

$$\begin{aligned} C_{n+1}(\Omega') + C_{n-1}(\Omega') &= \cos[(n+1) \cos^{-1} \Omega'] + \cos[(n-1) \cos^{-1} \Omega'] \\ &= 2 \cos(\cos^{-1} \Omega') \cos(n \cos^{-1} \Omega') \\ &= 2\Omega' C_n(\Omega'), \end{aligned} \quad (6.18)$$

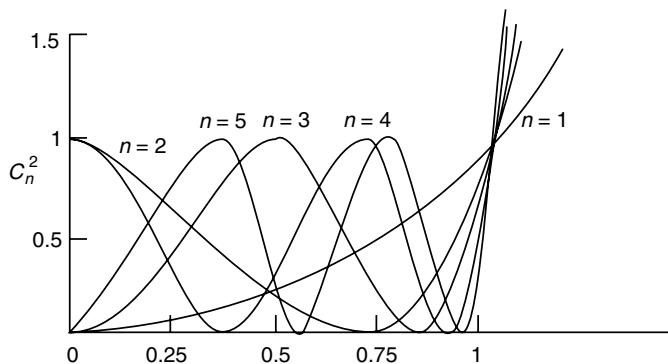


Fig. 6.3. Chebyshev functions for $n = 1, 2, \dots, 5$.

with $C_0(\Omega') = 1$ and $C_1(\Omega') = \Omega'$. We can then generate higher order Chebyshev polynomials through the recursive relation above; that is:

$$\left. \begin{aligned} C_2(\Omega') &= 2\Omega'^2 - 1 \\ C_3(\Omega') &= 4\Omega'^3 - 3\Omega' \\ &\vdots \\ C_{n+1}(\Omega') &= 2\Omega'C_n(\Omega') - C_{n-1}(\Omega') \end{aligned} \right\}. \quad (6.19)$$

Figure 6.3 depicts the Chebyshev functions for several values of n .

Since $C_n(\Omega') = 1$ at $\Omega' = \Omega'_p = 1$, we have that

$$A_p = A_{dB}(1) = 10 \log_{10}(1 + \epsilon^2) \quad (6.20)$$

and then

$$\epsilon = \sqrt{10^{0.1A_p} - 1}. \quad (6.21)$$

From Equations (6.15) and (6.16), when $\Omega' = \Omega'_r$, we find

$$A_r = A_{dB}(\Omega'_r) = 10 \log_{10} \left[1 + \epsilon^2 \cosh^2 \left(n \cosh^{-1} \Omega'_r \right) \right], \quad (6.22)$$

and thus the order of the normalized-lowpass Chebyshev filter that satisfies the required stopband attenuation is the smallest integer number that satisfies

$$n \geq \frac{\cosh^{-1} \sqrt{(10^{0.1A_r} - 1)/\epsilon^2}}{\cosh^{-1} \Omega'_r}. \quad (6.23)$$

Similar to the Butterworth case (see Equation (6.9)), we can now continue the approximation process by evaluating the zeros of $A(s')A(-s')$, with $s' = j\Omega'$. Since zero attenuation

can never occur in the stopband, these zeros are in the passband region $0 \leq \Omega' \leq 1$, and thus, from Equation (6.16), we have

$$\cos\left(n \cos^{-1} \frac{s'}{j}\right) = \pm \frac{j}{\epsilon}. \quad (6.24)$$

The above equation can be solved for s' by defining a complex variable p as

$$p = x_1 + jx_2 = \cos^{-1}\left(\frac{s'}{j}\right). \quad (6.25)$$

Replacing this value of p in Equation (6.24), we arrive at

$$\cos[n(x_1 + jx_2)] = \cos nx_1 \cosh nx_2 - j \sin nx_1 \sinh nx_2 = \pm \frac{j}{\epsilon}. \quad (6.26)$$

By equating the real parts of both sides of Equation (6.26), we can deduce that

$$\cos nx_1 \cosh nx_2 = 0, \quad (6.27)$$

and considering that

$$\cosh nx_2 \geq 1, \quad \text{for all } n, x_2, \quad (6.28)$$

we then have

$$\cos nx_1 = 0, \quad (6.29)$$

which yields the following $2n$ solutions:

$$x_{1i} = \frac{2i+1}{2n}\pi \quad (6.30)$$

for $i = 0, 1, \dots, (2n-1)$. Now, by equating the imaginary parts of both sides of Equation (6.26) and using the values of x_{1i} obtained in Equation (6.30), it follows that

$$\sin nx_{1i} = \pm 1 \quad (6.31)$$

$$x_2 = \frac{1}{n} \sinh^{-1}\left(\frac{1}{\epsilon}\right). \quad (6.32)$$

Since, from Equations (6.32) and (6.25), the zeros of $A(s')A(-s')$ are given by

$$s'_i = \sigma'_i \pm j\Omega'_i = j \cos(x_{1i} + jx_2) = \sin x_{1i} \sinh x_2 + j \cos x_{1i} \cosh x_2 \quad (6.33)$$

for $i = 0, 1, \dots, (2n-1)$, we have, from Equations (6.25) and (6.30), that

$$\sigma_i = \pm \sin\left[\frac{\pi}{2} \left(\frac{2i+1}{n}\right)\right] \sinh\left(\frac{1}{n} \sinh^{-1} \frac{1}{\epsilon}\right) \quad (6.34)$$

$$\Omega_i = \cos\left[\frac{\pi}{2} \left(\frac{2i+1}{n}\right)\right] \cosh\left(\frac{1}{n} \sinh^{-1} \frac{1}{\epsilon}\right). \quad (6.35)$$

The calculated zeros belong to $A(s')A(-s')$. Analogous to the Butterworth case, we associate the n zeros, p_i , with the negative real part to $A(s')$, in order to guarantee the filter stability.

The above equations indicate that the zeros of a Chebyshev approximation are placed on an ellipse in the s plane, since Equation (6.34) implies the following relation:

$$\left\{ \frac{\sigma_i}{\sinh[(1/n)\sinh^{-1}(1/\epsilon)]} \right\}^2 + \left\{ \frac{\Omega_i}{\cosh[(1/n)\sinh^{-1}(1/\epsilon)]} \right\}^2 = 1. \quad (6.36)$$

The transfer function of the Chebyshev filter is then given by

$$H'(s') = \frac{H'_0}{A(s')} = \frac{H'_0}{\prod_{i=1}^n (s' - p_i)}, \quad (6.37)$$

where H'_0 is chosen so that $A(s')$ satisfies Equation (6.15); that is (see also Figure 6.3):

$$H'_0 = \begin{cases} \prod_{i=1}^n (-p_i), & \text{for } n \text{ odd} \\ 10^{-0.05A_p} \prod_{i=1}^n (-p_i), & \text{for } n \text{ even.} \end{cases} \quad (6.38)$$

It is interesting to note that in the Butterworth case the frequency response is monotone in both passband and stopband, and is maximally flat at $\Omega = 0$. In the case of the Chebyshev filters, the smooth passband characteristics are exchanged for steeper transition bands for the same filter orders. In fact, for a given prescribed specification, Chebyshev filters usually require lower order transfer functions than Butterworth filters, owing to their equiripple behavior in the passband.

6.2.4 Elliptic approximation

The two approximations discussed so far, namely the lowpass Butterworth and Chebyshev approximations, lead to transfer functions whose numerator is a constant and the denominator is a polynomial in s . These are called all-pole filters, because all their zeros are located at infinity. When going from the Butterworth to the Chebyshev filters, we have traded off monotonicity and maximal flatness in the passband for higher attenuation in the stopband. At this point, it is natural to wonder whether we could also exchange the monotonicity in the stopband possessed by the Butterworth and Chebyshev filters for an even steeper transition band without increasing the filter order. This is indeed the case, as approximations with finite-frequency zeros can have transition bands with very steep slopes.

In practice, there are transfer function approximations with finite zeros which have equiripple characteristics in the passband and in the stopband, with the advantage that their coefficients can be computed using closed formulas. These filters are usually called

elliptic filters, as their closed-form equations are derived based on elliptic functions, but they are also known as Cauer or Zolotarev filters (Daniels, 1974).

This section covers the lowpass elliptic filter approximation (the derivations are not detailed here, as they are beyond the scope of this book). In the following, we describe an algorithm to calculate the coefficients of elliptic filters which is based on the procedure described in the benchmark book (Antoniou, 1993).

Consider the following lowpass filter transfer function:

$$|H(j\Omega')| = \frac{1}{\sqrt{1 + R_n^2(\Omega')}} \quad (6.39)$$

where

$$R_n(\Omega') = \begin{cases} C_e \prod_{i=1}^{n/2} \left[\frac{\Omega'^2 - (\Omega_r'^2/\Omega_i'^2)^2}{\Omega'^2 - \Omega_i'^2} \right], & \text{for } n \text{ even} \\ C_o \Omega' \prod_{i=1}^{(n-1)/2} \left[\frac{\Omega'^2 - (\Omega_r'^2/\Omega_i'^2)^2}{\Omega'^2 - \Omega_i'^2} \right], & \text{for } n \text{ odd} \end{cases}. \quad (6.40)$$

The computation of $R_n(\Omega')$ requires the use of some elliptic functions.

All frequencies in Equation (6.39) are normalized. The normalization procedure for the elliptic approximation is rather distinct from the one for the Butterworth and Chebyshev filters. Here, the frequency normalization factor is given by

$$\Omega_c = \sqrt{\Omega_p \Omega_r} \quad (6.41)$$

In that manner, we have that

$$\Omega'_p = \frac{\Omega_p}{\Omega_c} = \sqrt{\frac{\Omega_p}{\Omega_r}} \quad (6.42)$$

$$\Omega'_r = \frac{\Omega_r}{\Omega_c} = \sqrt{\frac{\Omega_r}{\Omega_p}}. \quad (6.43)$$

Defining

$$k = \frac{\Omega'_p}{\Omega'_r} = \frac{1}{\Omega_r'^2}, \quad (6.44)$$

$$q_0 = \frac{1}{2} \left[\frac{1 - (1 - k^2)^{1/4}}{1 + (1 - k^2)^{1/4}} \right], \quad (6.45)$$

$$q = q_0 + 2q_0^5 + 15q_0^9 + 150q_0^{13}, \quad (6.46)$$

and

$$\epsilon = \sqrt{\frac{10^{0.1A_p} - 1}{10^{0.1A_r} - 1}}, \quad (6.47)$$

the specifications are satisfied if the filter order n is chosen through the following relation:

$$n \geq \frac{\log_{10}(16/\epsilon^2)}{\log_{10}(1/q)}. \quad (6.48)$$

Having the filter order n , we can then determine the following parameters before proceeding with the computation of the filter coefficients:

$$\Theta = \frac{1}{2n} \ln \frac{10^{0.05A_p} + 1}{10^{0.05A_p} - 1} \quad (6.49)$$

$$\sigma = \left| \frac{2q^{1/4} \sum_{j=0}^{\infty} (-1)^j q^{j(j+1)} \sinh[(2j+1)\Theta]}{1 + 2 \sum_{j=1}^{\infty} (-1)^j q^{j^2} \cosh(2j\Theta)} \right| \quad (6.50)$$

$$W = \sqrt{(1 + k\sigma^2) \left(1 + \frac{\sigma^2}{k}\right)}. \quad (6.51)$$

Also, for $i = 1, 2, \dots, l$, where $l = n/2$ for n even and $l = (n-1)/2$ for n odd, we compute

$$\Omega'_i = \frac{2q^{1/4} \sum_{j=0}^{\infty} (-1)^j q^{j(j+1)} \sin[(2j+1)\pi u/n]}{1 + 2 \sum_{j=1}^{\infty} (-1)^j q^{j^2} \cos(2j\pi u/n)} \quad (6.52)$$

$$V_i = \sqrt{(1 - k\Omega'^2_i) \left(1 - \frac{\Omega'^2_i}{k}\right)}, \quad (6.53)$$

where

$$\left. \begin{aligned} u &= i, && \text{for } n \text{ odd} \\ u &= i - \frac{1}{2}, && \text{for } n \text{ even} \end{aligned} \right\}. \quad (6.54)$$

The infinite summations in Equations (6.50) and (6.52) converge extremely quickly, and only two or three terms are sufficient to reach a very accurate result.

The transfer function of a normalized-lowpass elliptic filter can be written as

$$H'(s') = \frac{H'_0}{(s' + \sigma)^m} \prod_{i=1}^l \frac{s'^2 + b_{2i}}{s'^2 + a_{1i}s' + a_{2i}}, \quad (6.55)$$

where

$$\left. \begin{array}{ll} m = 0 \text{ and } l = n/2, & \text{for } n \text{ even} \\ m = 1 \text{ and } l = (n - 1)/2, & \text{for } n \text{ odd} \end{array} \right\}. \quad (6.56)$$

The coefficients of the above transfer function are calculated based on the parameters obtained from Equations (6.44)–(6.53) as

$$b_{2i} = \frac{1}{\Omega_i'^2} \quad (6.57)$$

$$a_{2i} = \frac{(\sigma V_i)^2 + (\Omega_i' W)^2}{(1 + \sigma^2 \Omega_i'^2)^2} \quad (6.58)$$

$$a_{1i} = \frac{2\sigma V_i}{1 + \sigma^2 \Omega_i'^2} \quad (6.59)$$

$$H'_0 = \begin{cases} \sigma \prod_{i=1}^l \frac{a_{2i}}{b_{2i}}, & \text{for } n \text{ odd} \\ 10^{-0.05A_p} \sigma \prod_{i=1}^l \frac{a_{2i}}{b_{2i}}, & \text{for } n \text{ even} \end{cases}. \quad (6.60)$$

Following this procedure, the resulting minimum stopband attenuation is slightly better than the specified value, being precisely given by

$$A_r = 10 \log_{10} \left(\frac{10^{0.1A_p} - 1}{16q^n} + 1 \right). \quad (6.61)$$

6.2.5 Frequency transformations

The approximation methods presented so far are meant for designing normalized-lowpass filters. In this subsection, we address the issue of how a transfer function of a general lowpass, highpass, symmetric bandpass, or symmetric bandstop filter can be transformed into a normalized-lowpass transfer function, and vice versa. The procedure used here, the so-called frequency transformation technique, consists of replacing the variable s' in the normalized-lowpass filter by an appropriate function of s . In the following, we make a detailed analysis of the normalized-lowpass \leftrightarrow bandpass transformation. The analyses of the other transformations are similar, and their expressions are summarized later in Table 6.2.

A normalized-lowpass transfer function $H'(s')$ can be transformed into a symmetric bandpass transfer function by applying the following substitution of variables:

$$s' \leftrightarrow \frac{1}{a} \frac{s^2 + \Omega_0^2}{Bs}, \quad (6.62)$$

where Ω_0 is the central frequency of the bandpass filter, B is the filter passband width, and a is a normalization parameter that depends upon the filter type as follows:

$$\Omega_0 = \sqrt{\Omega_{p1}\Omega_{p2}} \quad (6.63)$$

$$B = \Omega_{p2} - \Omega_{p1} \quad (6.64)$$

$$a = \frac{1}{\Omega'_p} = \begin{cases} 1, & \text{for any Butterworth or Chebyshev filter} \\ \sqrt{\frac{\Omega_r}{\Omega_p}}, & \text{for a lowpass elliptic filter} \\ \sqrt{\frac{\Omega_p}{\Omega_r}}, & \text{for a highpass elliptic filter} \\ \sqrt{\frac{\Omega_{r2} - \Omega_{r1}}{\Omega_{p2} - \Omega_{p1}}}, & \text{for a bandpass elliptic filter} \\ \sqrt{\frac{\Omega_{p2} - \Omega_{p1}}{\Omega_{r2} - \Omega_{r1}}}, & \text{for a bandstop elliptic filter} \end{cases} . \quad (6.65)$$

The value of a is different from unity for the elliptic filters because, in this case, the normalization is not $\Omega'_p = 1$, but $\sqrt{\Omega'_p\Omega'_r} = 1$ (see Equations (6.41)–(6.43)).

The frequency transformation in Equation (6.62) has the following properties:

- The frequency $s' = j0$ is transformed into $s = \pm j\Omega_0$.
- Any complex frequency $s' = -j\Omega'$, corresponding to an attenuation of A_{dB} in the normalized-lowpass filter, is transformed into the two distinct frequencies

$$\Omega_1 = -\frac{1}{2}aB\Omega' + \sqrt{\frac{1}{4}a^2B^2\Omega'^2 + \Omega_0^2} \quad (6.66)$$

$$\bar{\Omega}_1 = -\frac{1}{2}aB\Omega' - \sqrt{\frac{1}{4}a^2B^2\Omega'^2 + \Omega_0^2}, \quad (6.67)$$

where Ω_1 is a positive frequency and $\bar{\Omega}_1$ is a negative frequency, both corresponding to the attenuation A_{dB} .

In addition, a complex frequency $s' = j\Omega'$, which also corresponds to an attenuation of A_{dB} , is transformed into two frequencies with the same attenuation level; that is:

$$\Omega_2 = \frac{1}{2}aB\Omega' + \sqrt{\frac{1}{4}a^2B^2\Omega'^2 + \Omega_0^2} \quad (6.68)$$

$$\bar{\Omega}_2 = \frac{1}{2}aB\Omega' - \sqrt{\frac{1}{4}a^2B^2\Omega'^2 + \Omega_0^2}, \quad (6.69)$$

and it can be seen that $\bar{\Omega}_1 = -\Omega_2$ and $\bar{\Omega}_2 = -\Omega_1$.

The positive frequencies Ω_1 and Ω_2 are the ones we are interested in analyzing. They can be expressed in a single equation as follows:

$$\Omega_{1,2} = \mp \frac{1}{2}aB\Omega' + \sqrt{\frac{1}{4}a^2B^2\Omega'^2 + \Omega_0^2}, \quad (6.70)$$

from which we get

$$\Omega_2 - \Omega_1 = aB\Omega' \quad (6.71)$$

$$\Omega_1\Omega_2 = \Omega_0^2. \quad (6.72)$$

These relationships indicate that, in this kind of transformation, for each frequency with attenuation A_{dB} there is another frequency geometrically symmetric with respect to the central frequency Ω_0 with the same attenuation.

- From the above, the cutoff frequency of the normalized-lowpass filter Ω'_p is mapped into the frequencies

$$\Omega_{p1,2} = \mp \frac{1}{2}aB\Omega'_p + \sqrt{\frac{1}{4}a^2B^2\Omega'^2_p + \Omega_0^2}, \quad (6.73)$$

such that

$$\Omega_{p2} - \Omega_{p1} = aB\Omega'_p \quad (6.74)$$

$$\Omega_{p1}\Omega_{p2} = \Omega_0^2. \quad (6.75)$$

- Similarly, the stopband edge frequency Ω'_r of the normalized-lowpass prototype is transformed into the frequencies

$$\Omega_{r1,2} = \mp \frac{1}{2}aB\Omega'_r + \sqrt{\frac{1}{4}a^2B^2\Omega'^2_r + \Omega_0^2}, \quad (6.76)$$

such that

$$\Omega_{r2} - \Omega_{r1} = aB\Omega'_r \quad (6.77)$$

$$\Omega_{r1}\Omega_{r2} = \Omega_0^2. \quad (6.78)$$

The above analysis leads to the conclusion that this normalized-lowpass \leftrightarrow bandpass transformation works for bandpass filters which are geometrically symmetric with respect to the central frequency. However, bandpass filter specifications are not usually geometrically symmetric in practice. Fortunately, we can generate geometrically symmetric bandpass specifications satisfying the minimum stopband attenuation requirements by the following procedure (see Figure 6.4):

- (i) Compute $\Omega_0^2 = \Omega_{p1}\Omega_{p2}$.
- (ii) Compute $\bar{\Omega}_{r1} = \Omega_0^2/\Omega_{r2}$, and if $\bar{\Omega}_{r1} > \Omega_{r1}$, replace Ω_{r1} with $\bar{\Omega}_{r1}$, as illustrated in Figure 6.4.

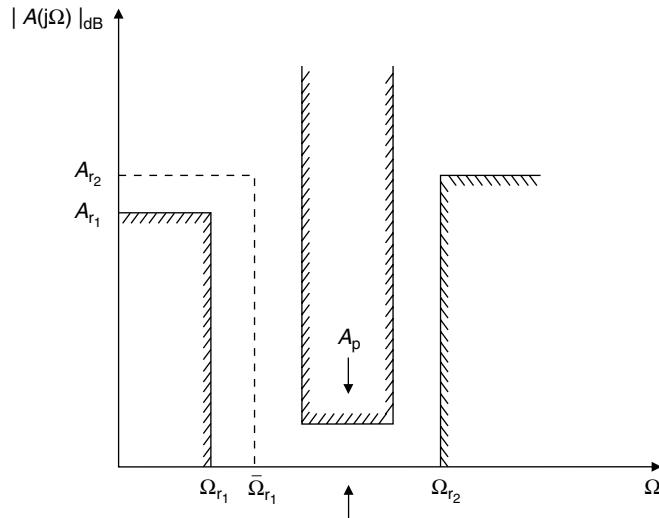


Fig. 6.4. Nonsymmetric bandpass filter specifications.

- (iii) If $\bar{\Omega}_{r_1} \leq \Omega_{r_1}$, then compute $\bar{\Omega}_{r_2} = \frac{\Omega_0^2}{\Omega_{r_1}}$, and replace Ω_{r_2} with $\bar{\Omega}_{r_2}$.
- (iv) If $A_{r_1} \neq A_{r_2}$, choose $A_r = \max\{A_{r_1}, A_{r_2}\}$.

Once the geometrically symmetric bandpass filter specifications are available, we need to determine the normalized frequencies Ω'_p and Ω'_r , in order to have the corresponding normalized-lowpass filter completely specified. According to Equations (6.74) and (6.77), they can be computed as follows:

$$\Omega'_p = \frac{1}{a} \quad (6.79)$$

$$\Omega'_r = \frac{1}{a} \frac{\Omega_{r_2} - \Omega_{r_1}}{\Omega_{p_2} - \Omega_{p_1}}. \quad (6.80)$$

It is worth noting that bandstop filter specifications must also be geometrically symmetric. In this case, however, in order to satisfy the minimum stopband attenuation requirements, the stopband edges must be preserved, while the passband edges should be modified analogously according to the procedure described above.

A summary of all types of transformation, including the respective correspondence among the lowpass prototype frequencies and desired filter specifications, is shown in Table 6.2.

The general procedure to approximate a standard analog filter using frequency transformations can be summarized as follows:

- (i) Determine the specifications for the lowpass, highpass, bandpass, or bandstop analog filter.
- (ii) When designing a bandpass or bandstop filter, make sure the specifications are geometrically symmetric following the proper procedure described earlier in this subsection.

Table 6.2.

Analog frequency transformations.

Transformation	Normalization	Denormalization
lowpass(Ω) \leftrightarrow lowpass(Ω')	$\Omega'_p = \frac{1}{a}$ $\Omega'_r = \frac{1}{a} \frac{\Omega_r}{\Omega_p}$	$s' \leftrightarrow \frac{1}{a} \frac{s}{\Omega_p}$
highpass(Ω) \leftrightarrow lowpass(Ω')	$\Omega'_p = \frac{1}{a}$ $\Omega'_r = \frac{1}{a} \frac{\Omega_p}{\Omega_r}$	$s' \leftrightarrow \frac{1}{a} \frac{\Omega_p}{s}$
bandpass(Ω) \leftrightarrow lowpass(Ω')	$\Omega'_p = \frac{1}{a}$ $\Omega'_r = \frac{1}{a} \frac{\Omega_{r_2} - \Omega_{r_1}}{\Omega_{p_2} - \Omega_{p_1}}$	$s' \leftrightarrow \frac{1}{a} \frac{s^2 + \Omega_0^2}{Bs}$
bandstop(Ω) \leftrightarrow lowpass(Ω')	$\Omega'_p = \frac{1}{a}$ $\Omega'_r = \frac{1}{a} \frac{\Omega_{p_2} - \Omega_{p_1}}{\Omega_{r_2} - \Omega_{r_1}}$	$s' \leftrightarrow \frac{1}{a} \frac{Bs}{s^2 + \Omega_0^2}$

- (iii) Determine the normalized-lowpass specifications equivalent to the desired filter, following the relationships seen in Table 6.2.
- (iv) Perform the filter approximation using the Butterworth, Chebyshev, or elliptic methods.
- (v) Denormalize the prototype using the frequency transformations given on the right-hand side of Table 6.2.

Sometimes the approximation of analog filters can present poor numerical conditioning, especially when the desired filter has a narrow transition and/or passband. In this case, design techniques employing transformed variables are available (Daniels, 1974; Sedra & Brackett, 1978), which can improve the numerical conditioning by separating the roots of the polynomials involved.

Example 6.1. Design a bandpass filter satisfying the specification below using the Butterworth, Chebyshev, and elliptic approximation methods:

$$\left. \begin{aligned} A_p &= 1.0 \text{ dB} \\ A_r &= 40 \text{ dB} \\ \Omega_{r_1} &= 1394\pi \text{ rad/s} \\ \Omega_{p_1} &= 1510\pi \text{ rad/s} \\ \Omega_{p_2} &= 1570\pi \text{ rad/s} \\ \Omega_{r_2} &= 1704\pi \text{ rad/s} \end{aligned} \right\}. \quad (6.81)$$

Solution

Since $\Omega_{p_1}\Omega_{p_2} \neq \Omega_{r_1}\Omega_{r_2}$, the first step in the design is to determine the geometrically symmetric bandpass filter following the procedure described earlier in this subsection. In that manner, we get

$$\Omega_{r_2} = \bar{\Omega}_{r_2} = \frac{\Omega_0^2}{\Omega_{r_1}} = 1700.6456\pi \text{ rad/s.} \quad (6.82)$$

Finding the corresponding lowpass specifications based on the transformations in Table 6.2, we have

$$\Omega'_p = \frac{1}{a} \quad (6.83)$$

$$\Omega'_r = \frac{1}{a} \frac{\bar{\Omega}_{r_2} - \Omega_{r_1}}{\Omega_{p_2} - \Omega_{p_1}} = \frac{1}{a} 5.1108, \quad (6.84)$$

where

$$a = \begin{cases} 1, & \text{for the Butterworth and Chebyshev filters} \\ 2.2607, & \text{for the elliptic filter} \end{cases}. \quad (6.85)$$

- *Butterworth approximation:* From the specifications above, we can compute ϵ from Equation (6.6), and, having ϵ , the minimum filter order required to satisfy the specifications from Equation (6.8):

$$\epsilon = 0.5088 \quad (6.86)$$

$$n = 4. \quad (6.87)$$

From Equation (6.12), the zeros of the normalized Butterworth polynomial when $n = 4$ are given by

$$\left. \begin{array}{l} s'_{1,2} = -1.0939 \pm j0.4531 \\ s'_{3,4} = -0.4531 \pm j1.0939 \\ s'_{5,6} = 1.0939 \pm j0.4531 \\ s'_{7,8} = 0.4531 \pm j1.0939 \end{array} \right\}. \quad (6.88)$$

Selecting the ones with negative real part to be the poles of $H'(s')$, this normalized transfer function becomes

$$H'(s') = 1.9652 \frac{1}{s'^4 + 3.0940s'^3 + 4.7863s'^2 + 4.3373s' + 1.9652}. \quad (6.89)$$

Table 6.3.

Characteristics of the Butterworth bandpass filter. Gain constant: $H_0 = 2.4809 \times 10^9$.

Denominator coefficients	Filter poles
$a_0 = 2.9971 \times 10^{29}$	$p_1 = -41.7936 + j4734.9493$
$a_1 = 7.4704 \times 10^{24}$	$p_2 = -41.7936 - j4734.9493$
$a_2 = 5.1331 \times 10^{22}$	$p_3 = -102.1852 + j4793.5209$
$a_3 = 9.5851 \times 10^{17}$	$p_4 = -102.1852 - j4793.5209$
$a_4 = 3.2927 \times 10^{15}$	$p_5 = -104.0058 + j4878.9280$
$a_5 = 4.0966 \times 10^{10}$	$p_6 = -104.0058 - j4878.9280$
$a_6 = 9.3762 \times 10^7$	$p_7 = -43.6135 + j4941.1402$
$a_7 = 5.8320 \times 10^2$	$p_8 = -43.6135 - j4941.1402$
$a_8 = 1.0$	

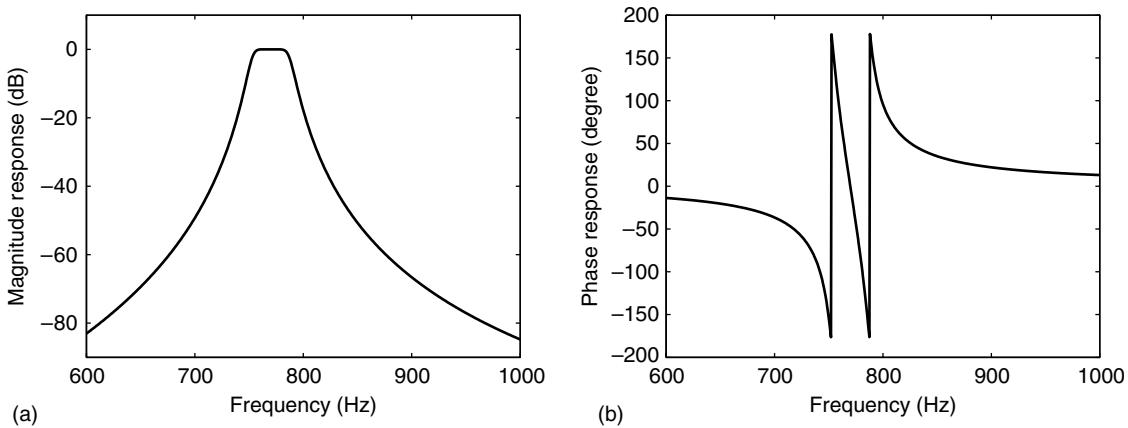


Fig. 6.5. Bandpass Butterworth filter: (a) magnitude response; (b) phase response.

The design is completed by applying the lowpass to bandpass transformation in Table 6.2. The resulting bandpass transfer function is then given by

$$H(s) = H_0 \frac{s^4}{a_8 s^8 + a_7 s^7 + a_6 s^6 + a_5 s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}, \quad (6.90)$$

where the filter coefficients and poles are listed in Table 6.3.

Figure 6.5 depicts the frequency response of the designed Butterworth bandpass filter.

- *Chebyshev approximation:* From the normalized specifications in Equations (6.81) and (6.82), one can compute ϵ and n based on Equations (6.21) and (6.23) respectively,

Table 6.4.

Characteristics of the Chebyshev bandpass filter. Gain constant: $H_0 = 3.2905 \times 10^6$.

Denominator coefficients	Filter poles
$a_0 = 1.2809 \times 10^{22}$	$p_1 = -22.8490 + j4746.8921$
$a_1 = 1.0199 \times 10^{17}$	$p_2 = -22.8490 - j4746.8921$
$a_2 = 1.6434 \times 10^{15}$	$p_3 = -46.5745 + j4836.9104$
$a_3 = 8.7212 \times 10^9$	$p_4 = -46.5745 - j4836.9104$
$a_4 = 7.0238 \times 10^7$	$p_5 = -23.7255 + j4928.9785$
$a_5 = 1.8630 \times 10^2$	$p_6 = -23.7255 - j4928.9785$
$a_6 = 1.0$	

resulting in

$$\epsilon = 0.5088 \quad (6.91)$$

$$n = 3. \quad (6.92)$$

Then, from Equations (6.24)–(6.35), we have that the poles of the normalized transfer function are

$$\left. \begin{aligned} s'_{1,2} &= -0.2471 \mp j0.9660 \\ s'_3 &= -0.4942 - j0.0 \end{aligned} \right\}. \quad (6.93)$$

This implies that the normalized-lowpass filter has the following transfer function:

$$H'(s') = 0.4913 \frac{1}{s'^3 + 0.9883s'^2 + 1.2384s' + 0.4913}. \quad (6.94)$$

The denormalized design is obtained by applying the lowpass to bandpass transformation. The resulting transfer function is of the form

$$H(s) = H_0 \frac{s^3}{a_6s^6 + a_5s^5 + a_4s^4 + a_3s^3 + a_2s^2 + a_1s + a_0}, \quad (6.95)$$

where all filter coefficients and poles are listed in Table 6.4.

Figure 6.6 depicts the frequency response of the resulting Chebyshev bandpass filter.

- *Elliptic approximation:* From Equation (6.85), for this elliptic approximation, we have that $a = 2.2607$, and then the normalized specifications are

$$\Omega'_p = 0.4423 \quad (6.96)$$

$$\Omega'_r = 2.2607. \quad (6.97)$$

From Equation (6.48), the minimum order required for the elliptic approximation to satisfy the specifications is $n = 3$. Therefore, from Equations (6.55)–(6.60), the

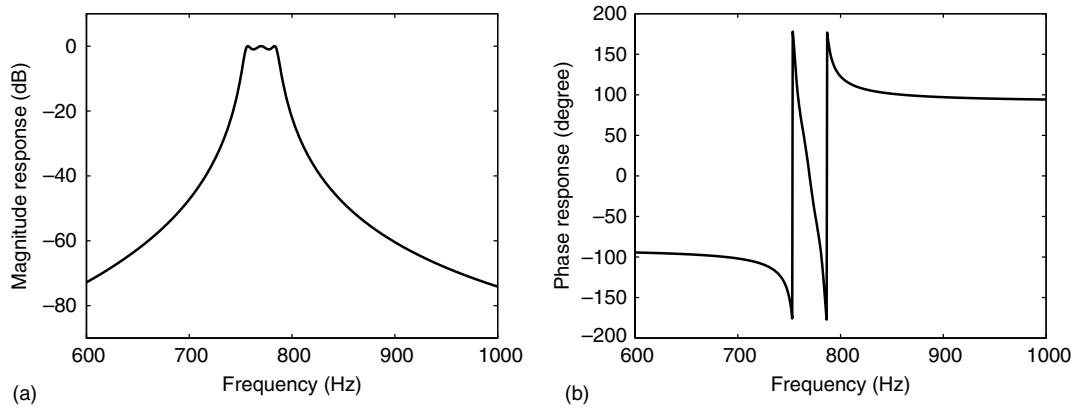


Fig. 6.6. Bandpass Chebyshev filter: (a) magnitude response; (b) phase response.

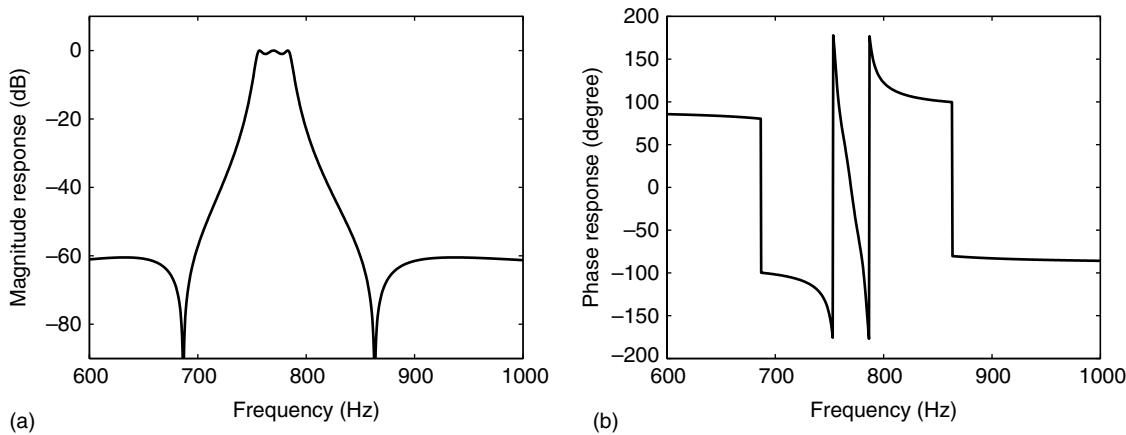


Fig. 6.7. Bandpass elliptic filter: (a) magnitude response; (b) phase response.

normalized-lowpass filter has the transfer function

$$H'(s') = 6.3627 \times 10^{-3} \frac{s'^2 + 6.7814}{s'^3 + 0.4362s'^2 + 0.2426s' + 0.0431}. \quad (6.98)$$

The denormalized-bandpass design is then obtained by applying the lowpass to bandpass transformation given in Table 6.2, with $a = 2.2607$. The resulting bandpass transfer function is given by

$$H(s) = H_0 \frac{b_5 s^5 + b_3 s^3 + b_1 s}{a_6 s^6 + a_5 s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}, \quad (6.99)$$

where all filter coefficients, zeros, and poles are listed in Table 6.5.

Figure 6.7 depicts the frequency response of the resulting elliptic bandpass filter. \triangle

Table 6.5.

Characteristics of the elliptic bandpass filter. Gain constant: $H_0 = 2.7113 \times 10^6$.

Numerator coefficients	Denominator coefficients
$b_0 = 0.0$	$a_0 = 1.2809 \times 10^{22}$
$b_1 = 5.4746 \times 10^{14}$	$a_1 = 1.0175 \times 10^{17}$
$b_2 = 0.0$	$a_2 = 1.6434 \times 10^{15}$
$b_3 = 4.8027 \times 10^7$	$a_3 = 8.7008 \times 10^9$
$b_4 = 0.0$	$a_4 = 7.0238 \times 10^7$
$b_5 = 1.0$	$a_5 = 1.8586 \times 10^2$
$b_6 = 0.0$	$a_6 = 1.0$
Filter zeros	Filter poles
$z_1 = +j4314.0061$	$p_1 = -22.4617 + j4746.6791$
$z_2 = -j4314.0061$	$p_2 = -22.4617 - j4746.6791$
$z_3 = +j5423.6991$	$p_3 = -47.1428 + j4836.9049$
$z_4 = -j5423.6991$	$p_4 = -47.1428 - j4836.9049$
$z_5 = 0.0$	$p_5 = -23.3254 + j4929.2035$
	$p_6 = -23.3254 - j4929.2035$

6.3 Continuous-time to discrete-time transformations

As mentioned at the beginning of this chapter, a classical procedure for designing IIR digital filters is to design an analog prototype first and then transform it into a digital filter. In this section we study two methods of carrying out this transformation, namely the impulse-invariance method and the bilinear transformation method.

6.3.1 Impulse-invariance method

The intuitive way to implement a digital filtering operation, having an analog prototype as starting point, is the straightforward digitalization of the convolution operation, as follows. The output $y_a(t)$ of an analog filter having impulse response $h_a(t)$ when excited by a signal $x_a(t)$ is

$$y_a(t) = \int_{-\infty}^{\infty} x_a(\tau) h_a(t - \tau) d\tau. \quad (6.100)$$

One possible way to implement this operation in the discrete-time domain is to divide the time axis into slices of size T , replacing the integral by a summation of the areas of rectangles of width T and height $x_a(mT)h_a(t - mT)$, for all integers m . Equation (6.100)

then becomes

$$y_a(t) = \sum_{m=-\infty}^{\infty} x_a(mT)h_a(t - mT)T. \quad (6.101)$$

The sampled version of $y_a(t)$ is obtained by substituting t by nT , yielding

$$y_a(nT) = \sum_{m=-\infty}^{\infty} x_a(mT)h_a(nT - mT)T. \quad (6.102)$$

This is clearly equivalent to obtaining the samples $y_a(nT)$ of $y_a(t)$ by filtering the samples $x_a(nT)$ with a digital filter having impulse response $h(n) = h_a(nT)$. That is, the impulse response of the equivalent digital filter would be a sampled version of the impulse response of the analog filter, using the same sampling rate for the input and output signals.

Roughly speaking, if the Nyquist criterion is met by the filter impulse response during the sampling operation, the discrete-time prototype has the same frequency response as the continuous-time one. In addition, a sampled version of a stable analog impulse response is clearly also stable. These are the main properties of this method of generating IIR filters, called the impulse-invariance method. In what follows, we analyze the above properties more precisely, in order to get a better understanding of the main strengths and limitations of this method.

We can begin by investigating the properties of the digital filter with impulse response $h(n) = h_a(nT)$ in the frequency domain. From Equation (6.102), the discrete-time Fourier transform of $h(n)$ is

$$H(e^{j\Omega T}) = \frac{1}{T} \sum_{l=-\infty}^{\infty} H_a(j\Omega + j\Omega_s l), \quad (6.103)$$

where $H_a(s)$, $s = \sigma + j\Omega$, is the analog transfer function and $\Omega_s = 2\pi/T$ is the sampling frequency. That is, the digital frequency response is equal to the analog one replicated at intervals $l\Omega_s$. One important consequence of this fact is that if $H_a(j\Omega)$ has much energy for $\Omega > \Omega_s/2$, there will be aliasing and, therefore, the digital frequency response will be a severely distorted version of the analog one.

Another way of seeing this is that the digital frequency response is obtained by folding the analog frequency response, for $-\infty < \Omega < \infty$, on the unit circle of the $z = e^{j\Omega T}$ plane, with each interval $[\sigma + j(l - \frac{1}{2})\Omega_s, \sigma + j(l + \frac{1}{2})\Omega_s]$, for all integers l , corresponding to one full turn over the unit circle of the z plane. This limits the usefulness of the impulse-invariance method to the design of transfer functions whose magnitude responses decrease monotonically at high frequencies. For example, its use in the direct design of highpass, bandstop, or even elliptic lowpass and bandpass filters is strictly forbidden, and other methods should be considered for designing such filters.

Stability of the digital filter can also be inferred from the stability of the analog prototype by analyzing Equation (6.103). In fact, based on that equation, we can interpret the impulse-invariance method as a mapping from the s domain to the z domain such that each slice of the s plane given by the interval $[\sigma + j(l - \frac{1}{2})\Omega_s, \sigma + j(l + \frac{1}{2})\Omega_s]$, for all integers l , where

$\sigma = \operatorname{Re}\{s\}$, is mapped into the same region of the z plane. Also, the left side of the s plane (that is, where $\sigma < 0$) is mapped into the interior of the unit circle, implying that if the analog transfer function is stable (all poles on the left side of the s plane), then the digital transfer function is also stable (all poles inside the unit circle of the z plane).

In practice, the impulse-invariance transformation is not implemented through Equation (6.103), as a simpler procedure can be deduced by expanding an N th-order $H_a(s)$ as follows:

$$H_a(s) = \sum_{l=1}^N \frac{r_l}{s - p_l}, \quad (6.104)$$

where it is assumed that $H_a(s)$ does not have multiple poles. The corresponding impulse response is given by

$$h_a(t) = \sum_{l=1}^N r_l e^{p_l t} u(t), \quad (6.105)$$

where $u(t)$ is the unit step function. If we now sample that impulse response, the resulting sequence is

$$h_d(n) = h_a(nT) = \sum_{l=1}^N r_l e^{p_l nT} u(nT) \quad (6.106)$$

and the corresponding discrete-time transfer function is given by

$$H_d(z) = \sum_{l=1}^N \frac{r_l z}{z - e^{p_l T}}. \quad (6.107)$$

This equation shows that a pole $s = p_l$ of the continuous-time filter corresponds to a pole of the discrete-time filter at $z = e^{p_l T}$. In that way, if p_l has a negative real part, then $e^{p_l T}$ is inside the unit circle, generating a stable digital filter when we use the impulse-invariance method.

In order to obtain the same passband gain for the continuous- and discrete-time filters, for any value of the sampling period T , we should use the following expression for $H_d(z)$:

$$H_d(z) = \sum_{l=1}^N T \frac{r_l z}{z - e^{p_l T}}, \quad (6.108)$$

which corresponds to

$$h_d(n) = T h_a(nT). \quad (6.109)$$

Thus, the overall impulse-invariance method consists of writing the analog transfer function $H_a(s)$ in the form of Equation (6.104), determining the poles p_l and corresponding residues r_l , and generating $H_d(z)$ according to Equation (6.108).

Example 6.2. Transform the continuous-time lowpass transfer function given by

$$H(s) = \frac{1}{s^2 + s + 1} \quad (6.110)$$

into a discrete-time transfer function using the impulse-invariance transformation method with $\Omega_s = 10$ rad/s. Plot the corresponding analog and digital magnitude responses.

Solution

A second-order lowpass transfer function can be written as

$$\begin{aligned} H(s) &= \frac{\Omega_0^2}{s^2 + (\Omega_0/Q)s + \Omega_0^2} \\ &= \frac{\Omega_0^2}{\sqrt{\frac{\Omega_0^2}{Q^2} - 4\Omega_0^2}} \left[\frac{1}{s + (\Omega_0/2Q) - \sqrt{(\Omega_0^2/4Q^2) - \Omega_0^2}} \right. \\ &\quad \left. - \frac{1}{s + (\Omega_0/2Q) + \sqrt{(\Omega_0^2/4Q^2) - \Omega_0^2}} \right]. \end{aligned} \quad (6.111)$$

Its poles are located at

$$p_1 = p_2^* = -\frac{\Omega_0}{2Q} + j\sqrt{\Omega_0^2 - \frac{\Omega_0^2}{4Q^2}} \quad (6.112)$$

and the corresponding residues are given by

$$r_1 = r_2^* = \frac{-j\Omega_0^2}{\sqrt{4\Omega_0^2 - (\Omega_0^2/Q^2)}}. \quad (6.113)$$

Applying the impulse-invariance method with $T = 2\pi/10$, the resulting discrete-time transfer function is given by

$$\begin{aligned} H(z) &= \frac{2jT r_1 \sin(\text{Im}\{p_1\}T) e^{\text{Re}\{p_1\}T} z}{z^2 - 2 \cos(\text{Im}\{p_1\}T) e^{\text{Re}\{p_1\}T} z + e^{2\text{Re}\{p_1\}T}} \\ &= \frac{0.274\,331\,03\,z}{z^2 - 1.249\,825\,52\,z + 0.533\,488\,09}. \end{aligned} \quad (6.114)$$

The magnitude responses corresponding to the analog and digital transfer functions are depicted in Figure 6.8. As can be seen, the frequency responses are similar except for the limited stopband attenuation of the discrete-time filter, which is due to the aliasing effect.

△

We should again emphasize that the impulse-invariance method is suitable only for continuous-time prototypes with frequency responses that decrease monotonically at high

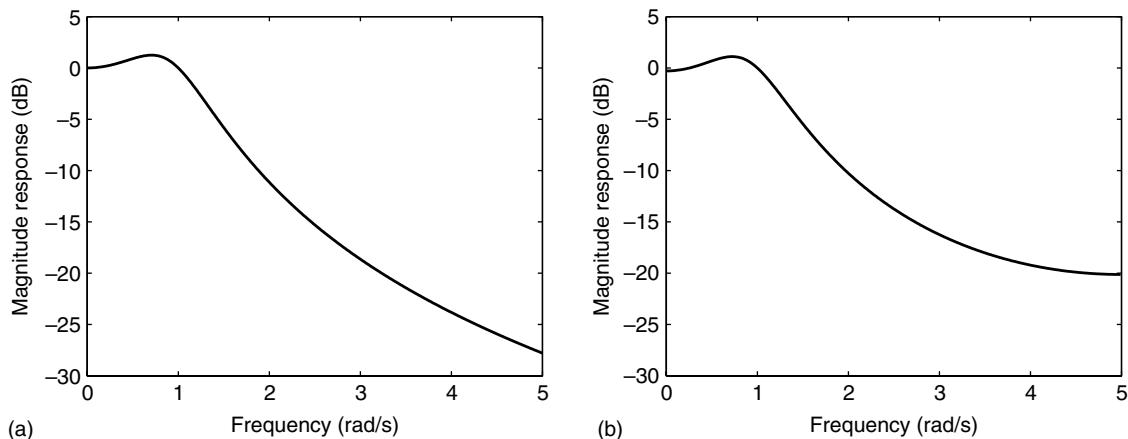


Fig. 6.8. Magnitude responses obtained with the impulse-invariance method: (a) continuous-time filter; (b) discrete-time filter.

frequencies, which limits its applicability a great deal. In the next section we analyze the bilinear transformation method, which overcomes some of the limitations of the impulse-invariance method.

6.3.2 Bilinear transformation method

The bilinear transformation method, like the impulse-invariance method, basically consists of mapping the left-hand side of the s plane into the interior of the unit circle of the z plane. The main difference between them is that in the bilinear transformation method the whole analog frequency range $-\infty < \Omega < \infty$ is squeezed into the unit circle $-\pi \leq \omega \leq \pi$, while in the impulse-invariance method the analog frequency response is folded around the unit circle indefinitely. The main advantage of the bilinear transformation method is that aliasing is avoided, thereby keeping the magnitude response characteristics of the continuous-time transfer function when generating the discrete-time transfer function.

The bilinear mapping is derived by first considering the key points of the s plane and analyzing their corresponding points in the z plane after the transformation. Hence, the left-hand side of the s plane should be uniquely mapped into the interior of the unit circle of the z plane, and so on, as given in Table 6.6.

In order to satisfy the second and third requirements of Table 6.6, the bilinear transformation must have the following form

$$s \rightarrow k \frac{f_1(z) - 1}{f_2(z) + 1}, \quad (6.115)$$

where $f_1(1) = 1$ and $f_2(-1) = -1$.

Table 6.6.

Correspondence of key points of the s and z planes using the bilinear transformation method.

s plane	\rightarrow	z plane
$\sigma \pm j\Omega$	\rightarrow	$r e^{\pm j\omega}$
$j0$	\rightarrow	1
$j\infty$	\rightarrow	-1
$\sigma > 0$	\rightarrow	$r > 1$
$\sigma = 0$	\rightarrow	$r = 1$
$\sigma < 0$	\rightarrow	$r < 1$
$j\Omega$	\rightarrow	$e^{j\omega}$
$-\infty < \Omega < \infty$	\rightarrow	$-\pi < \omega < \pi$

Sufficient conditions for the last three mapping requirements to be satisfied can be determined as follows:

$$s = \sigma + j\Omega = k \frac{(\operatorname{Re}\{f_1(z)\} - 1) + j\operatorname{Im}\{f_1(z)\}}{(\operatorname{Re}\{f_2(z)\} + 1) + j\operatorname{Im}\{f_2(z)\}}. \quad (6.116)$$

Equating the real parts of both sides of the above equation, we have that

$$\sigma = k \frac{(\operatorname{Re}\{f_1(z)\} - 1)(\operatorname{Re}\{f_2(z)\} + 1) + \operatorname{Im}\{f_1(z)\}\operatorname{Im}\{f_2(z)\}}{(\operatorname{Re}\{f_2(z)\} + 1)^2 + (\operatorname{Im}\{f_2(z)\})^2}, \quad (6.117)$$

and since $\sigma = 0$ implies that $r = 1$, the following relation is valid:

$$\frac{\operatorname{Re}\{f_1(e^{j\omega})\} - 1}{\operatorname{Im}\{f_1(e^{j\omega})\}} = -\frac{\operatorname{Im}\{f_2(e^{j\omega})\}}{\operatorname{Re}\{f_2(e^{j\omega})\} + 1}. \quad (6.118)$$

The condition $\sigma < 0$ is equivalent to

$$\frac{\operatorname{Re}\{f_1(re^{j\omega})\} - 1}{\operatorname{Im}\{f_1(re^{j\omega})\}} < -\frac{\operatorname{Im}\{f_2(re^{j\omega})\}}{\operatorname{Re}\{f_2(re^{j\omega})\} + 1}, \quad r < 1. \quad (6.119)$$

The last two lines of Table 6.6 show the correspondence between the analog frequency and the unit circle of the z plane.

If we want the orders of the discrete-time and continuous-time systems to remain the same after the transformation, then $f_1(z)$ and $f_2(z)$ must be first-order polynomials. In addition, if we wish to satisfy the conditions imposed by Equation (6.115), we must choose $f_1(z) = f_2(z) = z$. It is straightforward to verify that Equation (6.118), as well as the inequality (6.119), is automatically satisfied with this choice for $f_1(z)$ and $f_2(z)$.

The bilinear transformation is then given by

$$s \rightarrow k \frac{z - 1}{z + 1}, \quad (6.120)$$

which, for $s = j\Omega$ and $z = e^{j\omega}$, is equivalent to

$$j\Omega \rightarrow k \frac{e^{j\omega} - 1}{e^{j\omega} + 1} = k \frac{e^{j\omega/2} - e^{-j\omega/2}}{e^{j\omega/2} + e^{-j\omega/2}} = jk \frac{\sin(\omega/2)}{\cos(\omega/2)} = jk \tan \frac{\omega}{2}; \quad (6.121)$$

that is:

$$\Omega \rightarrow k \tan \frac{\omega}{2}. \quad (6.122)$$

For small frequencies, $\tan(\omega/2) \approx \omega/2$. Hence, to keep the magnitude response of the digital filter approximately the same as the prototype analog filter at low frequencies, we should have for small frequencies $\Omega = \omega\Omega_s/(2\pi)$ and, therefore, we should choose $k = \Omega_s/\pi = 2/T$. In conclusion, the bilinear transformation of a continuous-time transfer function into a discrete-time transfer function is implemented through the following mapping:

$$H(z) = H_a(s)|_{s=\frac{2}{T}\frac{z-1}{z+1}}; \quad (6.123)$$

therefore, the bilinear transformation maps analog frequencies into digital frequencies as follows:

$$\Omega \rightarrow \frac{2}{T} \tan \frac{\omega}{2}. \quad (6.124)$$

For high frequencies, this relationship is highly nonlinear, as seen in Figure 6.9a, corresponding to a large distortion in the frequency response of the digital filter when compared to the analog prototype. The distortion in the magnitude response, also known as the warping effect, can be visualized as in Figure 6.9b.

The warping effect caused by the bilinear transformation can be compensated for by prewarping the frequencies given at the specifications before the analog filter is actually designed. For example, suppose we wish to design a lowpass digital filter with cutoff frequency ω_p and stopband edge ω_r . The prewarped specifications Ω_{ap} and Ω_{ar} of the lowpass analog prototype are then given by

$$\Omega_{ap} = \frac{2}{T} \tan \frac{\omega_p}{2} \quad (6.125)$$

$$\Omega_{ar} = \frac{2}{T} \tan \frac{\omega_r}{2}. \quad (6.126)$$

Following the same line of thought, we may apply prewarping to as many frequencies of interest as specified for the digital filter. If these frequencies are given by ω_i , for $i = 1, 2, \dots, n$, then the frequencies to be included in the analog filter specifications are

$$\Omega_{ai} = \frac{2}{T} \tan \frac{\omega_i}{2} \quad (6.127)$$

for $i = 1, 2, \dots, n$.

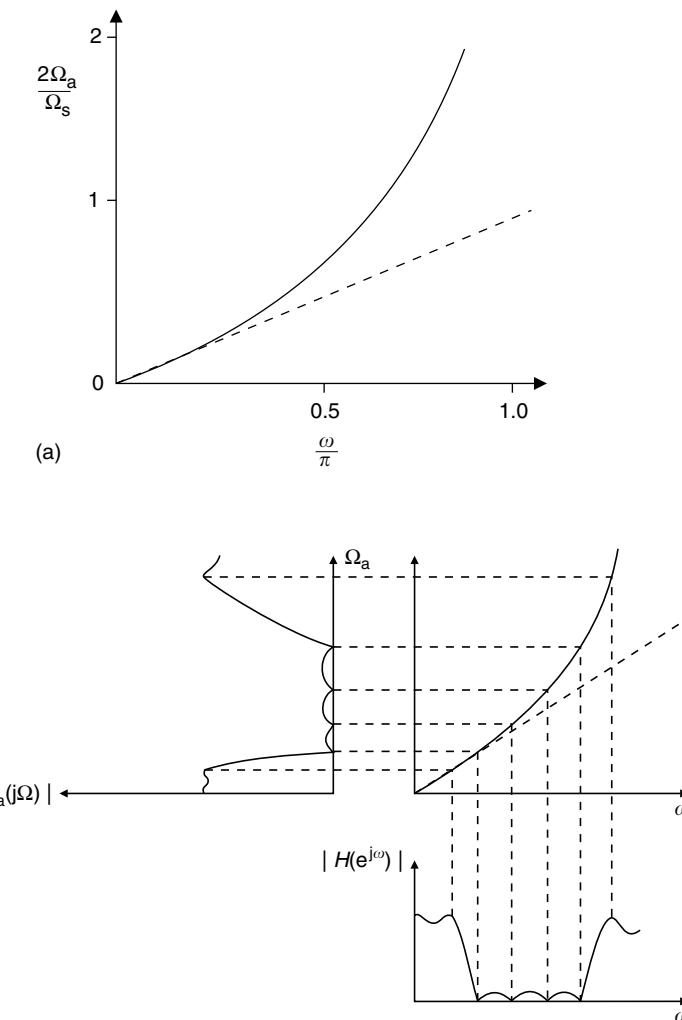


Fig. 6.9. Bilinear transformation method: (a) relation between the analog and digital frequencies; (b) warping effect in the magnitude response of a bandstop filter.

Hence, the design procedure using the bilinear transformation method can be summarized as follows:

- Prewarp all the prescribed frequency specifications ω_i , obtaining Ω_{a_i} , for $i = 1, 2, \dots, n$.
- Generate $H_a(s)$, following the procedure given in Section 6.2.5, satisfying the specifications for the frequencies Ω_{a_i} .
- Obtain $H_d(z)$, by replacing s with $(2/T)[(z - 1)/(z + 1)]$ in $H_a(s)$.

With the bilinear transformation, we can design Butterworth, Chebyshev, and elliptic digital filters starting with a corresponding analog prototype. The bilinear transformation

method always generates stable digital filters as long as the prototype analog filter is stable. Using the prewarping procedure, the method keeps the magnitude characteristics of the prototype but introduces distortions to the phase response.

Example 6.3. Design a digital elliptic bandpass filter satisfying the following specifications:

$$\left. \begin{array}{l} A_p = 0.5 \text{ dB} \\ A_r = 65 \text{ dB} \\ \Omega_{r_1} = 850 \text{ rad/s} \\ \Omega_{p_1} = 980 \text{ rad/s} \\ \Omega_{p_2} = 1020 \text{ rad/s} \\ \Omega_{r_2} = 1150 \text{ rad/s} \\ \Omega_s = 10000 \text{ rad/s} \end{array} \right\}. \quad (6.128)$$

Solution

First, we have to normalize the frequencies above to the range of digital frequencies using the expression $\omega = \Omega 2\pi / \Omega_s$. Since $\Omega_s = 10000 \text{ rad/s}$, we have that

$$\left. \begin{array}{l} \omega_{r_1} = 0.5341 \text{ rad/sample} \\ \omega_{p_1} = 0.6158 \text{ rad/sample} \\ \omega_{p_2} = 0.6409 \text{ rad/sample} \\ \omega_{r_2} = 0.7226 \text{ rad/sample} \end{array} \right\}. \quad (6.129)$$

Then, by applying Equation (6.127), the prewarped frequencies become

$$\left. \begin{array}{l} \Omega_{a_{r_1}} = 870.7973 \text{ rad/s} \\ \Omega_{a_{p_1}} = 1012.1848 \text{ rad/s} \\ \Omega_{a_{p_2}} = 1056.4085 \text{ rad/s} \\ \Omega_{a_{r_2}} = 1202.7928 \text{ rad/s} \end{array} \right\}. \quad (6.130)$$

By making $\Omega_{a_{r_1}} = 888.9982 \text{ rad/s}$ to obtain a geometrically symmetric filter, from Table 6.2 we have that

$$\Omega_0 = 1034.0603 \text{ rad/s} \quad (6.131)$$

$$B = 44.2237 \text{ rad/s} \quad (6.132)$$

$$a = 2.6638 \quad (6.133)$$

$$\Omega'_p = 0.3754 \quad (6.134)$$

$$\Omega'_r = 2.6638 \quad (6.135)$$

From the filter specifications, the required order for the analog elliptic normalized-lowpass filter is $n = 3$ and the resulting normalized transfer function is

$$H'(s') = 4.0426 \times 10^{-3} \frac{s'^2 + 9.4372}{s'^3 + 0.4696s'^2 + 0.2162s' + 0.0382}. \quad (6.136)$$

The denormalized design is then obtained by applying the lowpass-to-bandpass transformation given in Table 6.2, with $a = 2.6638$. After applying the bilinear transformation the resulting digital bandpass transfer function becomes

$$H(z) = H_0 \frac{b_6z^6 + b_5z^5 + b_4z^4 + b_3z^3 + b_2z^2 + b_1z + b_0}{a_6z^6 + a_5z^5 + a_4z^4 + a_3z^3 + a_2z^2 + a_1z + a_0}, \quad (6.137)$$

where all filter coefficients, zeros, and poles are listed in Table 6.7.

Figure 6.10 depicts the frequency response of the resulting digital elliptic bandpass filter. \triangle

From the previous discussion, we can observe that, as we perform the mapping $s \rightarrow z$, either opting for the impulse-invariance method or the bilinear transformation method, we are essentially folding the continuous-time frequency axis around the z -domain unit circle. For that matter, any digital frequency response is periodic, and the interval $-\Omega_s/2 \leq \Omega \leq \Omega_s/2$, or equivalently, $-\pi \leq \omega \leq \pi$, is the so-called fundamental period. We should bear in mind that in the expressions developed in this subsection the unfolded analog frequencies Ω are related to the digital frequencies ω , which are restricted to the interval $-\pi \leq \omega \leq \pi$. Therefore, all digital filter specifications should be normalized to the interval $[-\pi, \pi]$ using

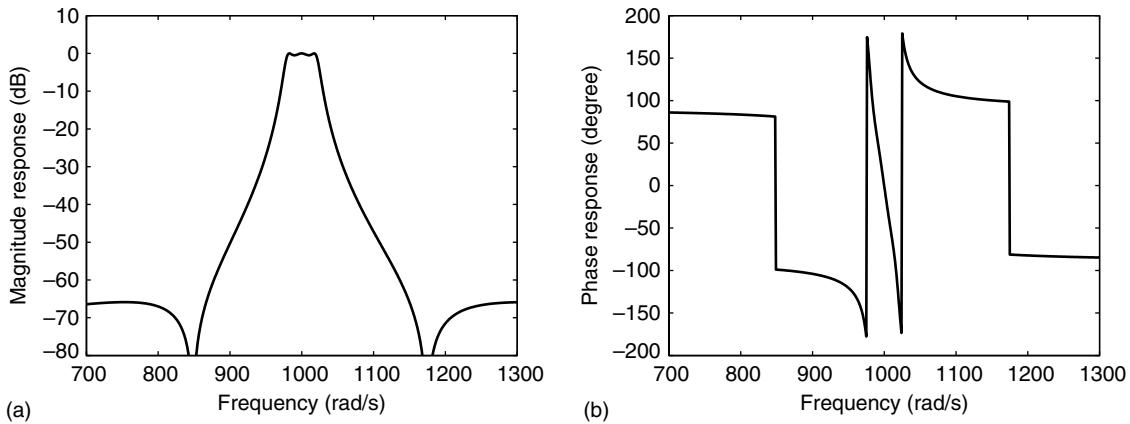


Fig. 6.10. Digital elliptic bandpass filter: (a) magnitude response; (b) phase response.

Table 6.7.

Characteristics of the digital elliptic bandpass filter. Gain constant: $H_0 = 1.3461 \times 10^{-4}$.

Numerator coefficients	Denominator coefficients
$b_0 = -1.0$	$a_0 = 0.9691$
$b_1 = 3.2025$	$a_1 = -4.7285$
$b_2 = -3.5492$	$a_2 = 10.6285$
$b_3 = 0.0$	$a_3 = -13.7261$
$b_4 = 3.5492$	$a_4 = 10.7405$
$b_5 = -3.2025$	$a_5 = -4.8287$
$b_6 = 1.0$	$a_6 = 1.0$
Filter zeros	Filter poles
$z_1 = 0.7399 + j0.6727$	$p_1 = 0.7982 + j0.5958$
$z_2 = 0.7399 - j0.6727$	$p_2 = 0.7982 - j0.5958$
$z_3 = 0.8613 + j0.5081$	$p_3 = 0.8134 + j0.5751$
$z_4 = 0.8613 - j0.5081$	$p_4 = 0.8134 - j0.5751$
$z_5 = 1.0 + j0.0$	$p_5 = 0.8027 + j0.5830$
$z_6 = -1.0 + j0.0$	$p_6 = 0.8027 - j0.5830$

the expression

$$\omega = \Omega \frac{2\pi}{\Omega_s}. \quad (6.138)$$

In the discussion that follows, we assume that $\Omega_s = 2\pi$ rad/s, unless specified otherwise.

6.4 Frequency transformation in the discrete-time domain

Usually, in the approximation of a continuous-time filter, we begin by designing a normalized lowpass filter and then, through a frequency transformation, the filter with the specified magnitude response is obtained. In the design of digital filters, we can also start by designing a digital lowpass filter and then apply a frequency transformation in the discrete-time domain.

The procedure consists of replacing the variable z by an appropriate function $g(z)$ to generate the desired magnitude response. The function $g(z)$ needs to meet some constraints to be a valid transformation (Constantinides, 1970), namely:

- The function $g(z)$ must be a ratio of polynomials, since the digital filter transfer function must remain a ratio of polynomials after the transformation.
- The mapping $z \rightarrow g(z)$ must be such that the filter stability is maintained; that is, stable filters generate stable transformed filters and unstable filters generate unstable

transformed filters. This is equivalent to saying that the transformation maps the interior of the unit circle onto the interior of the unit circle and the exterior of the unit circle onto the exterior of the unit circle.

It can be shown that a function $g(z)$ satisfying the above conditions is of the form

$$g(z) = \pm \left[\prod_{i=1}^n \frac{(z - \alpha_i)}{(1 - z\alpha_i^*)} \frac{(z - \alpha_i^*)}{(1 - z\alpha_i)} \right] \left(\prod_{i=n+1}^m \frac{z - \alpha_i}{1 - z\alpha_i} \right), \quad (6.139)$$

where α_i^* is complex conjugate of α_i and α_i is real for $n + 1 \leq i \leq m$.

In the following subsections, we analyze special cases of $g(z)$ that generate lowpass-to-lowpass, lowpass-to-highpass, lowpass-to-bandpass, and lowpass-to-bandstop transformations.

6.4.1 Lowpass-to-lowpass transformation

One necessary condition for a lowpass-to-lowpass transformation is that a magnitude response must keep its original values at $\omega = 0$ and $\omega = \pi$ after the transformation. Therefore, we must have

$$g(1) = 1 \quad (6.140)$$

$$g(-1) = -1. \quad (6.141)$$

Another necessary condition is that the frequency response should only be warped between $\omega = 0$ and $\omega = \pi$; that is, a full turn around the unit circle in z must correspond to a full turn around the unit circle in $g(z)$.

One possible $g(z)$ in the form of Equation (6.139) that satisfies these conditions is

$$g(z) = \frac{z - \alpha}{1 - \alpha z}, \quad (6.142)$$

where α is real such that $|\alpha| < 1$.

Assuming that the passband edge frequency of the original lowpass filter is given by ω_p , and that we wish to transform the original filter into a lowpass filter with cutoff frequency at ω_{p1} , that is, $g(e^{j\omega_{p1}}) = e^{j\omega_p}$, the following relation must be valid:

$$e^{j\omega_p} = \frac{e^{j\omega_{p1}} - \alpha}{1 - \alpha e^{j\omega_{p1}}} \quad (6.143)$$

and then

$$\alpha = \frac{e^{-j[(\omega_p - \omega_{p1})/2]} - e^{j[(\omega_p - \omega_{p1})/2]}}{e^{-j[(\omega_p + \omega_{p1})/2]} - e^{j[(\omega_p + \omega_{p1})/2]}} = \frac{\sin[(\omega_p - \omega_{p1})/2]}{\sin[(\omega_p + \omega_{p1})/2]}. \quad (6.144)$$

The desired transformation is then implemented by replacing z by $g(z)$ given in Equation (6.142) with α calculated as indicated in Equation (6.144).

6.4.2 Lowpass-to-highpass transformation

If ω_{p_1} is the highpass filter band edge and ω_p is the lowpass filter cutoff frequency, then the lowpass-to-highpass transformation function is given by

$$g(z) = -\frac{z + \alpha}{\alpha z + 1}, \quad (6.145)$$

where

$$\alpha = -\frac{\cos[(\omega_p + \omega_{p_1})/2]}{\cos[(\omega_p - \omega_{p_1})/2]}. \quad (6.146)$$

6.4.3 Lowpass-to-bandpass transformation

The lowpass-to-bandpass transformation is accomplished if the following mappings occur:

$$g(1) = -1 \quad (6.147)$$

$$g(e^{-j\omega_{p_1}}) = e^{j\omega_p} \quad (6.148)$$

$$g(e^{j\omega_{p_2}}) = e^{j\omega_p} \quad (6.149)$$

$$g(-1) = -1, \quad (6.150)$$

where ω_{p_1} and ω_{p_2} are the band edges of the bandpass filter and ω_p is the band edge of the lowpass filter. Since the bandpass filter has two passband edges, we need a second-order function $g(z)$ to accomplish the lowpass-to-bandpass transformation. After some manipulation, it can be inferred that the required transformation and its parameters are given by (Constantinides, 1970)

$$g(z) = -\frac{z^2 + \alpha_1 z + \alpha_2}{\alpha_2 z^2 + \alpha_1 z + 1}, \quad (6.151)$$

with

$$\alpha_1 = -\frac{2\alpha k}{k + 1} \quad (6.152)$$

$$\alpha_2 = \frac{k - 1}{k + 1}, \quad (6.153)$$

where

$$\alpha = \frac{\cos[(\omega_{p_2} + \omega_{p_1})/2]}{\cos[(\omega_{p_2} - \omega_{p_1})/2]} \quad (6.154)$$

$$k = \cot[(\omega_{p_2} - \omega_{p_1})/2] \tan(\omega_p/2). \quad (6.155)$$

6.4.4 Lowpass-to-bandstop transformation

The lowpass-to-bandstop transformation function $g(z)$ is given by

$$g(z) = \frac{z^2 + \alpha_1 z + \alpha_2}{\alpha_2 z^2 + \alpha_1 z + 1}, \quad (6.156)$$

with

$$\alpha_1 = -\frac{2\alpha}{k+1} \quad (6.157)$$

$$\alpha_2 = \frac{1-k}{1+k}, \quad (6.158)$$

where

$$\alpha = \frac{\cos[(\omega_{p_2} + \omega_{p_1})/2]}{\cos[(\omega_{p_2} - \omega_{p_1})/2]} \quad (6.159)$$

$$k = \tan[(\omega_{p_2} - \omega_{p_1})/2] \tan(\omega_p/2). \quad (6.160)$$

6.4.5 Variable-cutoff filter design

An interesting application for the frequency transformations, first proposed in Constantides (1970), is to design highpass and lowpass filters with variable cutoff frequency with the cutoff frequency being directly controlled by a single parameter α . This method can be best understood through Example 6.4.

Example 6.4. Consider the lowpass notch filter

$$H(z) = 0.004 \frac{z^2 - \sqrt{2}z + 1}{z^2 - 1.8z + 0.96} \quad (6.161)$$

whose zeros are located at $z = (\sqrt{2}/2)(1 \pm j)$. Transform this filter into a highpass notch with a zero at frequency $\omega_{p_1} = \pi/6$ rad/sample. Plot the magnitude responses before and after the frequency transformation.

Solution

Using the lowpass-to-highpass transformation given in Equation (6.145), the highpass transfer function is of the form

$$H(z) = H_0 \frac{(\alpha^2 + \sqrt{2}\alpha + 1)(z^2 + 1) + (\sqrt{2}\alpha^2 + 4\alpha + \sqrt{2})z}{(0.96\alpha^2 + 1.8\alpha + 1)z^2 + (1.8\alpha^2 + 3.92\alpha + 1.8)z + (\alpha^2 + 1.8\alpha + 0.96)} \quad (6.162)$$

with $H_0 = 0.004$. The parameter α can control the position of the zeros of the highpass notch filter. For instance, in this example, as the original zero is at $\omega_p = \pi/4$ rad/sample

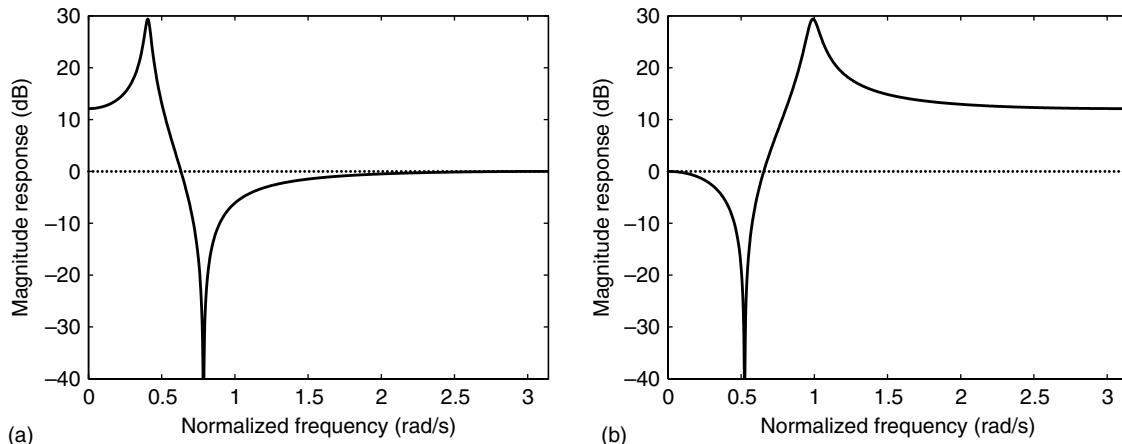


Fig. 6.11. Magnitude responses of notch filters: (a) lowpass notch filter; (b) highpass notch filter.

and the desired zero is at $\omega_{p1} = \pi/6$ rad/sample, the parameter α should be, as given in Equation (6.146), equal to

$$\alpha = -\frac{\cos[(\frac{\pi}{4} + \frac{\pi}{6})/2]}{\cos[(\frac{\pi}{4} - \frac{\pi}{6})/2]} = -0.8002. \quad (6.163)$$

The magnitude responses corresponding to the lowpass and highpass transfer functions are seen in Figure 6.11. Notice how the new transfer function has indeed a zero at the desired position. \triangle

6.5 Magnitude and phase approximation

In this section we discuss the approximation of IIR digital filters using optimization techniques aimed at the simultaneous approximation of the magnitude and phase responses. The same approach is useful in designing continuous-time filters and FIR filters. However, in the case of FIR filters, more efficient approaches exist, as we have seen in Sections 5.6.2 and 5.6.3.

6.5.1 Basic principles

Assume that $H(z)$ is the transfer function of an IIR digital filter. Then $H(e^{j\omega})$ is a function of the filter coefficients, which are usually grouped into a single vector γ , and of the independent variable $\theta = \omega$.

The frequency response of a digital filter can be expressed as a function of the filter parameters γ and θ , namely $F(\gamma, \theta)$, and a desired frequency response is usually referred to as $f(\theta)$.

The complete specification of an optimization problem involves: definition of an objective function (also known as a cost function), determination of the form of the transfer function $H(z)$ and its coefficients γ , and the solution methods for the optimization problem. These three items are further discussed below.

- *Choosing the objective function:* A widely used type of objective function in filter design is the weighted L_p norm, defined as (Deczky, 1972)

$$\|L(\gamma)\|_p = \left(\int_0^\pi W(\theta) |F(\gamma, \theta) - f(\theta)|^p d\theta \right)^{1/p} \quad (6.164)$$

where $W(\theta) > 0$ is the so-called weight function.

Problems based on the L_p -norm minimization criteria with different values of p lead, in general, to different solutions. An appropriate choice for the value of p depends on the type of error which is acceptable for the given application. For example, when we wish to minimize the mean-square value of the error between the desired and the designed responses, we should choose $p = 2$. Another problem is the minimization of the maximum deviation between the desired specification and the designed filter by searching the space of parameters. This case, which is known as the Chebyshev or minimax criterion, corresponds to $p \rightarrow \infty$. This important result derived from the optimization theory can be stated more formally as Theorem 6.1 (Deczky, 1972).

Theorem 6.1. *For a given coefficient space P and a given angle space X_θ , there is a unique optimal minimax approximation $F(\gamma_\infty^*, \theta)$ for $f(\theta)$. In addition, if the best L_p approximation for the function $f(\theta)$ is denoted by $F(\gamma_p^*, \theta)$, then it can be demonstrated that*

$$\lim_{p \rightarrow \infty} \gamma_p^* = \gamma_\infty^*. \quad (6.165)$$

◇

This result shows that we can use any minimization program based on the L_p norm to find a minimax (or approximately minimax) solution, by progressively calculating the L_p optimal solution with, for instance, $p = 2, 4, 6$, and so on, indefinitely.

Specifically, the minimax criterion for a continuous frequency function is best defined as

$$\|L(\gamma^*)\|_\infty = \min_{\gamma \in P} \max_{\theta \in X_\theta} \{W(\theta) |F(\gamma, \theta) - f(\theta)|\}. \quad (6.166)$$

In practice, due to several computational aspects, it is more convenient to use a simplified objective function given by

$$L_{2p}(\gamma) = \sum_{k=1}^K W(\theta_k) (F(\gamma, \theta_k) - f(\theta_k))^{2p}, \quad (6.167)$$

where, by minimizing $L_{2p}(\boldsymbol{\gamma})$, we also minimize $\|L(\boldsymbol{\gamma})\|_{2p}$. In this case, the minimax solution is obtained by minimizing $L_{2p}(\boldsymbol{\gamma})$, for $p = 1, 2, 3$, and so on, indefinitely.

The points θ_k are the angles chosen to sample the desired and the prototype frequency responses. These points, lying on the unit circle, do not need to be equally spaced. In fact, we usually choose θ_k such that there are denser grids in the regions where the error function has more variations.

The sort of filter designs described here can be applied to a large class of problems, in particular the design of filters with arbitrary magnitude response, phase equalizers, and filters with simultaneous specifications of magnitude and phase responses. The last two classes are illustrated below.

- *Phase equalizer*: The transfer function of a phase equalizer is (see Section 4.7.1, Equation (4.105))

$$H_l(z) = \prod_{i=1}^M \frac{a_{2i}z^2 + a_{1i}z + 1}{z^2 + a_{1i}z + a_{2i}}. \quad (6.168)$$

Since its magnitude is 1, the objective function becomes

$$L_{2p}\tau(\boldsymbol{\gamma}, \tau_0) = \sum_{k=1}^K W(\theta_k) (\tau_l(\boldsymbol{\gamma}, \theta_k) - \tau_s(\theta_k) + \tau_0)^{2p} \quad (6.169)$$

where τ_s is the group delay of the original digital filter, $\tau_l(\boldsymbol{\gamma}, \theta_k)$ is the equalizer group delay and τ_0 is a constant delay, whose value minimizes $\sum_{k=1}^K (\tau_s(\theta_k) - \tau_0)^{2p}$.

- *Simultaneous approximation of magnitude and group-delay responses*: For this type of approximation, the objective function can be given by

$$\begin{aligned} L_{2p,2qM,\tau}(\boldsymbol{\gamma}, \tau_0) &= \delta \sum_{k=1}^K W_M(\theta_k) (M(\boldsymbol{\gamma}, \theta_k) - f(\theta_k))^{2p} \\ &\quad + (1 - \delta) \sum_{r=1}^R W_\tau(\theta_k) (\tau(\boldsymbol{\gamma}, \theta_r) + \tau(\theta_r) - \tau_0)^{2p}, \end{aligned} \quad (6.170)$$

where $0 \leq \delta \leq 1$ and $\tau(\theta_r)$ is the group delay which we wish to equalize.

Usually, in the simultaneous approximation of magnitude and group-delay responses, the numerator of $H(z)$ is forced to have zeros on the unit circle or in reciprocal pairs, such that the group delay is a function of the poles of $H(z)$ only. The task of the zeros would be to shape the magnitude response.

- *Choosing the form of the transfer function*: One of the most convenient ways to describe an IIR $H(z)$ is the cascade form decomposition, because the filter stability can be easily tested and controlled. In this case, the coefficient vector $\boldsymbol{\gamma}$, according to Equation (4.32), is of the form

$$\boldsymbol{\gamma} = (\gamma'_{11}, \gamma'_{21}, m_{11}, m_{21}, \dots, \gamma'_{1i}, \gamma'_{2i}, m_{1i}, m_{2i}, \dots, H_0). \quad (6.171)$$

Unfortunately, the expressions for the magnitude and group delay of $H(z)$, as a function of the coefficients of the second-order sections, are very complicated. The same comment holds for the expressions of the partial derivatives of $H(z)$ with respect to the coefficients, which are also required in the optimization algorithm.

An alternative solution is to use the poles and zeros of the second-order sections represented in polar coordinates as parameters. In this case, the coefficient vector γ becomes

$$\gamma = (r_{z1}, \phi_{z1}, r_{p1}, \phi_{p1}, \dots, r_{zi}, \phi_{zi}, r_{pi}, \phi_{pi}, \dots, k_0) \quad (6.172)$$

and the magnitude and group-delay responses are expressed as

$$M(\gamma, \omega) = k_0 \prod_{i=1}^m \left\{ \frac{\left[1 - 2r_{zi} \cos(\omega - \phi_{zi}) + r_{zi}^2 \right]^{\frac{1}{2}}}{\left[1 - 2r_{pi} \cos(\omega - \phi_{pi}) + r_{pi}^2 \right]^{\frac{1}{2}}} \right\} \left\{ \frac{\left[1 - 2r_{zi} \cos(\omega + \phi_{zi}) + r_{zi}^2 \right]^{\frac{1}{2}}}{\left[1 - 2r_{pi} \cos(\omega + \phi_{pi}) + r_{pi}^2 \right]^{\frac{1}{2}}} \right\} \quad (6.173)$$

and

$$\begin{aligned} \tau(\gamma, \omega) = & \sum_{i=1}^N \left[\frac{1 - r_{pi} \cos(\omega - \phi_{pi})}{1 - 2r_{pi} \cos(\omega - \phi_{pi}) + r_{pi}^2} + \frac{1 - r_{pi} \cos(\omega + \phi_{pi})}{1 - 2r_{pi} \cos(\omega + \phi_{pi}) + r_{pi}^2} \right. \\ & \left. - \frac{1 - r_{zi} \cos(\omega - \phi_{zi})}{1 - 2r_{zi} \cos(\omega - \phi_{zi}) + r_{zi}^2} - \frac{1 - r_{zi} \cos(\omega + \phi_{zi})}{1 - 2r_{zi} \cos(\omega + \phi_{zi}) + r_{zi}^2} \right] \end{aligned} \quad (6.174)$$

respectively.

In an optimization problem such as this, the first- and second-order derivatives should be determined using closed-form formulas to speed up the convergence process. In fact, the use of numerical approximation to calculate such derivatives would make the optimization procedure too complex. The partial derivatives of the magnitude and group delay with respect to the radii and angles of the poles and zeros, which are required in the optimization processes, are given below:

$$\frac{\partial M}{\partial r_{zi}} = M(\gamma, \omega) \left[\frac{r_{zi} - \cos(\omega - \phi_{zi})}{1 - 2r_{zi} \cos(\omega - \phi_{zi}) + r_{zi}^2} + \frac{r_{zi} - \cos(\omega + \phi_{zi})}{1 - 2r_{zi} \cos(\omega + \phi_{zi}) + r_{zi}^2} \right] \quad (6.175)$$

$$\frac{\partial M}{\partial \phi_{zi}} = -M(\gamma, \omega) \left[\frac{r_{zi} \sin(\omega - \phi_{zi})}{1 - 2r_{zi} \cos(\omega + \phi_{zi}) + r_{zi}^2} - \frac{r_{zi} \sin(\omega + \phi_{zi})}{1 - 2r_{zi} \cos(\omega + \phi_{zi}) + r_{zi}^2} \right] \quad (6.176)$$

$$\frac{\partial \tau}{\partial r_{pi}} = \left\{ \frac{(1+r_{pi}^2) \cos(\omega - \phi_{pi}) - 2r_{pi}}{\left[1 - 2r_{pi} \cos(\omega - \phi_{pi}) + r_{pi}^2\right]^2} + \frac{(1+r_{pi}^2) \cos(\omega + \phi_{pi}) - 2r_{pi}}{\left[1 - 2r_{pi} \cos(\omega + \phi_{pi}) + r_{pi}^2\right]^2} \right\} \quad (6.177)$$

$$\frac{\partial \tau}{\partial \phi_{pi}} = \left\{ \frac{r_{pi}(1-r_{pi}^2) \sin(\omega - \phi_{pi})}{\left[1 - 2r_{pi} \cos(\omega - \phi_{pi}) + r_{pi}^2\right]^2} - \frac{r_{pi}(1-r_{pi}^2) \sin(\omega + \phi_{pi})}{\left[1 - 2r_{pi} \cos(\omega + \phi_{pi}) + r_{pi}^2\right]^2} \right\}. \quad (6.178)$$

We also need $\partial M / \partial r_{pi}$, $\partial M / \partial \phi_{pi}$, $\partial \tau / \partial r_{zi}$, and $\partial \tau / \partial \phi_{zi}$, which are similar to the expressions above. These derivatives are part of the expressions of the partial derivatives of the objective function with respect to the filter poles and zeros which are the derivatives used in the optimization algorithms employed. These derivatives are:

$$\frac{\partial L_{2p}M(\boldsymbol{\gamma})}{\partial r_{zi}} = \sum_{k=1}^K 2pW_M(\theta_k) \frac{\partial M}{\partial r_{zi}} (M(\boldsymbol{\gamma}, \theta_k) - f(\theta_k))^{2p-1} \quad (6.179)$$

and

$$\frac{\partial L_{2p}\tau(\boldsymbol{\gamma})}{\partial r_{zi}} = \sum_{k=1}^K 2pW_\tau(\theta_k) \frac{\partial \tau}{\partial r_{zi}} (\tau(\boldsymbol{\gamma}, \theta_k) - f(\theta_k))^{2p-1}. \quad (6.180)$$

Analogously, we need the expressions for $\partial L_{2p}M(\boldsymbol{\gamma}) / \partial \phi_{zi}$, $\partial L_{2p}M(\boldsymbol{\gamma}) / \partial r_{pi}$, $\partial L_{2p}M(\boldsymbol{\gamma}) / \partial \phi_{pi}$, $\partial L_{2p}\tau(\boldsymbol{\gamma}) / \partial \phi_{zi}$, $\partial L_{2p}\tau(\boldsymbol{\gamma}) / \partial r_{pi}$, and $\partial L_{2p}\tau(\boldsymbol{\gamma}) / \partial \phi_{pi}$, which are similar to the expressions given above.

It is important to note that we are interested only in generating stable filters. Since we are performing a search for the minimum of the error function on the parameter space Γ , the region in which the optimal parameter should be searched is a restricted subspace $\Gamma_s = \{\boldsymbol{\gamma} \mid r_{pi} < 1, \text{ for all } i\}$.

- *Choosing the optimization procedure:* There are several optimization methods suitable for solving the problem of filter approximation. Choosing the best method depends heavily on the designer's experience in dealing with this problem and on the available computer resources. The optimization algorithms used are such that they will converge only if the error function has a local minimum in the interior of the subspace Γ_s and not on the boundaries of Γ_s . In the present case, this is not a cause for concern because the magnitude and group delay of digital filters become large when a pole approaches the unit circle (Deczky, 1972), and, as a consequence, there is no local minimum corresponding to poles on the unit circle. In this manner, if we start the search from the interior of Γ_s (that is, with all the poles strictly inside the unit circle) and constrain our search to the subspace Γ_s , then a local minimum not located at the boundary of Γ_s will be reached for certain.

Owing to the importance of this step for setting up a procedure for designing IIR digital filters, for the sake of completeness and clarity of presentation, its discussion is left to the next subsection, which is devoted exclusively to it.

6.5.2 Multivariable function minimization method

An n -variable function $F(\mathbf{x})$ can be approximated by a quadratic function in a small region around a given operating point. For instance, in a region close to a point \mathbf{x}_k , we can write that

$$F(\mathbf{x}_k + \boldsymbol{\delta}_k) \approx F(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k)\boldsymbol{\delta}_k + \frac{1}{2}\boldsymbol{\delta}_k^T\mathbf{H}(\mathbf{x}_k)\boldsymbol{\delta}_k, \quad (6.181)$$

where

$$\mathbf{g}^T(\mathbf{x}_k) = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \dots, \frac{\partial F}{\partial x_n} \right] \quad (6.182)$$

is the gradient vector of $F(\mathbf{x})$ at the operating point \mathbf{x}_k and $\mathbf{H}(\mathbf{x}_k)$ is the Hessian matrix of $F(\mathbf{x})$, defined as

$$\mathbf{H}(\mathbf{x}_k) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}. \quad (6.183)$$

Clearly, if $F(\mathbf{x})$ is a quadratic function, then the right-hand side of Equation (6.181) is minimized when

$$\boldsymbol{\delta}_k = -\mathbf{H}^{-1}(\mathbf{x}_k)\mathbf{g}(\mathbf{x}_k). \quad (6.184)$$

If, however, the function $F(\mathbf{x})$ is not quadratic and the operating point is far away from a local minimum, then we can devise an algorithm which iteratively searches the minimum in the direction of $\boldsymbol{\delta}_k$ as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k = \mathbf{x}_k - \alpha_k \mathbf{H}^{-1}(\mathbf{x}_k)\mathbf{g}(\mathbf{x}_k), \quad (6.185)$$

where the convergence factor, α_k , is a scalar that minimizes, in the k th iteration, $F(\mathbf{x}_k + \boldsymbol{\delta}_k)$ in the direction of $\boldsymbol{\delta}_k$. There are several procedures for determining the value of α_k (Fletcher, 1980; Luenberger, 1984), which can be considered as two classes: exact and inexact line searches. As a general rule of thumb, an inexact line search should be used when the operating point is far from a local minimum, because in these conditions it is appropriate to trade accuracy for faster results. However, when the parameters approach a minimum, accuracy becomes an important issue, and an exact line search is the best choice.

The minimization procedure described above is widely known as the Newton method. The main drawbacks related to this method are the need for computation of the second-order derivatives of the objective function $F(\mathbf{x})$ with respect to the parameters in \mathbf{x} and the necessity of inverting the Hessian matrix.

For these two reasons, the most widely used methods for the solution of the simultaneous approximation of magnitude and phase are the so-called quasi-Newton methods (Fletcher, 1980; Luenberger, 1984). These methods are characterized by an attempt to build the inverse of the Hessian matrix, or an approximation of it, using the data obtained during the optimization process. The updated approximation of the Hessian inverse is used in each step of the algorithm in order to define the next direction in which to search for the minimum of the objective function.

A general structure of an optimization algorithm suitable for designing digital filters is given below, where \mathbf{P}_k is used as an estimate of the Hessian inverse.

(i) Algorithm initialization:

Set the iteration counter as $k = 0$.

Choose the initial vector \mathbf{x}_0 corresponding to a stable filter.

Use the identity as the first estimate of the Hessian inverse; that is, $\mathbf{P}_k = \mathbf{I}$.

Compute $F_0 = F(\mathbf{x}_0)$.

(ii) Convergence check:

Check if convergence was achieved by using an appropriate criterion. For example, a criterion would be to verify if $F_k < \epsilon$, where ϵ is a predefined error threshold. An alternative criterion is to verify that $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2 < \epsilon'$.

If the algorithm has converged, go to step (iv), otherwise go on to step (iii).

(iii) Algorithm iteration:

Compute $\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k)$.

Set $\mathbf{s}_k = -\mathbf{P}_k \mathbf{g}_k$.

Compute α_k that minimizes $F(\mathbf{x})$ in the direction of \mathbf{s}_k .

Set $\boldsymbol{\delta}_k = \alpha_k \mathbf{s}_k$.

Upgrade the coefficient vector, $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k$.

Compute $F_{k+1} = F(\mathbf{x}_{k+1})$.

Update \mathbf{P}_k , generating \mathbf{P}_{k+1} (see discussion below).

Increment k and return to step (ii).

(iv) Data output:

Display $\mathbf{x}^* = \mathbf{x}_k$ and $F^* = F(\mathbf{x}^*)$.

We should note that the way that the estimate \mathbf{P}_k of the Hessian inverse is updated was omitted from the above algorithm. In fact, what distinguishes the different quasi-Newton methods is solely the way that \mathbf{P}_k is updated. The most widely known quasi-Newton method is the Davidson–Fletcher–Powell method (Fletcher, 1980; Luenberger, 1984), used in Deczky (1972). Such an algorithm updates \mathbf{P}_k in the form

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T \Delta \mathbf{g}_k} - \frac{\mathbf{P}_k \Delta \mathbf{g}_k \Delta \mathbf{g}_k^T \mathbf{P}_k}{\Delta \mathbf{g}_k^T \mathbf{P}_k \Delta \mathbf{g}_k}, \quad (6.186)$$

where $\Delta \mathbf{g}_k = \mathbf{g}_k - \mathbf{g}_{k-1}$.

However, our experience has shown that the Broyden–Fletcher–Goldfarb–Shannon (BFGS) method (Fletcher, 1980) is more efficient. This algorithm updates \mathbf{P}_k in the form

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \left(1 + \frac{\Delta \mathbf{g}_k^T \mathbf{P}_k \Delta \mathbf{g}_k}{\Delta \mathbf{g}_k^T \boldsymbol{\delta}_k}\right) \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\Delta \mathbf{g}_k^T \boldsymbol{\delta}_k} - \frac{\boldsymbol{\delta}_k \Delta \mathbf{g}_k^T \mathbf{P}_k + \mathbf{P}_k \Delta \mathbf{g}_k \boldsymbol{\delta}_k^T}{\Delta \mathbf{g}_k^T \boldsymbol{\delta}_k}, \quad (6.187)$$

with $\Delta \mathbf{g}_k$ as before.

It is important to note that, in general, filter designers do not need to implement an optimization routine, as they can employ optimization routines already available in a number of computer packages. What the designers are required to do is to express the objective function and optimization problem in a way that can be input to the chosen optimization routine.

6.5.3 Alternative methods

Some methods in addition to phase equalization have been proposed for the simultaneous approximation of magnitude and group delay (Charalambous & Antoniou, 1980; Cortelazzo & Lightner, 1984; Saramäki & Neüvo, 1984).

The work of Charalambous & Antoniou (1980) presents a fast algorithm for phase equalization, satisfying a minimax criterion.

The work in Saramäki & Neüvo (1984) emphasizes the design of digital filters with a reduced number of multiplications. It employs an all-pole IIR filter in cascade with a linear-phase FIR filter. The IIR filter section is designed with the constraint of keeping the group delay equiripple, whereas the FIR filter is designed so that the overall cascade meets the desired magnitude response specifications. The work does not address any issue related to the speed of convergence of the algorithms involved. The reduction in the number of multipliers results from proper trading between the orders of the IIR and FIR filters. The drawback to this approach seems to be the large dynamic range of the resulting IIR and FIR filter internal signals, which may require very accurate coefficients in their implementation.

The work of Cortelazzo & Lightner (1984) presents a systematic procedure for the simultaneous approximation of magnitude and group delay based on the concept of multicriterion optimization. The minimization procedure searches for a solution satisfying the minimax criterion. In this work, the methodology is discussed, but no reference to the efficiency of the optimization algorithms is given. In fact, only low-order examples are presented due to the long computational time required to find a solution, indicating that this methodology will only be useful when a more efficient algorithm is proposed.

In recent years there has been great effort in this field to find specific methods tailored to design IIR digital filters which satisfy magnitude and phase specifications simultaneously. In Holford and Agathoklis (1996), for example, IIR filters with almost linear phase in the passband are designed by applying a technique called model reduction to the state-space form of an FIR filter prototype. This type of design leads to filters with high selectivity in the magnitude response, whereas the phase is kept almost linear. Another approach generalizes the traditional designs with prescribed magnitude and phase specifications based on L_p norms or minimax objective functions to filters with equiripple passbands and peak constrained stopbands (Sullivan & Adams, 1998; Lu, 1999).

Table 6.8.Characteristics of the initial bandpass filter. Gain constant: $H_0 = 0.0588$.

Filter zeros ($r_{Z_i}; \phi_{Z_i}$ (rad))	Filter poles ($r_{P_i}; \phi_{P_i}$ (rad))
$r_{Z_1} = 1.0; \phi_{Z_1} = 0.1740$	$r_{P_1} = 0.8182; \phi_{P_1} = 0.3030$
$r_{Z_2} = 1.0; \phi_{Z_2} = -0.1740$	$r_{P_2} = 0.8182; \phi_{P_2} = -0.3030$
$r_{Z_3} = 0.7927; \phi_{Z_3} = 0.5622$	$r_{P_3} = 0.8391; \phi_{P_3} = 0.4837$
$r_{Z_4} = 0.7927; \phi_{Z_4} = -0.5622$	$r_{P_4} = 0.8391; \phi_{P_4} = -0.4837$
$r_{Z_5} = 1.0; \phi_{Z_5} = 0.9022$	$r_{P_5} = 0.8346; \phi_{P_5} = 0.6398$
$r_{Z_6} = 1.0; \phi_{Z_6} = -0.9022$	$r_{P_6} = 0.8346; \phi_{P_6} = -0.6398$
$r_{Z_7} = 1.0; \phi_{Z_7} = 2.6605$	$r_{P_7} = 0.8176; \phi_{P_7} = 0.8053$
$r_{Z_8} = 1.0; \phi_{Z_8} = -2.6605$	$r_{P_8} = 0.8176; \phi_{P_8} = -0.8053$

Example 6.5. Design a bandpass filter satisfying the specifications below:

$$\left. \begin{array}{ll} M(\omega) = 1, & \text{for } 0.2\pi < \omega < 0.5\pi \\ M(\omega) = 0, & \text{for } 0 < \omega < 0.1\pi \text{ and } 0.6\pi < \omega < \pi \\ \tau(\omega) = L, & \text{for } 0.2\pi < \omega < 0.5\pi \end{array} \right\}, \quad (6.188)$$

where L is constant.

Solution

Since it is a simultaneous magnitude and phase approximation, the objective function is given by Equation (6.170), with the expressions for magnitude and group delay and their derivatives given in Equations (6.169)–(6.180). We can start the design with an eighth-order transfer function with the characteristics given in Table 6.8.

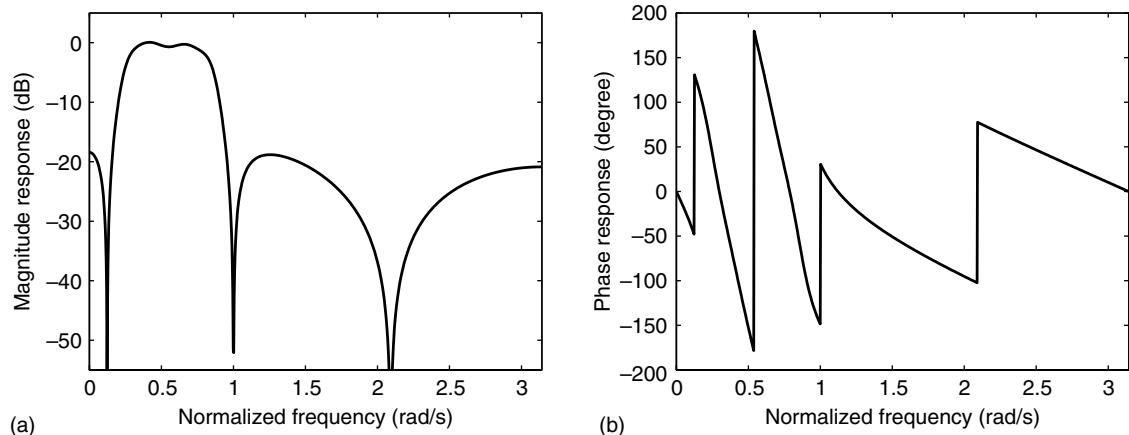
This initial filter is designed with the objective of approximating the desired magnitude specifications, and its average delay in the passband is used to estimate an initial value for L . In order to solve this optimization problem, we used a quasi-Newton program based on the BFGS method. Keeping the order of the starting filter at $n = 8$, we ran 100 iterations without obtaining noticeable improvements. We then increased the numerator and denominator orders by 2; that is, we made $n = 10$. After a few iterations the solution described in Table 6.9 was achieved.

Figure 6.12 illustrates the resulting frequency response. The attenuations at the first stopband edges are 18.09 dB and 18.71 dB. The attenuations at the second stopband edges are 18.06 dB and 19.12 dB. The passband attenuations at the edges are 0.69 dB and 0.71 dB, the two passband peaks have gains of 0.50 dB and 0.41 dB, whereas the attenuation at the passband minimum point is 0.14 dB. The group-delay values at the beginning of the passband, at the passband minimum, and at the end of the passband are 14.02 s, 12.09 s, and 14.38 s respectively. \triangle

Table 6.9.

Characteristics of the resulting bandpass filter. Gain constant: $H_0 = 0.058\,772\,50$.

Filter zeros ($r_{z_i}; \phi_{z_i}$ (rad))	Filter poles ($r_{p_i}; \phi_{p_i}$ (rad))
$r_{z_1} = 1.0; \phi_{z_1} = 0.1232$	$r_{p_1} = 0.0; \phi_{p_1} = 0.0$
$r_{z_2} = 1.0; \phi_{z_2} = -0.1232$	$r_{p_2} = 0.0; \phi_{p_2} = 0.0$
$r_{z_3} = 0.7748; \phi_{z_3} = 0.5545$	$r_{p_3} = 0.9072; \phi_{p_3} = 0.2443$
$r_{z_4} = 0.7748; \phi_{z_4} = -0.5545$	$r_{p_4} = 0.9072; \phi_{p_4} = -0.2443$
$r_{z_5} = 1.2907; \phi_{z_5} = 0.5545$	$r_{p_5} = 0.8654; \phi_{p_5} = 0.4335$
$r_{z_6} = 1.2907; \phi_{z_6} = -0.5545$	$r_{p_6} = 0.8654; \phi_{p_6} = -0.4335$
$r_{z_7} = 1.0; \phi_{z_7} = 1.0006$	$r_{p_7} = 0.8740; \phi_{p_7} = 0.6583$
$r_{z_8} = 1.0; \phi_{z_8} = -1.0006$	$r_{p_8} = 0.8740; \phi_{p_8} = -0.6583$
$r_{z_9} = 1.0; \phi_{z_9} = 2.0920$	$r_{p_9} = 0.9152; \phi_{p_9} = 0.8604$
$r_{z_{10}} = 1.0; \phi_{z_{10}} = -2.0920$	$r_{p_{10}} = 0.9152; \phi_{p_{10}} = -0.8604$



(a)

(b)

Fig. 6.12. Optimized bandpass filter: (a) magnitude response; (b) phase response.

6.6 Time-domain approximation

In some applications, time-domain specifications are given to the filter designer. In these cases the objective is to design a transfer function $H(z)$ such that the corresponding impulse response h_n is as close as possible to a given sequence g_n , for $n = 0, 1, \dots, (K - 1)$, where

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} = h_0 + h_1 z^{-1} + h_2 z^{-2} + \dots \quad (6.189)$$

Since $H(z)$ has $(M + N + 1)$ coefficients, if $K = (M + N + 1)$ there is at least one transfer function available which satisfies the specifications. This solution can be obtained through optimization, as follows.

By equating

$$H(z) = g_0 + g_1 z^{-1} + \cdots + g_{M+N} z^{-(M+N)} + \cdots \quad (6.190)$$

and considering the z -transform products as convolutions in the time domain, we can write, from Equations (6.189) and (6.190), that

$$\sum_{n=0}^N a_n g_{i-n} = \begin{cases} b_i, & \text{for } i = 0, 1, \dots, M \\ 0, & \text{for } i > M \end{cases}. \quad (6.191)$$

Now, assuming that $g_n = 0$, for all $n < 0$, this equation can be rewritten in matrix form as

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_M & g_{M-1} & g_{M-2} & \cdots & g_{M-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{M+N} & g_{M+N-1} & g_{M+N-2} & \cdots & g_M \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ a_N \end{bmatrix}, \quad (6.192)$$

which can be partitioned as

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_M & g_{M-1} & g_{M-2} & \cdots & g_{M-N} \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad (6.193)$$

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} g_{M+1} & \cdots & g_{M-N+1} \\ \vdots & \ddots & \vdots \\ g_{K-1} & \cdots & g_{K-N-1} \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad (6.194)$$

or

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{g}_2 \quad \mathbf{G}_3 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix}, \quad (6.195)$$

where \mathbf{g}_2 is a column vector and \mathbf{G}_3 is an $N \times N$ matrix. If \mathbf{G}_3 is nonsingular, then the coefficients \mathbf{a} are given by

$$\mathbf{a} = -\mathbf{G}_3^{-1} \mathbf{g}_2. \quad (6.196)$$

If \mathbf{G}_3 is singular of rank $R < N$, then there are infinite solutions, one of which is obtained by forcing the first $(N - R)$ entries of \mathbf{a} to be null.

With \mathbf{a} available, \mathbf{b} can be computed as

$$\mathbf{b} = \mathbf{G}_1 \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix}. \quad (6.197)$$

The main differences between the filters designed with different values of M and N , while keeping $K = (M + N + 1)$ constant, are the values of h_k , for $k > K$.

6.6.1 Approximate approach

A solution that is in general satisfactory is obtained by replacing the null vector in Equation (6.195) by a vector $\hat{\epsilon}$ whose magnitude should be minimized. In that manner, Equation (6.195) becomes

$$\begin{bmatrix} \mathbf{b} \\ \hat{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{g}_2 \\ \mathbf{G}_3 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix}. \quad (6.198)$$

Given the prescribed g_n and the values of N and M , we then have to find a vector \mathbf{a} such that $(\hat{\epsilon}^T \hat{\epsilon})$ is minimized, with

$$\hat{\epsilon} = \mathbf{g}_2 + \mathbf{G}_3 \mathbf{a}. \quad (6.199)$$

The value of \mathbf{a} which minimizes $(\hat{\epsilon}^T \hat{\epsilon})$ is the normal equation solution (Evans and Fischel, 1973):

$$\mathbf{G}_3^T \mathbf{G}_3 \mathbf{a} = -\mathbf{G}_3^T \mathbf{g}_2. \quad (6.200)$$

If the rank of \mathbf{G}_3 is N , then the rank of $\mathbf{G}_3^T \mathbf{G}_3$ is also N , and, therefore, the solution is unique, being given by

$$\mathbf{a} = -(\mathbf{G}_3^T \mathbf{G}_3)^{-1} \mathbf{G}_3^T \mathbf{g}_2. \quad (6.201)$$

On the other hand, if the rank of \mathbf{G}_3 is $R < N$, then we should force $a_i = 0$, for $i = 0, 1, \dots, (R - 1)$, as before, and redefine the problem as described in Burrus and Parks (1970).

It is important to point out that the procedure described above does not lead to a minimum squared error in the specified samples. In fact, the squared error is given by

$$\mathbf{e}^T \mathbf{e} = \sum_{n=0}^K (g_n - h_n)^2, \quad (6.202)$$

where g_n and h_n are the desired and obtained impulse responses respectively.

In order to obtain \mathbf{b} and \mathbf{a} which minimize $\mathbf{e}^T \mathbf{e}$, we need an iterative process, such as the one proposed in Evans and Fischel (1973). The time-domain approximation can also be formulated as a system identification problem, as addressed in Jackson (1996).

Example 6.6. Design a digital filter characterized by $M = 3$ and $N = 4$ such that its impulse response approximates the following sequence:

$$g_n = \frac{1}{3} \left[\frac{1}{4^{n+1}} + e^{-n-1} + \frac{1}{(n+2)} \right] u(n) \quad (6.203)$$

for $n = 0, 1, \dots, 7$.

Solution

Using $M = 3$ and $N = 4$, one gets

$$\mathbf{G}_1 = \begin{bmatrix} g_0 & 0 & 0 & 0 & 0 \\ g_1 & g_0 & 0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 & 0 \\ g_3 & g_2 & g_1 & g_0 & 0 \end{bmatrix} \quad (6.204)$$

$$\mathbf{g}_2 = [g_4 \ g_5 \ g_6 \ g_7]^T \quad (6.205)$$

$$\mathbf{G}_3 = \begin{bmatrix} g_3 & g_2 & g_1 & g_0 \\ g_4 & g_3 & g_2 & g_1 \\ g_5 & g_4 & g_3 & g_2 \\ g_6 & g_5 & g_4 & g_3 \end{bmatrix}. \quad (6.206)$$

As \mathbf{G}_3 is nonsingular, we can use Equations (6.199) and (6.197) to determine the transfer function

$$H(z) = \frac{0.3726z^3 - 0.6446z^2 + 0.3312z - 0.0466}{z^4 - 2.2050z^3 + 1.6545z^2 - 0.4877z + 0.0473}, \quad (6.207)$$

which has the exact desired impulse response for $n = 0, 1, \dots, 7$.

The impulse response corresponding to the transfer function above is depicted in Figure 6.13, together with the prescribed impulse response. As can be seen, the responses are the same in the first few iterations, and they become distinct for $n > 7$, as expected, because we have only eight coefficients to adjust. \triangle

6.7 Do-it-yourself: IIR filter approximations

Experiment 6.1

The elliptic bandstop filter specified in Example 6.3 can be readily designed in MATLAB as follows:

```
Ap = 0.5; Ar = 65;
wr1 = 850/5000; wr2 = 1150/5000;
wp1 = 980/5000; wp2 = 1020/5000;
```

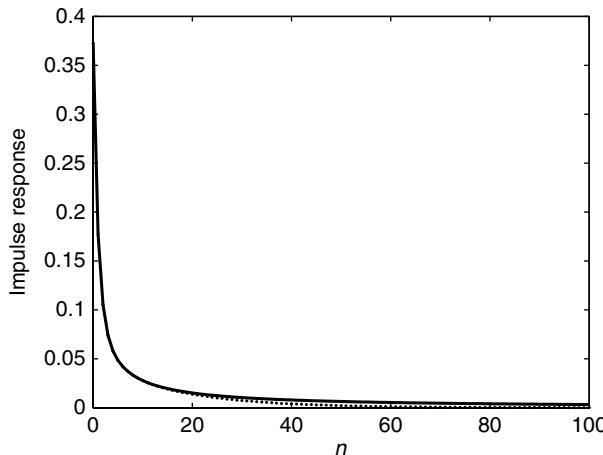


Fig. 6.13. Impulse responses: desired (solid line) and obtained (dotted line).

```
wp = [wp1 wp2]; wr = [wr1 wr2];
[n,wn] = ellipord(wp,wr,Ap,Ar);
[b,a] = ellip(n,Ap,Ar,wp);
```

In this script, commands `ellipord` and `ellip` require a frequency normalization such that the $\bar{\Omega}_s = 2$, thus explaining all divisions by $\Omega_s/2 = 5000$.

Similar Butterworth or Chebyshev filters can be designed using `butterord-butter` or `chebyord-cheby` commands, respectively.

The group delay response, determined with the `grpdelay` command, for the elliptic filter is seen in Figure 6.14. This figure indicates that two similar frequencies within the filter passband can suffer quite different delays. For instance, frequencies $f_1 = 980$ rad/s and $f_2 = 990$ rad/s are delayed in approximately 300 and 150 samples respectively, corresponding in this case to a difference of about

$$\Delta t = \frac{300 - 150}{F_s} = \frac{150}{\Omega_s/2\pi} = 94 \text{ ms.}$$

Figure 6.15 compares the input and output signals for each frequency f_1 and f_2 as determined by the following script:

```
Fs = 10000/(2*pi); Ts = 1/Fs; time = 0:Ts:(1-Ts);
f1 = 980; f2 = 990;
x1 = cos(f1.*time); y1 = filter(b,a,x1);
x2 = cos(f2.*time); y2 = filter(b,a,x2);
```

When the input signal presents a richer spectral component, this delay difference may cause severe distortion on the output signal. In such cases, a delay equalizer must be employed or the designer should opt for an FIR filter with perfectly linear phase. \triangle

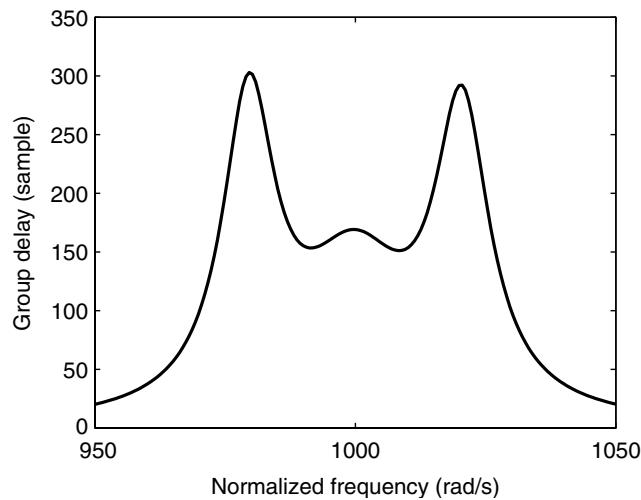


Fig. 6.14. Group delay response in passband of elliptic filter.

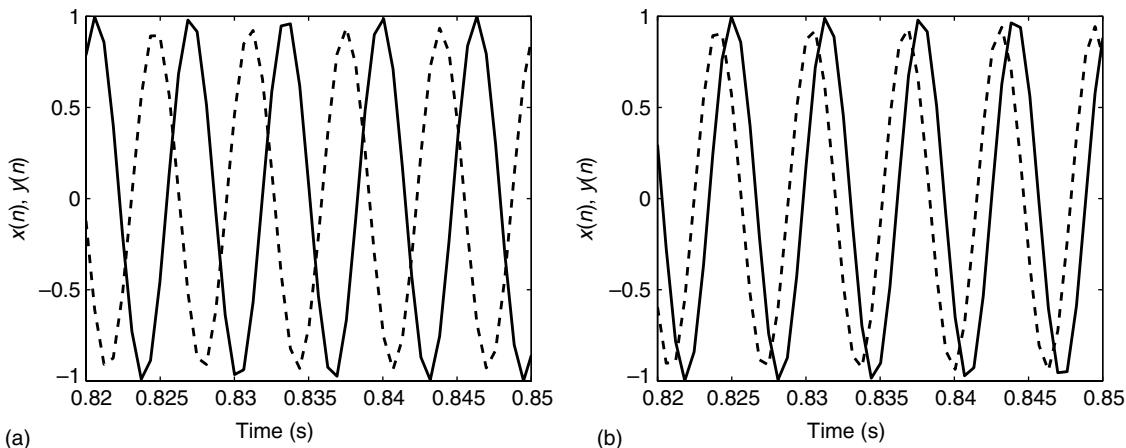


Fig. 6.15. Input (solid line) and output (dashed line) signals for elliptic bandpass filter: (a) $f_1 = 980$ rad/s; (b) $f_2 = 990$ rad/s.

Experiment 6.2

Consider the analog transfer function of the normalized-lowpass Chebyshev filter in Example 6.1, repeated here for convenience:

$$H_a(s) = 0.4913 \frac{1}{s^3 + 0.9883s^2 + 1.2384s + 0.4913}. \quad (6.208)$$

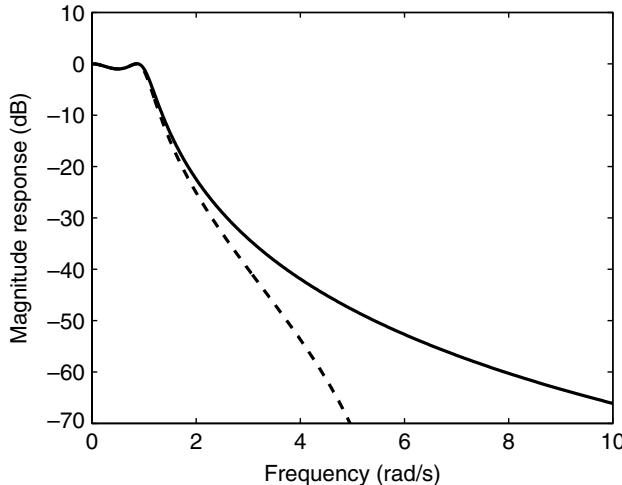


Fig. 6.16. Analog (solid line) and digital (dashed line) magnitude responses related by bilinear transformation method with $F_s = 2$ Hz.

The corresponding discrete-time transfer function $H(z)$ obtained with the bilinear transformation method with $F_s = 2$ Hz can be determined in MATLAB using the command lines

```
b = [0.4913]; a = [1 0.9883 1.2384 0.4913]; Fs = 2;
[bd,ad] = bilinear(b,a,Fs);
```

where `bd` and `ad` respectively receive the numerator and denominator coefficients of $H(z)$ such that

$$H(z) = \frac{0.0058(z^3 + 3z^2 + 3z + 1)}{z^3 - 2.3621z^2 + 2.0257z - 0.6175}. \quad (6.209)$$

The magnitude responses of $H_a(s)$ and $H(z)$ are shown in Figure 6.16.

Interestingly enough, the bilinear transformation can be implemented as a coefficient mapping between the two transfer functions. If we write

$$H_a(s) = \frac{\hat{b}_N s^n + \hat{b}^{N-1} s^{N-1} + \dots + \hat{b}_1 s + \hat{b}_0}{\hat{a}_N s^n + \hat{a}^{N-1} s^{N-1} + \dots + \hat{a}_1 s + \hat{a}_0} \quad (6.210)$$

$$H(z) = \frac{b_N z^n + b^{N-1} z^{N-1} + \dots + b_1 z + b_0}{a_N z^n + a^{N-1} z^{N-1} + \dots + a_1 z + a_0} \quad (6.211)$$

and define the coefficient vectors

$$\hat{\mathbf{a}} = [\hat{a}_N \ \hat{a}_{N-1} \ \dots \ \hat{a}_0]^T; \quad \hat{\mathbf{b}} = [\hat{b}_N \ \hat{b}_{N-1} \ \dots \ \hat{b}_0]^T \quad (6.212)$$

$$\mathbf{a} = [a_N \ a_{N-1} \ \dots \ a_0]^T; \quad \mathbf{b} = [b_N \ b_{N-1} \ \dots \ b_0]^T, \quad (6.213)$$

then one may write that (Pšenička *et al.*, 2002):

$$\mathbf{a} = \mathbf{P}_{N+1} \Delta_{N+1} \hat{\mathbf{a}} \quad (6.214)$$

$$\mathbf{b} = \mathbf{P}_{N+1} \Delta_{N+1} \hat{\mathbf{b}}, \quad (6.215)$$

where

$$\Delta_{N+1} = \text{diag} \left[\left(\frac{2}{T} \right)^N, \left(\frac{2}{T} \right)^{N-1}, \dots, \left(\frac{2}{T} \right), 1 \right] \quad (6.216)$$

and \mathbf{P}_{N+1} is an $(N + 1) \times (N + 1)$ Pascal matrix with the following properties:

- All elements in the first row are equal to 1.
- Elements of the first column are determined as

$$P_{i,1} = (-1)^{i-1} \frac{N!}{(N-i+1)!(i-1)!} \quad (6.217)$$

for $i = 1, 2, \dots, (N + 1)$.

- The remaining elements are given by

$$P_{i,j} = P_{i-1,j} + P_{i-1,j-1} + P_{i,j-1} \quad (6.218)$$

for $i, j = 2, 3, \dots, (N + 1)$.

In this experiment, since $F_s = 2$ and $N = 3$ we get

$$\mathbf{P}_{N+1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 3 & -1 & -1 & 3 \\ -1 & 1 & -1 & 1 \end{bmatrix}, \quad \Delta_{N+1} = \begin{bmatrix} 4^3 & 0 & 0 & 0 \\ 0 & 4^2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.219)$$

such that

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 64 & 16 & 4 & 1 \\ -192 & -16 & 4 & 3 \\ 192 & -16 & -4 & 3 \\ -64 & 16 & -4 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0.9883 \\ 1.2384 \\ 0.4913 \end{bmatrix} = \begin{bmatrix} 85.2577 \\ -201.3853 \\ 172.7075 \\ -52.6495 \end{bmatrix} \quad (6.220)$$

$$\begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 64 & 16 & 4 & 1 \\ -192 & -16 & 4 & 3 \\ 192 & -16 & -4 & 3 \\ -64 & 16 & -4 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.4913 \end{bmatrix} = \begin{bmatrix} 0.4913 \\ 1.4739 \\ 1.4739 \\ 0.4913 \end{bmatrix}, \quad (6.221)$$

which correspond to the same discrete-time transfer function as before, after proper normalization forcing $a_N = 1$. \triangle

6.8 IIR filter approximation with MATLAB

- **butter:** Designs Butterworth analog and digital filters.

Input parameters:

- The filter order n .
- The filter normalized (between 0 and 1) cutoff frequency wp , which is the frequency where the magnitude response assumes the value -3 dB. If such a parameter is a two-element vector $[w1, w2]$, then the command returns an order $2n$ digital bandpass filter with passband $w1 \leq \omega \leq w2$.
- The filter type specified by a string. The options are 'high' for a highpass filter, and 'stop' for a $2n$ th-order bandstop filter with stopband $w1 \leq \omega \leq w2$.
- For analog filters, add the string 's'.

There are three possibilities for output parameters. The choice is automatic depending on the number of parameters being requested:

- direct-form parameters $[b, a]$, where b is the vector of numerator coefficients and a is the vector of denominator coefficients;
- zero-pole parameters $[z, p, k]$, where z is the set of filter zeros, p is the set of filter poles, and k is the constant gain;
- state-space parameters $[A, B, C, D]$.

Example (direct-form highpass filter):

```
n=11; wp=0.2;
[b,a]=butter(n,wp,'high');
```

- **buttord:** Order selection for Butterworth filters.

Input parameters:

- passband edge frequency ω_p ;
- stopband edge frequency ω_r ;
- passband ripple in decibels A_p ;
- stopband attenuation in decibels A_r ;
- for analog filters, add the string 's'.

Output parameters:

- the filter order n ;
- the corresponding cutoff frequency for lowpass and highpass filters, and a pair of frequencies for bandpass and bandstop filters corresponding to the passband and stopband edges respectively.

Example:

```
wp=0.1; wr=0.15; Ap=1; Ar=20;
[n,wn]=buttord(wp,wr,Ap,Ar);
```

- **buttap:** Determines the analog prototype of a Butterworth lowpass filter.

Input parameter: the filter order n .

Output parameters: the pole-zero parameters $[z, p, k]$, where z is the set of filter zeros, p is the set of filter poles, and k is the constant gain.

Example:

```
n=9; [z,p,k]=buttap(n);
```

- **cheby1:** Designs Chebyshev analog and digital filters.

For input parameters, output parameters, and a similar example, see the command **butter**.

- **cheb1ord:** Order selection for Chebyshev filters.

For input parameters, output parameters, and a similar example, see the command **buttord**.

- **cheb1ap:** Determines the analog prototype of a Chebyshev lowpass filter.

For input and output parameters, see the command **buttap**. For Chebyshev filters, however, the passband ripple level should also be provided.

Example:

```
n=9; Ap=1.5;
[z,p,k]=cheb1ap(n,Ap);
```

- **cheby2:** Designs inverse Chebyshev analog and digital filters (see Exercise 6.19).

For input parameters, output parameters, and a similar example, see the command **butter**.

- **cheb2ord:** Order selection for inverse Chebyshev filters.

For input parameters, output parameters, and a similar example, see the command **buttord**.

- **cheb2ap:** Determines the analog prototype of an inverse Chebyshev lowpass filter.

For input and output parameters, see the command **buttap**. For inverse Chebyshev filters, however, the stopband attenuation level should also be provided.

Example:

```
n=11; Ar=30;
[z,p,k]=cheb2ap(n,Ar);
```

- **ellip:** Designs elliptic analog and digital filters.

Input parameters:

– The filter order n .

– The desired passband ripple A_p and minimum stopband attenuation A_r in decibels.

– The filter normalized cutoff frequency, $0 < w_p < 1$, which is the frequency value where the magnitude response assumes the value A_p dB. If such a parameter is

a two-element vector $[w_1, w_2]$, then the command returns a $2n$ th-order digital bandpass filter with passband $w_1 \leq \omega \leq w_2$.

- For analog filters, add the string 's'.

Output parameters: as for `butter` command.

Example:

```
n=5; Ap=1; Ar=25; wp=0.2;
[b,a]=ellip(n,Ap,Ar,wp);
```

- `ellipord`: Order selection for elliptic filters.

For input parameters, output parameters, and a similar example, see the command `buttord`.

- `ellipap`: Determines the analog prototype of an elliptic lowpass filter.

For input and output parameters, see the command `buttap`. For elliptic filters, however, the passband ripple and the stopband attenuation levels should also be provided.

Example:

```
n=9; Ap=1.5; Ar=30;
[z,p,k]=ellipap(n,Ap,Ar);
```

- `lp2lp`: Transforms a normalized analog lowpass filter prototype with cutoff frequency 1 rad/s into a lowpass filter with a given cutoff frequency.

Input parameters:

- direct-form coefficients `b`, `a`, where `b` is the vector of numerator coefficients and `a` is the vector of denominator coefficients; or state-space parameters `[A, B, C, D]`;
- desired normalized cutoff frequency in radians per second.

Output parameters: the direct-form coefficients `bt`, `at`; or the state-space parameters `[At, Bt, Ct, Dt]`.

Example (direct-form filter):

```
n=9; [z,p,k]=buttap(n);
b=poly(z)*k; a=poly(p); wp=0.3;
[bt,at]=lp2lp(b,a,wp);
```

- `lp2hp`: Transforms a normalized analog lowpass filter prototype with cutoff frequency 1 rad/s into a highpass filter with a given cutoff frequency.

For input and output parameters, see the command `lp2lp`.

Example:

```
n=9; [z,p,k]=buttap(n);
b=poly(z)*k; a=poly(p); wp=1.3;
[bt,at]=lp2hp(b,a,wp);
```

- `lp2bp`: Transforms a normalized analog lowpass filter prototype with cutoff frequency 1 rad/s into a bandpass filter with given center frequency and passband width.

For input and output parameters, see the command `1p2lp`. For bandpass filters, however, one must specify the center frequency and the passband width, instead of the desired cutoff frequency.

Example:

```
n=9; [z,p,k]=buttap(n);
b=poly(z)*k; a=poly(p); wc=1; Bw=0.2;
[bt,at]=lp2bp(b,a,wc,Bw);
```

- `1p2bs`: Transforms a normalized analog lowpass filter prototype with cutoff frequency 1 rad/s into a bandstop filter with given center frequency and stopband width.

For input parameters, output parameters, and a similar example, see the command `1p2bp`. For bandstop filters, however, one must specify the center frequency and the stopband width, instead of the passband width.

- `impinvar`: Maps the analog s plane into the digital z plane using the impulse-invariance method. It then transforms analog filters into their discrete-time equivalents, whose impulse response is equal to the analog impulse response scaled by a factor $1/F_s$.

Input parameters:

- direct-form coefficients b , a , where b is the vector of numerator coefficients and a is the vector of denominator coefficients;
- the scaling factor F_s , whose default value, if none is provided, is 1 Hz.

Output parameters: the direct-form coefficients bt and at .

Example:

```
n=9; [z,p,k]=buttap(n); b=poly(z)*k; a=poly(p); Fs=5;
[bt,at]=impinvar(b,a,Fs);
```

- `bilinear`: Maps the analog s plane into the digital z plane using the bilinear transformation. It then transforms analog filters into their discrete-time equivalents. This command requires the denominator order to be greater or equal to the numerator order.

Input parameters:

- direct-form coefficients b , a , where b is the vector of numerator coefficients and a is the vector of denominator coefficients; or zero-pole parameters $[z, p, k]$, where z is the set of filter zeros, p is the set of filter poles, and k is the constant gain; or state-space parameters $[A, B, C, D]$;
- the sampling frequency in hertz;
- a frequency, in hertz, for which the frequency response remains unchanged before and after the transformation.

Output parameters: the direct-form coefficients bt, at ; the zero-pole parameters $[z, p, k]$; or the state-space parameters $[A_t, B_t, C_t, D_t]$.

Example (direct-form filter with prewarping):

```
n=9; [z,p,k]=buttap(n);
b=poly(z)*k; a=poly(p); Fs=5; Fp=2;
[bt,at]=bilinear(b,a,Fs,Fp);
```

- `invfreqz`: Performs digital filter design using a given frequency response. By default, this command minimizes a quadratic (equation-error) functional that may lead to unstable solutions. The minimization of an output-error functional, similar to that given in Section 6.5.1, is performed, using the equation-error solution as an initial guess, when the parameters' maximum number of iterations or tolerance are provided by the user.

Input parameters:

- A vector with the desired complex frequency response and the corresponding frequency vector.
- The desired numerator and denominator orders.
- A weight vector for each frequency specified.
- A maximum number of iterations for the numerical algorithm to achieve convergence. This parameter forces the minimization of an output-error functional.
- A tolerance for convergence checking. This parameter forces the minimization of an output-error functional.

Output parameter: the direct-form coefficients b , a .

Example (Butterworth digital filter with $n = 2$ and $\omega_p = 0.2$):

```
n=2; b=[0.0675 0.1349 0.0675];
a=[1.0000 -1.1430 0.4128]; [h,w]=freqz(b,a,64);
[bt,at]=invfreqz(h,w,n,n);
```

- `invfreqs`: Performs analog filter design using a given frequency response. For input parameters, output parameters, and a similar example, see the command `invfreqz`.
- As mentioned in Chapter 5, MATLAB provides the command `sptool` that integrates most of the commands above into a unique interface that greatly simplifies the design of standard IIR digital filters, using several methods discussed previously.
- The interested reader may also refer to the MATLAB literature for the commands `lpc`, `maxflat`, `prony`, `stmcb`, and `yulewalk` for more specific design procedures of IIR digital filters.

6.9 Summary

In this chapter, we have covered the classical approximation methods for analog filters as well as two methods of transforming a continuous-time transfer function into a discrete-time transfer function. The transformation methods addressed were impulse-invariance and bilinear transformations. Although other transformation methods exist, those presented in this chapter are the most widely used in digital signal processing.

Transformation methods in the discrete-time domain were also addressed. It was shown that some of these transformations are useful in the design of variable-cutoff filters.

The simultaneous approximation of magnitude and phase responses was studied and an optimization procedure was described.

Then, the time-domain approximation problem was presented, and some methods aimed at the minimization of the mean-squared error between the prescribed impulse response and the resulting one were briefly discussed. Finally, hands-on experiments on designing IIR filters were presented in a Do-it-yourself section, followed by a summary of related MATLAB commands.

6.10 Exercises

- 6.1 Determine the normalized specifications for the analog, lowpass, Chebysev filter corresponding to the highpass filter:

$$\begin{aligned}A_p &= 0.2 \text{ dB} \\A_r &= 50 \text{ dB} \\\Omega_r &= 400 \text{ Hz} \\\Omega_p &= 440 \text{ Hz.}\end{aligned}$$

- 6.2 Determine the normalized specifications for the analog, lowpass, elliptic filter corresponding to the bandpass filter:

$$\begin{aligned}A_p &= 2 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_{r_1} &= 400 \text{ Hz} \\\Omega_{p_1} &= 500 \text{ Hz} \\\Omega_{p_2} &= 600 \text{ Hz} \\\Omega_{r_2} &= 700 \text{ Hz.}\end{aligned}$$

- 6.3 Design an analog elliptic filter satisfying the following specifications:

$$\begin{aligned}A_p &= 1.0 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_p &= 1000 \text{ Hz} \\\Omega_r &= 1209 \text{ Hz.}\end{aligned}$$

- 6.4 Design a lowpass Butterworth filter satisfying the following specifications:

$$\begin{aligned}A_p &= 0.5 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_p &= 100 \text{ Hz} \\\Omega_r &= 150 \text{ Hz} \\\Omega_s &= 500 \text{ Hz.}\end{aligned}$$

6.5 Design a bandstop elliptic filter satisfying the following specifications:

$$\begin{aligned}A_p &= 0.5 \text{ dB} \\A_r &= 60 \text{ dB} \\\Omega_{p_1} &= 40 \text{ Hz} \\\Omega_{r_1} &= 50 \text{ Hz} \\\Omega_{r_2} &= 70 \text{ Hz} \\\Omega_{p_2} &= 80 \text{ Hz} \\\Omega_s &= 240 \text{ Hz.}\end{aligned}$$

6.6 Design highpass Butterworth, Chebyshev, and elliptic filters that satisfy the following specifications:

$$\begin{aligned}A_p &= 1.0 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_r &= 5912.5 \text{ rad/s} \\\Omega_p &= 7539.8 \text{ rad/s} \\\Omega_s &= 50265.5 \text{ rad/s.}\end{aligned}$$

- 6.7 Design three bandpass digital filters, one with a central frequency of 770 Hz, a second of 852 Hz, and a third of 941 Hz. For the first filter, the stopband edges are at frequencies 697 and 852 Hz. For the second filter, the stopband edges are 770 and 941 Hz. For the third filter, the edges are at 852 and 1209 Hz. In all three filters, the minimum stopband attenuation is 40 dB and use $\Omega_s = 8 \text{ kHz}$.
- 6.8 Plot the zero-pole constellation for the three filters designed in Exercise 6.7 and visualize the resulting magnitude response in each case.
- 6.9 Create an input signal composed of three sinusoidal components of frequencies 770 Hz, 852 Hz, and 941 Hz, in MATLAB with $\Omega_s = 8 \text{ kHz}$. Use the three filters designed in Exercise 6.7 to isolate each component in a different signal.
- 6.10 The transfer function

$$H(s) = \frac{\kappa}{(s^2 + 1.4256s + 1.23313)(s + 0.6265)}$$

corresponds to a lowpass normalized Chebyshev filter with passband ripple $A_p = 0.5 \text{ dB}$.

- (a) Determine κ such that the filter gain at DC is 1.
- (b) Design a highpass digital filter with cutoff frequency $\omega_p = \frac{\pi}{3} \text{ rad/s}$, sampling frequency $\omega_s = \pi \text{ rad/s}$, and passband ripple of 0.5 dB using the bilinear transformation.
- (c) Suggest a possible realization for the resulting transfer function.

- 6.11 Transform the continuous-time highpass transfer function given by

$$H(s) = \frac{s^2}{s^2 + s + 1}$$

into a discrete-time transfer function using the impulse-invariance transformation method with $\Omega_s = 10$ rad/s. Plot the resulting analog and digital magnitude responses.

- 6.12 Repeat Exercise 6.11 using the bilinear transformation method and compare results from both exercises.
 6.13 Given the analog transfer function

$$H(s) = \frac{1}{(s^2 + 0.76722s + 1.33863)(s + 0.76722)},$$

design transfer functions corresponding to discrete-time filters using both the impulse-invariance method and the bilinear transformation. Choose $\Omega_s = 12$ rad/s. Compare the resulting frequency responses with the one from the analog filter.

- 6.14 Repeat Exercise 6.13 using $\Omega_s = 24$ rad/s and compare the results achieved in each case.
 6.15 Repeat Exercise 6.13 using MATLAB commands `impinvar` and `bilinear`.
 6.16 Determine the original analog transfer function corresponding to

$$H(z) = \frac{4z}{z - e^{-0.4}} - \frac{z}{z - e^{-0.8}},$$

assuming that the following method was employed in the analog to discrete-time mapping, with $T = 4$:

- (a) Impulse invariance method.
 (b) Bilinear transformation method.

- 6.17 Determine the original analog transfer function corresponding to

$$H(z) = \frac{2z^2 - (e^{-0.2} + e^{-0.4})z}{(z - e^{-0.2})(z - e^{0.4})},$$

assuming that the following method was employed in the analog to discrete-time mapping, with $T = 2$:

- (a) Impulse invariance method.
 (b) Bilinear transformation method.

- 6.18 Design a digital filter corresponding to the filter in Exercise 6.3, with $\Omega_s = 8$ kHz. Then transform the designed filter into a highpass filter satisfying the specifications of Exercise 6.6, using the frequency transformation of Section 6.4.
 6.19 This exercise describes the inverse Chebyshev approximation. The attenuation function of a lowpass inverse Chebyshev filter is characterized as

$$|A(j\Omega')|^2 = 1 + E(j\Omega')E(-j\Omega')$$

$$E(s')E(-s') = \frac{\epsilon'^2}{C_n^2(j/s')}$$

where $C_n(\Omega')$ is a Chebyshev function of order n and

$$\epsilon' = \sqrt{10^{0.14_r} - 1}.$$

The inverse Chebyshev approximation is maximally flat at $\Omega' = 0$ and has a number of transmission zeros at the stopband placed at the inverse of the roots of the corresponding Chebyshev polynomial. Using the equations above, the stopband edge is placed at $\Omega'_r = 1$ rad/s, and this property should be considered when applying denormalization.

- (a) Develop expressions for the passband edge, the transmission zeros, and the poles of a normalized filter of order n .
 - (b) Design the filter of Exercise 6.6 using the inverse Chebyshev approximation.
- 6.20 Show that the lowpass-to-bandpass and lowpass-to-bandstop transformations proposed in Section 6.4 are valid.
- 6.21 Apply the lowpass-to-highpass transformation to the filter designed in Exercise 6.4 and plot the resulting magnitude response.
- 6.22 Revisit Example 6.4, now forcing the highpass zero at $\omega_{p_1} = 2\pi/3$. Plot the magnitude responses before and after the transformation.
- 6.23 Given the transfer function

$$H(z) = 0.06 \frac{z^2 + \sqrt{2}z + 1}{z^2 - 1.18z + 0.94},$$

describe a frequency transformation to a bandpass filter with zeros at $\pi/6$ and $2\pi/3$.

- 6.24 Design a phase equalizer for the elliptic filter of Exercise 6.6 with the same order as the filter.
- 6.25 Design a lowpass filter satisfying the following specifications:

$$\begin{aligned} M(\Omega T) &= 1.0, & \text{for } 0.0\Omega_s < \Omega < 0.1\Omega_s \\ M(\Omega T) &= 0.5, & \text{for } 0.2\Omega_s < \Omega < 0.5\Omega_s \\ \tau(\Omega T) &= 4.0, & \text{for } 0.0\Omega_s < \Omega < 0.1\Omega_s. \end{aligned}$$

- 6.26 The desired impulse response for a filter is given by $g(n) = 1/2^n$. Design a recursive filter such that its impulse response $h(n)$ equals $g(n)$ for $n = 0, 1, \dots, 5$.
- 6.27 Plot and compare the magnitude responses associated with the ideal and approximated impulse responses in Exercise 6.26.
- 6.28 Design a filter with 10 coefficients such that its impulse response approximates the following sequence:

$$g_n = \left(\frac{1}{6^n} + 10^{-n} + \frac{0.05}{n+2} \right) u(n).$$

Choose a few key values for M and N , and discuss which choice leads to the smallest mean-squared error after the tenth sample.

- 6.29 Compare the magnitude responses associated with the filters designed in Exercise 6.28 for several values of M and N .

- 6.30 Repeat Experiment 6.2 using $F_s = 10$ Hz. Compare the magnitude response of the resulting discrete-time transfer function with the one obtained in the experiment.
- 6.31 Determine the Pascal matrix \mathbf{P}_{N+1} defined in Experiment 6.2 for $N = 4$ and $N = 5$.
- 6.32 Design an IIR digital filter to reduce the amount of noise in the two sinusoidal components in Experiment 1.3. Evaluate your specifications by processing `x_noisy`, as defined in that experiment, with the designed filter and verifying the output signal-to-noise ratio.

7.1 Introduction

In previous chapters we were introduced to some design techniques for FIR and IIR digital filters. Some of these techniques can also be used in other applications related to the general field of digital signal processing. In the present chapter we consider the very practical problem of estimating the power spectral density (PSD) of a given discrete-time signal $y(n)$. This problem appears in several applications, such as radar/sonar systems, music transcription, speech modeling, and so on. In general, the problem is often solved by first estimating the autocorrelation function associated with the data at hand, followed by a Fourier transform to obtain the desired spectral description of the process, as suggested by the Wiener–Khinchin theorem to be described in this chapter.

There are several algorithms for performing spectral estimation. Each one has different characteristics with respect to computational complexity, precision, frequency resolution, or other statistical aspects. We may classify all algorithms as nonparametric or parametric methods. Nonparametric methods do not assume any particular structure behind the available data, whereas parametric schemes consider that the process follows some pattern characterized by a specific set of parameters pertaining to a given model. In general, parametric approaches tend to be simpler and more accurate, but they depend on some a priori information regarding the problem at hand.

This chapter is organized as follows: Section 7.2 presents basic concepts of estimation theory that are used to characterize the nonparametric methods presented in Section 7.3, including the periodogram algorithm, its many variations, and the minimum-variance method, which is based on a concept of estimating the power spectrum around any prescribed frequency as in a zoom operation. Section 7.4 introduces the general theory of system modeling, characterizing the autocorrelation function for distinct system classes by the so-called Yule–Walker equations. Section 7.5 focuses on the PSD estimation for autoregressive systems, including the so-called covariance, autocorrelation, and Burg methods. Section 7.6 maps the linear prediction method onto a more general problem of finding a deterministic relationship between two stochastic signals, whose solution is called a Wiener filter. The Wiener solution finds application beyond the realm of spectral estimation. Some discussions about more advanced methods of spectral estimation not covered in this book are presented in Section 7.7. The chapter then closes with a Do-it-yourself section describing the PSD estimation of a synthetic signal.

7.2 Estimation theory

The problem of estimation is commonly classified in two groups: we refer to the classic estimation problem, when we are attempting to determine the value of a fixed (deterministic) unknown value; if we are attempting to estimate some statistic of a random parameter, then the problem is referred to as Bayesian estimation.

Consider the classic estimation problem. Let Θ be a real-valued parameter to be estimated from the available set of data $\mathbf{y} = \{y(0), y(1), \dots, y(L-1)\}$ associated with a random process $\{Y\}$.

The bias $B(\hat{\Theta})$ of an estimate $\hat{\Theta}$ of a deterministic parameter Θ is defined as

$$B(\hat{\Theta}) = E\{\hat{\Theta}\} - \Theta. \quad (7.1)$$

If $B(\hat{\Theta}) = 0$, then the estimate $\hat{\Theta}$ is called unbiased, otherwise it is referred to as biased. Other important characteristics of an estimator are its variance, standard deviation, and mean squared error (MSE), respectively defined as

$$\text{var}\{\hat{\Theta}\} = E \left\{ (\hat{\Theta} - E\{\hat{\Theta}\})^2 \right\} \quad (7.2)$$

$$\sigma_{\hat{\Theta}} = \sqrt{\text{var}\{\hat{\Theta}\}} \quad (7.3)$$

and

$$\text{MSE}\{\hat{\Theta}\} = E \left\{ (\hat{\Theta} - \Theta)^2 \right\}. \quad (7.4)$$

It is straightforward to show that

$$\text{MSE}\{\hat{\Theta}\} = \text{var}\{\hat{\Theta}\} + B^2(\hat{\Theta}) \quad (7.5)$$

in such a way that for an unbiased estimate we have that

$$\text{MSE}\{\hat{\Theta}\} = \text{var}\{\hat{\Theta}\}. \quad (7.6)$$

An estimate $\hat{\Theta}$ is called consistent if it converges (in probability) to the true parameter value, namely

$$\lim_{L \rightarrow \infty} \text{Prob}\{|\hat{\Theta} - \Theta| > \epsilon\} = 0, \quad (7.7)$$

where ϵ is a small positive number, with the associated variance also converging to zero. Consistency evaluates the estimator performance in the limiting case of sufficiently large L and is a desirable feature that an estimator should have.

In general, we aim at an unbiased estimator with small variance or, equivalently, with small MSE, although there is another viewpoint where allowing a bias in order to achieve the minimum MSE, can be better than an unbiased minimum variance estimator (Kay & Eldar, 2008). In practice, however, we can reduce bias at the cost of increasing the variance, and vice versa. The natural way of reducing both of them simultaneously is increasing the amount of data L . It can be shown that the variance of any unbiased estimate is bounded from below by the so-called Cramer–Rao lower bound (Van Trees, 1968) given by

$$\text{var}\{\hat{\Theta}\} \geq \frac{1}{E\{(\partial \ln p(\mathbf{y}, \Theta)/\partial \Theta)^2\}}, \quad (7.8)$$

where $p(\mathbf{y}, \Theta)$ is the probability density of the observations \mathbf{y} including its dependence on Θ , also known as the likelihood function of Θ . An unbiased estimate is referred to as efficient if its variance achieves the Cramer–Rao limit. In that sense, we say that the estimate uses all available data in an efficient manner. It should be stressed that we cannot always guarantee the existence of an efficient estimator for a particular problem.

7.3 Nonparametric spectral estimation

The nonparametric spectral estimation relies on the Wiener–Khinchin theorem that defines the relationship between the autocorrelation function of a process and its Fourier transform, which corresponds to the PSD of the process. This section explains how to estimate the PSD or the autocorrelation sequence of an ergodic WSS process $\{Y\}$ from a limited number of measured data. Note that such an estimate is actually a random process $\{P\}$, distinct from the original random process $\{Y\}$. In practice, whenever we perform an estimate we compute a realization of this random process $\{P\}$ using the available realization of the random process $\{Y\}$. To emphasize this, we will use lowercase letters $y(n)$ to represent the samples of the process $\{Y\}$. In addition, it is always interesting to analyze the statistical characteristics (bias, variance, etc.) of our estimating process $\{P\}$.

7.3.1 Periodogram

The periodogram PSD estimator for the process $\{Y\}$ is defined by

$$\hat{\Gamma}_{Y,P}(e^{j\omega}) = \frac{1}{L} \left| \sum_{n=0}^{L-1} y(n)e^{-j\omega n} \right|^2, \quad (7.9)$$

where the subscript P comes from the name of the associated estimator. This is equivalent to using a rectangular window on the signal $y(n)$ for the time interval $0 \leq n \leq (L-1)$, squaring the absolute value of the Fourier transform of the truncated sequence, and normalizing the

result by a factor L , to obtain a density measure of the spectral power. For $y(n)$ real, a simple algebraic development of Equation (7.9) yields

$$\begin{aligned}
 \hat{\Gamma}_{Y,P}(e^{j\omega}) &= \frac{1}{L} \left(\sum_{m=0}^{L-1} y(m)e^{-j\omega m} \right) \left(\sum_{n=0}^{L-1} y(n)e^{j\omega n} \right) \\
 &= \frac{1}{L} \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} y(m)y(n)e^{-j\omega(m-n)} \\
 &= \frac{1}{L} \left[y(0)y(L-1)e^{-j\omega(-L+1)} \right. \\
 &\quad + (y(0)y(L-2) + y(1)y(L-1)) e^{-j\omega(-L+2)} \\
 &\quad + \cdots + (y^2(0) + y^2(1) + \cdots + y^2(L-1)) e^{-j\omega(0)} + \cdots \\
 &\quad + (y(L-1)y(1) + y(L-2)y(0)) e^{-j\omega(L-2)} \\
 &\quad \left. + y(L-1)y(0)e^{-j\omega(L-1)} \right] \\
 &= \sum_{v=-L+1}^{L-1} \hat{R}_{Y,b}(v)e^{-j\omega v}, \tag{7.10}
 \end{aligned}$$

with

$$\hat{R}_{Y,b}(v) = \frac{1}{L} \sum_{n=0}^{L-1-|v|} y(n)y(n+|v|) \tag{7.11}$$

for $v = -(L-1), -(L-2), \dots, 0, \dots, (L-2), (L-1)$, and the subscript b standing for biased. Therefore, one may interpret that the periodogram estimator is based on the Wiener–Kinchin theorem applied to the estimated autocorrelation function given in Equation (7.11).

By taking the expected value of Equation (7.11) and considering that $\{Y\}$ is WSS, we get

$$E \left\{ \hat{R}_{Y,b}(v) \right\} = \frac{1}{L} \sum_{n=0}^{L-1-|v|} E \{ y(n)y(n+|v|) \} = \frac{L-|v|}{L} R_Y(v). \tag{7.12}$$

Hence, the autocorrelation estimation associated with the periodogram method is, on average, the result of multiplying the Bartlett window

$$w_B(v) = \begin{cases} \frac{L-|v|}{L}, & \text{if } |v| \leq (L-1) \\ 0, & \text{otherwise} \end{cases} \tag{7.13}$$

to the true autocorrelation function $R_Y(v)$. In the frequency domain, the averaged periodogram PSD becomes the convolution of the true PSD function with the Fourier transform

$W_B(e^{j\omega})$ of the Bartlett window, that is

$$E\{\hat{\Gamma}_{Y,P}(e^{j\omega})\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_B(e^{j(\omega-\psi)}) \Gamma_Y(e^{j\psi}) d\psi \quad (7.14)$$

where

$$W_B(e^{j\omega}) = \frac{1}{L} \left[\frac{\sin(\omega L/2)}{\sin(\omega/2)} \right]^2. \quad (7.15)$$

An important characteristic of the periodogram, as given in Equation (7.9), is that it is readily implemented with the FFT algorithm presented in Section 3.5. For finite L , the autocorrelation estimation $\hat{R}_{Y,b}(\nu)$ defined in Equation (7.11) is clearly biased for $\nu \neq 0$, with the bias increasing for larger values of $|\nu|$. In addition, $\hat{R}_{Y,b}(\nu)$ becomes zero for all lags such that $|\nu| > (L - 1)$. The variance of $\hat{R}_{Y,b}(\nu)$ also tends to increase with $|\nu|$, since the averaging sum is performed for fewer values of n .

If the $\{Y\}$ process is a white noise, such that $\Gamma_Y(e^{j\omega}) = \sigma_Y^2$, the periodogram estimate is unbiased even for finite L , since

$$E\{\hat{\Gamma}_{Y,P}(e^{j\omega})\} = \frac{\sigma_Y^2}{2\pi} \int_{-\pi}^{\pi} W_B(e^{j(\omega-\psi)}) d\psi = \sigma_Y^2 w_B(0) = \sigma_Y^2 = \Gamma_Y(e^{j\omega}). \quad (7.16)$$

In general, however, it can be verified that the periodogram PSD is biased for finite L , unbiased in the limiting case $L \rightarrow \infty$, and presents a constant variance, regardless of the value of L (Kay, 1988), thus constituting a nonconsistent estimator.

7.3.2 Periodogram variations

A large set of data can be partitioned into L/K blocks of length K each, yielding several PSD estimates that can be averaged to decrease the variance associated with the periodogram algorithm. This approach, however, decreases the amount of data used in each estimate, decreasing the associated spectral resolution. In that sense, the averaged periodogram method exchanges variance for resolution in the resulting PSD estimate.

Another variation for the standard periodogram estimation employs a non-rectangular window function on the whole data set $\{y(0), y(1), \dots, y(L - 1)\}$ being processed. This emphasizes the amplitude of the PSD peaks, better detecting sinusoidal components in $\{Y\}$, but also widens these same peaks, which can make neighboring peaks appear to be seen as a single one.

One can avoid the bias in the periodogram autocorrelation by using $(L - |\nu|)$ instead of L in the denominator of Equation (7.11), yielding the new autocorrelation estimate

$$\hat{R}_{Y,u}(\nu) = \frac{1}{L - |\nu|} \sum_{n=0}^{L-1-|\nu|} y(n)y(n + |\nu|) \quad (7.17)$$

such that

$$E \left\{ \hat{R}_{Y,u}(\nu) \right\} = \frac{1}{L - |\nu|} \sum_{n=0}^{L-1-|\nu|} E \{ y(n)y(n+|\nu|) \} = R_Y(\nu), \quad (7.18)$$

where the subscript u stands for unbiased. Replacing $\hat{R}_{Y,b}(\nu)$ with $\hat{R}_{Y,u}(\nu)$ in the last line of Equation (7.10), however, may lead to negative values in the resulting PSD function (Kay, 1988). This can be overcome by introducing a window function $w(n)$, of length $(2K + 1)$ with $K < L$, in the computation of the new PSD estimate:

$$\hat{\Gamma}_{Y,BT}(e^{j\omega}) = \sum_{\nu=-K}^K w(\nu) \hat{R}_{Y,u}(\nu) e^{-j\omega\nu}, \quad (7.19)$$

which constitutes the so-called Blackman–Tukey (BT) spectral estimator. Letting $K \ll L$ removes the noisier samples of the autocorrelation function in the computation of $\hat{\Gamma}_{Y,BT}(e^{j\omega})$, decreasing the variance in the resulting estimator, at the expense of a slight bias increase originated from the fact that we are using less data.

In the attempt to avoid negative PSD values, the associated literature presents several window functions $w(n)$ for the BT spectral estimator, generating several different trade-offs between bias and variance.

Example 7.1. Consider a signal $y(n)$ formed by three sinusoidal components as given by

$$y(n) = \sin \left(2\pi \frac{f_1}{f_s} n \right) + \sin \left(2\pi \frac{f_2}{f_s} n \right) + 5 \sin \left(2\pi \frac{f_3}{f_s} n \right), \quad (7.20)$$

with $f_1 = 45$ Hz, $f_2 = 55$ Hz, and $f_3 = 75$ Hz, sampled at $f_s = 400$ samples/s during a time interval of 200 ms. Use the periodogram, averaged periodogram, periodogram with windowed data, and BT methods to estimate the PSD function for this signal.

Solution

The four periodogram-based methods described in this section were implemented to estimate the PSD of the provided signal $y(n)$. The resulting PSD estimates are shown in Figure 7.1, using the normalized frequency range, where the vertical dotted lines indicate the frequencies of the sinusoidal components. More details on the use of these methods using MATLAB may be found in the Do-it-yourself section at the end of this chapter. For this example the number of samples is $L = 80$ and, where required, $K = 20$. In Figure 7.1a it is verified that the standard method (Equations (7.10) and (7.11)) yields a slightly biased estimation, since the 45 and 55 Hz peaks are slightly off mark, with the neighboring sidelobes of the 75 Hz peak almost masking the 55 Hz peak. The averaged periodogram estimate, where we average four blocks of size $K = 20$, is shown in Figure 7.1b, where we clearly observe the resulting variance reduction and a loss in spectral resolution. The averaged periodogram can be shown to be asymptotically unbiased. However, since the number of

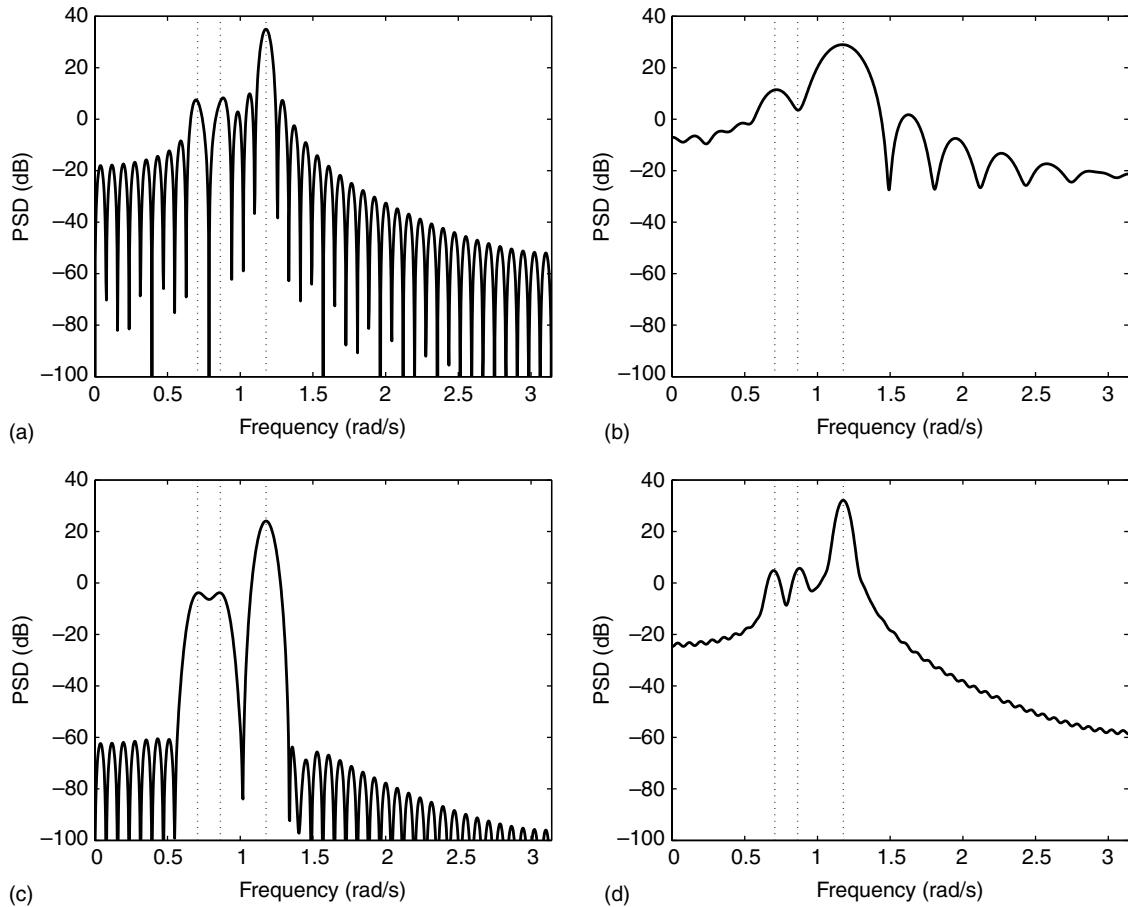


Fig. 7.1. PSD estimates for periodogram-based methods with true normalized frequencies indicated by vertical dotted lines: (a) standard periodogram; (b) averaged periodogram; (c) periodogram with windowed data; (d) BT method.

data samples utilized in each periodogram is smaller than $L = 80$, there is a loss of resolution in the order of $L/K = 4$, as expected. In the present case, a periodogram using the entire data record should have a resolution smaller than $2\pi/L$, which corresponds to a 5 Hz frequency resolution. Using $K = 20$, however, changes the resolution by a factor of four, to a 20 Hz resolution, explaining why the peak at 55 Hz is not observed in Figure 7.1b. The PSD estimate that applies a Hamming window to the data before calculating the auto-correlation function is able to avoid the masking effect, as seen in Figure 7.1c, but also widens the mainlobe of each peak, in this case almost merging the 45 and 55 Hz peaks into a single one. Finally, Figure 7.1d depicts the PSD estimate from the BT method, given in Equation (7.19), using the Hamming window directly on the estimated auto-correlation function, illustrating the excellent bias-variance compromise achieved by this technique. \triangle

7.3.3 Minimum-variance spectral estimator

In this subsection we derive yet another method to estimate the spectrum of a given signal based on an estimate of the signal power at arbitrary frequencies.

The nonparametric spectral estimator methods presented so far are based on the periodogram as described in Equation (7.9). This equation indicates that a rectangular window is applied to the signal $y(n)$ placed at the time interval $0 \leq n \leq (L - 1)$, then its Fourier transform is computed, followed by a scaled modulus calculation.

The minimum-variance approach estimates the spectrum at frequency ω_c by filtering the signal with a narrow bandpass filter centered at ω_c and estimating the power at the output of the filter. The filter should be optimized for having minimum energy outside the passband. Supposing that the impulse response of the filter with center frequency ω_c is $w_{\omega_c}^*(n)$ and has length L , we have that the output of the filter is

$$y_{MV,\omega_c}(n) = \sum_{l=0}^{L-1} w_{\omega_c}^*(l)y(n-l), \quad (7.21)$$

where the subscript MV stands for minimum variance. The reader should note that in order to derive the MV spectral estimator we allow $w_{\omega_c}(n)$ to be complex. Therefore, the power at the filter output is

$$E\{|y_{MV,\omega_c}(n)|^2\} = R_{MV,\omega_c}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |W_{\omega_c}^*(e^{-j\omega})|^2 \Gamma_Y(e^{j\omega}) d\omega, \quad (7.22)$$

where $W_{\omega_c}^*(e^{-j\omega})$ is the Fourier transform of the impulse response $w_{\omega_c}^*(n)$ of the passband filter centered at frequency ω_c . If the filter passband $\Delta\omega$ is narrow enough and has unit gain, then Equation (7.22) can be written as

$$E\{|y_{MV,\omega_c}(n)|^2\} \approx \frac{\Delta\omega}{2\pi} \Gamma_Y(e^{j\omega_c}), \quad (7.23)$$

which implies that the power of $y_{MV,\omega_c}(n)$ is proportional to the PSD of $y(n)$ around the frequency ω_c .

Our aim is to obtain an accurate estimate of $\Gamma_Y(e^{j\omega_c})$ for any value of ω_c , by cleverly designing the appropriate filter. The key idea is to minimize the power of $y_{MV,\omega_c}(n)$ over the entire frequency range $-\pi \leq \omega \leq \pi$, while keeping the filter gain at the arbitrary central frequency ω_c equal to one. In this way we are able to reduce as much as possible the power contributions from frequencies away from the central frequency. Therefore, the objective of the minimum-variance spectral estimator is to minimize

$$\xi_{\omega_c} = E\{|y_{MV,\omega_c}(n)|^2\} \quad (7.24)$$

subject to

$$\sum_{l=0}^{L-1} w_l^* e^{-j\omega_c l} = 1 \quad (7.25)$$

for a given $-\pi \leq \omega_c \leq \pi$, where we have made $w_{\omega_c}(l) = w_l$ in order to simplify the notation. By defining the auxiliary vectors

$$\mathbf{y}(n) = [y(n) \ y(n-1) \ \cdots \ y(n-L+1)]^T \quad (7.26)$$

$$\mathbf{w} = [w_0 \ w_1 \ \cdots \ w_{L-1}]^T \quad (7.27)$$

$$\mathbf{e}(e^{j\omega_c}) = \left[1 \ e^{-j\omega_c} \ e^{-j2\omega_c} \ \cdots \ e^{-j(L-1)\omega_c} \right]^T \quad (7.28)$$

one may incorporate the constraint into the objective function using a Lagrange multiplier λ , generating an equivalent problem that can be stated as the minimization of

$$\bar{\xi}_{\omega_c} = E\{\mathbf{w}^{*T} \mathbf{y}(n) \mathbf{y}^{*T}(n) \mathbf{w}\} + \lambda(\mathbf{w}^{*T} \mathbf{e}(e^{j\omega_c}) - 1), \quad (7.29)$$

where $[\cdot]^{*T}$ stands for complex conjugated and transposed.

The gradient of $\bar{\xi}_{\omega_c}$ with respect to \mathbf{w}^* should be equal to¹

$$\nabla_{\mathbf{w}^*} \bar{\xi}_{\omega_c} = \mathbf{R}_Y \mathbf{w} + \lambda \mathbf{e}(e^{j\omega_c}), \quad (7.30)$$

where $\mathbf{R}_Y = E[\mathbf{y}(n) \mathbf{y}^{*T}(n)]$. For a positive definite matrix \mathbf{R}_Y , the value of \mathbf{w} that satisfies $\nabla_{\mathbf{w}^*} \bar{\xi}_{\omega_c} = \mathbf{0}$ is unique and characterizes a minimum of $\bar{\xi}_{\omega_c}$. If we denote this optimal solution by $\tilde{\mathbf{w}}$, we have that

$$\mathbf{R}_Y \tilde{\mathbf{w}} + \lambda \mathbf{e}(e^{j\omega_c}) = \mathbf{0}. \quad (7.31)$$

Premultiplying the above equation by $\mathbf{e}^{*T}(e^{j\omega_c}) \mathbf{R}_Y^{-1}$, it follows that

$$\mathbf{e}^{*T}(e^{j\omega_c}) \tilde{\mathbf{w}} + \lambda \mathbf{e}^{*T}(e^{j\omega_c}) \mathbf{R}_Y^{-1} \mathbf{e}(e^{j\omega_c}) = \mathbf{0} \quad (7.32)$$

and then

$$\lambda = -\frac{1}{\mathbf{e}^{*T}(e^{j\omega_c}) \mathbf{R}_Y^{-1} \mathbf{e}(e^{j\omega_c})}, \quad (7.33)$$

considering that the constraint of Equation (7.25) holds for $\tilde{\mathbf{w}}$.

Therefore, according to Equation (7.31), the minimum-variance solution is given by

$$\tilde{\mathbf{w}} = \frac{1}{\mathbf{e}^{*T}(e^{j\omega_c}) \mathbf{R}_Y^{-1} \mathbf{e}(e^{j\omega_c})} \mathbf{R}_Y^{-1} \mathbf{e}(e^{j\omega_c}). \quad (7.34)$$

¹ In a real function with complex variables w and w^* one can treat these variables as independent, and the stationary point may be found by setting the derivative of the function with respect to w^* equal to zero. In the case the equality constraint is a function of w , the differentiation should be performed with respect to w , not w^* .

For this solution, the minimum value of the original objective function becomes

$$\begin{aligned}\xi_{\omega_c \min} &= \min \left\{ E\{|y_{MV, \omega_c}(n)|^2\} \right\} \\ &= E\{\tilde{\mathbf{w}}^{*T} \mathbf{y}(n) \mathbf{y}^{*T}(n) \tilde{\mathbf{w}}\} \\ &= \frac{1}{\mathbf{e}^{*T}(\mathbf{e}^{j\omega_c}) \mathbf{R}_Y^{-1} \mathbf{e}(\mathbf{e}^{j\omega_c})}. \end{aligned} \quad (7.35)$$

This solution is valid for any $\omega = \omega_c$, so that, according to the relationship (7.23), one gets

$$\hat{\Gamma}_{Y, MV}(\mathbf{e}^{j\omega}) = \frac{2\pi}{\Delta\omega} \xi_{\omega \min} = \frac{2\pi}{\Delta\omega \left(\mathbf{e}^{*T}(\mathbf{e}^{j\omega}) \mathbf{R}_Y^{-1} \mathbf{e}(\mathbf{e}^{j\omega}) \right)}. \quad (7.36)$$

Given that L is the window length, one may approximate the window bandwidth by

$$\Delta\omega \approx \frac{2\pi}{L} \quad (7.37)$$

and the minimum-variance spectrum estimator is determined by

$$\hat{\Gamma}_{Y, MV}(\mathbf{e}^{j\omega}) \approx \frac{L}{\mathbf{e}^{*T}(\mathbf{e}^{j\omega}) \mathbf{R}_Y^{-1} \mathbf{e}(\mathbf{e}^{j\omega})}. \quad (7.38)$$

Example 7.2. Repeat Example 7.1 using the MV method and comment on the results.

Solution

The MV method was implemented using the data length $L = 80$ and the biased autocorrelation function given in Equation (7.11), similar to Example 7.1. The resulting MV PSD estimate is shown in Figure 7.2, where we observe a very smooth behavior, characteristic of

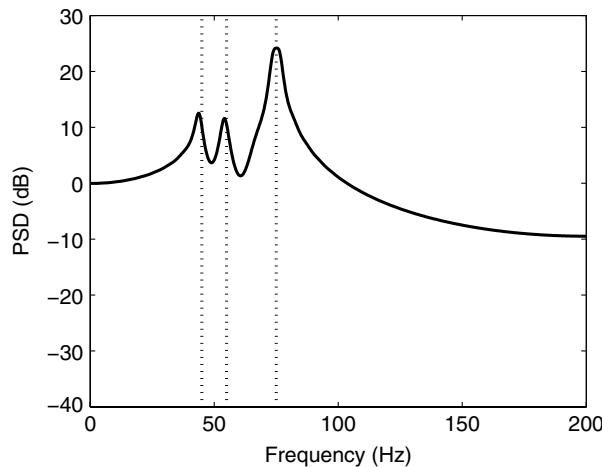


Fig. 7.2.

PSD estimate for MV method with true normalized frequencies indicated by vertical dotted lines.

the small-variance solution, with sharp peaks corresponding to the three sinusoidal components. However, one is still able to notice some significant bias on the estimated frequencies for the low-power 45 and 55 Hz sinusoids. \triangle

7.4 Modeling theory

In many applications one can use closed-form models to represent a given stochastic process. In such cases, the periodogram is not the method of choice since it does not yield any structural model about the process. This section describes some classical modeling tools to be employed in applications allowing parametric estimation.

7.4.1 Rational transfer-function models

The modeling problem refers to finding a compact description (usually as an input–output relationship) of a given process.

The so-called Wold decomposition theorem states that any WSS process can be expressed as the sum of a random process and a deterministic process. The deterministic component can be perfectly determined for all $n \geq 0$ based on the knowledge of its infinite past for $n < 0$.

The classic concept of system modeling consists of writing a WSS process as the output signal of a linear, time-invariant, and causal system to a white-noise input. In that sense, the modeling problem is commonly solved in a two-step procedure. The first step consists of choosing a particular input–output model that seems to fit the available data. Then, in a second stage, we determine the parameter values of the particular model previously chosen.

In the modeling context, FIR filters, which are characterized by the input–output relationship

$$y(n) = b_0x(n) + b_1x(n-1) + \cdots + b_Mx(n-M) \quad (7.39)$$

are commonly referred to as moving average (MA) systems, since the output can be seen as a weighted average of the input samples within the interval $n, (n-1), \dots, (n-M)$ that shifts with n .

Autoregressive (AR) models are characterized by a purely recursive input–output relationship described by

$$y(n) = x(n) - a_1y(n-1) - a_2y(n-2) - \cdots - a_Ny(n-N). \quad (7.40)$$

Combining the two models above, we get the so-called ARMA model described by

$$\begin{aligned} y(n) = & b_0x(n) + b_1x(n-1) + \cdots + b_Mx(n-M) \\ & - a_1y(n-1) - a_2y(n-2) - \cdots - a_Ny(n-N), \end{aligned} \quad (7.41)$$

which we associate with an IIR filter.

Taking Equations (7.39), (7.40), and (7.41) into the z -transform domain, we notice that the MA, AR, and ARMA are respectively associated with the following transfer functions:

$$\begin{aligned} H_{\text{MA}}(z) &= b_0 + b_1 z^{-1} + \cdots + b_M z^{-M} \\ &= \frac{b_0 z^M + b_1 z^{M-1} + \cdots + b_M}{z^M} \end{aligned} \quad (7.42)$$

$$\begin{aligned} H_{\text{AR}}(z) &= \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \\ &= \frac{z^N}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots + a_N} \end{aligned} \quad (7.43)$$

$$\begin{aligned} H_{\text{ARMA}}(z) &= \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \\ &= z^{N-M} \frac{b_0 z^M + b_1 z^{M-1} + \cdots + b_M}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots + a_N}. \end{aligned} \quad (7.44)$$

The MA, AR, and ARMA nomenclature originated in the field of system modeling and control, where the input signal is often assumed to be a white noise. Other than this assumption, we can readily interchange MA or ARMA models by FIR or IIR filters respectively, and AR models are to be considered special cases of IIR filters. In the associated literature, MA and AR systems are also referred to as all-zero and all-pole respectively. Note that these are somewhat misleading names that indicate the absence of poles or zeros in each case. This is so because Equations (7.39) and (7.40) do show that these models may present poles or zeros, accordingly, which are located at the origin of the complex plane.

Theorem 7.1 (System Decomposition Theorem). *Any stable ARMA system can be decomposed as a minimum-phase stable ARMA system cascaded to an all-pass system with constant-gain.*

In addition, any minimum-phase and stable ARMA model can be approximated by an infinite-order AR system (Oppenheim & Schafer, 1989). \diamond

Proof A minimum-phase stable system is one that has all its zeros and poles inside the unit circle. Now consider the zero–pole constellation for a general stable transfer function $H(z)$. In this case, all poles are inside the unit circle and the zeros can be anywhere in the complex plane. For each zero z_i outside the unit circle, multiply $H(z)$ by a factor $[1 - (1/z_i)z^{-1}]/[1 - (1/z_i)z^{-1}]$, which corresponds to a zero–pole pair inside the unit circle. Pairing the z_i zero with its inverse $1/z_i$ forms a constant-gain all-pass system. The

$1 - (1/z_i)z^{-1}$ term in the numerator corresponds to a zero within the unit circle. Therefore, the original transfer function can be written as the cascade of an all-pass system (combining the constant-gain contributions of all original zeros outside the unit circle and the corresponding reciprocal poles) and a minimum-phase stable ARMA system (which includes all original poles and minimum-phase zeros and the newly introduced minimum-phase zeros).

Let us now consider only the minimum-phase stable ARMA transfer function as

$$H(z) = b_0 \frac{\prod_{i=1}^M (1 - z_i z^{-1})}{\prod_{j=1}^N (1 - p_j z^{-1})} \quad (7.45)$$

with all $|z_i| < 1$ and $|p_j| < 1$.

Representing just the ARMA poles with an AR system is trivial. On the other hand, for each zero such that $|z_i| < 1$, using the summing formula for an infinite-length geometric series, we may write that

$$1 - z_i z^{-1} = \frac{1}{1 + \frac{z_i z^{-1}}{1 - z_i z^{-1}}} = \frac{1}{1 + \sum_{k=1}^{\infty} z_i^k z^{-k}}. \quad (7.46)$$

Hence, each minimum-phase zero can also be modeled as an AR system with infinite order and so can the complete ARMA model. It can also be demonstrated that the effect of the minimum-phase zero can be approximated with arbitrary precision if we truncate the above summation to a sufficient number of terms. This is verified in Example 7.3 below. \square

This result indicates that the above minimum-phase and stable ARMA–AR equivalence, except for an all-pass factor, applies to a large set of linear systems. A similar result follows for minimum-phase and stable ARMA systems and MA models, the demonstration of which is left as an end-of-chapter exercise for the interested reader. The ARMA–AR and ARMA–MA equivalences indicate that we do not need to be sure about which model to employ in a particular application. As indicated above, the price to be paid for employing an MA or an AR model instead of an ARMA model is that we may be forced to work with a high-order model.

Example 7.3. Approximate the ARMA system

$$H(z) = \frac{1 + 0.3z^{-1}}{1 - 0.9z^{-1}} \quad (7.47)$$

by an N th-order AR model.

Solution

Since the ARMA filter is minimum-phase, using Equation (7.46) we can write that

$$\begin{aligned}
 H(z) &= \frac{1}{(1 - 0.9z^{-1}) \left[1 + \sum_{k=1}^{\infty} (-0.3)^k z^{-k} \right]} \\
 &= \frac{1}{1 + \left[\sum_{k=1}^{\infty} (-0.3)^k z^{-k} \right] - 0.9z^{-1} - 0.9 \left[\sum_{k=1}^{\infty} (-0.3)^k z^{-(k+1)} \right]} \\
 &= \frac{1}{1 + \left[\sum_{k=1}^{\infty} (-0.3)^k z^{-k} \right] - 0.9 \left[\sum_{k=0}^{\infty} (-0.3)^k z^{-(k+1)} \right]} \\
 &= \frac{1}{1 + \left[\sum_{k=1}^{\infty} (-0.3)^k z^{-k} \right] - 0.9 \left[\sum_{k'=1}^{\infty} \frac{(-0.3)^{k'}}{(-0.3)} z^{-k'} \right]} \\
 &= \frac{1}{1 + \left[\sum_{k=1}^{\infty} (-0.3)^k z^{-k} \right] + 3 \left[\sum_{k'=1}^{\infty} (-0.3)^{k'} z^{-k'} \right]} \\
 &= \frac{1}{1 + 4 \sum_{k=1}^{\infty} (-0.3)^k z^{-k}}, \tag{7.48}
 \end{aligned}$$

which corresponds to an infinite-order AR system with denominator coefficients

$$a_i = 4(-0.3)^i \Rightarrow \begin{cases} a_1 = -1.2 \\ a_2 = 0.36 \\ a_3 = -0.108 \\ \vdots \end{cases}. \tag{7.49}$$

Truncating the last summation in Equation (7.48), we get a finite-order AR approximation

$$H(z) \approx \frac{1}{1 + 4 \sum_{k=1}^N (-0.3)^k z^{-k}}. \tag{7.50}$$

Figure 7.3 plots the magnitude response for the original ARMA system and the corresponding approximations for $N = 1, 2, 3$. For $N \geq 4$ the AR magnitude response becomes quite similar to the original one. \triangle

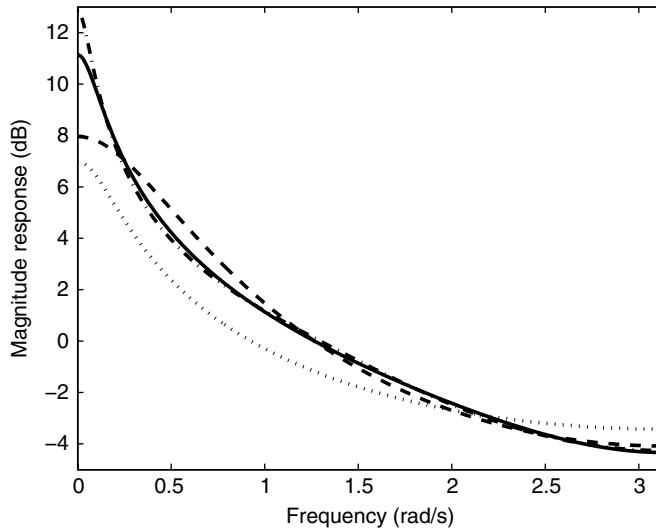


Fig. 7.3. Magnitude responses of original ARMA system (solid line) and AR approximations of orders $N = 1$ (dotted line), $N = 2$ (dashed line), and $N = 3$ (dash-dotted line).

7.4.2 Yule–Walker equations

For the MA model, multiplying Equation (7.39) by $y(n - \nu)$ and taking the expected value, we get

$$\begin{aligned} E\{y(n)y(n - \nu)\} &= E\left\{\sum_{j=0}^M b_j x(n-j)y(n-\nu)\right\} \\ &= \sum_{j=0}^M b_j E\{x(n-j)y(n-\nu)\}. \end{aligned} \quad (7.51)$$

The general term $E\{x(n-j)y(n-\nu)\}$ becomes null for $\nu > j$, since for a causal system the output at time $(n - \nu)$ is independent of the future input $x(n - j)$, and then

$$E\{x(n-j)y(n-\nu)\} = E\{x(n-j)\}E\{y(n-\nu)\} = 0, \quad (7.52)$$

assuming a zero-mean white noise $x(n)$. For $\nu \leq j$, however, we obtain

$$\begin{aligned} E\{x(n-j)y(n-\nu)\} &= E\left\{x(n-j)\sum_{l=0}^M b_l x(n-l-\nu)\right\} \\ &= \sum_{l=0}^M b_l E\{x(n-j)x(n-l-\nu)\} \\ &= b_{j-\nu}\sigma_X^2 \end{aligned} \quad (7.53)$$

for a white noise $x(n)$ with variance σ_X^2 . Substituting Equation (7.53) into (7.51), for the MA model, it is possible to show that

$$R_Y(v) = \begin{cases} \left(\sum_{j=v}^M b_j b_{j-v} \right) \sigma_X^2, & \text{for } v = 0, 1, \dots, M \\ 0, & \text{for } v > M \end{cases}. \quad (7.54)$$

For the AR model, multiplying Equation (7.40) by $y(n-v)$ and taking the expected value, we get

$$\begin{aligned} E\{y(n)y(n-v)\} &= E\{x(n)y(n-v)\} - E\left\{\sum_{i=1}^N a_i y(n-i)y(n-v)\right\} \\ &= E\{x(n)y(n-v)\} - \sum_{i=1}^N a_i E\{y(n-i)y(n-v)\} \end{aligned} \quad (7.55)$$

and then

$$R_Y(v) = \begin{cases} \sigma_X^2 - \sum_{i=1}^N a_i R_Y(v-i), & \text{for } v = 0 \\ - \sum_{i=1}^N a_i R_Y(v-i), & \text{for } v > 0, \end{cases} \quad (7.56)$$

since, for a causal AR model with a zero mean output, we have that (see Exercise 7.13)

$$E\{x(n)y(n-v)\} = \begin{cases} \sigma_X^2, & \text{for } v = 0 \\ 0, & \text{for } v > 0. \end{cases} \quad (7.57)$$

For the general ARMA model, a similar relationship can be determined (Kay, 1988):

$$R_Y(v) = \begin{cases} \left(\sum_{j=v}^M b_j h(j-v) \right) \sigma_X^2 - \sum_{i=1}^N a_i R_Y(v-i), & \text{for } v = 0, 1, \dots, M \\ - \sum_{i=1}^N a_i R_Y(v-i), & \text{for } v > M, \end{cases} \quad (7.58)$$

where $h(n)$ is the corresponding ARMA model impulse response.

Equations (7.54), (7.56), and (7.58), relating the output autocorrelation function to the model coefficients, are the so-called Yule–Walker equations for the MA, AR, and ARMA models respectively.

Example 7.4. Determine $R_Y(v)$ for the AR system

$$H(z) = \frac{1}{1 - 0.9z^{-1}} \quad (7.59)$$

assuming a unit-variance white-noise input.

Solution

For the given AR system, we use $N = 1$ in Equation (7.56), resulting in the following relationships for the correlation function:

$$R_Y(v) = \begin{cases} \sigma_X^2 - a_1 R_Y(v-1), & \text{for } v = 0 \\ -a_1 R_Y(v-1), & \text{for } v > 0 \end{cases} \quad (7.60)$$

such that, for $v = 0, 1$:

$$\begin{cases} R_Y(0) = \sigma_X^2 - a_1 R_Y(-1) \\ R_Y(1) = -a_1 R_Y(0) \end{cases}. \quad (7.61)$$

Since $R_Y(-1) = R_Y(1)$, we have that

$$R_Y(0) = \frac{\sigma_X^2}{1 - (-a_1)^2} \quad (7.62)$$

and then

$$R_Y(v) = \frac{\sigma_X^2 (-a_1)^{|v|}}{1 - (-a_1)^2}, \quad (7.63)$$

which, for $\sigma_X^2 = 1$ and $a_1 = -0.9$, is depicted in Figure 7.4. \triangle

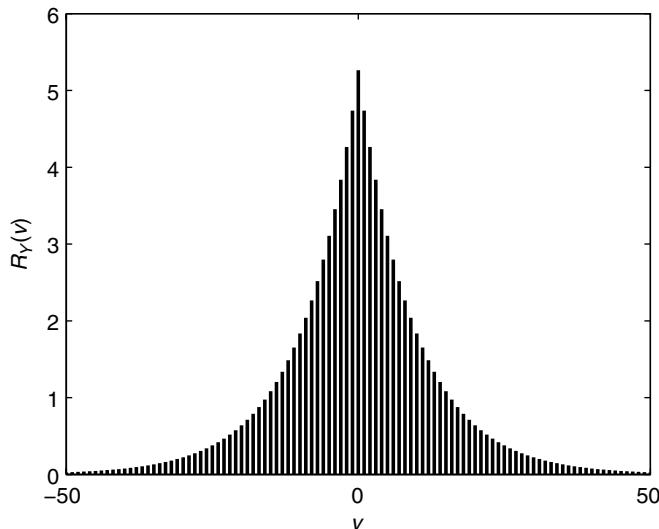


Fig. 7.4.

Autocorrelation function for AR system $H(z) = 1/(1 - 0.9z^{-1})$.

7.5 Parametric spectral estimation

The periodogram-based spectral estimators perform a primary estimation of the autocorrelation function according to the Wiener–Khinchin theorem. In such a case, the autocorrelation estimation is restricted to a lag interval delimited by the amount of available data. Outside that interval, the estimation becomes zero, leading to a biased PSD estimation.

Using parametric spectral estimation, we first model the random process at hand, and the resulting PSD estimate is obtained as the power spectrum of that model. The MA model also forces the estimated autocorrelation to become null for large lag values, as indicated by the Yule–Walker Equations (7.54).

Therefore, PSD estimation based on MA models presents similar properties to the periodogram-based methods. By employing an AR or ARMA model, however, we do not impose any null constraint on the resulting autocorrelation function. In fact, for these types of model, the autocorrelation function is automatically adjusted to better fit the statistical behavior of the available data. The result is a PSD estimation with better statistical properties. In addition, the nonparametric PSD methods require large amounts of data, whereas the parametric methods are more suitable to applications where the data length is short.

Among the three types of model seen before, the AR model presents interesting characteristics that suit very well the practical problem of system modeling, namely:

- It does not force the autocorrelation function to become zero for large lag values, as opposed to the MA model.
- Apart from an all-pass component, it can be used to approximate any stable ARMA system, as indicated by the system decomposition theorem.
- Its Yule–Walker equations consist of a linear relationship between the autocorrelation values and the model coefficients, as opposed to the general ARMA model.
- The resulting linear system presents a special structure that yields the solution via a simplified numerical algorithm.

All these aspects together justify the wide acceptance of AR modeling in practice, in particular for the PSD estimation problem.

7.5.1 Linear prediction

This subsection describes the classical approach, commonly referred to as linear prediction (LP), to estimate the parameters of an AR model for a particular data set. There are several ways of introducing the LP problem. We follow what we consider is the most didactic one. Later, we show that the following algebraic development is associated with the AR modeling problem with an interesting frequency-domain interpretation.

Consider that we have the knowledge of a set of discrete-time data $\{y(0), y(1), \dots\}$ from a particular process. This data may have come from measurements of a stock price, some bacteria population, or a speech signal, for instance. The idea behind the LP problem is to

estimate the value of $y(n)$ as a linear combination of N past samples of the process; that is:

$$\hat{y}(n) = \hat{a}_1 y(n-1) + \hat{a}_2 y(n-2) + \cdots + \hat{a}_N y(n-N) = \sum_{i=1}^N \hat{a}_i y(n-i), \quad (7.64)$$

where \hat{a}_i , for $i = 1, 2, \dots, N$, are the so-called LP coefficients and N is the LP model order. By comparing the LP estimation with the true signal value, we can form the estimation error

$$e(n) = y(n) - \hat{y}(n), \quad (7.65)$$

and the associated MSE is given by

$$\xi = E\{e^2(n)\}, \quad (7.66)$$

which, from Equations (7.64) and (7.65), constitutes a quadratic function of the LP coefficients \hat{a}_i . One may then determine that

$$\begin{aligned} \frac{\partial \xi}{\partial \hat{a}_i} &= \frac{\partial E\{e^2(n)\}}{\partial \hat{a}_i} \\ &= E \left\{ \frac{\partial e^2(n)}{\partial \hat{a}_i} \right\} \\ &= E \left\{ 2e(n) \frac{\partial e(n)}{\partial \hat{a}_i} \right\} \\ &= E \left\{ 2e(n) \frac{\partial \left(y(n) - \sum_{j=1}^N \hat{a}_j y(n-j) \right)}{\partial \hat{a}_i} \right\} \\ &= -2E\{e(n)y(n-i)\} \end{aligned} \quad (7.67)$$

and, by replacing $e(n)$ by its expression in Equation (7.65), it follows that

$$\begin{aligned} \frac{\partial \xi}{\partial \hat{a}_i} &= -2E \left\{ \left(y(n) - \sum_{j=1}^N \hat{a}_j y(n-j) \right) y(n-i) \right\} \\ &= -2 \left\{ E\{y(n)y(n-i)\} - \sum_{j=1}^N \hat{a}_j E\{y(n-j)y(n-i)\} \right\} \\ &= -2R_Y(i) + 2 \sum_{j=1}^N \hat{a}_j R_Y(i-j), \end{aligned} \quad (7.68)$$

assuming that the process $\{Y\}$ is WSS.

By forming the gradient vector $\nabla_{\hat{\mathbf{a}}}\xi$ with respect to the LP coefficient vector $\hat{\mathbf{a}} = [\hat{a}_1 \ \hat{a}_2 \ \dots \ \hat{a}_N]^T$ and equating it to zero, we are able to determine the MSE stationary point $\hat{\mathbf{a}}^*$ as the solution of the following linear system:

$$\begin{bmatrix} R_Y(0) & R_Y(-1) & \cdots & R_Y(1-N) \\ R_Y(1) & R_Y(0) & \cdots & R_Y(2-N) \\ \vdots & \vdots & \ddots & \vdots \\ R_Y(N-1) & R_Y(N-2) & \cdots & R_Y(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1^* \\ \hat{a}_2^* \\ \vdots \\ \hat{a}_N^* \end{bmatrix} = \begin{bmatrix} R_Y(1) \\ R_Y(2) \\ \vdots \\ R_Y(N) \end{bmatrix}, \quad (7.69)$$

which is the so-called Wiener–Hopf equation, written in short as

$$\mathbf{R}_Y \hat{\mathbf{a}}^* = \mathbf{p}_Y, \quad (7.70)$$

where \mathbf{R}_Y is the autocorrelation matrix for the process $\{Y\}$ and \mathbf{p}_Y is the right-hand side vector in Equation (7.69).

Using Equation (7.68), we may determine the MSE second-order differentiations as

$$\begin{aligned} \frac{\partial^2 \xi}{\partial \hat{a}_i \partial \hat{a}_k} &= \frac{\partial(\partial \xi / \partial \hat{a}_i)}{\partial \hat{a}_k} \\ &= \frac{\partial \left(-2R_Y(i) + 2 \sum_{j=1}^N \hat{a}_j R_Y(i-j) \right)}{\partial \hat{a}_k} \\ &= 2R_Y(i-k) \end{aligned} \quad (7.71)$$

in such a manner that the MSE Hessian matrix is given by

$$\mathbf{H} = \left[\frac{\partial^2 \xi}{\partial \hat{a}_i \partial \hat{a}_k} \right]_{i,k} = [2R_Y(i-k)]_{i,k} = 2\mathbf{R}_Y, \quad (7.72)$$

which, in general, is positive definite. This indicates that the stationary point $\hat{\mathbf{a}}^*$ as given in Equation (7.70) is associated with the MSE global minimum.

Analyzing Equations (7.64) and (7.65) in the z -transform domain, for a given signal $y(n)$ we get

$$E(z) = \left(1 - \hat{a}_1 z^{-1} - \hat{a}_2 z^{-2} - \dots - \hat{a}_N z^{-N} \right) Y(z), \quad (7.73)$$

or, equivalently:

$$H(z) = \frac{Y(z)}{E(z)} = \frac{1}{1 - \hat{a}_1 z^{-1} - \hat{a}_2 z^{-2} - \dots - \hat{a}_N z^{-N}}. \quad (7.74)$$

If the model order N is high enough, then one may achieve the best possible prediction and the error process $\{E\}$ becomes a white noise. Thus, the LP problem corresponds to an AR modeling of $y(n)$ with the signal $e(n)$ as input. In fact, the Wiener–Hopf Equation (7.69) can be obtained directly from the AR Yule–Walker equations by considering $e(n) \equiv x(n)$ and letting $-a_i = \hat{a}_i$ in Equation (7.56), for $v = 1, 2, \dots, N$. AR modeling and LP can be seen

as a pair of inverse problems: given an AR system, using the Yule–Walker equations one is able to infer the autocorrelation function of the output process. In the LP context, given the autocorrelation samples, we can estimate the AR system that better fits these samples in the minimum MSE sense.

Example 7.5. Determine the second-order linear predictor for a WSS process characterized by

$$R_Y(v) = 4(0.5)^{|v|}. \quad (7.75)$$

Solution

From Equation (7.75), we have that

$$\left. \begin{aligned} R_Y(0) &= 4 \\ R_Y(1) &= 2 \\ R_Y(2) &= 1 \end{aligned} \right\}, \quad (7.76)$$

which results in the Wiener–Hopf equation

$$\begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} \hat{a}_1^* \\ \hat{a}_2^* \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}. \quad (7.77)$$

The solution of the above equation is given by $\hat{a}_1^* = 0.5$ and $\hat{a}_2^* = 0$.

The solution obtained corresponds in fact to the first-order AR system

$$H(z) = \frac{1}{1 - 0.5z^{-1}}, \quad (7.78)$$

whose autocorrelation function, as determined in Equation (7.63), is of the form

$$R_Y(v) = \frac{\sigma_X^2(\hat{a}_1^*)^{|v|}}{1 - (\hat{a}_1^*)^2} = \frac{\sigma_X^2(0.5)^{|v|}}{1 - 0.5^2} = \frac{4\sigma_X^2(0.5)^{|v|}}{3}, \quad (7.79)$$

with, in this case, σ_X^2 determined by

$$\sigma_X^2 = R_Y(0)(1 - (\hat{a}_1^*)^2) = 4(1 - 0.5^2) = 3, \quad (7.80)$$

turning Equation (7.79) into (7.75), as expected. \triangle

An interesting frequency-domain interpretation for the LP problem arises from the development

$$E\{e^2(n)\} = R_E(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_E(e^{j\omega}) d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\Gamma_Y(e^{j\omega})}{|H(e^{j\omega})|^2} d\omega. \quad (7.81)$$

For a particular process, $\Gamma_Y(e^{j\omega})$ is a fixed nonnegative function and, for an AR system, $|H(e^{j\omega})|^2$ is a strictly positive function. Hence, the last integral in Equation (7.81) is equivalent to the area below the curve defined by the ratio of these two functions. The best we can do to minimize this area, thus minimizing the corresponding MSE, is to choose the

LP coefficients \hat{a}_i in such a way that the denominator $|H(e^{j\omega})|^2$ becomes large when the numerator is large, small when the numerator is small, and so on. This procedure leads to an error process closer to a white noise. In that sense, the optimal MSE solution is such that $|H(e^{j\omega})|^2$ follows the shape of $\Gamma_Y(e^{j\omega})$, as closely as possible, considering the limited number N of variables. In practice, the LP solution is such that the AR power spectrum becomes a smoothed version of the PSD function of the process $\{Y\}$, and the level of approximation required by the application at hand determines the optimal value of N to be used.

The implementations of the Wiener–Hopf equation differ in the algorithm for estimating the autocorrelation function. Each variation of estimation method results in a slightly different AR model, as described in the following subsections.

7.5.2 Covariance method

In the covariance method, the data set is windowed and the estimation error is minimized only within the window interval. In that manner, only part of the available data is used in estimating the autocorrelation function. This yields a different autocorrelation estimation for each window location k , disregarding the common WSS assumption, such that

$$\hat{R}_{Y,m}(\mu, \nu) = \frac{1}{K} \sum_{n=k}^{K+k-1} y(n-\mu)y(n-\nu), \quad (7.82)$$

where $K < N$ is the window length and subscript m stands for modified. The value of K should allow one to shift the data window N samples to the left and to the right within the complete data set. This estimation has the advantage of being unbiased, since

$$\begin{aligned} E\left\{\hat{R}_{Y,m}(\mu, \nu)\right\} &= E\left\{\frac{1}{K} \sum_{n=k}^{K+k-1} y(n-\mu)y(n-\nu)\right\} \\ &= \frac{1}{K} \sum_{n=k}^{K+k-1} E\{y(n-\mu)y(n-\nu)\} \\ &= R_Y(\mu, \nu) \end{aligned} \quad (7.83)$$

with low variance for large values of $(\mu - \nu)$, since all autocorrelation values are the average of the same number K of nonzero terms. This modified estimator results in a Wiener–Hopf equation of the form

$$\begin{bmatrix} R_Y(1,1) & R_Y(1,2) & \cdots & R_Y(1,N) \\ R_Y(2,1) & R_Y(2,2) & \cdots & R_Y(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ R_Y(N,1) & R_Y(N,2) & \cdots & R_Y(N,N) \end{bmatrix} \begin{bmatrix} \hat{a}_1^* \\ \hat{a}_2^* \\ \vdots \\ \hat{a}_N^* \end{bmatrix} = \begin{bmatrix} R_Y(1,0) \\ R_Y(2,0) \\ \vdots \\ R_Y(N,0) \end{bmatrix}, \quad (7.84)$$

where in an actual implementation we utilize $\hat{R}_{Y,m}(\mu, \nu)$ instead of the ideal values $R_Y(\mu, \nu)$. Note that the autocorrelation matrix is symmetric and non-Toeplitz, which can be inverted by using the so-called Cholesky or LU (lower–upper) decompositions (Strang, 1980). The

AR model obtained from the covariance method is not guaranteed to be stable (see Exercise 7.18), but practical problems often lead to stable poles and unbiased PSD estimation (Kay, 1988).

7.5.3 Autocorrelation method

In the autocorrelation method for AR modeling, the LP problem (7.69) is solved by using the biased autocorrelation estimation $\hat{R}_{Y,b}(\nu)$ as given in Equation (7.11). It can be shown that the resulting AR model is guaranteed to be stable, but the associated PSD estimation is biased and presents low resolution (Kay, 1988). The unbiased autocorrelation estimation $\hat{R}_{Y,u}(\nu)$ using all available data, as defined in Equation (7.17), leads to an ill-conditioned autocorrelation matrix, which introduces large variance in the final PSD estimation.

Owing to the even symmetry in the $\hat{R}_{Y,b}(\nu)$ estimation, the corresponding autocorrelation matrix is symmetric, Toeplitz, and positive definite, in such a way that the so-called Levinson–Durbin recursion, described in Section 7.5.4, can be used to invert it and solve the linear system of equations.

Example 7.6. Consider the output signal $y(n)$ of the Hamming-window FIR filter designed in Example 5.3, as detailed in Table 5.6, to a white-noise input with zero mean and unit variance. Find the N th-order AR model for such a signal using the autocorrelation method for different values of N .

Solution

This problem constitutes an interesting challenge for the AR model, since it is expected to approximate a system with all zeros on the unit circle and all poles at the origin of the complex plane. Using the autocorrelation method with $N = 10, 20, 50$, as described in this section, results in the AR power spectra shown in Figure 7.5 along with the original FIR

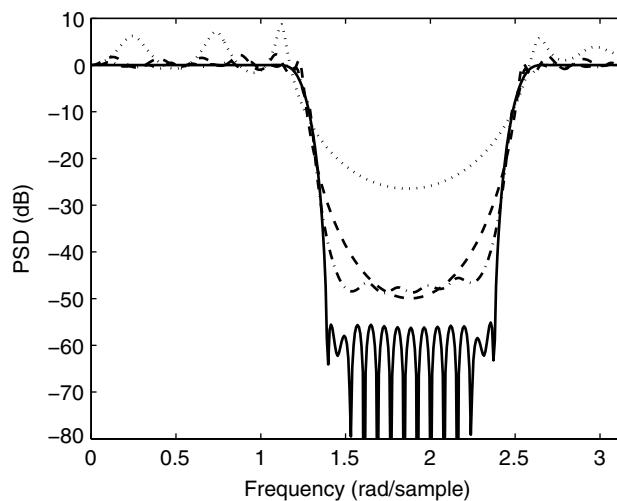


Fig. 7.5.

Power response for original MA system (solid line) and respective AR approximations using autocorrelation method: $N = 10$ (dotted line), $N = 20$ (dashed line), $N = 50$ (dash-dotted line).

response (solid line). In this figure, all AR spectra were normalized to 0 dB at $\omega = 0$. Results clearly indicate that larger values of N yield AR models with power response closer to the ideal one. \triangle

7.5.4 Levinson–Durbin algorithm

We can generate the extended Wiener–Hopf equation of (7.69) by incorporating the AR Yule–Walker equation (7.56) for $v = 0$ into the standard form, obtaining

$$\begin{bmatrix} R_Y(0) & R_Y(-1) & R_Y(-2) & \dots & R_Y(-N) \\ R_Y(1) & R_Y(0) & R_Y(-1) & \dots & R_Y(1-N) \\ R_Y(2) & R_Y(1) & R_Y(0) & \dots & R_Y(2-N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_Y(N) & R_Y(N-1) & R_Y(N-2) & \dots & R_Y(0) \end{bmatrix} \begin{bmatrix} 1 \\ -\hat{a}_{1,[N]}^* \\ -\hat{a}_{2,[N]}^* \\ \vdots \\ -\hat{a}_{N,[N]}^* \end{bmatrix} = \begin{bmatrix} \sigma_{X,[N]}^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad (7.85)$$

where the sub-index $[N]$ relates the associated variable to the N th-order model. In this case, we must note that the value of $R_Y(v)$ is independent of the model order, whereas the estimation error is highly dependent on N , and so is its variance σ_X^2 . Using the matrix notation introduced in Equation (7.70), the extended Wiener–Hopf equation can also be written in a compact form as

$$\mathbf{R}_{Y,[N+1]} \begin{bmatrix} 1 \\ -\hat{\mathbf{a}}_{[N]}^* \end{bmatrix} = \begin{bmatrix} \sigma_{X,[N]}^2 \\ \mathbf{0} \end{bmatrix}. \quad (7.86)$$

The Levinson–Durbin algorithm determines the LP coefficients of the N th-order model from the $(N - 1)$ th-order model. This recursion is then initialized for the first-order model and then iterates up to the desired order N . To obtain the order-recursive relationships, following the approach in Deller *et al.* (2000), consider the upgrade from the second- to the third-order AR model, assuming a symmetric autocorrelation matrix in Equation (7.85), such that $R_Y(v) = R_Y(-v)$, for all v .

Assume, at first, that we can express the third-order extended coefficient vector in Equation (7.86) as

$$\begin{bmatrix} 1 \\ -\hat{\mathbf{a}}_{[3]}^* \end{bmatrix} = \begin{bmatrix} 1 \\ -\hat{a}_{1,[3]}^* \\ -\hat{a}_{2,[3]}^* \\ -\hat{a}_{3,[3]}^* \end{bmatrix} = \begin{bmatrix} 1 \\ -\hat{a}_{1,[2]}^* \\ -\hat{a}_{2,[2]}^* \\ 0 \end{bmatrix} - k_3 \begin{bmatrix} 0 \\ -\hat{a}_{2,[2]}^* \\ -\hat{a}_{1,[2]}^* \\ 1 \end{bmatrix}, \quad (7.87)$$

where k_3 is an auxiliary parameter to be determined. Using this expression, the extended Wiener–Hopf equation for $N = 3$ becomes

$$\mathbf{R}_{Y,[4]} \begin{bmatrix} 1 \\ -\hat{a}_{1,[3]}^* \\ -\hat{a}_{2,[3]}^* \\ -\hat{a}_{3,[3]}^* \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{Y,[3]} & R_Y(-3) \\ R_Y(3) & R_Y(2) & R_Y(1) & R_Y(0) \end{bmatrix} \begin{bmatrix} 1 \\ -\hat{a}_{1,[2]}^* \\ -\hat{a}_{2,[2]}^* \\ 0 \end{bmatrix}$$

$$\begin{aligned}
& -k_3 \begin{bmatrix} R_Y(0) & R_Y(-1) & R_Y(-2) & R_Y(-3) \\ R_Y(1) & & & \\ R_Y(2) & & \mathbf{R}_{Y,[3]} & \\ R_Y(3) & & & \end{bmatrix} \begin{bmatrix} 0 \\ -\hat{a}_{2,[2]}^* \\ -\hat{a}_{1,[2]}^* \\ 1 \end{bmatrix} \\
& = \begin{bmatrix} \mathbf{R}_{Y,[3]} \begin{bmatrix} 1 \\ -\hat{a}_{1,[2]}^* \\ -\hat{a}_{2,[2]}^* \\ q_{[2]} \end{bmatrix} \\ -k_3 \begin{bmatrix} \mathbf{R}_{Y,[3]} \begin{bmatrix} q_{[2]} \\ -\hat{a}_{2,[2]}^* \\ -\hat{a}_{1,[2]}^* \\ 1 \end{bmatrix} \\ \sigma_{X,[3]}^2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \end{aligned} \tag{7.88}$$

with

$$\begin{aligned}
q_{[2]} &= R_Y(3) - R_Y(2)\hat{a}_{1,[2]}^* - R_Y(1)\hat{a}_{2,[2]}^* \\
&= R_Y(-3) - R_Y(-2)\hat{a}_{1,[2]}^* - R_Y(-1)\hat{a}_{2,[2]}^*. \tag{7.89}
\end{aligned}$$

Hence, we can write that

$$\begin{bmatrix} \sigma_{X,[2]}^2 \\ 0 \\ 0 \\ q_{[2]} \end{bmatrix} - k_3 \begin{bmatrix} q_{[2]} \\ 0 \\ 0 \\ \sigma_{X,[2]}^2 \end{bmatrix} = \begin{bmatrix} \sigma_{X,[3]}^2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{7.90}$$

or, equivalently:

$$\begin{cases} \sigma_{X,[2]}^2 - k_3 q_{[2]} = \sigma_{X,[3]}^2 \\ q_{[2]} - k_3 \sigma_{X,[2]}^2 = 0 \end{cases}. \tag{7.91}$$

Solving the system above, we get

$$k_3 = \frac{q_{[2]}}{\sigma_{X,[2]}^2} = \frac{R_Y(3) - R_Y(2)\hat{a}_{1,[2]}^* - R_Y(1)\hat{a}_{2,[2]}^*}{\sigma_{X,[2]}^2} \tag{7.92}$$

$$\sigma_{X,[3]}^2 = (1 - k_3^2)\sigma_{X,[2]}^2 \tag{7.93}$$

and, from Equation (7.87):

$$\begin{cases} \hat{a}_{1,[3]}^* = \hat{a}_{1,[2]}^* - k_3 \hat{a}_{2,[2]}^* \\ \hat{a}_{2,[3]}^* = \hat{a}_{2,[2]}^* - k_3 \hat{a}_{1,[2]}^* \\ \hat{a}_{3,[3]}^* = k_3 \end{cases}. \tag{7.94}$$

Equations (7.92), (7.93), and (7.94) illustrate that the third-order AR model can be determined from the second-order parameters $q_{[2]}$, $\sigma_{X,[2]}^2$, $\hat{a}_{1,[2]}^*$, and $\hat{a}_{2,[2]}^*$. Generalizing this recursion, we obtain the iterative Levinson–Durbin algorithm with computational complexity proportional to N^2 and described by:

- (i) For a given data set, determine $R_Y(0), R_Y(1), \dots, R_Y(N)$.
- (ii) Set $\sigma_{X,[0]}^2 = R_Y(0)$.
- (iii) For $i = 1, 2, \dots, N$, compute:

$$k_i = \frac{R_Y(i) - \sum_{j=1}^{i-1} \hat{a}_{j,[i-1]}^* R_Y(i-j)}{\sigma_{X,[i-1]}^2} \quad (7.95)$$

$$\sigma_{X,[i]}^2 = (1 - k_i^2) \sigma_{X,[i-1]}^2 \quad (7.96)$$

$$\hat{a}_{j,[i]}^* = \begin{cases} \hat{a}_{j,[i-1]}^* - k_i \hat{a}_{i-j,[i-1]}^*, & \text{for } j = 1, 2, \dots, (i-1) \\ k_i, & \text{for } j = i \end{cases}. \quad (7.97)$$

The auxiliary parameters k_i , for $i = 1, 2, \dots, N$, are known as the reflection coefficients, and they constitute an alternative description of the N th-order AR model, as opposed to the LP coefficients $\hat{a}_{i,[N]}^*$.

7.5.5 Burg's method

The LP model given in Equation (7.64) employs N past samples of the available signal to estimate the current sample $y(n)$. The estimation error between $y(n)$ and the optimal LP estimate is also known as the N th-order forward estimation error and is denoted by

$$x_{f,[N]}(n) = y(n) - \sum_{i=1}^N \hat{a}_{i,[N]}^* y(n-i). \quad (7.98)$$

If we feed the LP model with a reversed-in-time version of the process $\{Y\}$, then future samples are combined to form an estimate of $y(n-N)$. Since $R_Y(v)$ is an even function, the corresponding Wiener–Hopf equations, and consequently the optimal LP model, remain the same. In such a case, the error between the past sample $y(n-N)$ and its estimate is referred to as the backward LP error and is denoted by

$$x_{b,[N]}(n) = y(n-N) - \sum_{i=1}^N \hat{a}_{i,[N]}^* y(n-N+i). \quad (7.99)$$

Applying the Levinson–Durbin order-recursion of Equation (7.97) to the optimal LP coefficients in Equation (7.98), we have that

$$\begin{aligned}
x_{f,[N]}(n) &= y(n) - \left(\sum_{i=1}^{N-1} \hat{a}_{i,[N]}^* y(n-i) \right) - \hat{a}_{N,[N]}^* y(n-N) \\
&= y(n) - \left(\sum_{i=1}^{N-1} \hat{a}_{i,[N-1]}^* y(n-i) \right) \\
&\quad + \left(\sum_{i=1}^{N-1} k_N \hat{a}_{N-i,[N-1]}^* y(n-i) \right) - k_N y(n-N) \\
&= x_{f,[N-1]}(n) - k_N \left[y(n-N) - \left(\sum_{j=1}^{N-1} \hat{a}_{j,[N-1]}^* y(n+j-N) \right) \right] \\
&= x_{f,[N-1]}(n) - k_N x_{b,[N-1]}(n-1).
\end{aligned} \tag{7.100}$$

A similar development for the backward estimation error, as defined in Equation (7.99), results in

$$\begin{aligned}
x_{b,[N]}(n) &= y(n-N) - \left(\sum_{i=1}^{N-1} \hat{a}_{i,[N]}^* y(n-N+i) \right) - \hat{a}_{N,[N]}^* y(n) \\
&= y(n-N) - \left(\sum_{i=1}^{N-1} \hat{a}_{i,[N-1]}^* y(n-N+i) \right) \\
&\quad + \left(\sum_{i=1}^{N-1} k_N \hat{a}_{N-i,[N-1]}^* y(n-N+i) \right) - k_N y(n) \\
&= x_{b,[N-1]}(n-1) - k_N \left[y(n) - \left(\sum_{j=1}^{N-1} \hat{a}_{j,[N-1]}^* y(n-j) \right) \right] \\
&= x_{b,[N-1]}(n-1) - k_N x_{f,[N-1]}(n).
\end{aligned} \tag{7.101}$$

We can then define the mean value of the forward and backward prediction-error powers for the i th-order LP model as

$$\xi_{B,[i]} = \frac{\xi_{f,[i]} + \xi_{b,[i]}}{2}, \tag{7.102}$$

where

$$\xi_{f,[i]} = \frac{1}{L-i} \sum_{n=i}^{L-1} x_{f,[i]}^2(n) \tag{7.103}$$

$$\xi_{B,[i]} = \frac{1}{L-i} \sum_{n=i}^{L-1} x_{B,[i]}^2(n). \quad (7.104)$$

Burg's method determines the reflection coefficients k_i that minimize $\xi_{B,[i]}$.

Differentiating $\xi_{f,[i]}$ with respect to k_i we get

$$\begin{aligned} \frac{\partial \xi_{f,[i]}}{\partial k_i} &= \frac{2}{L-i} \sum_{n=i}^{L-1} x_{f,[i]}(n) \frac{\partial x_{f,[i]}(n)}{\partial k_i} \\ &= \frac{2}{L-i} \sum_{n=i}^{L-1} x_{f,[i]}(n) \frac{\partial (x_{f,[i-1]}(n) - k_i x_{b,[i-1]}(n-1))}{\partial k_i} \\ &= -\frac{2}{L-i} \sum_{n=i}^{L-1} x_{f,[i]}(n) x_{b,[i-1]}(n-1). \end{aligned} \quad (7.105)$$

Analogously, differentiating $\xi_{b,[i]}$ with respect to k_i , results in

$$\begin{aligned} \frac{\partial \xi_{b,[i]}}{\partial k_i} &= \frac{2}{L-i} \sum_{n=i}^{L-1} x_{b,[i]}(n) \frac{\partial x_{b,[i]}(n)}{\partial k_i} \\ &= \frac{2}{L-i} \sum_{n=i}^{L-1} x_{b,[i]}(n) \frac{\partial (x_{b,[i-1]}(n-1) - k_i x_{f,[i-1]}(n))}{\partial k_i} \\ &= -\frac{2}{L-i} \sum_{n=i}^{L-1} x_{b,[i]}(n) x_{f,[i-1]}(n). \end{aligned} \quad (7.106)$$

Therefore, according to Equations (7.100), (7.101), (7.105), and (7.106), we can write that

$$\begin{aligned} \frac{\partial \xi_{B,[i]}}{\partial k_i} &= \frac{(\partial \xi_{f,[i]}/\partial k_i) + (\partial \xi_{b,[i]}/\partial k_i)}{2} \\ &= -\frac{1}{L-i} \sum_{n=i}^{L-1} [(x_{f,[i-1]}(n) - k_i x_{b,[i-1]}(n-1)) x_{b,[i-1]}(n-1)] \\ &\quad - \frac{1}{L-i} \sum_{n=i}^{L-1} [(x_{b,[i-1]}(n-1) - k_i x_{f,[i-1]}(n)) x_{f,[i-1]}(n)]. \end{aligned} \quad (7.107)$$

Equating this result to zero yields

$$k_i = \frac{2 \sum_{n=i}^{L-1} x_{f,[i-1]}(n) x_{b,[i-1]}(n-1)}{\sum_{n=i}^{L-1} (x_{b,[i-1]}^2(n-1) + x_{f,[i-1]}^2(n))}, \quad (7.108)$$

which is the reflection coefficient used in Burg's method. This estimate is employed in the Levinson–Durbin recursions of Equation (7.97), to determine the corresponding LP model and the desired PSD estimation.

It is commonplace in the spectral estimation literature to associate Burg's method with a maximum entropy PSD estimation; see Exercise 1.32. As mentioned in the exercise, maximizing the entropy of a random variable is equivalent to maximizing its uncertainty. Therefore, this condition would be the least restrictive constraint on the associated extrapolation of the autocorrelation function for any lag v . Such a property, however, is intrinsic to the AR model of Gaussian processes and, therefore, should be associated with all AR-modeling methods described in Section 7.5 (Kay, 1988; Cover & Thomas, 2006). In this sense, the main feature of Burg's method is that it yields a system model based directly on the lattice-form reflection coefficients k_i , as given in Equation (7.108), instead of the standard LP coefficients which characterize a direct-form AR model.

Example 7.7. Find an AR model for the same process as given in Example 7.6 using the covariance, autocorrelation, and Burg's method with $N = 40$.

Solution

The PSD estimates from the autocorrelation (dashed line) and Burg's (dash-dotted line) methods are shown in Figure 7.6 for $N = 40$. The PSD estimate from the covariance method is indistinguishable from that from Burg's method, which, owing to the unbiased estimate for the autocorrelation function, is superior to that yielded by the autocorrelation method. \triangle

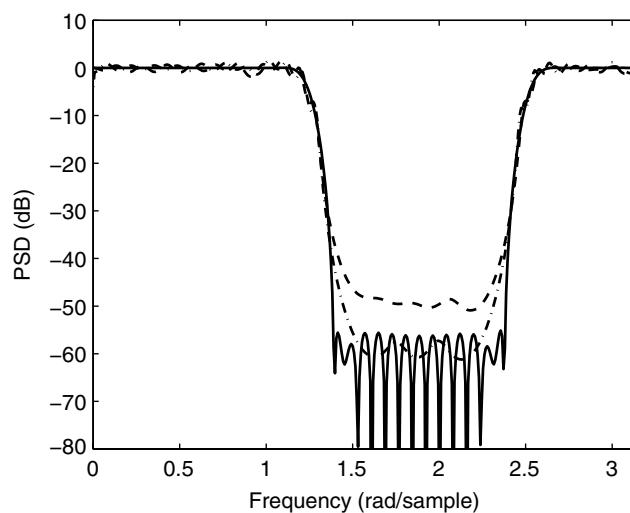


Fig. 7.6.

Power response for original MA system (solid line) and respective AR approximations of order $N = 40$: autocorrelation (dashed line) and Burg and covariance (dash-dotted line) methods.

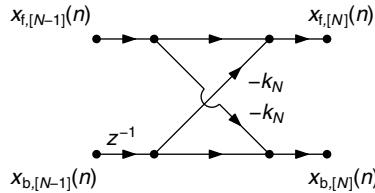


Fig. 7.7. Basic cell equivalent to one-level Levinson–Durbin recursion.

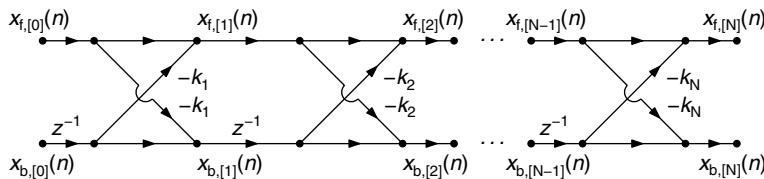


Fig. 7.8. Two-multiplier lattice structure equivalent to N -level Levinson–Durbin recursion.

7.5.6 Relationship of the Levinson–Durbin algorithm to a lattice structure

In this subsection we describe how the parameters of the Levinson–Durbin recursion can be related to a digital filter structure known as lattice structure. The lattice structures will be further discussed in Section 12.2.

Equations (7.100) and (7.101)

$$\left. \begin{aligned} x_{f,[N]}(n) &= x_{f,[N-1]}(n) - k_N x_{b,[N-1]}(n-1) \\ x_{b,[N]}(n) &= x_{b,[N-1]}(n-1) - k_N x_{f,[N-1]}(n) \end{aligned} \right\}, \quad (7.109)$$

repeated here for the reader’s convenience, relate the N th-order estimation errors with their $(N-1)$ th-order counterparts. Taking these relationships to the z domain, we obtain

$$\left. \begin{aligned} X_{f,[N]}(z) &= X_{f,[N-1]}(z) - k_N z^{-1} X_{b,[N-1]}(z) \\ X_{b,[N]}(z) &= z^{-1} X_{b,[N-1]}(z) - k_N X_{f,[N-1]}(z) \end{aligned} \right\}, \quad (7.110)$$

which can be readily associated with the basic digital-filter cell depicted in Figure 7.7. Concatenating several of these cells yields the two-multiplier lattice structure seen in Figure 7.8. This lattice structure can be useful, for example, when one wants to estimate the prediction errors for several model orders, to aid in determining a good compromise for an AR model order.

7.6 Wiener filter

In this section we generalize the linear prediction problem to the case where we wish to characterize the stochastic relationship between two processes represented by the realizations

$y(n)$ and $x(n)$. The resulting linear model is commonly referred to as the Wiener filter. In this context, assume that an estimate $\hat{y}(n)$ of $y(n)$ can be determined as a linear combination of samples of $x(n)$; that is:

$$\begin{aligned}\hat{y}(n) &= \hat{w}_0x(n) + \hat{w}_1x(n-1) + \cdots + \hat{w}_Mx(n-M) \\ &= \sum_{i=0}^M \hat{w}_i x(n-i),\end{aligned}\quad (7.111)$$

where \hat{w}_i , for $i = 0, 1, \dots, M$, are the so-called Wiener filter coefficients and M is the filter order. The error signal in this case is given by

$$e(n) = y(n) - \hat{y}(n) \quad (7.112)$$

and the associated MSE is given by

$$\xi = E\{e^2(n)\}. \quad (7.113)$$

Similar to the linear prediction case, Equation (7.113) represents a quadratic function of the Wiener filter coefficients \hat{w}_i . It is then possible to deduce that

$$\begin{aligned}\frac{\partial \xi}{\partial \hat{w}_i} &= \frac{\partial E\{e^2(n)\}}{\partial \hat{w}_i} \\ &= E \left\{ \frac{\partial e^2(n)}{\partial \hat{w}_i} \right\} \\ &= E \left\{ 2e(n) \frac{\partial e(n)}{\partial \hat{w}_i} \right\} \\ &= -2E\{e(n)x(n-i)\}\end{aligned}\quad (7.114)$$

and, by replacing $e(n)$ by its expression using Equations (7.111) and (7.112), it follows that

$$\begin{aligned}\frac{\partial \xi}{\partial \hat{w}_i} &= -2E \left\{ \left(y(n) - \sum_{j=0}^M \hat{w}_j x(n-j) \right) x(n-i) \right\} \\ &= -2 \left\{ E\{y(n)x(n-i)\} - \sum_{j=0}^M \hat{w}_j E\{x(n-j)x(n-i)\} \right\} \\ &= -2p_{YX}(i) + 2 \sum_{j=0}^M \hat{w}_j R_X(i-j),\end{aligned}\quad (7.115)$$

where $p_{YX}(i)$ is the cross-correlation between $y(n)$ and $x(n-i)$, assuming that the processes $\{Y\}$ and $\{X\}$ are jointly WSS. By forming the gradient vector $\nabla_{\hat{w}} \xi$ with respect to the Wiener filter coefficients $\hat{w} = [\hat{w}_0 \ \hat{w}_1 \ \cdots \ \hat{w}_M]^T$ and equating it to zero, it is possible to compute

the MSE stationary point $\hat{\mathbf{w}}^*$ by solving the system

$$\begin{bmatrix} R_X(0) & R_X(-1) & \dots & R_X(-M) \\ R_X(1) & R_X(0) & \dots & R_X(-M+1) \\ \vdots & \vdots & \ddots & \vdots \\ R_X(M) & R_X(M-1) & \dots & R_X(0) \end{bmatrix} \begin{bmatrix} \hat{w}_0^* \\ \hat{w}_1^* \\ \vdots \\ \hat{w}_M^* \end{bmatrix} = \begin{bmatrix} p_{YX}(0) \\ p_{YX}(1) \\ \vdots \\ p_{YX}(M) \end{bmatrix}; \quad (7.116)$$

that is:

$$\hat{\mathbf{w}}^* = \mathbf{R}_X^{-1} \mathbf{p}_{YX}, \quad (7.117)$$

where \mathbf{R}_X is the autocorrelation matrix for the process $\{X\}$ and \mathbf{p}_{YX} is the cross-correlation vector in the right-hand side of Equation (7.116).

The minimum MSE yielded by the Wiener solution is given by (see Exercise 7.30)

$$\xi_{\min} = E\{y^2(n)\} - \mathbf{p}_{YX}^T \mathbf{R}_X^{-1} \mathbf{p}_{YX}. \quad (7.118)$$

Example 7.8. Let $y(n)$ and $v(n)$ be realizations of two first-order AR processes characterized by

$$y(n) = \alpha_1 w_1(n) + 0.8y(n-1) \quad (7.119)$$

$$v(n) = \alpha_2 w_2(n) - 0.8v(n-1), \quad (7.120)$$

where $w_1(n)$ and $w_2(n)$ are uncorrelated white noise signals. Assume that $y(n)$, $v(n)$, $w_1(n)$, and $w_2(n)$ all have unit variance. Determine the second-order Wiener filter for the jointly WSS processes $\{Y\}$ and $\{X\}$, where

$$x(n) = \frac{1}{2}y(n) + \frac{\sqrt{3}}{2}v(n). \quad (7.121)$$

Solution

From Equation (7.63), as $\sigma_{W_1}^2 = \sigma_{W_2}^2 = 1$, the autocorrelations of $y(n)$ and $v(n)$ are such that

$$E\{y(n)y(n-m)\} = \alpha_1^2 \frac{(0.8)^{|m|}}{1-0.8^2} \quad (7.122)$$

$$E\{v(n)v(n-m)\} = \alpha_2^2 \frac{(-0.8)^{|m|}}{1-0.8^2}, \quad (7.123)$$

and since $\sigma_Y^2 = \sigma_V^2 = 1$, one has $\alpha_1^2 = \alpha_2^2 = (1-0.8^2)$. In addition, since $w_1(n)$ and $w_2(n)$ are uncorrelated, so are $y(n)$ and $v(n)$, and then

$$R_X(0) = \left(\frac{1}{2}\right)^2 R_Y(0) + \left(\frac{\sqrt{3}}{2}\right)^2 R_V(0) = \frac{1}{4} + \frac{3}{4} = 1 \quad (7.124)$$

$$R_X(1) = \left(\frac{1}{2}\right)^2 R_Y(1) + \left(\frac{\sqrt{3}}{2}\right)^2 R_V(1) = \frac{1}{4}(0.8) + \frac{3}{4}(-0.8) = -0.4 \quad (7.125)$$

$$R_{YX}(0) = \frac{1}{2}R_Y(0) = 0.5 \quad (7.126)$$

$$R_{YX}(1) = \frac{1}{2}R_Y(1) = 0.4. \quad (7.127)$$

Therefore, as given by Equation (7.116), the Wiener filter coefficients are the solution of

$$\begin{bmatrix} 1 & -0.4 \\ -0.4 & 1 \end{bmatrix} \hat{\mathbf{w}}^* = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}, \quad (7.128)$$

which corresponds to the Wiener filter

$$W(z) = \frac{11}{14} + \frac{10}{14}z^{-1}. \quad (7.129)$$

△

The Wiener filter determines the best closed-form linear relationship, in the MSE sense, between the realizations $y(n)$ and $x(n)$ of two joint WSS processes. In that context, the LP problem can be seen as a special case where $y(n)$ represents a future sample $x(n+1)$ of the other process. Owing to its generality, the Wiener filter finds wide application in a variety of problems such as system identification, channel equalization, and noise cancellation, besides, of course, linear prediction.

7.7 Other methods for spectral estimation

The list of PSD estimation methods is almost endless. The material included in this chapter represents perhaps only the tip of an immense iceberg, nicely treated by excellent textbooks such as Kay (1988), Hayes (1996), and Stoica and Moses (1997).

For AR modeling, for instance, we may modify Burg's method by incorporating a window function onto the reflection coefficient estimate given in Equation (7.108). The result is a PSD estimate less prone to bias due to phase dependence on sinusoidal components of the input signal (Kay, 1988). Another technique, also based on the Levinson–Durbin algorithm, is due to Itakura and Saito (1971), and employs a reflection coefficient estimate given by

$$k_i = \frac{2 \sum_{n=i}^{L-1} x_{f,[i-1]}(n)x_{b,[i-1]}(n-1)}{\sqrt{\left(\sum_{n=i}^{L-1} x_{b,[i-1]}^2(n-1)\right)\left(\sum_{n=i}^{L-1} x_{f,[i-1]}^2(n)\right)}}. \quad (7.130)$$

Alternatively, the same idea of combining forward and backward prediction errors can be applied directly to the LP coefficients, originating a modified version of the covariance method. In general, all these AR-modeling modifications yield minor differences on the resulting PSD estimation and, therefore, may be seen as interesting variations of the methods described here.

As mentioned before, this chapter has focused on AR models due to the simplicity of the associated algorithms, their capacity of modeling ARMA systems, and to the excellent quality of the obtained PSD estimates. Algorithms for ARMA and MA models do exist and can be effectively employed when the spectrum at hand presents a significant number of zeros. In that sense, working directly with ARMA or MA models, we avoid the high order required to approximate these functions by AR systems. The price to be paid is the computational complexity of the ARMA and MA modeling algorithms, which may converge to local-optimal solutions due to the highly nonlinear equations associated with these model classes.

The problem of PSD estimation may be particularized for several types of random process. The practical case of sinusoids embedded in noise is of crucial interest, since it is related to several communications or radar/sonar systems. Algorithms that are customized for this problem, including Prony's, MUSIC, or subspace methods, are classified as parametric, since they assume a particular structure for the data at hand. Owing to their general complexity, these methods are not covered in this book, and the reader is once again directed to the vast literature on the subject to find out more about them.

There is another very important group of methods that can be employed for PSD estimation. The so-called adaptive algorithms are often employed in real-time situations, where we do not have access to a significant amount of data in advance, or when the process presents nonstationary statistical properties. Adaptive filters repeatedly adjust their transfer-function coefficients, generating a time-varying model with an associated PSD function. Once again, this class of PSD estimation algorithms constitutes a very broad subject, whose associated literature may be accessed by the interested reader (Widrow & Stearns, 1985; Haykin, 1996; Diniz, 2008).

7.8 Do-it-yourself: spectral estimation

Experiment 7.1

In this experiment we consider an ARMA process z generated by applying a white Gaussian noise, with zero mean and unit variance, to a filter with transfer function

$$H(z) = \frac{z^2 - \sqrt{3}z + 1}{10(z^2 + 1.8z + 0.96)}. \quad (7.131)$$

This transfer function has zeros on the unit circle with angles ± 0.5236 radians and the poles are placed at angles ± 2.7352 radians with radius 0.9798. The corresponding power response is shown in Figure 7.9.

Assume that this ARMA process is corrupted by a sinusoidal component s of frequency $f_1 = fs/8$, with $fs = 1024$ Hz, such that 250 ms of the measured signal y can be determined in MATLAB as

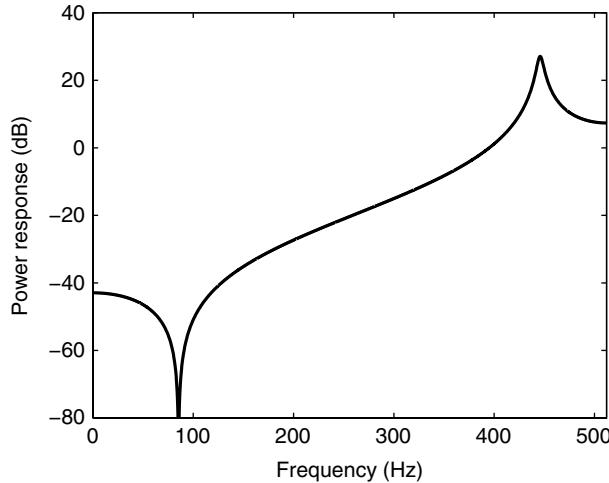


Fig. 7.9. Power response associated with $H(z)$ in Experiment 7.1.

```
fs = 1024; Ts = 1/fs; t = 0:Ts:(0.25-Ts); L = length(t);
f1 = 128; s = sin(2*pi*f1.*t);
x = randn(1,L); x = x-mean(x); x = x./sqrt(var(x));
b = [1 -sqrt(3) 1.0]; a = 10*[1 1.8 0.96]; z = filter(b,a,x);
y = z + s;
```

Therefore, the PSD associated with the process $\{Y\}$ incorporates a delta function with area 0.5 at $f = f_1$ to the power spectrum of $\{Z\}$ seen in Figure 7.9.

The PSD estimate Gamma_Y_P using the periodogram algorithm, based on an L_f -point FFT, with $L_f = 2048$, can be determined as

```
Ry = xcorr(y, 'biased');
Gamma_Y_P = 10*log10(abs(fft(Ry,Lf)));
```

The estimated PSD is shown in Figure 7.10a, where it can be observed that both the sinusoid frequency and the filter resonance frequency, indicated by the dotted vertical lines, are located at their proper positions.

By dividing the data into $nb = 4$ blocks of length L/nb , we may obtain distinct periodogram estimates, which can be averaged to generate the PSD estimate Gamma_Y_AP using the script

```
YBlock = reshape(y,L/nb,nb);
RYBlock = zeros(2*L/nb-1,nb);
for i = 1:nb,
    RYBlock(:,i) = xcorr(YBlock(:,i),'biased');
end;
Gamma_Y_AP = 10*log10(mean(abs(fft(RYBlock,LRY)')));
The result, as seen in Figure 7.10b, is a smoothed estimate less able to discriminate the sinusoidal component.
```

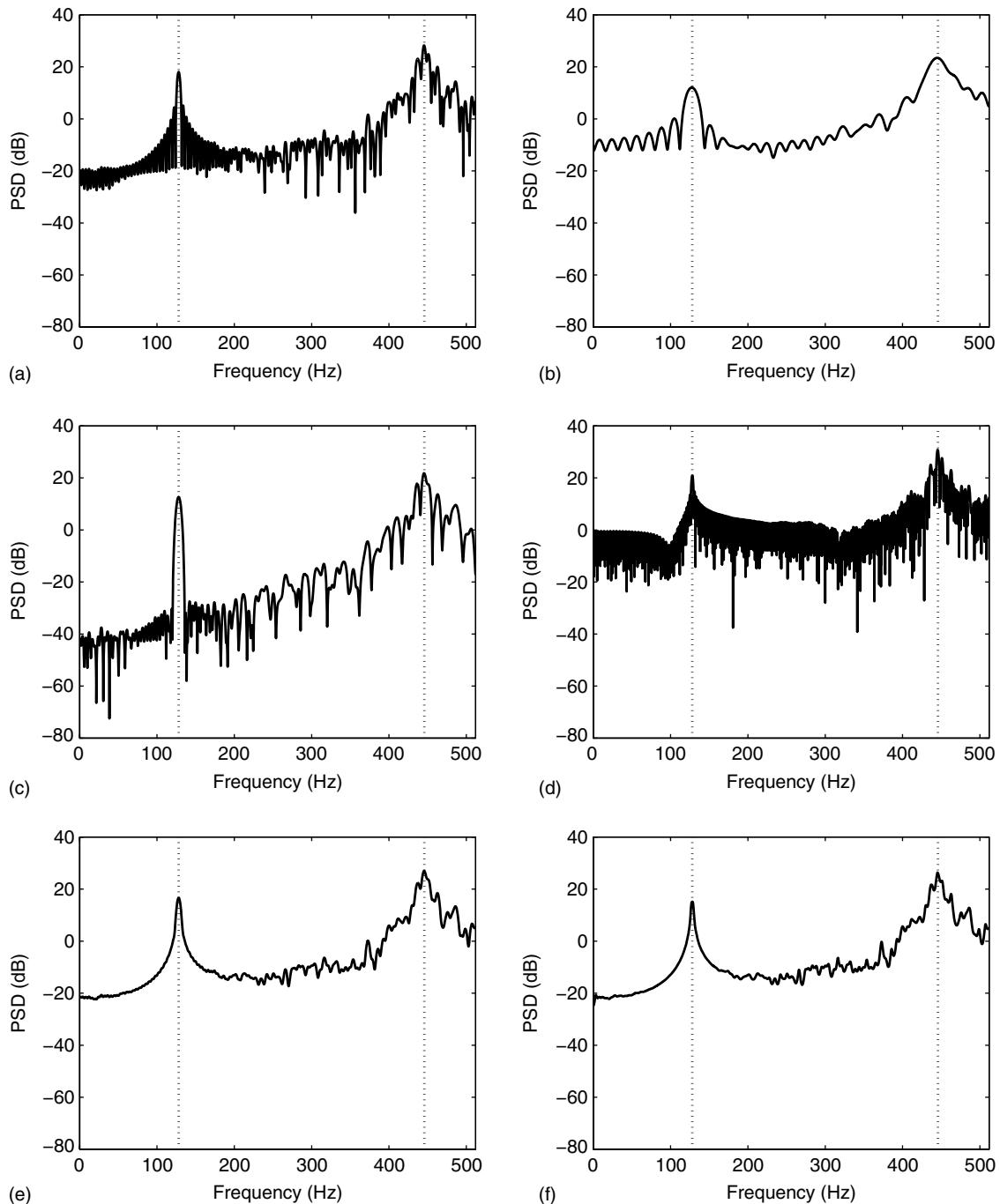


Fig. 7.10. PSD estimates for periodogram-based methods: (a) standard periodogram; (b) averaged periodogram; (c) windowed-data periodogram; (d) windowed-data periodogram with unbiased autocorrelation; (e) BT scheme; (f) minimum-variance method.

Another periodogram variation uses windowed data as given by

```
w = hamming(L)';
wy = w.*y;
Rwy = xcorr(wy,'biased');
Gamma_WY = 10*log10(abs(fft(Rwy,Lf)));
```

The PSD estimate for this is shown in Figure 7.10c. From this plot, one observes that the main effect of the data-windowing operation is to discriminate the sinusoid with less resolution and make the estimated curve a bit smoother, compared with the original periodogram estimate.

Using an unbiased estimate for the autocorrelation function of the windowed data, as given by

```
Rwyu = xcorr(wy,'unbiased');
Gamma_WYU = 10*log10(abs(fft(Rwyu,Lf)));
```

results in the PSD estimate seen in Figure 7.10d. As expected, this approach leads to a sharp peak in the vicinity of the sinusoid frequency with a very high variance around the entire frequency range.

Using the autocorrelation function Ry determined for the standard periodogram method, the BT estimator can be implemented as

```
RyBT = hamming(2*L-1)'.*Ry;
Gamma_BT = 10*log10(abs(fft(RyBT,Lf)));
```

As can be seen in Figure 7.10e, the PSD for the BT estimator is quite smooth and presents a broad peak associated with the sinusoidal component.

The MV method can be implemented in MATLAB according to the following script:

```
wy = y' - mean(y);
wy_pad = [zeros(L-1,1);wy;zeros(L-1,1)]';
for i=1:L,
    XU(:,i) = wy_pad(L-i+1:3*L-1-i);
end;
Rwy = XU'*XU/(L-1);
[eigvec,eigval] = eig(Rwy);
U = abs(fft(eigvec,L));
V = diag(inv(eigval + eps));
Gamma_MV = 10*log10(L)-10*log10(U*V);
```

This implementation follows the procedure described in Hayes (1996), which decomposes \mathbf{R}_Y^{-1} by its spectral decomposition (Diniz, 2008)

$$\mathbf{R}_Y^{-1} = \sum_{l=0}^{L-1} \frac{\mathbf{q}_l \mathbf{q}_l^*}{\lambda_l}, \quad (7.132)$$

where \mathbf{q}_l and λ_l , for $l = 1, 2, \dots, L$, are the eigenvectors and eigenvalues of \mathbf{R}_Y respectively. As a result, one can rewrite Equation (7.38) as

$$\Gamma_{Y,MV}(e^{j\omega}) = \frac{L}{\sum_{l=0}^{L-1} |e^{j\omega^T}(\mathbf{q}_l)|^2 / \lambda_l}. \quad (7.133)$$

The terms $\mathbf{e}^{*\top}(\mathbf{e}^{j\omega})\mathbf{q}_l$ represent the DFT of the l th eigenvector. In the script above, matrix \mathbf{U} stores the terms $|\text{FFT}[\mathbf{q}_l]|^2$ and the diagonal matrix \mathbf{V} stores the inverse of the eigenvalues of \mathbf{R}_Y^{-1} , with a conditioning factor eps to avoid division by zero. Using the MV method, the PSD estimate for a single run of the recorded data is depicted in Figure 7.10f, where we can observe a strong similarity, in this case, to the BT estimate. \triangle

Experiment 7.2

In this experiment we investigate the performances of some parametric methods for estimating the PSD of the random signal used in Experiment 7.1.

For the autocorrelation algorithm with order N , one may employ the script

```
Ry = xcorr(y,N,'biased');
Ryy = toeplitz(Ry(N+1:2*N));
pyy = Ry(N+2:2*N+1);
a = inv(Ryy)*pyy';
H_AC_dB = 20*log10(abs(freqz(1,[1; -a],LRy/2+1)));
```

with LRy as defined in Experiment 7.1, making the resulting PSD estimate compatible with the frequency vector freq determined in the same experiment. This command sequence is equivalent to

```
[b,E] = lpc(y,N);
```

with $b = [1; -a]$.

For other parametric approaches, we may resort to a powerful spectral-analysis tool in MATLAB whose usage is exemplified here for Burg's algorithm:

```
h = spectrum.burg(N,'UserDefined');
set(hopts,'NFFT',LRy);
HBurg = psd(h,y,hopts);
```

In most recent MATLAB versions, specifying the number of FFT points is made directly in the `psd` command. Therefore, the above lines shall be replaced by

```
h = spectrum.burg(N);
HBurg = psd(h,y,'NFFT',LRy);
```

In either case, the decibel scale can be enforced as

```
HBurg_dB = 10*log10(abs(HBurg.Data));
```

More details on the `spectrum` command can be found in Section 7.9.

Figure 7.11 shows the resulting PSDs for the autocorrelation and Burg's methods for $N = 4$ and $N = 8$ in each case, indicating the good performances achieved by both methods, particularly for larger values of N .

If, however, we move the sinusoid frequency f_2 closer to the ARMA pole frequencies, by making, for instance,

```
f2 = 3.3*fs/8; s2 = sin(2*pi*f2.*t);
y2 = z + s2;
```

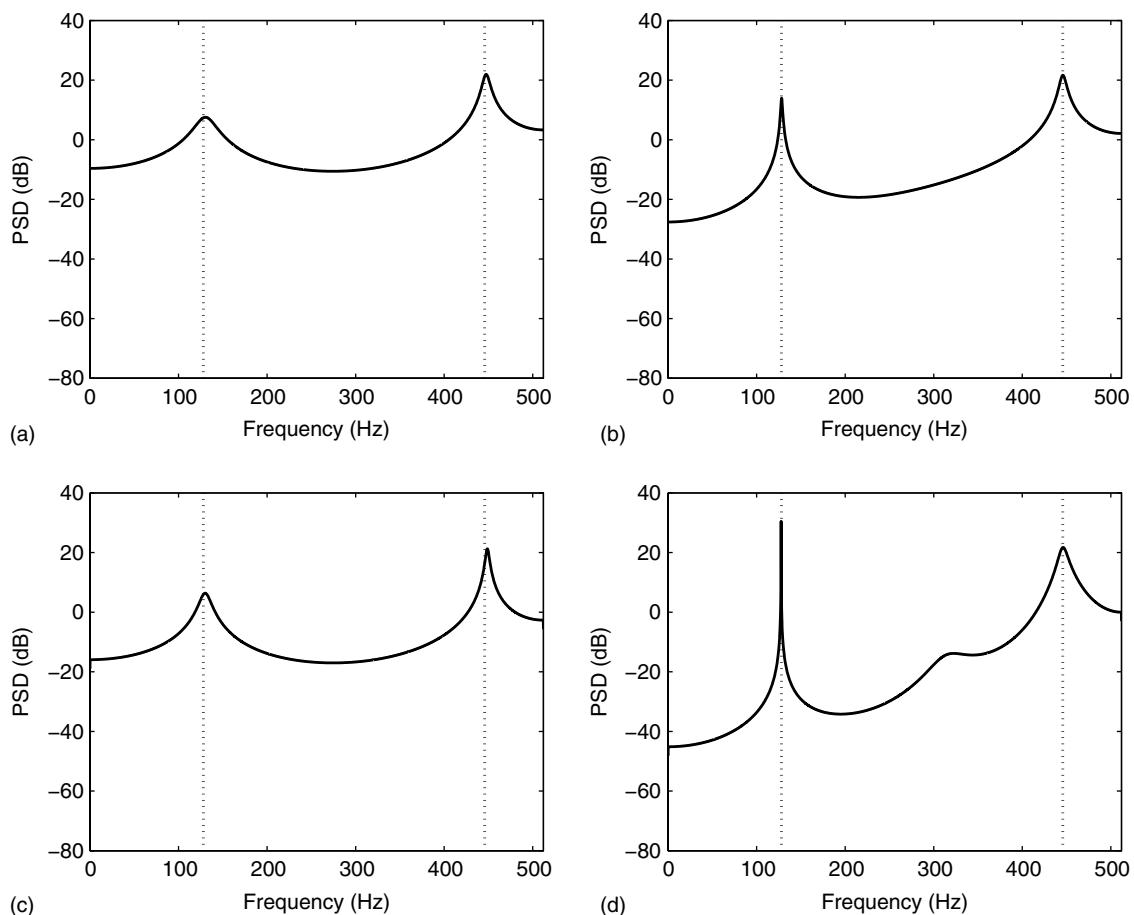


Fig. 7.11. PSD estimates for AR methods when $f_1 = f_s/8$: (a) autocorrelation with $N = 4$; (b) autocorrelation with $N = 8$; (c) Burg's algorithm with $N = 4$; (d) Burg's algorithm with $N = 8$.

then the resulting PSD estimates for the two methods with $N = 8$ and $N = 24$ are depicted in Figure 7.12. From these plots, one clearly notices how in this case both methods required a very large model order N to discriminate the two PSD peaks satisfactorily.

Figure 7.13 shows the pole-zero constellations for the AR model generated by the autocorrelation method with $N = 8$ and $N = 24$. In both cases, it can be observed that the AR estimate places some poles closer to the unit circle in the frequency range where the input signal has higher power, as expected. Although this phenomenon happened in both cases, when $N = 8$ the number of available poles was not enough to detect the presence of the sinusoid.

Figure 7.14 shows PSD estimates for 15 independent runs superimposed using the nonparametric BT and the parametric Burg (with $N = 24$) methods. As can be observed from the overlaid plots, the result for the nonparametric method, despite being able to locate better the sinusoid, has much higher variance than the parametric method. \triangle

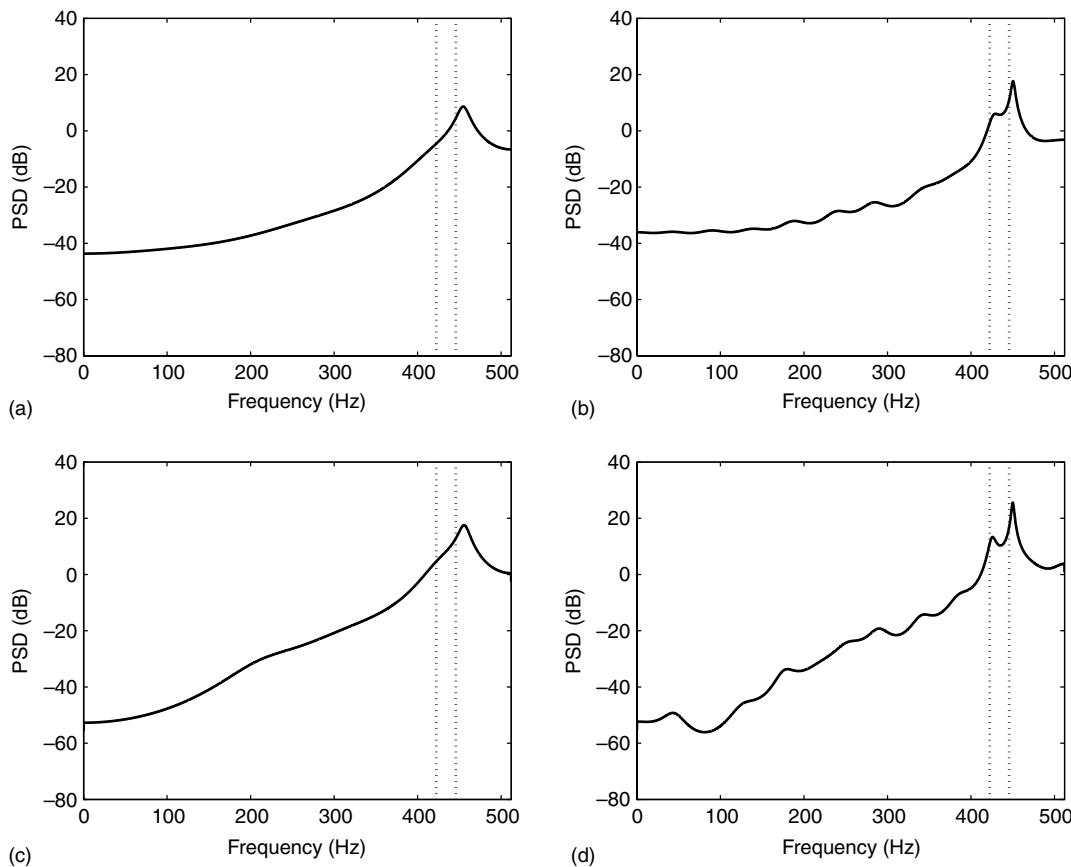


Fig. 7.12. PSD estimates for AR methods when $\epsilon_2 = 3.3\epsilon_s/8$: (a) autocorrelation with $N = 8$; (b) autocorrelation with $N = 24$; (c) Burg's algorithm with $N = 8$; (d) Burg's algorithm with $N = 24$.

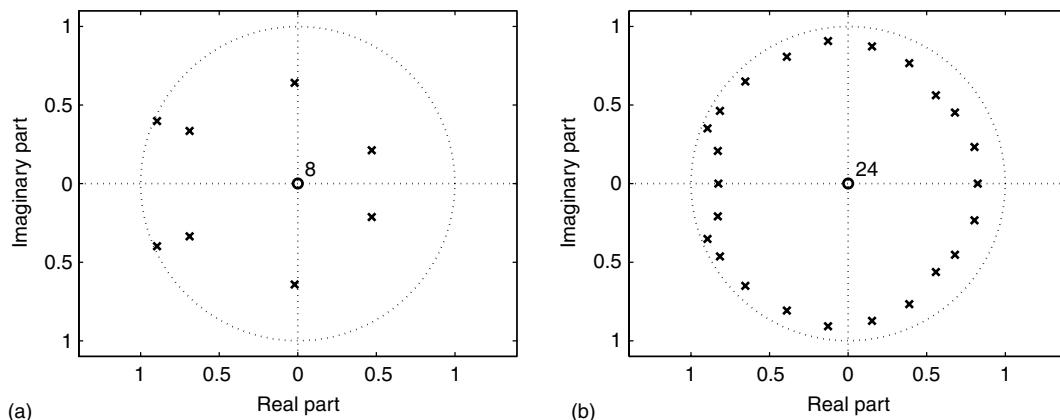


Fig. 7.13. Pole-zero plot for the AR models generated by the autocorrelation algorithm for Experiment 7.2: (a) $N = 8$; (b) $N = 24$.

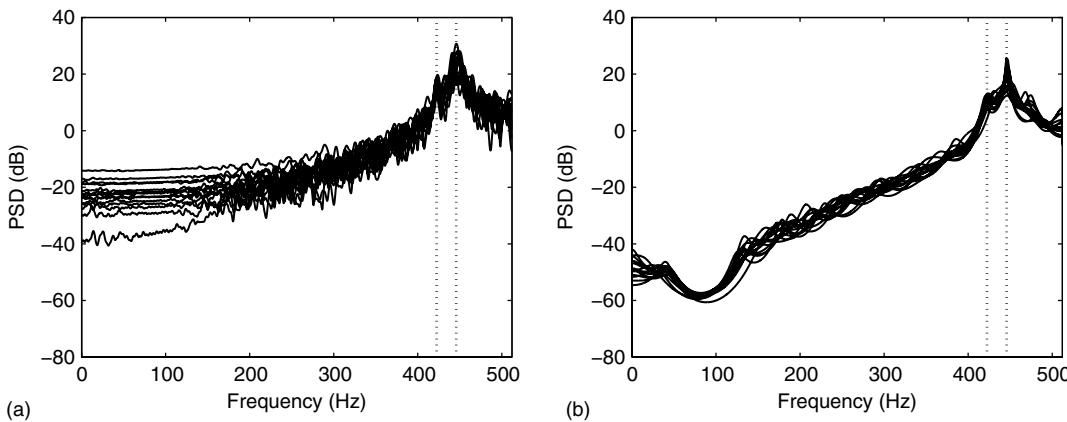


Fig. 7.14. PSD estimates for 15 independent runs in Experiment 7.2: (a) BT periodogram; (b) Burg's method with $N = 24$.

7.9 Spectral estimation with MATLAB

The main MATLAB tool for spectral estimation is the `spectrum` command. Below, we give a description of this command along with other auxiliary commands for spectral estimation, including commands from the MATLAB Spectral Estimation Toolbox.

- `aryule`: Determines the linear prediction coefficients of order N for a process $\{X\}$ solving the Yule–Walker equations using Levinson–Durbin algorithm.

Input parameters: process realization x and model order N .

Output parameters:

- denominator coefficients a of N th-order AR model;
- estimation error power E ;
- vector K of reflection coefficients.

Example:

```
x=filter(1,[1 1 0.8],randn(1000,1));
N=2;
[a,E,K] = aryule(x,N);
```

- `levinson`: Solves the Wiener–Hopf equation using the Levinson–Durbin algorithm.

Input parameters: vector Rx containing samples of the autocorrelation function of signal x starting at lag 0, and model order N .

Output parameters: the same as in the `aryule` command.

Example:

```
x=filter(1,[1 1 0.8],randn(1000,1));
N=2;
Rx=xcorr(x,N);
[a,E,K]=levinson(Rx(N+1:2*N+1),N);
```

which is equivalent to $[a, E] = lpc(x, N)$; or $[a, E, K] = aryule(x, N)$.

- **lpc:** Determines the linear prediction coefficients of order N for a process $\{X\}$ using the autocorrelation algorithm. Its usage is quite similar to the `aryule` command.
- **spectrum:** Performs the spectral estimation of a given process using algorithms such as periodogram (`periodogram`), windowed-data periodogram (`welch`), autocorrelation (`yulear`), covariance (`cov`), and the Burg (`burg`) methods. To specify the desired method, one must append the proper string (indicated before inside the parentheses) to the `spectrum` command. Once the estimator is obtained, the corresponding PSD function is determined through the `psd` command, as exemplified below.

Input parameter: for nonparametric algorithms, there is no required input argument, whereas for parametric methods one should provide the model order N.

Output parameter: the estimator h.

Example (periodogram method):

```
x=filter(1,[1 1 0.8],randn(1000,1));
hp=spectrum.periodogram;
PSDp=psd(hp,x);
plot(PSDp);
```

Example (Burg's method):

```
N=4;
hb=spectrum.burg(N);
PSDb=psd(hb,x);
plot(PSDb);
```

If one wants to specify the FFT length, the string 'UserDefined' should be included as an input argument and the auxiliary commands `psdopts` and `set` should be employed, as exemplified in the Do-it-yourself section of this chapter.

7.10 Summary

This chapter expanded the field of digital signal processing outside the scope of filter design. In that sense, the problem of estimating the power spectrum of a given random process was addressed. The basic ground for the estimation problem was laid down and the nonparametric periodogram method was introduced, along with several of its variations. The concept of the window function was applied in the context of smoothing the data or the autocorrelation function to yield better (with less bias or with reduced variance) PSD estimates. This chapter also addressed the minimum-variance nonparametric method, which, in general, results in smooth estimates of the PSD. The modeling problem, in which digital filters appear playing a major role, was then introduced as the basic tool for performing parametric spectral estimation. Our presentation focused on AR modeling owing to their general ability of mimicking MA and ARMA systems. Several methods for determining the AR model, including the so-called covariance, autocorrelation, and Burg approaches, were presented with different converging characteristics. The resulting PSD estimate can then be determined as the power response for the AR model obtained.

The chapter briefly addressed the important subject of Wiener filtering, where the optimum filter output produces an estimate of a reference signal by filtering another signal related to the reference. The objective function minimized by the optimum solution is the MSE between the estimated and reference signals. The chapter concluded with some reproducible experiments which allow the readers to test their learning by implementing the concepts presented.

7.11 Exercises

- 7.1 Use the periodogram algorithm to estimate the PSD of a length- L white-noise sequence generated by the MATLAB command `randn`. Average your results for N realizations of the white noise and verify the influences of L and N on the resulting estimate.
- 7.2 Use the periodogram algorithm to estimate the PSD of $L = 1024$ samples of the following signal:

$$x(n) = \cos \frac{2\pi}{20} n + x_1(n),$$

where

$$x_1(n) = -0.9x_1(n-1) + x_2(n),$$

where $x_2(n)$ is a white Gaussian noise with variance $\sigma_{X_2}^2 = 0.19$.

- 7.3 Use the periodogram algorithm to estimate the PSD of the following signal:

$$x(n) = \cos \frac{2\pi}{4} n + x_1(n),$$

where

$$x_1(n) = -0.9x_1(n-1) + x_2(n),$$

where $x_2(n)$ is a white Gaussian noise with variance $\sigma_{X_2}^2 = 0.19$. Use in this case $L = 512$, $L = 1024$, and $L = 2048$ and compare the results obtained in each case.

- 7.4 An AR process is generated by applying white Gaussian noise, with variance σ_X^2 , to a first-order filter with transfer function

$$H(z) = \frac{z}{z-a}.$$

This process has the autocorrelation matrix

$$\mathbf{R}_Y = \frac{\sigma_X^2}{1-a^2} \begin{bmatrix} 1 & a & \cdots & a^7 \\ a & 1 & \cdots & a^6 \\ \vdots & \vdots & \ddots & \vdots \\ a^7 & a^6 & \cdots & 1 \end{bmatrix}$$

whose inverse can be shown to be

$$\mathbf{R}_Y^{-1} = \frac{1}{\sigma_X^2} \begin{bmatrix} 1 & -a & \cdots & 0 & 0 \\ -a & 1+a^2 & \cdots & 0 & 0 \\ 0 & -a & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1+a^2 & -a \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix}.$$

For the signal above, calculate a closed-form solution for its minimum-variance estimate and comment on this solution when L approaches infinity.

- 7.5 In Exercise 7.4, consider $a = 0.8$ and plot the minimum-variance solution for window lengths equal to $L = 4$, $L = 10$, and $L = 50$. Compare the estimated PSD in each case with the actual PSD and comment on the results.
- 7.6 Use the minimum-variance method to estimate the PSD of $L = 256$ samples of the following signal:

$$y(n) = \sin \frac{\pi}{2} n + x_1(n),$$

where

$$x_1(n) = -0.8x_1(n-1) + x_2(n),$$

with $x_2(n)$ being a white Gaussian noise with variance $\sigma_{X_2}^2 = 0.36$.

- 7.7 Show that any stable ARMA system can be written as an MA model.
- 7.8 (a) Use the MATLAB command `roots` to determine the pole locations of the AR approximation, using $N = 1, 2, 3, 4$, for the ARMA system given in Example 7.3.
 (b) Find the AR impulse response for each value of N in the previous item and compare your results with the impulse response of the original ARMA system.
- 7.9 Find an N th-order AR approximation for the ARMA system

$$H(z) = \frac{1 - 1.5z^{-1}}{1 + 0.5z^{-1}}$$

and compare the resulting magnitude response to the original ARMA system. Observe that in this case the ARMA system presents a zero outside the unit circle.

- 7.10 Given two first-order AR processes generated by the same zero-mean white noise with unit variance whose respective poles are located at a_1 and $-a_1$, with $|a_1| < 1$, calculate the cross-correlation function between these AR processes.
- 7.11 Find an N th-order AR approximation for the ARMA system

$$H(z) = \frac{(1 - 0.8z^{-1})(1 - 0.9z^{-1})}{(1 - 0.5z^{-1})(1 + 0.5z^{-1})}$$

and compare the resulting magnitude response with the original ARMA system.

- 7.12 (a) Find an M th-order MA approximation for the ARMA system given in Example 7.3.
 (b) Use the `roots` command in MATLAB to determine the zero locations of the MA approximation for $M = 1, 2, 3, 4$.
 (c) Find the MA impulse response for each value of M and compare your results with the impulse response of the original ARMA system.
- 7.13 Show that, for a causal AR system with a zero-mean input $x(n)$ and output $y(n)$:

$$E\{x(n)y(n-\nu)\} = \begin{cases} \sigma_X^2, & \text{for } \nu = 0 \\ 0, & \text{for } \nu > 0. \end{cases}$$

- 7.14 Solve the Yule–Walker equations for a second-order AR system

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}},$$

determining $R_Y(\nu)$ for a white-noise input of variance σ_X^2 .

- 7.15 Solve the Yule–Walker equations provided by Equation (7.58) for the ARMA system given in Example 7.3, assuming a unit-variance white-noise input.
- 7.16 Determine the second-order optimal predictor in the minimum MSE sense for the output process of an MA system

$$H(z) = 1 + 2z^{-1} + 3z^{-2}$$

to a unit-variance white-noise input.

- 7.17 Show that the autocorrelation method yields an AR system whose poles are not outside the unit circle in the z plane.
- 7.18 Show, by a simple numerical example, that the covariance method may yield an AR system with poles outside the unit circle in the z plane.
- 7.19 Given two first-order AR processes, generated by the same zero-mean white noise with unit variance, whose respective poles are located at $z = 0.8$ and $z = -0.8$, generate a new process by adding the outcomes of these AR processes. Estimate the fourth-order AR model for the resulting process using the linear prediction method and comment on the results.
- 7.20 Given an MA process generated by applying a zero-mean white noise with unit variance to a system described by

$$H(z) = 0.921 - 1.6252z^{-1} + z^{-2},$$

estimate the third-order AR model for the resulting process using the linear prediction method and comment on the results.

- 7.21 Use the covariance method to estimate the PSD of $L = 1024$ samples for the following signal:

$$x(n) = \cos \frac{2\pi}{L} n + x_1(n),$$

where

$$x_1(n) = ax_1(n-1) + x_2(n),$$

with $x_2(n)$ being a white Gaussian noise with variance $1 - a^2$.

- (a) Choose $a = -0.9$ and $N = 4$.
- (b) Choose $a = -0.9$ and $N = 20$.
- (c) Choose $a = 0.9$ and $N = 4$.

Comment on your results.

7.22 Estimate the PSD of the signal

$$x(n) = e^{j(2\pi/8)n} + x_1(n)$$

with $x_1(n)$ being a white Gaussian noise with unit variance.

- 7.23 Solve Exercise 7.21 by estimating the AR parameters using the autocorrelation method.
- 7.24 Evaluate the number of floating-point operations (additions, subtractions, multiplications, and divisions) required by an N th-level Levinson–Durbin recursion, as described in Section 7.5.3.
- 7.25 Verify that the Burg reflection coefficients, as given in Equation (7.108), determine the minimum value of $\xi_{B,[i]}$ defined in Equation (7.102).
- 7.26 Solve Exercise 7.21 by estimating the AR parameters using the Levinson–Durbin recursions with the Burg reflection coefficients.
- 7.27 Show that the Burg reflection coefficients provided in Equation (7.108) are such that $|k_i| < 1$, for $i = 1, 2, \dots, N$. In addition, show that forward prediction-error power $\xi_{f,[i]}$ constitutes a decreasing sequence with respect to the system order $i = 1, 2, \dots, N$.
- 7.28 Determine closed-form expressions for the Levinson–Durbin reflection coefficients k_1 , k_2 , and k_3 as functions of $R_Y(v)$.
- 7.29 Estimate the PSD of the ARMA system output in Exercise 7.11 to a unit-variance white-noise input. Use the standard periodogram method and the autocorrelation method with $N = 1, 2, 3, 4$ and compare the results in each case.
- 7.30 Show that the minimum MSE value achieved by the Wiener solution is given by Equation (7.118).
- 7.31 A random process $x(n)$ is generated by applying a white noise $w(n)$ with unit variance as input to a system described by the following transfer function:

$$H(z) = \frac{1}{z^2 - 0.36}.$$

Compute the second-order Wiener filter that relates $x(n)$ to the output $y(n)$ of the filter

$$H_1(z) = \frac{1}{z + 0.6}$$

when the same $H_1(z)$ has $w(n)$ as input.

8.1 Introduction

In many applications of digital signal processing, it is necessary for different sampling rates to coexist within a given system. One common example is when two subsystems working at different sampling rates have to communicate and the sampling rates must be made compatible. Another case is when a wideband digital signal is decomposed into several nonoverlapping narrowband channels in order to be transmitted. In such a case, each narrowband channel may have its sampling rate decreased until its Nyquist limit is reached, thereby saving transmission bandwidth.

Here, we describe such systems which are generally referred to as multirate systems. Multirate systems are used in several applications, ranging from digital filter design to signal coding and compression, and have been increasingly present in modern digital systems.

First, we study the basic operations of decimation and interpolation, and show how arbitrary rational sampling-rate changes can be implemented with them. Then, we describe properties pertaining to the multirate systems, namely their valid inverse operations and the noble identities. With these properties introduced, the next step is to present the polyphase decompositions and the commutator models, which are key tools in multirate systems. The design of decimation and interpolation filters is also addressed. A step further is to deal with filter design techniques which use decimation and interpolation in order to achieve a prescribed set of filter specifications. In addition, some useful forms to represent single-input single-output (SISO) systems in block form are presented. The effects of multirate systems in random signals are also discussed in this chapter. Finally, MATLAB experiments and related functions which aid in the design and implementation of multirate systems are briefly described.

8.2 Basic principles

Intuitively, any sampling-rate change can be effected by recovering the band-limited analog signal $x_a(t)$ from its samples $x(m)$ and then resampling it with a different sampling rate, thus generating a different discrete version of the signal, $x'(n)$. Of course, the intermediate analog signal $x_a(t)$ must be filtered so that it can be resampled without aliasing. One possible way to do so is described here.

Suppose we have a digital signal $x(m)$ that was generated from an analog signal $x_a(t)$ with sampling period T_1 ; that is, $x(m) = x_a(mT_1)$, for $m \in \mathbb{Z}$. In order to avoid aliasing in the process, it is assumed that $x_a(t)$ is band-limited to $[-\pi/T_1, \pi/T_1]$. Therefore, replacing each sample of the signal by an impulse proportional to it, we have that the equivalent analog signal is

$$x_i(t) = \sum_{m=-\infty}^{\infty} x(m)\delta(t - mT_1), \quad (8.1)$$

whose spectrum is periodic with period $2\pi/T_1$. In order to recover the original analog signal $x_a(t)$ from $x_i(t)$, the repetitions of the spectrum must be discarded. Therefore, as seen in Section 1.6, $x_i(t)$ must be filtered with a filter $h(t)$ whose ideal frequency response $H(j\omega)$ is

$$H(j\omega) = \begin{cases} 1, & \omega \in [-\pi/T_1, \pi/T_1] \\ 0, & \text{otherwise} \end{cases} \quad (8.2)$$

and then

$$x_a(t) = x_i(t) * h(t) = \frac{1}{T_1} \sum_{m=-\infty}^{\infty} x(m) \operatorname{sinc}\left[\frac{\pi}{T_1}(t - mT_1)\right]. \quad (8.3)$$

Then, resampling $x_a(t)$ above with period T_2 to generate the digital signal $x'(n) = x_a(nT_2)$, for $n \in \mathbb{Z}$, we have that

$$x'(n) = \frac{1}{T_1} \sum_{m=-\infty}^{\infty} x(m) \operatorname{sinc}\left[\frac{\pi}{T_1}(nT_2 - mT_1)\right]. \quad (8.4)$$

This is the general equation governing sampling-rate changes. Observe that there is no restriction on the values of T_1 and T_2 . Of course, if $T_2 > T_1$ and aliasing is to be avoided, then the filter in Equation (8.2) must have a frequency response equal to zero for $\omega \notin [-\pi/T_2, \pi/T_2]$. As seen in Section 1.6, since Equation (8.4) consists of infinite summations involving the sinc function, it is not of practical use. In general, for rational sampling-rate changes, which covers most cases of interest, one can derive expressions working solely in the discrete-time domain. This is covered in the next sections, where three special cases are considered: decimation by an integer factor M , interpolation by an integer factor L , and sampling-rate change by a rational factor L/M .

8.3 Decimation

To decimate or subsample a digital signal $x(m)$ by a factor of M is to reduce its sampling rate M times. This is equivalent to keeping only every M th sample of the signal. This operation is represented as in Figure 8.1 and exemplified in Figure 8.2 for the case $M = 2$.

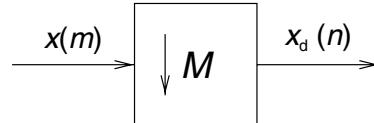


Fig. 8.1. Block diagram representing the decimation by a factor of M .

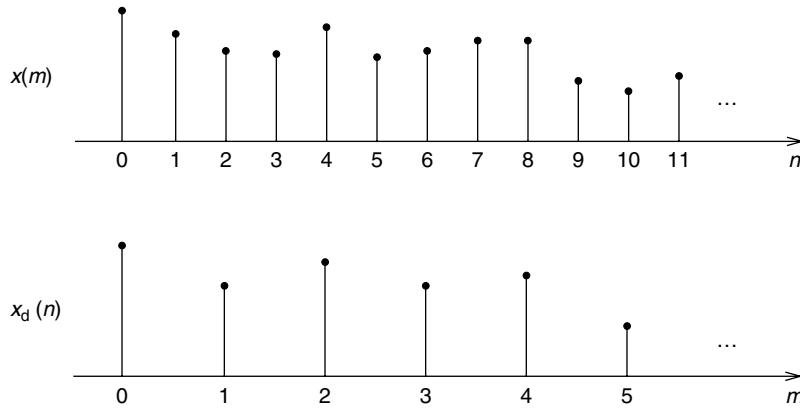


Fig. 8.2. Decimation by 2. $x(m) = \dots x(0) x(1) x(2) x(3) x(4) \dots$; $x_d(n) = \dots x(0) x(2) x(4) x(6) x(8) \dots$

The relation between the decimated signal and the original one is, therefore, very straightforward; that is:

$$x_d(n) = x(nM). \quad (8.5)$$

In the frequency domain, if the spectrum of $x(m)$ is $X(e^{j\omega})$, the spectrum of the decimated signal, $X_d(e^{j\omega})$, becomes

$$X_d(e^{j\omega}) = \frac{1}{M} \sum_{k=0}^{M-1} X\left[e^{j(\omega - 2\pi k)/M}\right]. \quad (8.6)$$

Such a result is reached by first defining $x'(m)$ as

$$x'(m) = \begin{cases} x(m), & m = nM, n \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}, \quad (8.7)$$

which can also be written as

$$x'(m) = x(m) \sum_{n=-\infty}^{\infty} \delta(m - nM) \quad (8.8)$$

The Fourier transform $X_d(e^{j\omega})$ is then given by

$$\begin{aligned}
 X_d(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x_d(n)e^{-j\omega n} \\
 &= \sum_{n=-\infty}^{\infty} x(nM)e^{-j\omega n} \\
 &= \sum_{n=-\infty}^{\infty} x'(nM)e^{-j\omega n} \\
 &= \sum_{l=-\infty}^{\infty} x'(l)e^{-j(\omega/M)l} \\
 &= X'(e^{j\omega/M}). \tag{8.9}
 \end{aligned}$$

But, from Equation (8.8) (see also Equation (2.237) and Exercise 2.15):

$$\begin{aligned}
 X'(e^{j\omega}) &= X(e^{j\omega}) \circledast \mathcal{F} \left\{ \sum_{n=-\infty}^{\infty} \delta(m - nM) \right\} \\
 &= X(e^{j\omega}) \circledast \frac{2\pi}{M} \sum_{k=0}^{M-1} \delta\left(\omega - \frac{2\pi k}{M}\right) \\
 &= \frac{1}{M} \sum_{k=0}^{M-1} X\left\{ e^{j[\omega - (2\pi k/M)]} \right\}. \tag{8.10}
 \end{aligned}$$

Then, from Equation (8.9),

$$X_d(e^{j\omega}) = X'\left(e^{j\omega/M}\right) = \frac{1}{M} \sum_{k=0}^{M-1} X\left[e^{j(\omega - 2\pi k)/M}\right], \tag{8.11}$$

which is the same as Equation (8.6).

As illustrated in Figure 8.3 for $M = 2$, Equation (8.6) means that the spectrum of $x_d(n)$ is composed of copies of the spectrum of $x(m)$ expanded by M and repeated with period 2π (which is equivalent to copies of the spectrum of $x(m)$ repeated with period $2\pi/M$ and then expanded by M). This implies that, in order to avoid aliasing after decimation, the bandwidth of the signal $x(m)$ must be limited to the interval $[-\pi/M, \pi/M]$. Therefore, the decimation operation is generally preceded by a lowpass filter (see Figure 8.4), which approximates the following frequency response:

$$H_d(e^{j\omega}) = \begin{cases} 1, & \omega \in [-\pi/M, \pi/M] \\ 0, & \text{otherwise} \end{cases} \tag{8.12}$$

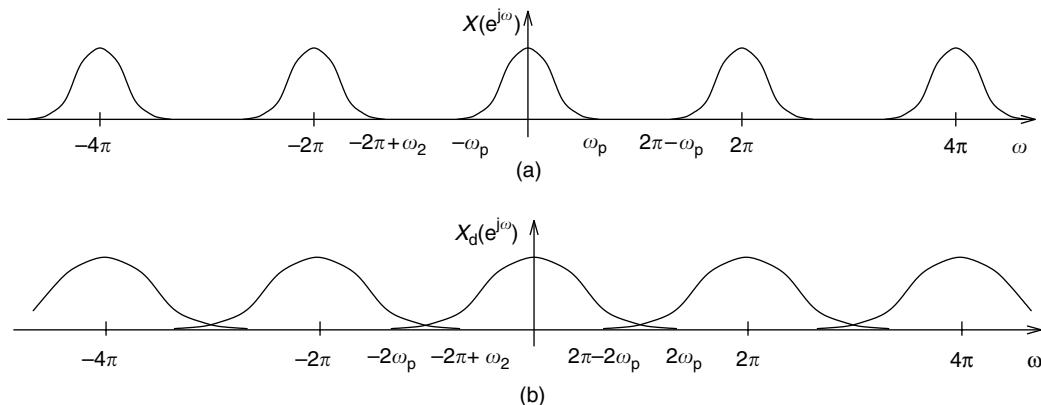


Fig. 8.3. Signal spectra of: (a) original digital signal; (b) decimated signal by a factor of 2.

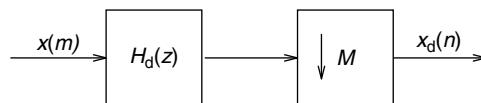


Fig. 8.4. General decimation operation.

If we then include the filtering operation, the decimated signal is obtained by retaining every M th sample of the convolution of the signal $x(m)$ with the filter impulse response $h_d(m)$; that is:

$$x_d(n) = \sum_{m=-\infty}^{\infty} x(m)h_d(nM - m). \quad (8.13)$$

Some important facts must be noted about the decimation operation (Crochiere & Rabiner, 1983):

- It is time varying; that is, if the input signal $x(m)$ is shifted, the output signal will not in general be a shifted version of the previous output. More precisely, let \mathcal{D}_M be the decimation-by- M operator. If $x_d(n) = \mathcal{D}_M\{x(m)\}$, then in general $\mathcal{D}_M\{x(m - k)\} \neq x_d(n - l)$, unless $k = rM$, when $\mathcal{D}_M\{x(m - k)\} = x_d(n - r)$. Because of this property, the decimation is referred to as a periodically time-invariant operation.
- Referring to Equation (8.13), one can see that, if the filter $H_d(z)$ is FIR, its outputs need only be computed every M samples, which implies that its implementation complexity is M times smaller than that of a usual filtering operation (Peled & Liu, 1985). This is not valid in general for IIR filters, because in such cases one needs all past outputs to compute the present output, unless the transfer function is of the type $H(z) = N(z)/D(z^M)$ (Martinez & Parks, 1979; Ansari & Liu, 1983).
- If the frequency range of interest for the signal $x(m)$ is $[-\omega_p, \omega_p]$, with $\omega_p < \pi/M$, one can afford aliasing outside this range. Therefore, the constraints upon the filter can be

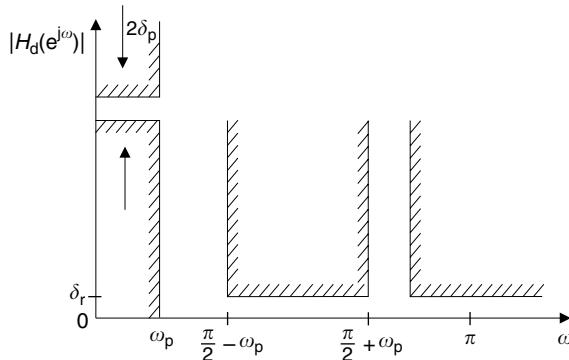


Fig. 8.5. Specifications of a decimation filter for $M = 4$.

relaxed, yielding the following specifications for $H_d(z)$:

$$H_d(e^{j\omega}) = \begin{cases} 1, & |\omega| \in [0, \omega_p] \\ 0, & |\omega| \in [(2\pi k/M) - \omega_p, (2\pi k/M) + \omega_p], \quad k = 1, 2, \dots, M-1. \end{cases} \quad (8.14)$$

The decimation filter can be efficiently designed using the optimum FIR approximation methods described in Chapter 5. In order to do so, one has to define the following parameters:

$$\left. \begin{array}{l} \delta_p : \text{passband ripple} \\ \delta_r : \text{stopband attenuation} \\ \omega_p : \text{passband cutoff frequency} \\ \omega_{r_1} = (2\pi/M) - \omega_p : \text{first stopband edge} \end{array} \right\}. \quad (8.15)$$

However, in general, it is more efficient to design a multiband filter according to Equation (8.14), as illustrated in Figure 8.5 and exemplified below.

Example 8.1. A signal that carries useful information only in the range $0 \leq \omega \leq 0.1\omega_s$ must be decimated by a factor of $M = 4$. Design a linear-phase decimation filter satisfying the following specifications:

$$\left. \begin{array}{l} \delta_p = 0.001 \\ \delta_r = 5 \times 10^{-5} \\ \Omega_s = 20\,000 \text{ Hz} \end{array} \right\}. \quad (8.16)$$

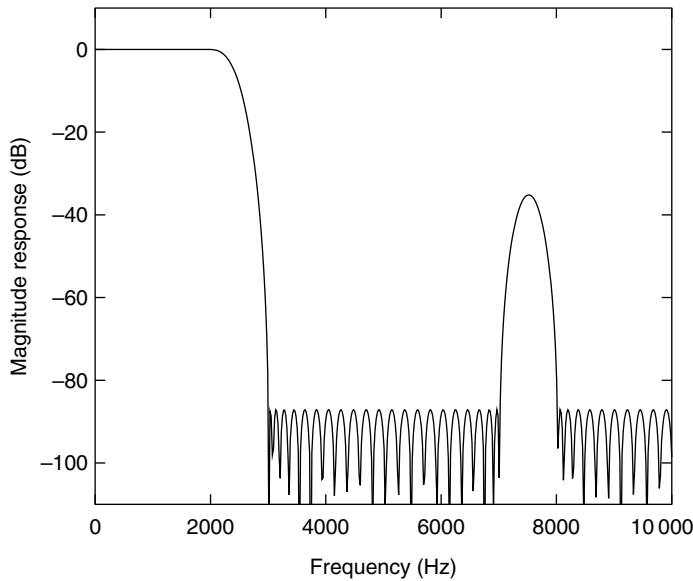
Solution

According to the specifications, the stopband edges of the decimation filter should be located at $((\Omega_s/4) - (\Omega_s/10))$ and $((\Omega_s/4) + (\Omega_s/10))$ in the first stopband, and $((\Omega_s/2) - (\Omega_s/10))$ in the second stopband. As a result, there is a “don’t care” band between $((\Omega_s/4) + (\Omega_s/10))$ and $((\Omega_s/2) - (\Omega_s/10))$. Designing the filter using the Chebyshev

Table 8.1.

Coefficient $h(0)$ to $h(42)$ for impulse response of the decimation filter.

$h(0) = 3.8208E-06$	$h(15) = 2.5170E-03$	$h(30) = -2.9906E-03$
$h(1) = -1.5078E-04$	$h(16) = 3.7016E-03$	$h(31) = 1.2798E-02$
$h(2) = -2.4488E-04$	$h(17) = 2.0456E-03$	$h(32) = 2.5575E-02$
$h(3) = -3.4356E-04$	$h(18) = -5.8022E-04$	$h(33) = 2.3561E-02$
$h(4) = -3.7883E-04$	$h(19) = -4.0164E-03$	$h(34) = 7.6551E-03$
$h(5) = 1.4857E-06$	$h(20) = -6.3092E-03$	$h(35) = -1.9703E-02$
$h(6) = 3.5092E-04$	$h(21) = -4.1002E-03$	$h(36) = -4.4867E-02$
$h(7) = 7.4044E-04$	$h(22) = 1.8340E-04$	$h(37) = -4.7659E-02$
$h(8) = 9.4756E-04$	$h(23) = 6.0511E-03$	$h(38) = -2.0785E-02$
$h(9) = 2.5364E-04$	$h(24) = 1.0189E-02$	$h(39) = 3.9424E-02$
$h(10) = -5.2335E-04$	$h(25) = 7.5145E-03$	$h(40) = 1.1942E-01$
$h(11) = -1.4509E-03$	$h(26) = 8.3120E-04$	$h(41) = 1.9216E-01$
$h(12) = -1.9966E-03$	$h(27) = -8.8128E-03$	$h(42) = 2.3804E-01$
$h(13) = -8.6587E-04$	$h(28) = -1.6037E-02$	
$h(14) = 6.3635E-04$	$h(29) = -1.3217E-02$	

**Fig. 8.6.**

Magnitude response of the decimation filter for $M = 4$.

approach from Section 5.6.2, we have that, in order to satisfy the specifications, the minimum required order is 85. The resulting magnitude response of one such filter is shown in Figure 8.6, where it can be observed that between 7000 and 8000 Hz is located a “don’t care” band as expected.

Table 8.1 shows the filter coefficients, where, since the filter is linear phase, only half of the coefficients are included. \triangle

8.4 Interpolation

To interpolate or upsample a digital signal $x(m)$ by a factor of L is to include $L - 1$ zeros between its samples. This operation is represented in Figure 8.7.

The interpolated signal is then given by

$$\hat{x}_i(n) = \begin{cases} x(n/L), & n = kL, k \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}. \quad (8.17)$$

The interpolation operation is exemplified in Figure 8.8 for the case $L = 2$.

In the frequency domain, if the spectrum of $x(m)$ is $X(e^{j\omega})$, it is straightforward to see that the spectrum of the interpolated signal $\hat{X}_i(e^{j\omega})$ becomes (Crochier & Rabiner, 1983)

$$\hat{X}_i(e^{j\omega}) = X(e^{j\omega L}). \quad (8.18)$$

Figure 8.9 shows the spectra of the signals $x(m)$ and $\hat{x}_i(n)$ for an interpolation factor of L .

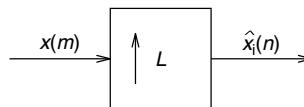


Fig. 8.7.

Interpolation by a factor of L .

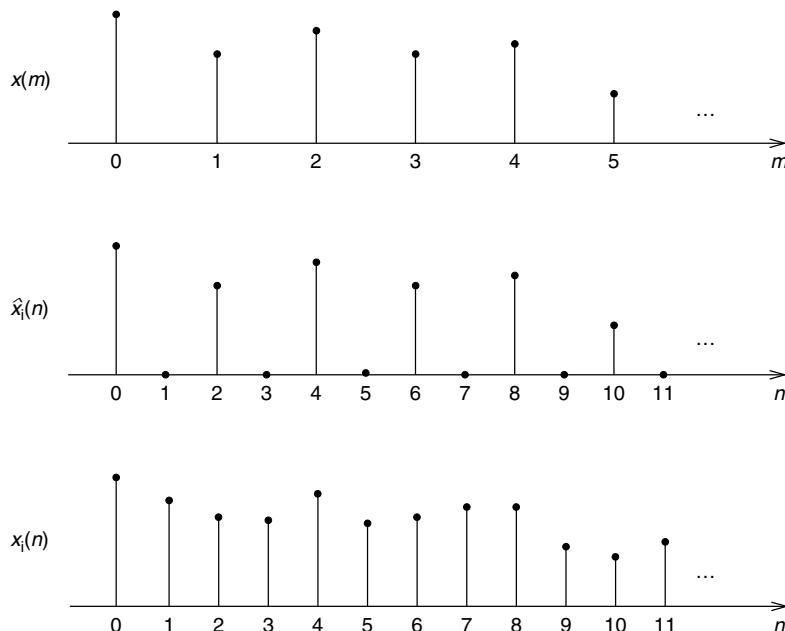


Fig. 8.8.

Interpolation by 2. $x(m)$: original signal; $\hat{x}_i(n)$: signal with zeros inserted between samples; $x_i(n)$: interpolated signal after filtering by $H_i(z)$. ($x(m) = \dots x(0) x(1) x(2) x(3) x(4) x(5) x(6) \dots$; $\hat{x}_i(n) = \dots x(0) 0 x(1) 0 x(2) 0 x(3) \dots$; $x_i(n) = \dots x_i(0) x_i(1) x_i(2) x_i(3) x_i(4) x_i(5) x_i(6) \dots$)

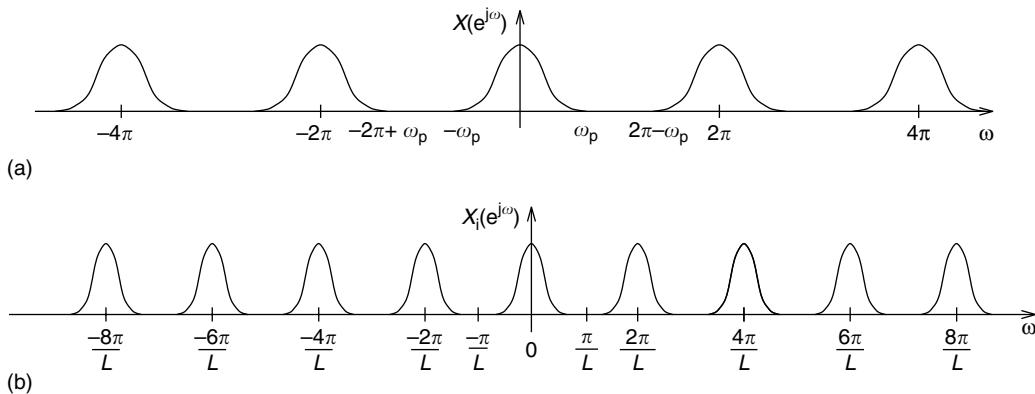


Fig. 8.9. Signal spectra of: (a) original digital signal; (b) interpolated signal by a factor of L .

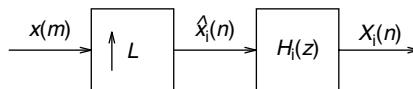


Fig. 8.10. General interpolation operation.

Since the spectrum of the original digital signal is periodic with period 2π , the spectrum of the interpolated signal has period $2\pi/L$. In order to obtain a smooth interpolated version of $x(m)$, the spectrum of the interpolated signal must be a compressed version of $X(e^{j\omega})$ in the frequency range $[-\pi, \pi]$, without any spectrum repetitions. This can be obtained by filtering out the repetitions of the spectrum of $\hat{x}_i(n)$ outside $[-\pi/L, \pi/L]$. Thus, the interpolation operation is generally followed by a lowpass filter (see Figure 8.10) which approximates the following frequency response:

$$H_i(e^{j\omega}) = \begin{cases} L, & \omega \in [-\pi/L, \pi/L] \\ 0, & \text{otherwise} \end{cases}. \quad (8.19)$$

The interpolation operation is thus equivalent to the convolution of the interpolation filter impulse response $h_i(n)$ with the signal $\hat{x}_i(n)$ defined in Equation (8.17). Considering that the only nonzero samples of $\hat{x}_i(n)$ are the ones having a multiple of L index, Equation (8.17) can be rewritten as

$$\hat{x}_i(kL) = \begin{cases} x(k), & k \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}. \quad (8.20)$$

With the help of the above equation, it is easy to see that, in the time domain, the filtered interpolated signal becomes

$$x_i(n) = \sum_{m=-\infty}^{\infty} \hat{x}_i(m)h(n-m) = \sum_{k=-\infty}^{\infty} x(k)h(n-kL). \quad (8.21)$$

Some important facts must be noted about the interpolation operation (Cochiere & Rabiner, 1983):

- As opposed to the decimation operation, the interpolation does not entail loss of information. More precisely, if \mathcal{I}_L is the interpolation-by- L operator, Equations (8.5) and (8.20) imply that $\mathcal{D}_L\{\mathcal{I}_L\{x(m)\}\} = x(m)$; that is, the interpolation operation is invertible. However, $\{\mathcal{I}_L\{x(m - k)\}\} = x_i(n - kL)$, which means that the interpolation is inherently time varying.
- Referring to Equation (8.21), one can see that the computation of the output of the filter $H_i(z)$ uses only one out of every L samples of the input signal, because the remaining samples are zero. This means that its implementation complexity can be made L times simpler than that of a usual filtering operation.
- If the signal $x(m)$ is band-limited to $[-\omega_p, \omega_p]$, the repetitions of the spectrum will only appear in a neighborhood of radius ω_p/L around the frequencies $2\pi k/L$, $k = 1, 2, \dots, L - 1$. Therefore, the constraints upon the filter can be relaxed as in the decimation case, yielding

$$H_i(e^{j\omega}) = \begin{cases} L, & |\omega| \in [0, \omega_p/L] \\ 0, & |\omega| \in [(2\pi k - \omega_p)/L, (2\pi k + \omega_p)/L], \quad k = 1, 2, \dots, L - 1. \end{cases} \quad (8.22)$$

The gain factor L in Equations (8.19) and (8.22) can be understood by noting that, since we are maintaining one out of every L samples of the signal, the average value of the signal decreases by a factor L and therefore, the gain of the interpolating filter must be L to compensate for this.

8.4.1 Examples of interpolators

Supposing $L = 2$, two common examples can be devised, as shown in Figure 8.11:

- Zero-order hold: $x(2n + 1) = x(2n)$. From Equation (8.21), this is equivalent to having $h(0) = h(1) = 1$; that is, $H_i(z) = 1 + z^{-1}$.
- Linear interpolator: $x(2n + 1) = \frac{1}{2}[x(2n) + x(2n + 2)]$. From Equation (8.21), this is equivalent to having $h(-1) = \frac{1}{2}$, $h(0) = 1$, and $h(1) = \frac{1}{2}$, that is, $H_i(z) = \frac{1}{2}(z + 2 + z^{-1})$.

Interesting examples of interpolators are the L th-band filters. They are filters that, when used as interpolators by L , keep the original samples of the signal to be interpolated. This can be stated more precisely by referring to Equation (8.21). There, if one decimates by L the interpolated signal $x_i(n)$ generated with an L th-band filter, one obtains the original signal $x(m)$. This is equivalent to saying that

$$x_i(mL) = x(m). \quad (8.23)$$

In this case, Equation (8.21) becomes

$$x_i(mL) = \sum_{k=-\infty}^{\infty} x(k)h(mL - kL) = x(m). \quad (8.24)$$

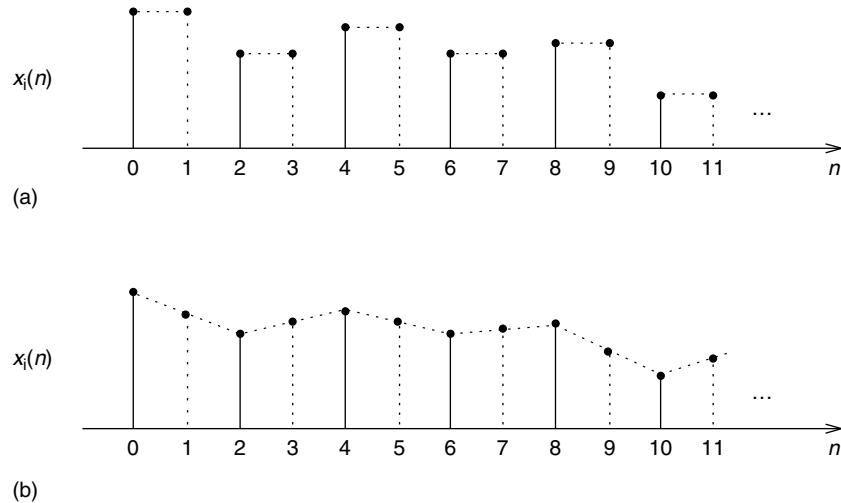


Fig. 8.11. Examples of interpolators: (a) zero-order hold; (b) linear interpolator.

This only happens if

$$h(mL - kL) = \begin{cases} 1, & m = k \\ 0, & m \neq k \end{cases}. \quad (8.25)$$

That is, the samples of $h(n)$ that are multiples of L are zero, except the one for $n = 0$, which should be equal to one.

Note that both the zeroth-order hold and the first-order hold are two-band filters, or, as is commonly said, half-band filters.

8.5 Rational sampling-rate changes

A rational sampling-rate change by a factor L/M can be implemented by cascading an interpolator by a factor of L with a decimator by a factor of M , as represented in Figure 8.12.

Since $H(z)$ is an interpolation filter, its cutoff frequency must be smaller than π/L . However, since it is also a decimation filter, its cutoff frequency must also be smaller than π/M . Therefore, it must approximate the following frequency response:

$$H(e^{j\omega}) = \begin{cases} L, & |\omega| \leq \min\{\pi/L, \pi/M\} \\ 0, & \text{otherwise} \end{cases}. \quad (8.26)$$

Similar to the cases of decimation and interpolation, the specifications of $H(z)$ can be relaxed if the bandwidth of the signal is smaller than ω_p . The relaxed specifications are the result of cascading the specifications in Equation (8.22) and the specifications in

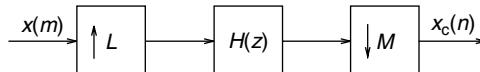


Fig. 8.12. Sampling rate change by a factor of L/M .

Equation (8.14) with ω_p replaced by ω_p/L . Since L and M can be assumed, without loss of generality, to be relatively prime, this yields

$$H(e^{j\omega}) = \begin{cases} L, & |\omega| < \min\{\omega_p/L, \pi/M\} \\ 0, & \min\{(2\pi/L) - (\omega_p/L), (2\pi/M) - (\omega_p/L)\} \leq |\omega| \leq \pi \end{cases}. \quad (8.27)$$

8.6 Inverse operations

At this point, a natural question to ask is: Are the decimation-by- M (\mathcal{D}_M) and interpolation-by- M (\mathcal{I}_M) operators inverses of each other? In other words, does $\mathcal{D}_M \mathcal{I}_M = \mathcal{I}_M \mathcal{D}_M =$ identity?

It is easy to see that $\mathcal{D}_M \mathcal{I}_M =$ identity, because the $(M - 1)$ zeros between samples inserted by the interpolation operation are removed by the decimation as long as the two operations are properly aligned, otherwise a null signal will result.

On the other hand, $\mathcal{I}_M \mathcal{D}_M$ is not the identity operator in general. This is so because the decimation operation removes $(M - 1)$ out of M samples of the signal and the interpolation operation inserts $(M - 1)$ zeros between samples. Then, their cascade is equivalent to replacing $(M - 1)$ out of M samples of the signal with zeros. However, if the decimation-by- M operation is preceded by a band-limiting filter for the interval $[-\pi/M, \pi/M]$ (see Equation (8.12)), and the interpolation operation is followed by the same filter (as illustrated in Figure 8.13), then $\mathcal{I}_M \mathcal{D}_M$ becomes the identity operation. This can be easily confirmed in the frequency domain, as the band-limiting filter avoids aliasing after decimation, which makes the decimation operation remain invertible. After interpolation by M , there are images of the spectrum of the signal in the intervals $[\pi k/M, \pi(k + 1)/M]$, for $k = -M, (-M + 1), \dots, (M - 1)$. However, the second band-limiting filter keeps only the image inside $[-\pi/M, \pi/M]$, which corresponds to the spectrum of the original signal.

We now discuss under which conditions the decimation and interpolation operations are commutative; that is, when the connection $\mathcal{D}_M \mathcal{I}_L$ as depicted in Figure 8.14a is equivalent to $\mathcal{I}_L \mathcal{D}_M$ shown in Figure 8.14b. We have already seen above that when $M = L$ they are not equivalent. Usually, these interconnections are not equivalent, unless M and L are relatively prime numbers. In the connection of Figure 8.14a, the output signal is given by

$$y(m) = \begin{cases} x(mM/L), & m = kL, k \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}, \quad (8.28)$$

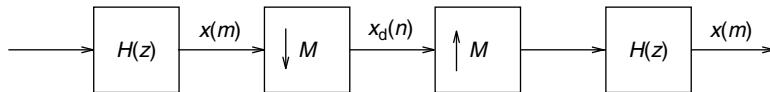


Fig. 8.13. Decimation followed by interpolation.

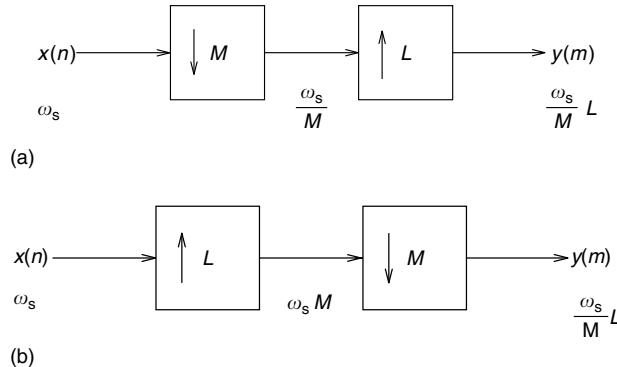


Fig. 8.14. Cascade operations: (a) decimation/interpolation; (b) interpolation/decimation.

whereas in the connection of Figure 8.14b the output signal is given by (see Exercise 8.2)

$$y(m) = \begin{cases} x(mM/L), & mM = kL, k \in \mathbb{Z} \\ 0, & \text{otherwise} \end{cases}. \quad (8.29)$$

Note that the condition in Equation (8.28), $m = kL, k \in \mathbb{Z}$, implies the condition in Equation (8.29); that is, $mM = kML = k'L, k' \in \mathbb{Z}$. On the other hand, the condition in Equation (8.29), $mM = kL, k \in \mathbb{Z}$, only implies that $m = k'L, k' \in \mathbb{Z}$ if M and L have no common multiple; that is, if they are relatively prime.

8.7 Noble identities

The noble identities are depicted in Figure 8.15. They have to do with the commutation of the filtering and decimation or interpolation operations, and are very useful in analyzing multirate systems and filter banks.

The identity in Figure 8.15a means that to decimate a signal by M and then filter it with $H(z)$ is equivalent to filtering the signal with $H(z^M)$ and then decimating the result by M . A filter $H(z^M)$ is one whose impulse response is equal to the impulse response of $H(z)$ with $(M - 1)$ zeros inserted between adjacent samples. Mathematically, it can be stated as

$$\mathcal{D}_M\{X(z)\}H(z) = \mathcal{D}_M\{X(z)H(z^M)\}, \quad (8.30)$$

where \mathcal{D}_M is the decimation-by- M operator.

The identity in Figure 8.15b means that to filter a signal with $H(z)$ and then interpolate it by M is equivalent to interpolating it by M and then filtering it with $H(z^M)$. Mathematically,

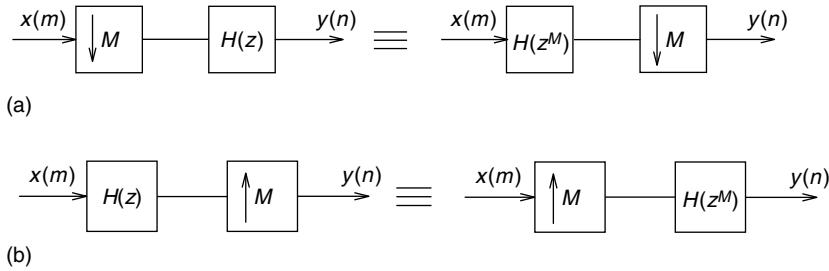


Fig. 8.15. Noble identities: (a) decimation; (b) interpolation.

it is stated as

$$\mathcal{I}_M \{X(z)H(z)\} = \mathcal{I}_M \{X(z)\}H(z^M), \quad (8.31)$$

where \mathcal{I}_M is the interpolation-by- M operator.

In order to prove the identity in Figure 8.15a, one begins by rewriting Equation (8.6), which gives the Fourier transform of the decimated signal $x_d(n)$ as a function of the input signal $x(m)$, in the z domain; that is:

$$X_d(z) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(z^{1/M} e^{-j2\pi k/M}\right). \quad (8.32)$$

For the decimator followed by filter $H(z)$, we have that

$$Y(z) = H(z)X_d(z) = \frac{1}{M} H(z) \sum_{k=0}^{M-1} X\left(z^{1/M} e^{-j2\pi k/M}\right). \quad (8.33)$$

For the filter $H(z^M)$ followed by the decimator, if $U(z) = X(z)H(z^M)$, then we have, from Equation (8.32), that

$$\begin{aligned} Y(z) &= \frac{1}{M} \sum_{k=0}^{M-1} U\left(z^{1/M} e^{-j2\pi k/M}\right) \\ &= \frac{1}{M} \sum_{k=0}^{M-1} X\left(z^{1/M} e^{-j2\pi k/M}\right) H\left(z e^{-j2\pi Mk/M}\right) \\ &= \frac{1}{M} \sum_{k=0}^{M-1} X\left(z^{1/M} e^{-j2\pi k/M}\right) H(z), \end{aligned} \quad (8.34)$$

which is the same as Equation (8.33), and the identity is proved.

Proof of the identity in Figure 8.15b is straightforward, as $H(z)$ followed by an interpolator gives $Y(z) = H(z^M)X(z^M)$, which is the same as the expression for an interpolator followed by $H(z^M)$.

8.8 Polyphase decompositions

The z transform $H(z)$ of a filter $h(n)$ can be written as

$$\begin{aligned}
 H(z) &= \sum_{k=-\infty}^{+\infty} h(k)z^{-k} \\
 &= \sum_{l=-\infty}^{+\infty} h(Ml)z^{-Ml} + \sum_{l=-\infty}^{+\infty} h(Ml+1)z^{-(Ml+1)} + \dots \\
 &\quad + \sum_{l=-\infty}^{+\infty} h(Ml+M-1)z^{-(Ml+M-1)} \\
 &= \sum_{l=-\infty}^{+\infty} h(Ml)z^{-Ml} + z^{-1} \sum_{l=-\infty}^{+\infty} h(Ml+1)z^{-Ml} + \dots \\
 &\quad + z^{-M+1} \sum_{l=-\infty}^{+\infty} h(Ml+M-1)z^{-Ml} \\
 &= \sum_{j=0}^{M-1} z^{-j} E_j(z^M).
 \end{aligned} \tag{8.35}$$

Equation (8.35) represents the polyphase decomposition (Vaidyanathan, 1993) of the filter $H(z)$ and

$$E_j(z) = \sum_{l=-\infty}^{+\infty} h(Ml+j)z^{-l} \tag{8.36}$$

are called the polyphase components of $H(z)$. In such a decomposition, the filter $H(z)$ is split into M filters: the first one with every sample of $h(m)$, whose indexes are multiples of M , the second one with every sample of $h(m)$, whose indexes are 1 plus a multiple of M , and so on.

Let us now analyze the basic operation of filtering followed by decimation represented in Figure 8.16a. Using the polyphase decomposition, such processing can be visualized as in Figure 8.16b, and applying the noble identity in Equation (8.30) we arrive at Figure 8.16c, which provides an interesting and useful interpretation of the operation represented in Figure 8.16a. In fact, Figure 8.16c shows that the whole operation is equivalent to filtering the samples of $x(m)$ whose indexes are equal to an integer k plus a multiple of M , with a filter composed of only the samples of $h(m)$ whose indexes are equal to the same integer k plus a multiple of M , for $k = 0, 1, \dots, M - 1$.

The polyphase decompositions also provide useful insights into the interpolation operation followed by filtering, as depicted in Figure 8.17. However, in this case, a variation

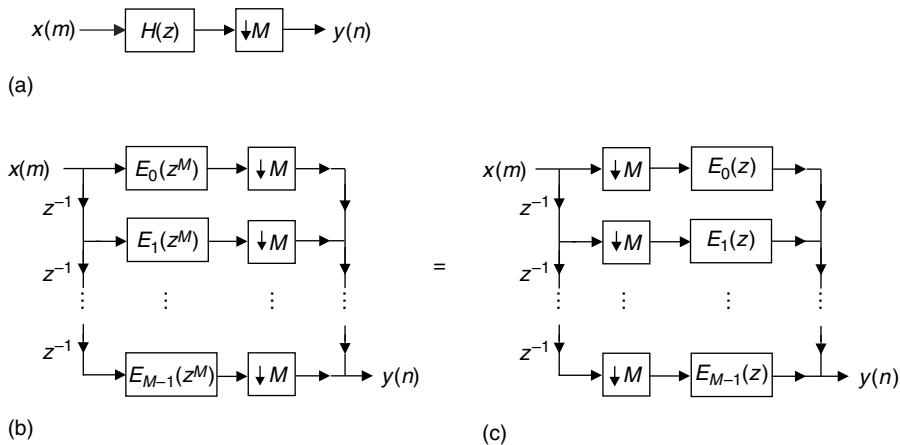


Fig. 8.16.

Decimation representations: (a) decimation by a factor of M ; (b) decimation using polyphase decompositions; (c) decimation using polyphase decompositions and the noble identities.

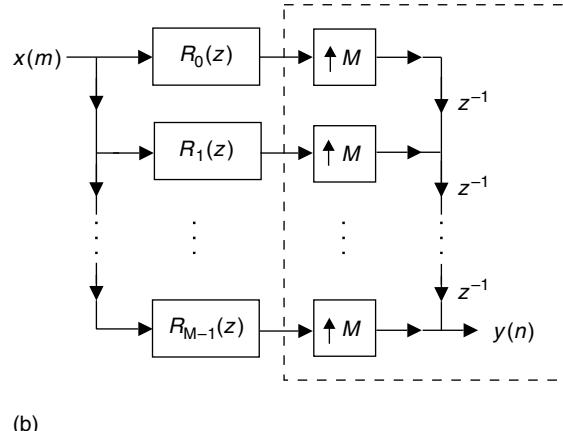
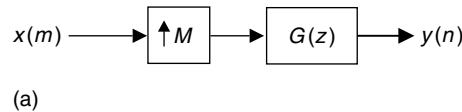


Fig. 8.17.

Interpolation representations: (a) interpolation by a factor of M ; (b) interpolation using polyphase decompositions and the noble identities.

of Equation (8.35) is usually employed. Defining $R_j(z) = E_{M-1-j}(z)$, the polyphase decomposition becomes

$$H(z) = \sum_{j=0}^{M-1} z^{-(M-1-j)} R_j(z^M). \quad (8.37)$$

Based on this equation, and applying the noble identity in Equation (8.31), the complete operation can be represented as depicted in Figure 8.17b.

8.9 Commutator models

The operations described respectively at the input and output of Figures 8.16c and 8.17b can also be interpreted in terms of rotary switches. These interpretations are referred to as commutator models. In them, the decimators and delays are replaced by rotary switches, as depicted in Figure 8.18 (Vaidyanathan, 1993).

In Figure 8.18a, the model with decimators and delays is noncausal, having “advances” instead of delays. However, in real-time systems, one wants to avoid noncausal operations. In these cases, the causal model of Figure 8.19 is usually preferred.

The operation depicted in Figure 8.19 is usually referred to as a serial-to-parallel converter. It can be expressed in matrix notation as

$$\mathbf{x}(m) = [x(mM) \quad x(mM - 1) \quad \dots \quad x(mM - M + 1)]^T. \quad (8.38)$$

Its inverse operation is the one depicted in Figure 8.18b. It is usually referred to as a parallel-to-serial converter.

Note that each $\mathbf{x}(m)$ is a block of M consecutive samples of $x(n)$. According to Equation (8.38), these blocks do not overlap, since there is no common sample between

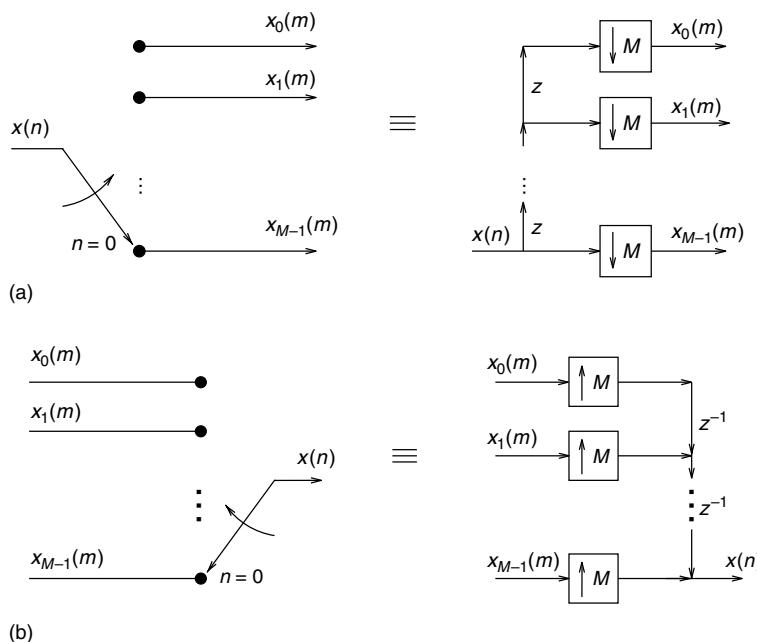


Fig. 8.18. Commutator models for: (a) decimation; (b) interpolation.

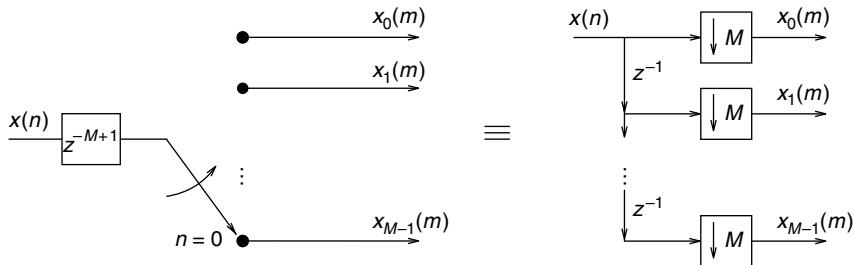


Fig. 8.19. Causal commutator model for decimation.

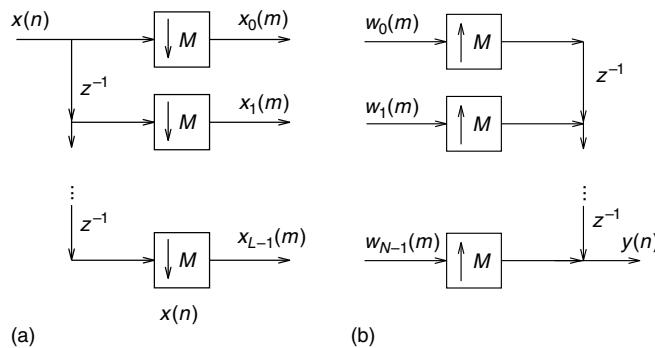


Fig. 8.20. Commutator models for: (a) division into overlapping blocks ($L > M$); (b) generating a signal by summation of overlapping blocks ($N > M$).

$x(m)$ and $x(m - 1)$. In addition, the last sample of $x(m)$ is consecutive to the first sample of $x(m - 1)$. This implies that indeed Equation (8.38) represents the splitting of $x(n)$ in nonoverlapping blocks of length M . Likewise, the inverse operation in Figure 8.18b is equivalent to putting the blocks side by side, recovering the signal $x(n)$.

If we generalize Equation (8.38) to

$$\mathbf{x}_L^M(m) = [x(mM) \quad x(mM - 1) \quad \cdots \quad x(mM - L + 1)]^T, \quad (8.39)$$

where $L > M$, then we have that there is an overlap between the samples of $\mathbf{x}_L^M(m)$ and $\mathbf{x}_L^M(m - 1)$. The last $(L - M)$ samples of $\mathbf{x}_L^M(m)$ are the same as the first $(L - M)$ samples of $\mathbf{x}_L^M(m - 1)$. That is, we divide the signal $x(n)$ into overlapping blocks. Note that, in this overlapping blocks case, the right-hand side of Figure 8.19 becomes Figure 8.20a.

We can express this operation more precisely if we define a unit delay operator $\mathcal{D}\{\cdot\}$ applied to a block $\mathbf{x}_L^M(m)$ as

$$\mathcal{D}\{\mathbf{x}_L^M(m)\} = [x(mM - 1) \quad x(mM - 2) \quad \cdots \quad x(mM - L)]^T. \quad (8.40)$$

Note that this delay operation displaces the start of the block by one sample, and thus $\mathcal{D}\{\mathbf{x}_L^M(m)\} \neq \mathbf{x}_L^M(m - 1)$. In fact:

$$\mathcal{D}^M\{\mathbf{x}_L^M(m)\} = \mathbf{x}_L^M(m - 1). \quad (8.41)$$

Using this definition, we can map a nonoverlapping block division to an overlapping block division for $M < L < 2M$ as follows:

$$\mathbf{x}_L^M(m) = \begin{bmatrix} \mathbf{I}_M & \mathbf{0} \\ \mathcal{D}^M \mathbf{I}_{L-M} & \mathbf{0} \end{bmatrix} \mathbf{x}_M^M(m) = \begin{bmatrix} \mathbf{I}_{L-M} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2M-L} \\ \mathcal{D}^M \mathbf{I}_{L-M} & \mathbf{0} \end{bmatrix} \mathbf{x}_M^M(m), \quad (8.42)$$

where \mathbf{I}_M is the $M \times M$ identity matrix.

Thus, if we consider the original signal $x(n)$ in vector form as the concatenation of the nonoverlapping blocks $\mathbf{x}_M^M(m)$; that is, if

$$\mathbf{x} = \left[\cdots \quad \mathbf{x}_M^{M^T}(m+1) \quad \mathbf{x}_M^{M^T}(m) \quad \mathbf{x}_M^{M^T}(m-1) \quad \cdots \right]^T, \quad (8.43)$$

then we can express the serial-to-parallel conversion in the overlapped case as

$$\mathbf{x}_L^M(m) = \underbrace{\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \dots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \cdots & \mathbf{0} & \mathbf{I}_{L-M} & \mathbf{0} & \mathbf{0} & \dots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{I}_{2M-L} & \mathbf{0} & \dots \\ \cdots & \mathbf{0} & \mathcal{D}^M \mathbf{I}_{L-M} & \mathbf{0} & \mathbf{0} & \dots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \cdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\mathbf{x}} \underbrace{\begin{bmatrix} \vdots \\ \mathbf{x}_M^M(m+1) \\ \vdots \\ \mathbf{x}_M^M(m) \\ \vdots \\ \mathbf{x}_M^M(m-1) \\ \vdots \end{bmatrix}}_{\mathbf{x}}$$
. (8.44)

Likewise, if we increase the number of branches in Figure 8.18b to $N > M$, we get Figure 8.20b. We can see that it is equivalent to generating a signal by summation of the overlapping blocks. The last $(N - M)$ samples of $\mathbf{x}_N^M(m)$ are added to the first $(N - M)$ samples of $\mathbf{x}_N^M(m-1)$. More precisely, this operation is equivalent to generating a block $\mathbf{y}_M^M(m)$ from $\mathbf{w}_N^M(m)$ as

$$\begin{aligned} \mathbf{y}_M^M(m) &= \begin{bmatrix} \mathbf{0} & \mathbf{I}_M \\ \mathcal{D}^M \mathbf{I}_{N-M} & \mathbf{0} \end{bmatrix} \mathbf{w}_N^M(m) \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} \\ \mathcal{D}^M \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} \end{bmatrix} \mathbf{w}_N^M(m). \end{aligned} \quad (8.45)$$

Thus, if we consider the output signal $y(n)$ in vector form as the concatenation of the nonoverlapping blocks $\mathbf{y}_M^M(m)$, namely

$$\mathbf{y} = \left[\cdots \quad \mathbf{y}_M^{M^T}(m+1) \quad \mathbf{y}_M^{M^T}(m) \quad \mathbf{y}_M^{M^T}(m-1) \quad \cdots \right]^T, \quad (8.46)$$

then by substituting Equation (8.45) into Equation (8.46) we can express the parallel-to-serial conversion in the overlapped case as

$$\begin{aligned}
 \mathbf{y} &= \begin{bmatrix} \vdots \\ \mathbf{y}_M^M(m+1) \\ \mathbf{y}_M^M(m) \\ \mathbf{y}_M^M(m-1) \\ \vdots \end{bmatrix} \\
 &= \begin{bmatrix} \ddots & \vdots & \cdots \\ \cdots & \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathcal{D}^M \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathcal{D}^M \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathcal{D}^M \mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} & \cdots \\ \cdots & \vdots & \ddots \end{bmatrix} \\
 &\times \begin{bmatrix} \vdots \\ \mathbf{w}_N^M(m+1) \\ \mathbf{w}_N^M(m) \\ \mathbf{w}_N^M(m-1) \\ \vdots \end{bmatrix}. \tag{8.47}
 \end{aligned}$$

8.10 Decimation and interpolation for efficient filter implementation

In Chapter 12, efficient structures for FIR filters will be analyzed in detail. In this section, we show how the concepts of decimation and interpolation can be employed to generate efficient FIR filter implementations with respect to the number of multiplications per output sample. First, we deal with the case of efficiently implementing a narrowband FIR filter. Then, we give a brief introduction to the frequency response masking approach, which will be dealt with in detail in Section 12.7.3.

8.10.1 Narrowband FIR filters

Consider the system in Figure 8.21, consisting of the cascade of a decimator and an interpolator by M .

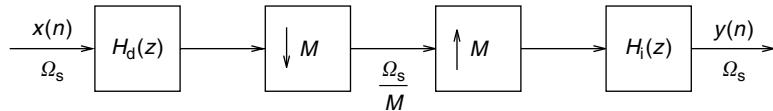


Fig. 8.21. Filter using decimation/interpolation.

From Equations (8.6) and (8.18), one can easily infer the relation between the Fourier transforms of $y(n)$ and $x(n)$, which is

$$Y(e^{j\omega}) = \frac{H_i(e^{j\omega})}{M} \left(\sum_{k=0}^{M-1} \left\{ X(e^{j[\omega - (2\pi k/M)]}) H_d(e^{j[\omega - (2\pi k/M)]}) \right\} \right). \quad (8.48)$$

Supposing that both the decimation filter $H_d(z)$, and the interpolation filter $H_i(z)$, have been properly designed, the spectrum repetitions in the above equation are canceled, yielding the following relation:

$$\frac{Y(e^{j\omega})}{X(e^{j\omega})} = \frac{H_d(e^{j\omega}) H_i(e^{j\omega})}{M} = H(e^{j\omega}). \quad (8.49)$$

This result shows that the cascading of the decimation and interpolation operations of the same order M is equivalent to just cascading the decimation and interpolation filters, provided that both bandwidths are smaller than π/M . At a first glance, this structure is entirely equivalent to cascading two filters, and it presents no special advantage. However, one must bear in mind that, in the implementation of the decimation operation, there is a reduction by M in the number of multiplications, and the same is true for the interpolation operation. Therefore, this structure can provide a dramatic reduction in the overall number of multiplications. Actually, this reduction increases with the value of M , and we should choose M as large as possible, such that the bandwidth of the desired filter remains smaller than π/M .

If we wish to design a filter with passband ripple δ_p and stopband ripple δ_r , it is sufficient to design interpolation and decimation filters, each having passband ripple $\delta_p/2$ and stopband ripple δ_r . This specification is justified by the fact that, if we assume that the aliasing effect caused by the decimation is negligible, the sole task of the interpolation filter is to eliminate the spectrum repetitions caused by the interpolator. Assuming the lowpass band has magnitude close to one, its repetitions will have the same gain. As such, the interpolation filter has to attenuate the repetition to the level of the prescribed stopband attenuation.

Example 8.2. Using the concepts of decimation and interpolation, design a lowpass filter satisfying the following specifications:

$$\left. \begin{array}{l} \delta_p = 0.001 \\ \delta_r = 1 \times 10^{-3} \\ \Omega_p = 0.025\Omega_s \\ \Omega_r = 0.045\Omega_s \\ \Omega_s = 2\pi \text{ rad/s} \end{array} \right\}. \quad (8.50)$$

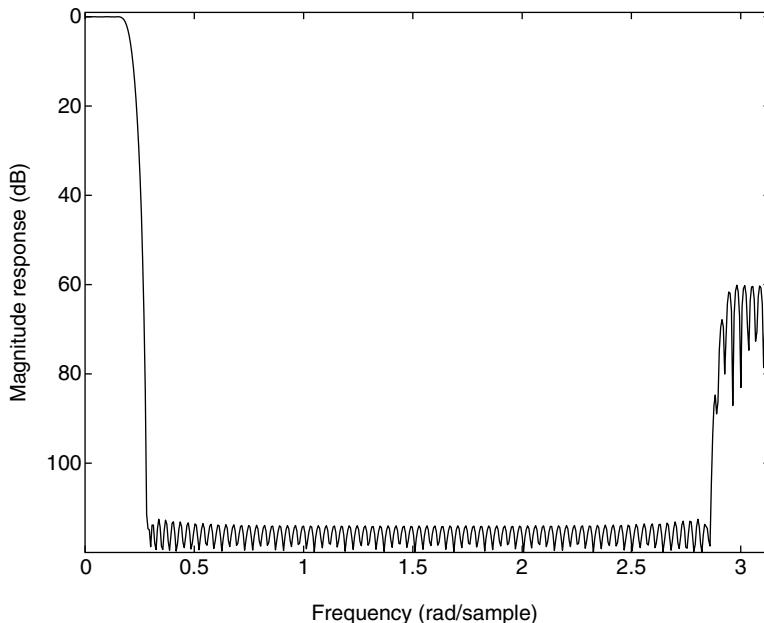


Fig. 8.22. Magnitude response of filter using decimation/interpolation.

Solution

With the given set of specifications, the maximum possible value of M is 11. Using the Chebyshev (minimax) method, $H_d(z)$ and $H_i(z)$ can be made identical, and they must be at least of order 177 each. The magnitude response for the cascade of $H_d(z)$, decimator, interpolator, and $H_i(z)$ is shown in Figure 8.22, and the corresponding coefficients of both subfilters are given in Table 8.2. Once again, only half of the coefficients are listed, as the filters have linear phase. As can be observed by the reader, the cascading of these two filters satisfies the problem specifications and in most of the stopband the attenuation is much higher than prescribed, leaving room for further reduction in complexity.

With the conventional approach, also using the Chebyshev method, the total number of multiplications per sample would be 87 (as a linear-phase filter of order 173 would be required). Using decimation and interpolation, the total number of multiplications per output sample is only 178/11 (89/11 for the decimation and 89/11 for the interpolation), a significant reduction in the overall complexity. In fact, even greater reductions in complexity can be achieved if the decimators and interpolators in Figure 8.21 are composed of several decimation stages followed by several interpolation stages. △

Although we have considered only lowpass design, the procedure of Figure 8.21 can also be used to design narrowband bandpass filters. All we have to do is to choose M such that the desired filter passband and transition bands are contained in an interval of the form $[i\pi/M, (i+1)\pi/M]$, for only one value of i (Cochiere & Rabiner, 1983). In such cases, the interpolation and decimation filters are bandpass (see Section 9.2.1). Highpass and bandstop filters can be implemented based on lowpass and bandpass designs.

Table 8.2.Interpolator and decimator filter coefficients $h(0)$ to $h(88)$.

$h(0) = 5.3448E-04$	$h(30) = 8.0055E-04$	$h(60) = 5.1321E-04$
$h(1) = 8.1971E-05$	$h(31) = 4.6245E-04$	$h(61) = -1.4963E-03$
$h(2) = 6.9925E-05$	$h(32) = 5.4968E-05$	$h(62) = -3.6515E-03$
$h(3) = 4.4127E-05$	$h(33) = -4.0663E-04$	$h(63) = -5.8547E-03$
$h(4) = 4.6053E-06$	$h(34) = -9.0159E-04$	$h(64) = -7.9954E-03$
$h(5) = -4.79628E-05$	$h(35) = -1.4053E-03$	$h(65) = -9.9540E-03$
$h(6) = -1.1146E-04$	$h(36) = -1.8894E-03$	$h(66) = -1.1607E-02$
$h(7) = -1.8289E-04$	$h(37) = -2.3239E-03$	$h(67) = -1.2831E-02$
$h(8) = -2.5796E-04$	$h(38) = -2.6777E-03$	$h(68) = -1.3510E-02$
$h(9) = -3.3170E-04$	$h(39) = -2.9218E-03$	$h(69) = -1.3539E-02$
$h(10) = -3.9818E-04$	$h(40) = -3.0295E-03$	$h(70) = -1.2831E-02$
$h(11) = -4.5128E-04$	$h(41) = -2.9798E-03$	$h(71) = -1.1320E-02$
$h(12) = -4.8464E-04$	$h(42) = -2.7575E-03$	$h(72) = -8.9673E-03$
$h(13) = -4.9255E-04$	$h(43) = -2.3572E-03$	$h(73) = -5.7634E-03$
$h(14) = -4.6997E-04$	$h(44) = -1.7801E-03$	$h(74) = -1.7298E-03$
$h(15) = -4.1343E-04$	$h(45) = -1.0415E-03$	$h(75) = 3.0794E-03$
$h(16) = -3.2098E-04$	$h(46) = -1.6455E-04$	$h(76) = 8.5785E-03$
$h(17) = -1.9311E-04$	$h(47) = 8.1790E-04$	$h(77) = 1.4651E-02$
$h(18) = -3.2495E-05$	$h(48) = 1.8613E-03$	$h(78) = 2.1156E-02$
$h(19) = 1.5538E-04$	$h(49) = 2.9146E-03$	$h(79) = 2.7927E-02$
$h(20) = 3.6273E-04$	$h(50) = 3.9205E-03$	$h(80) = 3.4782E-02$
$h(21) = 5.7917E-04$	$h(51) = 4.8191E-03$	$h(81) = 4.1531E-02$
$h(22) = 7.9256E-04$	$h(52) = 5.5495E-03$	$h(82) = 4.7975E-02$
$h(23) = 9.8902E-04$	$h(53) = 6.0542E-03$	$h(83) = 5.3925E-02$
$h(24) = 1.1542E-03$	$h(54) = 6.2813E-03$	$h(84) = 5.9197E-02$
$h(25) = 1.2735E-03$	$h(55) = 6.1893E-03$	$h(85) = 6.3630E-02$
$h(26) = 5.7490E-03$	$h(56) = 1.3336E-03$	$h(86) = 6.7084E-02$
$h(27) = 1.3229E-03$	$h(57) = 4.9473E-03$	$h(87) = 6.9450E-02$
$h(28) = 1.2330E-03$	$h(58) = 3.7883E-03$	$h(88) = 7.0652E-02$
$h(29) = 1.0589E-03$	$h(59) = 2.2958E-03$	

8.10.2 Wideband FIR filters with narrow transition bands

Another interesting application of interpolation is in the design of sharp cutoff filters with low computational complexity using the so-called frequency-response masking approach (Lim, 1986), which uses the fact that an interpolated filter has a transition band L times smaller than the prototype filter. The complete process is sketched in Figure 8.23, for an interpolation ratio of $L = 4$, and exemplified as follows.

Suppose, for instance, that we want to design a normalized-lowpass filter having $\omega_p = 5\pi/8$ and $\omega_r = 11\pi/16$, such that the transition width is $\pi/16$. Using the frequency-response masking approach, we design such a filter starting from a prototype half-band lowpass filter having $\omega_p = \pi/2$ and a transition bandwidth four times larger than the one

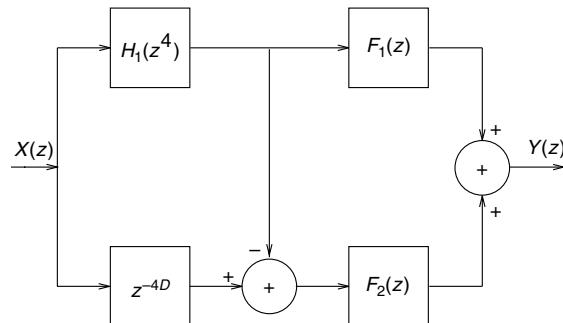
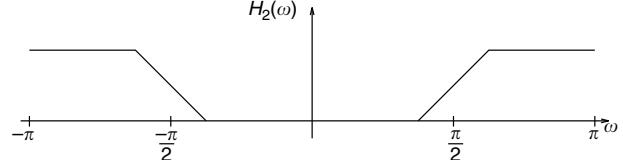
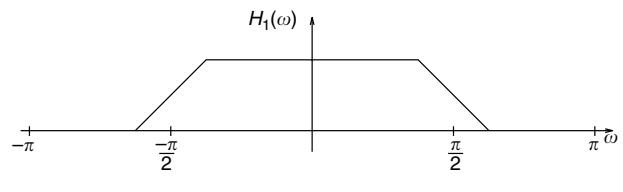
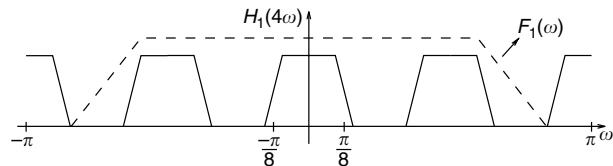


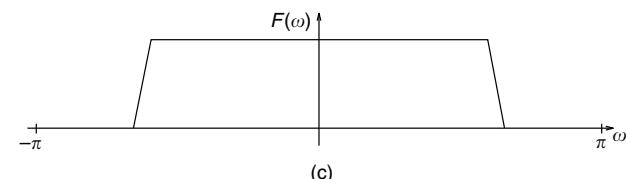
Fig. 8.23. Filter design with the frequency response masking approach using interpolation.



(a)



(b)



(c)

Fig. 8.24. (a) Prototype half-band filter $H_1(z)$ and its complementary filter $H_2(z)$; (b) frequency responses of $H_1(z)$ and $H_2(z)$ after interpolation by a factor of $L = 4$; (c) frequency response of the equivalent filter $F(z)$.

needed, in this case, $\pi/4$. Therefore, the implementation complexity of this prototype filter is much smaller than the original one. From this prototype, the complementary filter $H_2(z)$ is generated by a simple delay and subtraction as

$$H_2(z) = z^{-D} - H_1(z). \quad (8.51)$$

In Figure 8.23, $H_2(z^4)$ would represent the transfer function between the output of the summation following the delay z^{-4D} and $X(z)$.

The magnitude responses of $H_1(z)$ and $H_2(z)$ are illustrated in Figure 8.24a. After interpolation, their responses are related as shown in Figure 8.24b. The filters $F_1(z)$ and $F_2(z)$ are then used to select the parts of the interpolated spectra of $H_1(z)$ and $H_2(z)$ that will be used in composing the desired filter response $F(z) = Y(z)/X(z)$. It is interesting to note that $F_1(z)$ and $F_2(z)$, besides being interpolation filters, are allowed to have large transition bandwidths and, therefore, have very low implementation complexity. As can be seen from Figure 8.24c, one can then generate large bandwidth sharp cutoff filters with low implementation complexity.

8.11 Overlapped block filtering

In this section we use the division of a signal into overlapping blocks represented in Figures 8.20a and 8.20b in order to analyze filtering operations performed in blocks of a signal, either with or without overlap. We refer collectively to such operations as overlapped block filtering. We start this section by describing forms of representing block operations, emphasizing the constraints that the operations performed on the blocks should satisfy in order for the overall system to be linear and time invariant. We then discuss parallel implementations of FIR filters using overlapped block filtering. We finish this section by describing two fast FIR filtering methods from Vetterli (1988), Mou & Duhamel (1991), and Lin & Mitra (1996), using the overlapped block filtering framework. The concepts discussed in this section find applications in block-based digital communications as well as in parallel implementation of digital signal processing systems.

Consider the system represented in Figure 8.25. By referring to Figures 8.20a and 8.20b, we can see that in Figure 8.25 the input signal is divided into blocks of length L that have an overlap of $(L - M)$ samples. After processing, each length- L block is mapped into a length- N block. The output signal is generated by summing these blocks with an overlap of $(N - M)$ samples. The $N \times L$ matrix $\mathbf{C}(z)$ represents the linear mapping of an input block of length L to an output block of length N . Its element $C_{ij}(z)$ describes a time-invariant linear filtering operation performed on the sequence of elements j of each input block in order to generate the sequence of elements i of each output block.

A widely used example of overlapped block filtering is the overlap-and-add method analyzed in Section 3.4.2. By mapping Figure 3.4 to Figure 8.25, we can see that for the overlap-and-add method the decimation factor is N and the lengths of the input and output

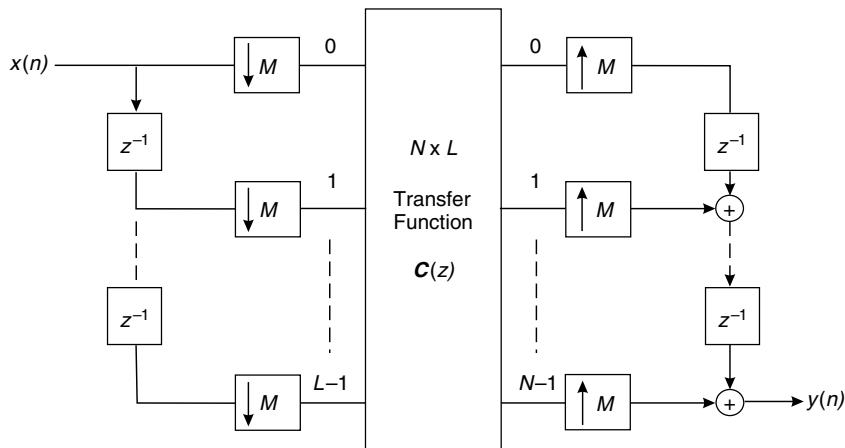


Fig. 8.25. Multirate representation of an overlapped block digital filter.

blocks are both equal to $(N + L - 1)$. The processing carried out by $\mathbf{C}(z)$ is the circular convolution with $h(n)$.

It is common to refer to the matrix $\mathbf{C}(z)$ as a multiple-input, multiple-output (MIMO) system. Likewise, the overall system in Figure 8.25, which maps $x(n)$ to $y(n)$, is often referred to as a SISO system.

It is important to note that in the overlapped block filtering scheme depicted in Figure 8.25 there is a decimation-by- M operation. Therefore, aliasing may occur in the process. In this case, one may not be able to describe the relation between the input and output as a linear filtering operation. Depending on the relative values of decimation factor M , overlap factor at the input ($L - M$), and overlap factor at the output ($N - M$), matrix $\mathbf{C}(z)$ has to satisfy different conditions in order to guarantee that the input–output relation is aliasing free. In the next subsection we analyze such conditions for the case of nonoverlapped input and output. Then, in Section 8.11.2, we analyze the more general overlapped input and overlapped output case.

8.11.1 Nonoverlapped case

In the nonoverlapped case the blocks do not overlap either at the input or at the reconstruction stage at the output. This happens when $L = M = N$.

Our aim is to use the scheme in Figure 8.25 to implement a shift-invariant system. The input–output relation of such a system can be expressed as in the z transform domain as

$$Y(z) = \frac{1}{M} [z^{-(N-1)} \dots z^{-1} 1] \mathbf{C}(z^M) \sum_{i=0}^{M-1} \begin{bmatrix} 1 \\ (zW_M)^{-1} \\ \vdots \\ (zW_M)^{-(L-1)} \end{bmatrix} X(zW_M^i). \quad (8.52)$$

Its proof is left as an exercise to the interested reader. Note that, in the summation on the right-hand side, the term for $i = 0$ is the shift-invariant component, while all other terms are due to aliasing. Remember that in the present nonoverlapped case we have that $L = M = N$.

If we want the system in Equation (8.52) to be shift invariant, then the aliasing terms in the summation have to be zero. It can be shown (Vaidyanathan, 1993) that this happens if and only if the matrix $\mathbf{C}(z)$ is pseudo-circulant; that is:

$$\begin{aligned}\mathbf{C}(z) &= \begin{bmatrix} C_{00}(z) & C_{01}(z) & \cdots & C_{0M-1}(z) \\ C_{10}(z) & C_{11}(z) & \cdots & C_{1M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ C_{M-10}(z) & C_{M-11}(z) & \cdots & C_{M-1M-1}(z) \end{bmatrix} \\ &= \begin{bmatrix} E_0(z) & E_1(z) & \cdots & E_{M-2}(z) & E_{M-1}(z) \\ z^{-1}E_{M-1}(z) & E_0(z) & \cdots & E_{M-3}(z) & E_{M-2}(z) \\ z^{-1}E_{M-2}(z) & z^{-1}E_{M-1}(z) & \cdots & E_{M-4}(z) & E_{M-3}(z) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ z^{-1}E_1(z) & z^{-1}E_2(z) & \cdots & z^{-1}E_{M-1}(z) & E_0(z) \end{bmatrix}. \quad (8.53)\end{aligned}$$

It is worth noting that we refer to this matrix as pseudo-circulant due to the presence of the delay terms z^{-1} in the lower triangular part of $\mathbf{C}(z)$. If they were not present the $\mathbf{C}(z)$ would be a circulant matrix.

Two important properties of pseudo-circulant matrices are (Vaidyanathan, 1993):

- A product of pseudo-circulant matrices is also pseudo-circulant. This implies that it is possible to exploit the decomposition of a pseudo-circulant matrix as a product of submatrices of the same type.
- If a matrix $\mathbf{C}(z)$ is pseudo-circulant and has an inverse, then its inverse is also pseudo-circulant.

In this case the overall transfer function becomes (Vaidyanathan, 1993)

$$H(z) = z^{-M+1}[E_0(z^M) + z^{-1}E_1(z^M) + \cdots + z^{-(M-1)}E_{M-1}(z^M)]; \quad (8.54)$$

that is, in the nonoverlapped case, the polyphase components of the overall transfer function correspond to the functions $E_i(z)$ in Equation (8.53).

In the following example, we consider the case of a 2×2 block digital filtering to illustrate the important pseudo-circulant requirement for shift invariance.

Example 8.3. Assume in Figure 8.25 that $L = M = N = 2$ and demonstrate that the SISO transfer function is time invariant when the transfer matrix $\mathbf{C}(z)$ is pseudo-circulant.

Solution

The output signal can be described using its polyphase components as

$$Y(z) = [z^{-1} \ 1] \begin{bmatrix} Y_0(z^2) \\ Y_1(z^2) \end{bmatrix}, \quad (8.55)$$

where, as can be observed by comparing Figure 8.25 to Figures 8.18 and 8.17, the polyphase components of $Y(z)$, denoted by $Y_i(z)$ for $i = 1, 2$, are the outputs of matrix $\mathbf{C}(z)$ when its inputs are the decimated polyphase components of the input signal. Therefore, these polyphase components can be expressed as

$$Y_0(z) = \frac{1}{2}[X_0(z^{1/2}) + X_0(z^{1/2}W_2)]C_{00}(z) + \frac{1}{2}[X_1(z^{1/2}) + X_1(z^{1/2}W_2)]C_{01}(z) \quad (8.56)$$

and

$$Y_1(z) = \frac{1}{2}[X_0(z^{1/2}) + X_0(z^{1/2}W_2)]C_{10}(z) + \frac{1}{2}[X_1(z^{1/2}) + X_1(z^{1/2}W_2)]C_{11}(z) \quad (8.57)$$

respectively.

The filter output can then be described as

$$\begin{aligned} Y(z) &= z^{-1}Y_0(z^2) + Y_1(z^2) \\ &= \frac{z^{-1}}{2} \left[(X_0(z) + X_0(zW_2))C_{00}(z^2) + (X_1(z) + X_1(zW_2))C_{01}(z^2) \right] \\ &\quad + \frac{1}{2} \left[(X_0(z) + X_0(zW_2))C_{10}(z^2) + (X_1(z) + X_1(zW_2))C_{11}(z^2) \right] \\ &= \frac{1}{2} \left[z^{-1}C_{00}(z^2)X_0(z) + z^{-1}C_{01}(z^2)X_1(z) + C_{10}(z^2)X_0(z) + C_{11}(z^2)X_1(z) \right] \\ &\quad + \frac{1}{2} \left[z^{-1}C_{00}(z^2)X_0(zW_2) + z^{-1}C_{01}(z^2)X_1(zW_2) \right. \\ &\quad \left. + C_{10}(z^2)X_0(zW_2) + C_{11}(z^2)X_1(zW_2) \right] \\ &= \frac{1}{2} \left[(z^{-1}C_{00}(z^2) + C_{10}(z^2))X_0(z) + (z^{-1}C_{01}(z^2) + C_{11}(z^2))X_1(z) \right] \\ &\quad + \frac{1}{2} \left[(z^{-1}C_{00}(z^2) + C_{10}(z^2))X_0(zW_2) + (z^{-1}C_{01}(z^2) + C_{11}(z^2))X_1(zW_2) \right] \\ &= \frac{1}{2} \left[(z^{-1}C_{00}(z^2) + C_{10}(z^2))(X_0(z) + X_0(-z)) \right. \\ &\quad \left. + (z^{-1}C_{01}(z^2) + C_{11}(z^2))(X_1(z) + X_1(-z)) \right]. \end{aligned} \quad (8.58)$$

According to Equation (8.6), the polyphase components of the input signal can be expressed as

$$\begin{aligned} X_0(z) &= \frac{1}{2}[X(z) + X(-z)] \\ X_1(z) &= \frac{z^{-1}}{2}[X(z) - X(-z)], \end{aligned} \quad (8.59)$$

which imply that

$$\begin{aligned} X_0(z) + X_0(-z) &= X(z) + X(-z) \\ X_1(z) + X_1(-z) &= z^{-1} [X(z) - X(-z)]. \end{aligned} \quad (8.60)$$

As a result, the z transform of the output signal can be expressed as

$$\begin{aligned} Y(z) &= \frac{1}{2} \left[\left(z^{-1} C_{00}(z^2) + C_{10}(z^2) \right) (X(z) + X(-z)) \right. \\ &\quad \left. + z^{-1} \left(z^{-1} C_{01}(z^2) + C_{11}(z^2) \right) (X(z) - X(-z)) \right]. \end{aligned} \quad (8.61)$$

If we choose $C_{00}(z) = C_{11}(z) = E_0(z)$ and $C_{01}(z) = zC_{10}(z) = E_1(z)$ (see Equation (8.53)), then

$$\begin{aligned} Y(z) &= \left[z^{-1} (C_{00}(z^2) + C_{11}(z^2)) + C_{10}(z^2) + z^{-2} C_{01}(z^2) \right] X(z) \\ &= z^{-1} \left[E_0(z^2) + z^{-1} E_1(z^2) \right] X(z), \end{aligned} \quad (8.62)$$

which has no aliasing component, meaning that the transfer function between the filter input and output is time invariant. Note that Equation (8.62) is in the same form as Equation (8.54). For a proof for the general case of $M \times M$ block representations the reader should refer to Vaidyanathan (1993). \triangle

8.11.2 Overlapped input and output

We now discuss more general forms of implementing shift-invariant systems using overlapped block filtering. In order to do so we will start from the nonoverlapped case analyzed in Section 8.11.1. There, the implementation is based on a pseudo-circulant matrix $\mathbf{C}(z)$. From such implementations, we use the matrix representations of the serial-to-parallel and parallel-to-serial converters with overlapping in Equations (8.42)–(8.47) in order to, from an $M \times M$ matrix $\mathbf{C}(z)$ (which implements a system without overlapping), generate an implementation corresponding to a factorization

$$\mathbf{C}(z) = \mathbf{P}_N^M(z) \hat{\mathbf{C}}(z) \mathbf{S}_L^M(z). \quad (8.63)$$

In this case, $\hat{\mathbf{C}}(z)$ is an $N \times L$ matrix that implements a system with overlapping, while $\mathbf{S}_L^M(z)$ and $\mathbf{P}_N^M(z)$ correspond to the serial-to-parallel and parallel-to-serial converters in Equations (8.42) and (8.45), respectively.

In order to derive an expression for the matrices $\mathbf{S}_L^M(z)$ and $\mathbf{P}_N^M(z)$ from Equations (8.42) and (8.45), we must first define the z transform of a signal block as in Equation (8.39); that is

$$\mathbf{X}_L^M(z) = \sum_{m=-\infty}^{\infty} \mathbf{x}_L^M(m) z^{-m}. \quad (8.64)$$

Also, applying the above definition to Equation (8.41), we have that

$$\mathcal{Z}\{\mathcal{D}^M\{\mathbf{x}_L^M(m)\}\} = \mathcal{Z}\{\mathbf{x}_L^M(m-1)\} = z^{-1}\mathbf{X}_L^M(z) \quad (8.65)$$

Therefore, in the z -transform domain, for $L < 2M$ and $N < 2M$, Equations (8.42) and (8.45) become

$$\mathbf{X}_L^M(z) = \begin{bmatrix} \mathbf{I}_{L-M} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2M-L} \\ z^{-1}\mathbf{I}_{L-M} & \mathbf{0} \end{bmatrix} \mathbf{X}_M^M(z) \quad (8.66)$$

and

$$\mathbf{Y}_M^M(z) = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} \\ z^{-1}\mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} \end{bmatrix} \mathbf{W}_N^M(z) \quad (8.67)$$

respectively.

Since $\mathbf{C}(z)$ represents the nonoverlapping block processing and $\hat{\mathbf{C}}(z)$ the overlapping block processing, we have that

$$\mathbf{Y}_M^M(z) = \mathbf{C}(z)\mathbf{X}_M^M(z) \quad (8.68)$$

$$\mathbf{W}_N^M(z) = \hat{\mathbf{C}}(z)\mathbf{X}_L^M(z). \quad (8.69)$$

Therefore, from Equations (8.63), (8.66), (8.67), (8.68), and (8.69), we conclude that the matrices $\mathbf{S}_L^M(z)$ and $\mathbf{P}_N^M(z)$ have the following general forms:

$$\mathbf{S}_L^M(z) = \begin{bmatrix} \mathbf{I}_{L-M} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2M-L} \\ z^{-1}\mathbf{I}_{L-M} & \mathbf{0} \end{bmatrix} \quad (8.70)$$

$$\mathbf{P}_N^M(z) = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{2M-N} & \mathbf{0} \\ z^{-1}\mathbf{I}_{N-M} & \mathbf{0} & \mathbf{I}_{N-M} \end{bmatrix}. \quad (8.71)$$

We illustrate the use of these matrices through a couple of workable examples.

Example 8.4. For the case where $L = M = 4$ and $N = 7$, implement the transfer function

$$H(z) = z^{-3}[E_0(z^4) + z^{-1}E_1(z^4) + z^{-2}E_2(z^4) + z^{-3}E_3(z^4)] \quad (8.72)$$

in a block form, by noting that overlapping is applied only at the output.

Solution

We should choose matrix $\hat{\mathbf{C}}(z)$ as

$$\hat{\mathbf{C}}(z) = \begin{bmatrix} E_3(z) & 0 & 0 & 0 \\ E_2(z) & E_3(z) & 0 & 0 \\ E_1(z) & E_2(z) & E_3(z) & 0 \\ E_0(z) & E_1(z) & E_2(z) & E_3(z) \\ 0 & E_0(z) & E_1(z) & E_2(z) \\ 0 & 0 & E_0(z) & E_1(z) \\ 0 & 0 & 0 & E_0(z) \end{bmatrix}. \quad (8.73)$$

With the above choice, from Equations (8.70) and (8.71), we respectively have that

$$\mathbf{S}_4^4(z) = \mathbf{I}_4 \quad (8.74)$$

and

$$\mathbf{P}_7^4(z) = \begin{bmatrix} \mathbf{0} & \mathbf{I}_1 & \mathbf{0} \\ z^{-1}\mathbf{I}_3 & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ z^{-1} & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & z^{-1} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & z^{-1} & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (8.75)$$

Note that $\mathbf{S}_4^4(z)$ is an identity matrix because there is no overlap between blocks at the input. With these choices, we have $\mathbf{C}(z) = \mathbf{P}_7^4(z)\hat{\mathbf{C}}(z)\mathbf{S}_4^4(z) = \mathbf{P}_7^4(z)\hat{\mathbf{C}}(z)$, so that

$$\mathbf{C}(z) = \begin{bmatrix} E_0(z) & E_1(z) & E_2(z) & E_3(z) \\ z^{-1}E_3(z) & E_0(z) & E_1(z) & E_2(z) \\ z^{-1}E_2(z) & z^{-1}E_3(z) & E_0(z) & E_1(z) \\ z^{-1}E_1(z) & z^{-1}E_2(z) & z^{-1}E_3(z) & E_0(z) \end{bmatrix}, \quad (8.76)$$

which is a pseudo-circulant matrix representing the overall transfer function of the block digital filter, according to Equation (8.52). \triangle

For the algorithm presented at Example 8.4, the input blocks are not overlapped, whereas the output blocks are overlapped. We now generalize this algorithm for an input block size $L = M$ and an output block size $N = 2M - 1$. The corresponding block transfer matrix, $\hat{\mathbf{C}}(z)$ of dimensions $(2M - 1) \times M$, should have the form

$$\hat{\mathbf{C}}(z) = \begin{bmatrix} E_{M-1}(z) & 0 & \cdots & 0 \\ E_{M-2}(z) & E_{M-1}(z) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ E_0(z) & E_1(z) & \cdots & E_{M-1}(z) \\ 0 & E_0(z) & \cdots & E_{M-2}(z) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_0(z) \end{bmatrix} \quad (8.77)$$

in order for the product $\hat{\mathbf{P}}_{2M-1}^M(z)\hat{\mathbf{C}}(z)\mathbf{I}_M$ to result in a pseudo-circulant matrix.

Example 8.5. We now consider the implementation of the transfer function of Example 8.4 for $L = 7$ and $N = 4 = M$. Note that overlapping is applied only at the input.

Solution

We should choose

$$\hat{\mathbf{C}}(z) = \begin{bmatrix} E_0(z) & E_1(z) & E_2(z) & E_3(z) & 0 & 0 & 0 \\ 0 & E_0(z) & E_1(z) & E_2(z) & E_3(z) & 0 & 0 \\ 0 & 0 & E_0(z) & E_1(z) & E_2(z) & E_3(z) & 0 \\ 0 & 0 & 0 & E_0(z) & E_1(z) & E_2(z) & E_3(z) \end{bmatrix}. \quad (8.78)$$

With the above choice, we have from Equations (8.70) and (8.71) that

$$\hat{\mathbf{S}}_7^4(z) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1 \\ z^{-1}\mathbf{I}_3 & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ z^{-1} & 0 & 0 & 0 \\ 0 & z^{-1} & 0 & 0 \\ 0 & 0 & z^{-1} & 0 \end{bmatrix} \quad (8.79)$$

$$\hat{\mathbf{P}}_4^4(z) = \mathbf{I}_4. \quad (8.80)$$

This case leads to the same $\mathbf{C}(z)$ as Example 8.4. Δ

In the structure in Example 8.5, the input blocks are overlapped, whereas the output blocks are not overlapped. This structure can also be generalized for $M = N$ and $L = 2M - 1$ by employing the block transfer function matrix $\hat{\mathbf{C}}(z)$ of dimensions $M \times (2M - 1)$ given by

$$\hat{\mathbf{C}}(z) = \begin{bmatrix} E_0(z) & E_1(z) & \cdots & E_{M-1}(z) & \cdots & 0 \\ 0 & E_0(z) & \cdots & E_{M-2}(z) & E_{M-1}(z) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_0(z) & E_1(z) & \cdots & E_{M-1}(z) \end{bmatrix}. \quad (8.81)$$

It is possible to generate several alternative structures for values of L, M, N different from those discussed here. For each case, a distinct form for $\hat{\mathbf{C}}(z)$ is required for proper generation of shift-invariant structure. However, determining the right $\hat{\mathbf{C}}(z)$ in order to achieve this is not trivial. It is important to emphasize the fact that a SISO system implemented in block form is not, in general, linear and time invariant. However, if the block system can be described by a transfer function matrix $\mathbf{C}(z)$ (meaning it is a linear time-invariant MIMO system), then the corresponding SISO system is necessarily linear and periodically time varying with period corresponding to the block size. If, in addition to this, $\mathbf{C}(z)$ is pseudo-circulant, then the SISO system is also linear and time invariant.

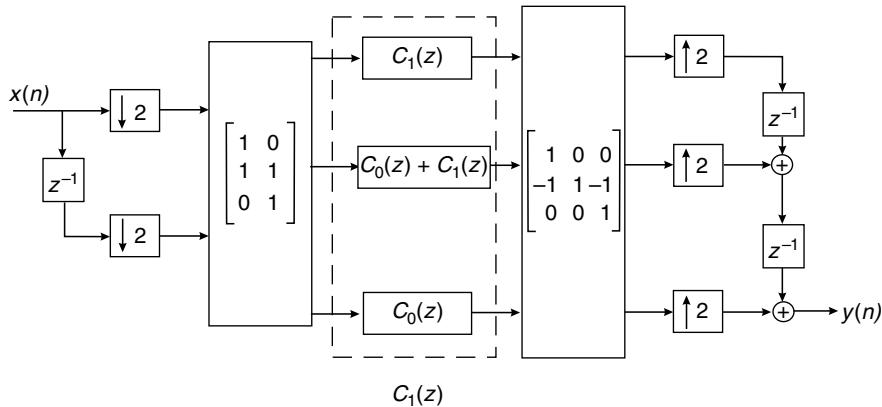


Fig. 8.26. Overlapped structure I for fast convolution. $C_i(z) = E_i(z)$, $i = 0, 1$.

8.11.3 Fast convolution structure I

Inspired by overlapped block implementation it is possible to derive the structure depicted in Figure 8.26. This configuration is called structure I, and corresponds to the case where $L = M = 2$, and $N = 3$. This structure was derived by decomposing matrix $\hat{C}(z)$ (see Example 8.4) as follows:

$$\begin{bmatrix} E_1(z) & 0 \\ E_0(z) & E_1(z) \\ 0 & E_0(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} E_1(z) & 0 & 0 \\ 0 & E_0(z) + E_1(z) & 0 \\ 0 & 0 & E_0(z) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}. \quad (8.82)$$

In Equation (8.82) each polynomial $E_i(z)$, for $i = 0, 1$, is a polyphase component of $H(z)$ and, therefore, corresponds to a filter operation with an FIR filter of about half the length of the original filter $H(z)$. This structure then shows how to implement an FIR filtering of a given order through three FIR filters of half order and operating at half rate. This can be done recursively, since each half-order FIR filter can be further decomposed into three other subfilters. Every time a new decomposition is applied the number of multiplications per sample is reduced, whereas the latency (delay) of the response increases. For more details on this, the reader is referred to Vetterli (1988) and Mou & Duhamel (1991).

8.11.4 Fast convolution structure II

Figure 8.27 shows structure II, which is exactly the transpose of overlapped block structure I. Note that, in the transposition of multirate systems, decimators become interpolators and vice versa.

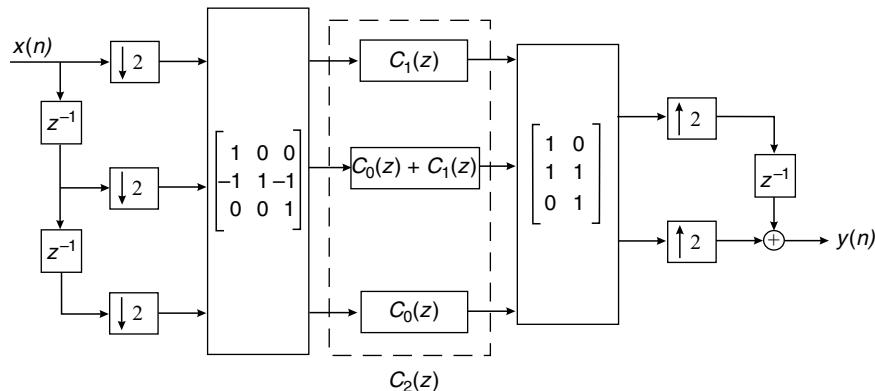


Fig. 8.27. Overlapped structure II for fast convolution. $C_i(z) = E_i(z)$, $i = 0, 1$.

Example 8.6. Determine if the transfer function matrix below can be the block representation of a linear and time-invariant system:

$$\mathbf{C}(z) = \begin{bmatrix} 1+z^{-1} & -1/2a \\ 2a & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1+2bz^{-1}+z^{-2} \end{bmatrix} \begin{bmatrix} 1 & (1+z^{-1})/2a \\ 0 & 1 \end{bmatrix}. \quad (8.83)$$

Solution

Let us start by computing $\mathbf{C}(z)$ to access its properties:

$$\begin{aligned} \mathbf{C}(z) &= \begin{bmatrix} 1+z^{-1} & -(1/2a) - (b/a)z^{-1} - (1/2a)z^{-2} \\ 2a & 0 \end{bmatrix} \begin{bmatrix} 1 & (1/2a)(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1+z^{-1} & -(1/2a) - (b/a)z^{-1} - (1/2a)z^{-2} + (1/2a)(1+2z^{-1}+z^{-2}) \\ 2a & 1+z^{-1} \end{bmatrix} \\ &= \begin{bmatrix} 1+z^{-1} & (1-b/a) \\ 2a & 1+z^{-1} \end{bmatrix}. \end{aligned} \quad (8.84)$$

In order for the matrix above to represent a linear and time-invariant system, it must be pseudo-circulant. Hence, the following condition must be satisfied:

$$\frac{1-b}{a} = 2a \Rightarrow b = 1 - 2a^2 \quad (8.85)$$

△

Example 8.7. (a) Propose a block implementation for a linear time-invariant transfer function using the matrices $\mathbf{S}_L^M(z)$ and $\mathbf{P}_N^M(z)$ in Equations (8.70) and (8.71), such that the input and output blocks have overlaps given by $L = 4$ and $N = 3$ respectively. The number of subchannels is $M = 2$.

- (b) Implement the transfer function below with the proposed structure and draw the overall realization

$$H(z) = z^{-4} + z^{-3} + 2z^{-2} + 4z^{-1}. \quad (8.86)$$

Solution

- (a) For this case $N = 3$, $M = 2$, and $L = 4$. Therefore:

$$\mathbf{S}_4^2(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ z^{-1} & 0 \\ 0 & z^{-1} \end{bmatrix} \quad (8.87)$$

$$\mathbf{P}_3^2(z) = \begin{bmatrix} 0 & 1 & 0 \\ z^{-1} & 0 & 1 \end{bmatrix}. \quad (8.88)$$

A possible and simple solution for $\hat{\mathbf{C}}(z)$, leading to the minimum overall delay, is given by

$$\hat{\mathbf{C}}(z) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ E_0(z) & E_1(z) & 0 & 0 \\ 0 & E_0(z) & E_1(z) & 0 \end{bmatrix}. \quad (8.89)$$

By post-multiplying the matrix $\hat{\mathbf{C}}(z)$ by $\mathbf{S}_4^2(z)$, it follows that

$$\hat{\mathbf{C}}(z)\mathbf{S}_4^2(z) = \begin{bmatrix} 0 & 0 \\ E_0(z) & E_1(z) \\ z^{-1}E_1(z) & E_0(z) \end{bmatrix}. \quad (8.90)$$

By pre-multiplying the resulting matrix by $\mathbf{P}_3^2(z)$, we have

$$\mathbf{C}(z) = \mathbf{P}_3^2(z)\hat{\mathbf{C}}(z)\mathbf{S}_4^2(z) = \begin{bmatrix} E_0(z) & E_1(z) \\ z^{-1}E_1(z) & E_0(z) \end{bmatrix}, \quad (8.91)$$

which is pseudo-circulant.

Figure 8.28 depicts the resulting structure. Note that, since the first row of $\hat{\mathbf{C}}(z)$ in Equation (8.89) has only zeros, then the structure has overlapping input and nonoverlapping output.

- (b) Since aliasing is canceled, the SISO transfer function described in block form is given by

$$\begin{aligned} H(z) &= [z^{-1} \ 1] \mathbf{C}(z^2) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \\ &= [z^{-1} \ 1] \begin{bmatrix} E_0(z^2) + z^{-1}E_1(z^2) \\ z^{-1}E_0(z^2) + z^{-2}E_1(z^2) \end{bmatrix} \\ &= 2z^{-1} [E_0(z^2) + z^{-1}E_1(z^2)]. \end{aligned} \quad (8.92)$$

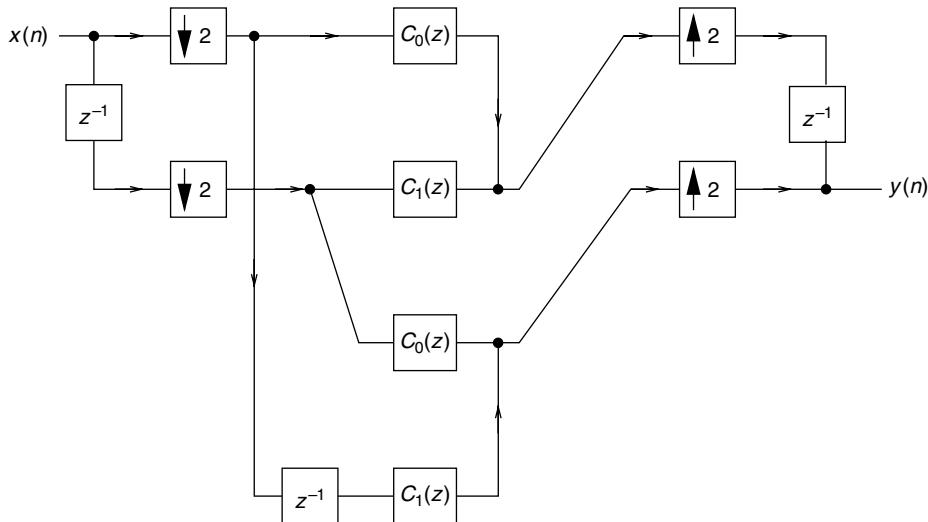


Fig. 8.28. Realization of Equation (8.91). $C_i(z) = E_i(z)$, $i = 0, 1$.

From the above expression and Equation (8.86), $E_0(z)$ and $E_1(z)$ become

$$E_0(z) = 2 + \frac{z^{-1}}{2} \quad (8.93)$$

$$E_1(z) = 1 + \frac{z^{-1}}{2}. \quad (8.94)$$

△

8.12 Random signals in multirate systems

The effects of multirate processing in random signals are the topic of this section. An important concept often present in rate change of stochastic signals is that of cyclostationary processes (Stark & Woods, 2002). A real random process $\{X\}$ is wide-sense cyclostationary (WSCS) with period M if its mean value and autocorrelation function satisfy

$$E\{X(n)\} = E\{X(n + kM)\} \quad (8.95)$$

and

$$R_X(n, k) = R_X(n + M, k + M) = E\{X(n + M)X(k + M)\} \quad (8.96)$$

for all n, k .

As the definitions state, the mean and the autocorrelation function are periodic with period M . Very often this property appears in several practical applications. Examples

include sampling in communication systems, modulation, multiplexing, and the interaction of WSS processes with multirate systems.

Let us assume now that the serial-to-parallel converter of Figure 8.19 retains M consecutive samples of a realization of a WSCS process without overlap (see Equation (8.39)) as follows:

$$\mathbf{X}_M^M(m) = [X(mM) \ X(mM - 1) \ \cdots \ X(mM - M + 1)]^T. \quad (8.97)$$

For a given random input vector, the autocorrelation matrix is defined as

$$\mathbf{R}_{\mathbf{X}_M^M}(m) = E\{\mathbf{X}_M^M(m)\mathbf{X}_M^M(m)^T\}. \quad (8.98)$$

As will be noted, the characteristics of the autocorrelation matrix play a key role in understanding the effects of multirate processing in random signals. Note that if the input process is WSCS with period M , the block vector $\mathbf{X}_M^M(m)$ is WSS; that is, the matrix $\mathbf{R}_{\mathbf{X}_M^M}(m)$ does not depend on m .

Let us assume now that an $M \times 1$ WSS vector $\mathbf{X}_M^M(m)$ is input to a transfer function matrix $\mathbf{C}(z)$ of dimensions $N \times M$, then the PSD of the output vector will be given by

$$\boldsymbol{\Gamma}_U(z) = \mathbf{C}(z)\boldsymbol{\Gamma}_{\mathbf{X}_M^M}(z)\mathbf{C}^T(z^{-1}), \quad (8.99)$$

where

$$\boldsymbol{\Gamma}_{\mathbf{X}_M^M}(z) = \sum_{v=-\infty}^{\infty} \mathbf{R}_{\mathbf{X}_M^M}(v)z^{-v} \quad (8.100)$$

is the PSD of the input signal vector. The expressions in (8.99) and (8.100) are M -dimensional generalizations of Equation (7.9).

A very important property of the PSD matrix formulation is that if the input process to a serial-to-parallel converter is WSS, then the PSD matrix $\boldsymbol{\Gamma}_{\mathbf{X}_M^M}(z)$ is pseudo-circulant (see Exercise 8.35 and Sathe and Vaidyanathan (1993)). Conversely, in the case the vector $\mathbf{X}_M^M(m)$, output by a serial-to-parallel converter, is WSS and its PSD matrix is pseudo-circulant, then the input process to the serial-to-parallel converter is WSS.

8.12.1 Interpolated random signals

If a WSS random signal is applied to the input of an interpolator as in Figure 8.7, then the signal random $\hat{X}(n)$ at the output is WSCS with period L . Its blocked output vector is given by

$$\begin{aligned} \hat{\mathbf{X}}_L^L(m) &= [\hat{X}(mL) \ \hat{X}(mL - 1) \ \cdots \ \hat{X}(mL - L + 1)]^T \\ &= [\hat{X}(mL) \ 0 \ \cdots \ 0]^T \end{aligned} \quad (8.101)$$

$$= [X(m) \ 0 \ \cdots \ 0]^T. \quad (8.102)$$

Hence its autocorrelation matrix is

$$\mathbf{R}_{\hat{\mathbf{X}}_L^L}(m) = E\{\hat{\mathbf{X}}_L^L(m)\hat{\mathbf{X}}_L^{L^T}(m)\} \\ = \begin{bmatrix} E\{\hat{X}^2(mL)\} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \quad (8.103)$$

$$= \begin{bmatrix} E\{X^2(m)\} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \quad (8.104)$$

Since $X(m)$ is WSS, then $E\{X^2(m)\}$ is constant for all m and, therefore, the correlation matrix is not a function of m ; that is, the vector $\hat{\mathbf{X}}_L^L(m)$ is WSS. This implies that its unblocked version $\hat{X}(n)$ is WSCS with period L .

8.12.2 Decimated random signals

Let us consider now the case where a random signal is applied to a decimator as in Figure 8.1. In this case, the decimated random signal $X_d(n)$ is the result of retaining every M th sample of the input random signal denoted as $X(nM)$. If we assume the general case where the input signal is WSCS with period N , the decimated process will also be WSCS but with a period P . In order to determine the value of P , we analyze the properties of the autocorrelation function of the decimated signal, that is:

$$R_{X_d}(n, l) = E\{X_d(n)X_d(l)\} \\ = E\{X(nM)X(lM)\}. \quad (8.105)$$

If the output signal is WSCS with period P , then

$$R_{X_d}(n + P, l + P) = E\{X_d(n + P)X_d(l + P)\} = E\{X((n + P)M)X((l + P)M)\}. \quad (8.106)$$

Considering that we assumed the input process WSCS with period N , the equality of (8.106) holds if $PM = iN$ for some integer i . Therefore, the period P should be

$$P = \frac{N}{\gcd(M, N)}, \quad (8.107)$$

where $\gcd(\cdot)$ stands for the greatest common divisor between two integer numbers.

Some special choices for M and N are worth mentioning:

- If $N = 1$, then $P = 1$, meaning that if the input to the decimator is WSS, then its output is also WSS.
- If N and M are prime numbers, then $P = N$.
- If $N = M$, then $P = 1$, indicating that a cyclostationary signal, when decimated by its cyclostationarity period, becomes WSS.

8.13 Do-it-yourself: multirate systems

Experiment 8.1

Consider a sinusoidal signal s of frequency $f_1 = 0.01$ Hz corrupted by noise and sampled at $F_s = 1$ samples/s for a time interval of 10 min. Assume that the noise component is the output of the filter

$$H_1(z) = \frac{1}{12} \sum_{i=0}^{11} (-1)^i z^{-i} \quad (8.108)$$

to a zero-mean and unit-variance Gaussian noise, such that

```
Fs = 1; Ts = 1/Fs; duration = 600;
time = 0:Ts:(duration-Ts); Ntime = length(time);
s = sin(2*pi*f1*time);
w = randn(1,Ntime); w = w-mean(w); w = w./sqrt(w*w');
h1 = [1 -1 1 -1 1 -1 1 -1 1 -1]./12;
wh1 = filter(h1,1,w);
x = s+wh1;
```

as depicted in the time and frequency domains in Figure 8.29. From this figure one notices the highpass characteristic of the noise component, as yielded by $H_1(z)$.

In order to simplify the sinusoidal storage or transmission, we may decimate signal x by $M=10$, after performing a proper lowpass filtering to minimize aliasing distortion, as given by

```
ordh2 = 20; h2 = ones(1,ordh2+1)./(ordh2+1);
xh2 = filter(h2,1,x);
M = 10; xdec = xh2(1:M:Ntime);
```

resulting in the x_{dec} signal characterized in Figure 8.30.

The sampling rate can be expanded back to its original value by introducing $(M-1)$ zeros between each two consecutive samples of x_{dec} , which in MATLAB can be performed as

```
xaux = [xdec; zeros(M-1,Ntime/M)];
xaux2 = reshape(xaux,1,Ntime);
```

This procedure causes spectral repetitions that must be removed by a proper lowpass filtering such as

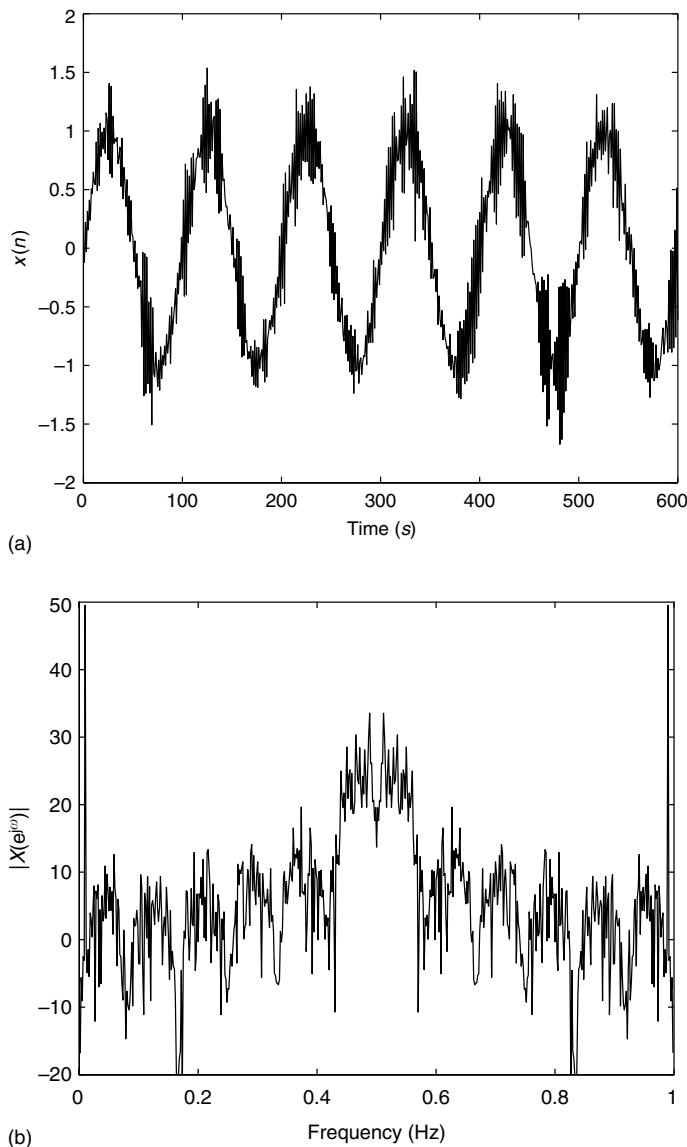


Fig. 8.29. Signal s corrupted by noise: (a) time domain; (b) frequency domain.

```

h3 = firpm(30,[0 0.01 0.09 0.5]*2,[1 1 0 0]);
xdec_int = filter(M*h3,1,xaux2);

```

Figure 8.31 depicts the filtered signal and its corresponding spectral representation, which can be readily compared to signal $xh2$, before the decimation operation.

In the last step of this experiment one can use a bandpass, instead of a lowpass, filter $h3$ to generate a modulated version of the original signal. This type of processing is employed in Experiment 11.1, which the student is motivated to read, to modulate a signal without an explicit multiplication by a high-frequency sinusoidal carrier.

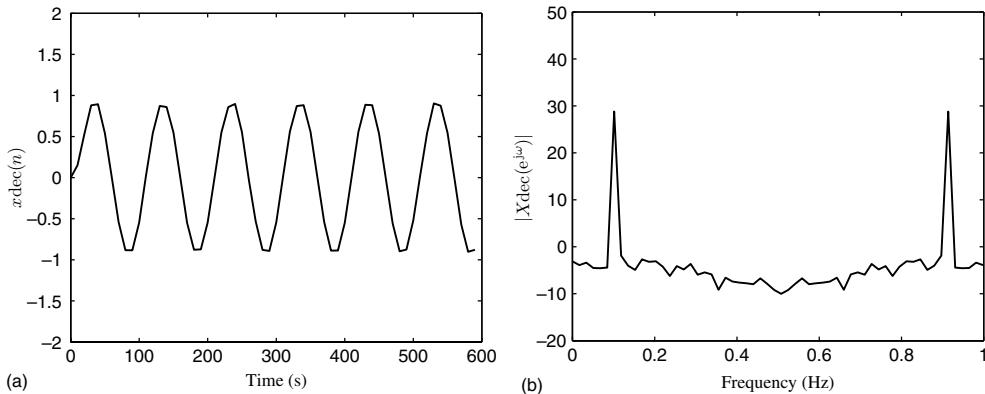


Fig. 8.30. Signal s corrupted by noise filtered and decimated by 10: (a) time domain; (b) frequency domain.

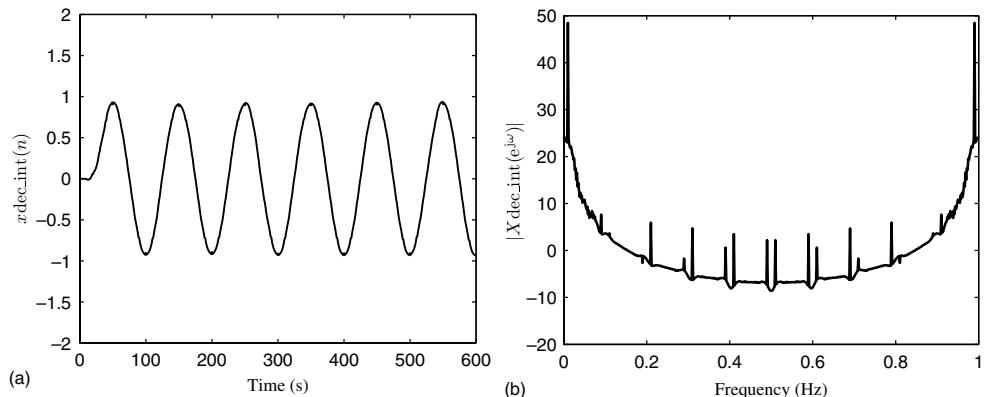


Fig. 8.31. Decimated signal from Figure 8.30 interpolated by 10 and filtered: (a) time domain; (b) frequency domain.

All rate-changing operations employed in the present experiment can be performed automatically with the MATLAB commands `decimate` and `interp`, which already include the corresponding lowpass filtering stage. \triangle

8.14 Multirate systems with MATLAB

The functions described below are part of the MATLAB Signal Processing Toolbox.

- `upfirdn`: Upsamples, applies a specified filter, and then downsamples a vector.
Input parameters:
 - the vector x containing the input signal;
 - the filter h to be applied after interpolation;
 - the interpolation factor p and the decimation factor q .

Output parameter: the vector y holding the filtered signal.

Example 1 (downsample a signal by a factor of 3):

```
x=rand(100,1); h=[1]; p=1; q=3;
y=upfirdn(x,h,p,q);
```

Example 2 (sampling-rate change by a factor of $\frac{5}{4}$ using a filter h):

```
x=rand(100,1); h=[1 2 3 4 5 4 3 2 1]/5; p=5; q=4;
y=upfirdn(x,h,p,q);
```

- **decimate**: Downsamples after lowpass filtering.

Input parameters:

- The vector x containing the input signal.
- The decimation factor r .
- The order n of the lowpass filter.
- The type of the lowpass filter. The default is a Chebyshev lowpass filter with cutoff $0.8f_s/2r$. ‘FIR’ specifies FIR filtering.

Output parameter: the vector y holding the downsampled signal.

Example 1 (downsample a signal by a factor of 3 using a 10th-order Chebyshev lowpass filter):

```
x=rand(100,1); r=3; n=10
y=decimate(x,r,10);
```

Example 2 (downsample a signal by a factor of 5 using a 50th-order FIR lowpass filter):

```
x=rand(1000,1); r=5; n=50
y=decimate(x,r,50,’FIR’);
```

- **interp**: Interpolates a signal.

Input parameters:

- the vector x containing the input signal.
- the interpolation factor r .
- the number of original sample values l used to compute each interpolated sample.
- the bandwidth α of the original signal.

Output parameters:

- the vector y holding the downsampled signal;
- the vector b holding the coefficients of the interpolating filter.

Example 1 (interpolate a signal by a factor of 3 using an FIR filter of order 12):

```
x=rand(100,1); r=3; l=4;
y=interp(x,r,l);
```

Example 2 (interpolate a signal band-limited to a quarter of the sampling rate by a factor of 4 using an FIR filter of order 12):

```
x=rand(100,1); r=4; l=3, alpha=0.5;
y=interp(x,r,l,alpha);
```

- **resample**: Changes the sampling rate of a signal.

Input parameters:

- The vector x containing the input signal;

- The interpolation factor p ;
- The decimation factor q ;
- The number n , which controls the number of original sample values used to compute each output sample. This number is equal to $2 * n * \max(1, q/p)$;
- The filter b used to filter the input signal, or, alternatively, the parameter β of the Kaiser window used to design the filter.
- The bandwidth α of the original signal.

Output parameters:

- the vector y holding the downsampled signal;
- the vector b holding the coefficients of the interpolating filter.

Example 1 (change the sampling rate of a signal by a factor of $\frac{5}{4}$):

```
x=rand(100,1); p=5; q=4;
y=resample(x,p,q);
```

Example 2 (change the sampling rate of a signal by a factor of $\frac{2}{3}$ using, for each output sample, 12 original samples of the input signal, and an FIR filter designed with a Kaiser window with $\beta=5$):

```
x=rand(500,1); p=2; q=3; n=4; beta=5;
[y,b]=resample(x,p,q,n,beta);
```

- **intfilt**: Interpolation and decimation FIR filter design.

Input parameters:

- the interpolation or decimation factor r ;
- the factor l , equal to $(n+2) / r$, where n is the filter order;
- the fraction α of the sampling frequency corresponding to the filter bandwidth;
- In cases where one wants to do Lagrangian interpolation, the filter order n and the parameter 'lagrange' are provided instead of l and α respectively.

Output parameter: the vector b containing the filter coefficients.

Example 1:

```
r=2; l=3; alpha=0.4;
b=intfilt(r,l,alpha);
```

Example 2 (Lagrangian interpolator of fifth order for a sequence with two zeros between nonzero samples):

```
r=3; n=5;
b=intfilt(r,n,'lagrange');
```

8.15 Summary

In this chapter we have studied the concepts of decimation and interpolation from the digital signal processing point of view. Suitable models for representing decimation and interpolation operations have been presented.

The specifications of the filters necessary in the decimation and interpolation processes have been studied and several design alternatives for those filters have been mentioned. We have also presented the use of interpolators and decimators in digital filter design.

The chapter also addressed the block implementation of digital filters by employing the overlapped blocking framework. In addition, we briefly discussed the effects of rate changes in random signals.

Finally, MATLAB experiments and functions which aid in the design and implementation of multirate systems were presented.

Although this subject goes far beyond that which we have discussed here, it is expected that the material presented is sufficient for the solution of many practical problems. For a more in-depth material, the reader is advised to resort to one of the many excellent books on the subject, for instance Crochiere & Rabiner (1983) and Vaidyanathan (1993).

8.16 Exercises

- 8.1 Deduce the two noble identities (Equations (8.30) and (8.31)) using an argument in the time domain.
- 8.2 Prove Equations (8.28) and (8.29).
- 8.3 The sequence

$$x = 0.125, 0.25, 0.5, 1, 2, 4$$

is filtered by a filter with transfer function

$$H(z) = \frac{1}{3}(1 + z^{-1} + z^{-2})$$

and the result is decimated by 2. The output is then upsampled by 2 and filtered by the same $H(z)$. Generate the resulting sequence and interpret the results.

- 8.4 Show that decimation is a time-varying, but periodically time-invariant, operation and that interpolation is a time-invariant operation.
- 8.5 Design two interpolation filters, one lowpass and the other multiband (Equation (8.22)), with specifications

$$\begin{aligned}\delta_p &= 0.0002 \\ \delta_r &= 0.0001 \\ \Omega_s &= 10\,000 \text{ rad/s} \\ L &= 10.\end{aligned}$$

Assume that we are interested in the information within $0 \leq \Omega \leq 0.2\Omega_s$. Compare the computational efficiency of the resulting filters.

- 8.6 Deduce the equivalence of Figures 8.17a and 8.17b from Equation (8.37).
- 8.7 Show the polyphase decomposition structure, as given in Equation (8.35), for an FIR filter with impulse response

$$h(n) = 0.25, 0.5, 0.5, 1, 1, 0.5, 0.5, 0.25$$

for $n = 0, 1, \dots, 7$. Try to minimize the number of multiplications.

- 8.8 Show the polyphase decomposition structure, as given in Equation (8.37), for an FIR filter with impulse response

$$h(n) = -0.375, 0.25, -0.5, 1, -1, 0.5, -0.25, 0.375$$

- for $n = 0, 1, \dots, 7$. Try to minimize the number of multiplications.
- 8.9 Show the polyphase decomposition structures, as given in Equations (8.35) and (8.37), for an FIR filter with impulse response

$$h(n) = a, b, c, d, e, -d, -c, -b, -a$$

- for $n = 0, 1, \dots, 8$, minimizing the number of multiplications.
- 8.10 Show that the zeroth polyphase component of an L th band filter is constant in the frequency domain.
- 8.11 Prove that for a linear-phase filter whose impulse response has length ML , its M polyphase components (Equation (8.35)) should satisfy

$$E_j(z) = \pm z^{-(L-1)} E_{M-1-j}(z^{-1}).$$

- 8.12 Show one efficient FIR structure for decimation and another for interpolation, based in the considerations given in Sections 8.3 and 8.4 respectively.
- 8.13 Repeat Exercise 8.12 for the case of an IIR filter.
- 8.14 Show, through an example, that it is more efficient to implement a decimator-by-50 through several decimation stages than using just one. Use the formula for estimating the order of a lowpass FIR filter presented in Exercise 5.25.
- 8.15 Design a lowpass filter using one decimation/interpolation stage satisfying the following specifications:

$$\begin{aligned}\delta_p &= 0.001 \\ \delta_r &\leq 0.0001 \\ \Omega_p &= 0.01\Omega_s \\ \Omega_r &= 0.02\Omega_s \\ \Omega_s &= 1 \text{ rad/s.}\end{aligned}$$

- Repeat the problem using two decimation/interpolation stages and compare the computational complexity of the results.
- 8.16 Design the bandpass filter for tone detection presented in Exercise 5.20, with center frequency of 700 Hz, using the concept of decimation/interpolation. Compare your results with those obtained in Exercise 5.20.
- 8.17 Design a filter satisfying the specifications in Section 8.10.2 using the frequency-response masking approach.
- 8.18 In the serial-to-parallel converter of Figure 8.25, see also Figure 8.19, assume $M = 2$ and $L = 3$ and that the input signal is a sequence given by

$$x(n) = 0, 0, a, b, c, d, e, f, g, h, i, 0, 0$$

for $n = 0, 1, \dots, 12$. Determine the output sequences.

- 8.19 In the parallel-to-serial converter of Figure 8.25, see also Figure 8.18, assume $M = 2$ and $N = 3$ and that the input signals are given by

$$x_1(n) = 0, 0, a$$

$$x_2(n) = b, c, d$$

$$x_3(n) = e, f, g$$

$$x_4(n) = h, i, 0.$$

Determine the output sequence.

- 8.20 In Exercise 8.7, a nonminimum delay overlapped solution is possible by choosing

$$\mathbf{C}_l(z) = \begin{bmatrix} D_0(z) & R_1(z) & 0 & 0 \\ 0 & R_0(z) & R_1(z) & 0 \\ 0 & 0 & R_0(z) & D_1(z) \end{bmatrix}.$$

Derive the corresponding solution and depict the resulting structure.

- 8.21 Given the matrix

$$\mathbf{C}(z) = \begin{bmatrix} R_0(z) & R_1(z) & R_2(z) \\ z^{-1}R_2(z) & R_0(z) & R_1(z) \\ z^{-1}R_1(z) & z^{-1}R_2(z) & R_0(z) \end{bmatrix},$$

verify if $\mathbf{C}^2(z)$ is pseudo-circulant.

- 8.22 Given the matrix

$$\mathbf{C}(z) = \begin{bmatrix} R_0(z) & R_1(z) \\ z^{-1}R_1(z) & R_0(z) \end{bmatrix},$$

show that its inverse is pseudo-circulant.

- 8.23 Assume you want to implement an FIR filter of length 16 using the fast convolution structure of Figure 8.27 described by Equation (8.82). Compute the delay and the number of multiplications per sample for decompositions ranging from 3 to 81 subfilters.
- 8.24 Implement the transfer function below using the fast convolution structure of Figure 8.27 with the subfilters of length one:

$$H(z) = 1 + z^{-1} + 2z^{-2} + 4z^{-3}.$$

- 8.25 Implement the transfer function below using the fast convolution structure of Figure 8.27 with the subfilters of length one:

$$H(z) = 0.25 + 0.5z^{-1} - 0.5z^{-2} - 0.25z^{-3}.$$

- 8.26 Design the filter satisfying the following specifications using the minimax approach and show its submatrices of overlapped blocking filtering, for $M = L = 4$ and $N = 2$:

$$\delta_p = 0.01$$

$$\delta_r = 0.005$$

$$\Omega_p = 0.05\Omega_s$$

$$\Omega_r = 0.1\Omega_s$$

$$\Omega_s = 2\pi \text{ rad/s.}$$

- 8.27 Design the filter of Exercise 8.26 with the minimax approach and show its submatrices of overlapped blocking filtering for $M = N = 4$ and for $L = 2$.
- 8.28 Design the filter of Exercise 8.26 with the WLS approach and derive its overlapped structure I of Figure 8.26.
- 8.29 Design the filter of Exercise 8.26 with the WLS approach and derive its overlapped structure II of Figure 8.27.
- 8.30 Design a filter satisfying the following specifications with the minimax and WLS approaches and show their implementations employing the overlapped structure I of Figure 8.26:

$$\delta_p = 0.01$$

$$\delta_r = 0.05$$

$$\Omega_p = 0.8 \frac{\Omega_s}{2}$$

$$\Omega_r = 0.6 \frac{\Omega_s}{2}$$

$$\Omega_s = 2\pi \text{ rad/s.}$$

- 8.31 Design a filter satisfying the following specifications with the minimax approach and show its submatrices of overlapped blocking filtering for $M = L = 2$ and for $N = 1$:

$$\delta_p = 0.01$$

$$\delta_r = 0.01$$

$$\Omega_{p_1} = 0.48 \frac{\Omega_s}{2}$$

$$\Omega_{p_2} = 0.55 \frac{\Omega_s}{2}$$

$$\Omega_{r_1} = 0.4 \frac{\Omega_s}{2}$$

$$\Omega_{r_2} = 0.6 \frac{\Omega_s}{2}$$

$$\Omega_s = 2\pi \text{ rad/s.}$$

- 8.32 Solve Exercise 8.31 using the WLS design approach.
- 8.33 Describe the overlap-and-save method in Section 3.4.3 using the overlapped block filtering framework seen in Figure 8.25. Specify the matrix $\mathbf{C}(z)$ in the function of the filter impulse response $h(n)$, its length, and the length of the input block.
- 8.34 Prove Equation (8.52). *Hint:* Use the noble identities.
- 8.35 Show that, for a realization of a WSS process $\{X\}$ applied as input process to a serial-to-parallel converter, the PSD matrix $\boldsymbol{\Gamma}_X(z)$ is pseudo-circulant.
- 8.36 Let us consider the case where a vector $\{\mathbf{X}(n)\}$ represents an $M \times 1$ WSS process, and that

$$Y_i(n) = W_i(n)X_i(n)$$

for $i = 0, 1, \dots, M - 1$. Show that $\{\mathbf{Y}(n)\}$ is WSS if and only if $W_i(n) = \kappa_i e^{j\phi_i n}$. This result indicates that the only time dependency between $Y_i(n)$ and $X_i(n)$ is at the exponent, where κ_i is a possibly complex constant and ϕ_i is a real constant.

- 8.37 Show that if you apply an input process WSCS with period N to a linear periodically time-varying system with period N , the output process will also be WSCS with period N .

9.1 Introduction

In Chapter 8 we dealt with multirate systems in general; that is, systems in which more than one sampling rate coexist. Operations of decimation, interpolation, and sampling-rate changes were studied, as well as some filter design techniques using multirate concepts.

In a number of applications, it is necessary to split a digital signal into several frequency bands. After such decomposition, the signal is represented by more samples than in the original stage. However, we can attempt to decimate each band, ending up with a digital signal decomposed into several frequency bands without increasing the overall number of samples. The question is whether it is possible to recover the original signal exactly from the decimated bands. Systems which decompose and reassemble the signals are generally called filter banks.

In this chapter, we deal with filter banks, showing several ways in which a signal can be decomposed into critically decimated frequency bands, and recovered from them with minimum error. We start with an analysis of M -band filter banks, giving conditions for perfect reconstruction. Then we perform both frequency- and time-domain analyses of filter banks, followed by a discussion on orthogonality. We also treat two-band perfect reconstruction filter banks, and present the special designs for quadrature mirror filters (QMFs) and conjugate quadrature filters (CQFs). In addition, we shift to M -band filter banks, analyzing block transforms, cosine-modulated filter banks, and lapped transforms. We finish the chapter with a Do-it-yourself section followed by a brief description of functions from the MATLAB Wavelet Toolbox which are useful for filter banks design and implementation.

9.2 Filter banks

In some applications, such as signal analysis, signal transmission, and signal coding, a digital signal $x(n)$ is decomposed into several frequency bands, as depicted in Figure 9.1.

In such cases, the signal in each of the bands $x_k(n)$, for $k = 0, 1, \dots, (M - 1)$, has at least the same number of samples as the original signal $x(n)$. This implies that, after the M -band decomposition, the signal is represented with at least M times more samples than the original one. However, there are many cases in which this expansion of the number of samples is highly undesirable. One such case is signal transmission (Vetterli

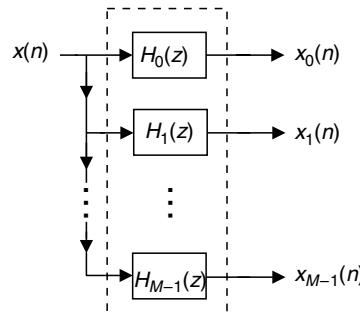


Fig. 9.1. Decomposition of a digital signal into M frequency bands.

and Kovačević, 1995), where more samples stand for more bandwidth and, consequently, increased transmission costs.

In the common case where the signal is uniformly split in the frequency domain – that is, each of the frequency bands $x_k(n)$ has the same bandwidth – a natural question to ask is: Since each band has bandwidth M times smaller than the one of the original signal, could the bands $x_k(n)$ be decimated by a factor of M (critically decimated) without destroying the original information? If this were possible, then one would have a digital signal decomposed into several frequency bands with the same overall number of samples as its original version.

In the next subsection we analyze the general problem of decimating a bandpass signal and performing its inverse operation.

9.2.1 Decimation of a bandpass signal

As was seen in Section 8.3, Equation (8.6), if the input signal $x(m)$ is lowpass and band-limited to $[-\pi/M, \pi/M]$, the aliasing after decimation by a factor of M can be avoided. However, if before decimation the signal is split into M uniform real frequency bands using the scheme in Figure 9.2, then the k th band will be confined to $[-(k + 1)\pi/M, -k\pi/M] \cup [k\pi/M, (k + 1)\pi/M]$ (see Section 2.4.7). This implies that band k , for $k \neq 0$, is necessarily not confined to $[-\pi/M, \pi/M]$. However, by examining Equation (8.6), one can see that aliasing is still avoided in this case. The only difference is that, after decimation, the spectrum contained in $[-(k + 1)\pi/M, -k\pi/M]$ is mapped into $[0, \pi]$, if k is odd, or into $[-\pi, 0]$, if k is even. Similarly, the spectrum contained in the interval $[k\pi/M, (k + 1)\pi/M]$ is mapped into $[-\pi, 0]$, if k is odd, or into $[0, \pi]$, if k is even (Crochiere & Rabiner, 1983). Then, the decimated band k of Figure 9.2 will be as shown in Figures 9.3a and 9.3b, for k odd and even respectively. Note that, from the above discussion, we see that both $\omega = 2l\pi/M$ and $\omega = -2l\pi/M$ in the original signal are mapped to $\omega = 0$ in the decimated signal. Therefore, in order to allow proper reconstruction of signals having components of the form $A_l \cos[(2l\pi/M)n]$, the ideal filters must have half the passband gain for $\omega = \pm 2l\pi/M$ (see Exercise 9.1).

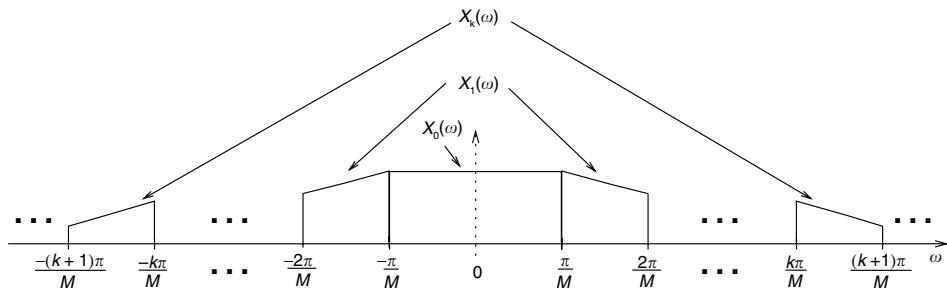


Fig. 9.2. Uniform split of a signal into M real bands.

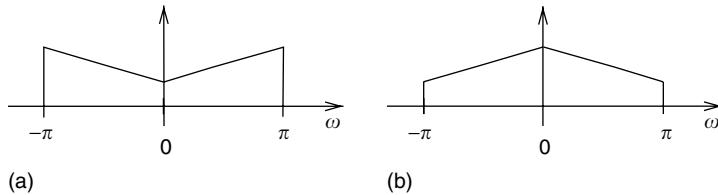


Fig. 9.3. Spectrum of band k decimated by a factor of M : (a) k odd; (b) k even.

9.2.2 Inverse decimation of a bandpass signal

We have just seen that a bandpass signal can be decimated by M without aliasing, provided that its spectrum is confined to $[-(k + 1)\pi/M, -k\pi/M] \cup [k\pi/M, (k + 1)\pi/M]$. The next natural question is: Can the original bandpass signal be recovered from its decimated version by an interpolation operation? The case of lowpass signals was examined in Section 8.6. Here we analyze the bandpass case.

The spectrum of a decimated bandpass signal is shown in Figure 9.3. After interpolation by M , the spectrum for k odd will be as given in Figure 9.4.

If we want to recover band k , as in Figure 9.2, it suffices to keep the region of the spectrum in Figure 9.4 within $[-(k + 1)\pi/M, -k\pi/M] \cup [k\pi/M, (k + 1)\pi/M]$. For k even, the procedure is entirely analogous.

As a general conclusion, the process of decimating and interpolating a bandpass signal is similar to the case of a lowpass signal, seen in Figure 8.13, with the difference that for the bandpass case $H(z)$ must be a bandpass filter with bandwidth $[-(k + 1)\pi/M, -k\pi/M] \cup [k\pi/M, (k + 1)\pi/M]$.

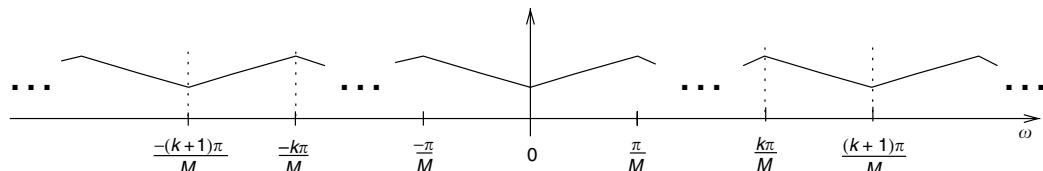


Fig. 9.4. Spectrum of band k after decimation and interpolation by a factor of M for k odd.

9.2.3 Critically decimated M -band filter banks

It is clear from the previous discussion that, if a signal $x(m)$ is decomposed into M nonoverlapping bandpass channels B_k , with $k = 0, 1, \dots, (M - 1)$, such that $\bigcup_{k=0}^{M-1} B_k = [-\pi, \pi]$, then it can be recovered by just summing these M channels. However, as conjectured above, exact recovery of the original signal may not be possible if each channel is decimated by M .

In Sections 9.2.1 and 9.2.2, we examined a way to recover the bandpass signal from its decimated version. In fact, all that is needed are interpolations followed by filters with passband $[-(k + 1)\pi/M, -k\pi/M] \cup [k\pi/M, (k + 1)\pi/M]$.

This whole process of decomposing a signal and restoring it from the frequency bands is depicted in Figure 9.5. We often refer to it as an M -band filter bank. The signals $u_k(m)$ occupy distinct frequency bands, which are collectively called sub-bands. If the input signal can be recovered exactly from its sub-bands, the structure is called an M -band perfect reconstruction filter bank. Figure 9.6 presents a perfect reconstruction filter bank for the two-band case.

However, the filters required for the M -band perfect reconstruction filter bank described above are not realizable; that is, at best they can be only approximated (see Chapter 5).

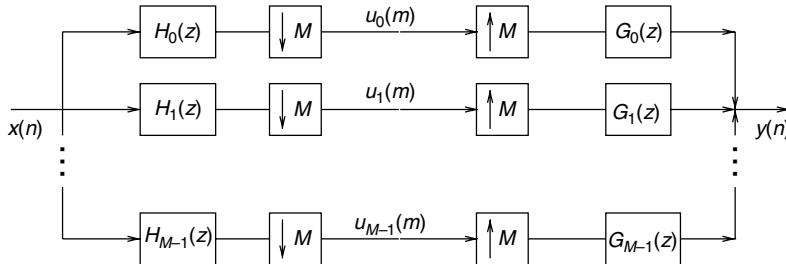


Fig. 9.5. Block diagram of an M -band filter bank.

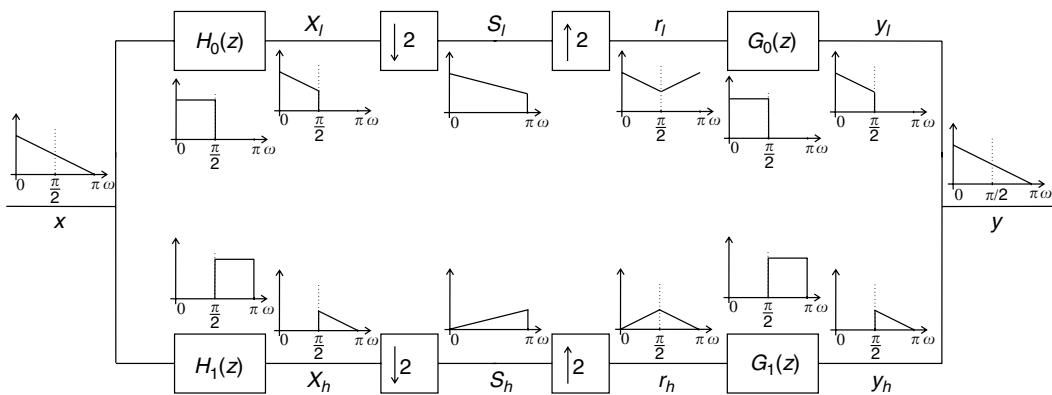


Fig. 9.6. A two-band perfect reconstruction filter bank using ideal filters.

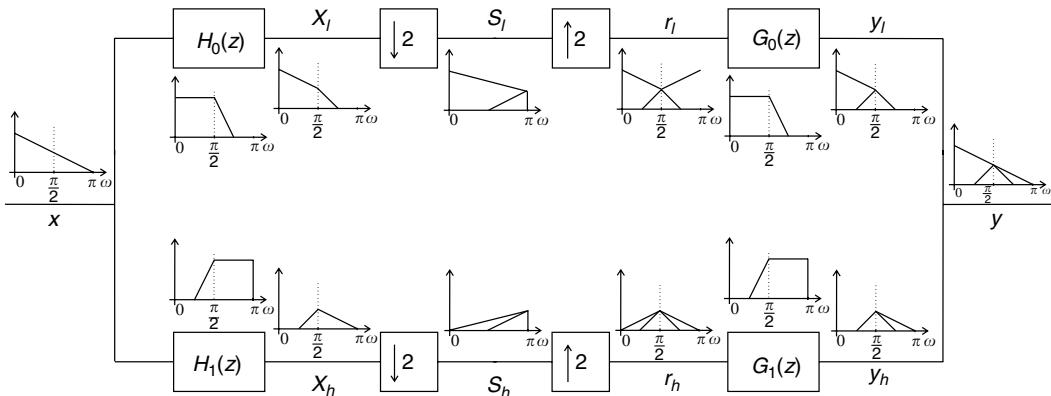


Fig. 9.7. Two-band filter bank using realizable filters.

Therefore, in a first analysis, the original signal would be only approximately recoverable from its decimated frequency bands.

Figure 9.7 depicts a two-band filter bank using realizable filters. One can see that, since the filters $H_0(z)$ and $H_1(z)$ are not ideal, the sub-bands $s_l(m)$ and $s_h(m)$ have aliasing. In other words, the signals $x_l(n)$ and $x_h(n)$ cannot be correctly recovered from $s_l(m)$ and $s_h(m)$ respectively. Nevertheless, by closely examining Figure 9.7, one can see that, since $y_l(n)$ and $y_h(n)$ are added in order to obtain $y(n)$, the aliased components of $y_l(n)$ can be combined with those of $y_h(n)$. In that manner, in principle, there is no reason why these aliased components could not be made to cancel each other, yielding $y(n)$ equal to $x(n)$. In such a case, the original signal could be recovered from its sub-band components. This setup can be used not only for the two-band case, but for the general M -band case as well (Vaidyanathan, 1993).¹

In an M -band filter bank as shown in Figure 9.5, the filters $H_k(z)$ and $G_k(z)$ are usually referred to as the analysis and synthesis filters of the filter bank. In the remainder of this chapter we will examine methods to design the analysis filters $H_k(z)$ and the synthesis filters $G_k(z)$ so that perfect reconstruction can be achieved, or, at least, arbitrarily approximated.

9.3 Perfect reconstruction

9.3.1 M -band filter banks in terms of polyphase components

As we will see next, polyphase decompositions can give us a valuable insight into the properties of M -channel filter banks. By substituting each of the filters $H_k(z)$ and $G_k(z)$ by

¹ By examining Figure 9.7, one can get the impression that the aliasing cannot be perfectly canceled, because only nonnegative quantities are being added. However, only the magnitudes of the signals are shown, and in additions of the spectra the phase should obviously be considered as well as the magnitude.

their polyphase components, according to Equations (8.35) and (8.37), we have that

$$H_k(z) = \sum_{j=0}^{M-1} z^{-j} E_{kj}(z^M) \quad (9.1)$$

$$G_k(z) = \sum_{j=0}^{M-1} z^{-(M-1-j)} R_{jk}(z^M), \quad (9.2)$$

where $E_{kj}(z)$ is the j th polyphase component of $H_k(z)$ and $R_{jk}(z)$ is the j th polyphase component of $G_k(z)$. By defining the matrices $\mathbf{E}(z)$ and $\mathbf{R}(z)$ as those having entries $E_{ij}(z)$ and $R_{ij}(z)$, for $i,j = 0, 1, \dots, (M-1)$, we have that the polyphase decompositions in Equations (9.1) and (9.2) can be expressed as

$$\begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = \mathbf{E}(z^M) \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(M-1)} \end{bmatrix} \quad (9.3)$$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \\ \vdots \\ G_{M-1}(z) \end{bmatrix} = \mathbf{R}^T(z^M) \begin{bmatrix} z^{-(M-1)} \\ z^{-(M-2)} \\ \vdots \\ 1 \end{bmatrix}. \quad (9.4)$$

Therefore, the M -band filter bank in Figure 9.5 can be represented as in Figure 9.8a (Vaidyanathan, 1993), which turns into Figure 9.8b once the noble identities are applied.

In signal processing it is often advantageous to split a sequence $x(k)$ into several frequency bands prior to processing. As Figure 9.9 illustrates, the analysis filters $H_i(z)$, for $i = 0, 1, \dots, (M-1)$, are comprised of a lowpass filter $H_0(z)$, several bandpass filters $H_i(z)$, for $i = 1, 2, \dots, (M-2)$, and a highpass filter $H_{M-1}(z)$. Ideally, these filters have nonoverlapping passbands.

Since the bandwidth of each analysis filter output is M times smaller than in the original signal, we can decimate each $x_i(k)$ by a factor of L smaller than or equal to M and still avoid aliasing. In addition, for $L \leq M$, it is possible to retain all information contained in the input signal by properly designing the analysis filters in conjunction with the synthesis filters $G_i(z)$, for $i = 0, 1, \dots, (M-1)$. On the other hand, if $L > M$ there is a loss of information due to aliasing, which does not allow the recovery of the original signal. For $L = M$, we refer to the filter bank as maximally (or critically) decimated. For $L < M$, the filter bank is called oversampled (or noncritically sampled), since the set of sub-bands comprises more samples than the input signal.

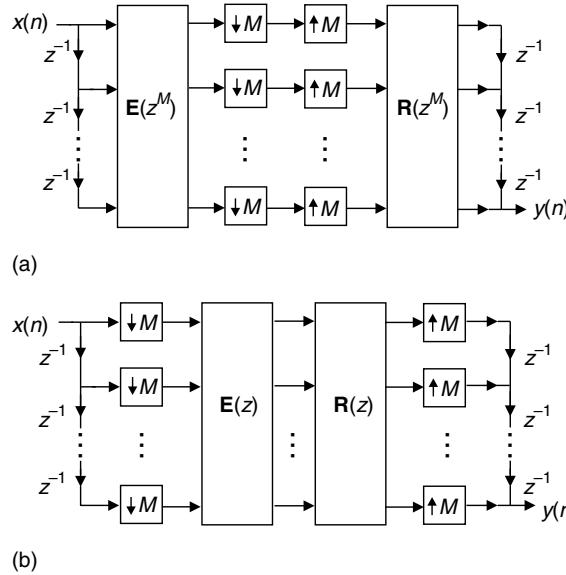


Fig. 9.8. M -band filter bank in terms of the polyphase components: (a) before application of the noble identities; (b) after application of the noble identities.

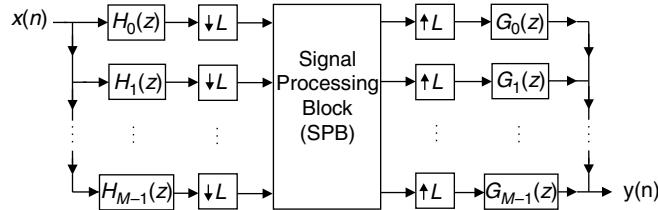
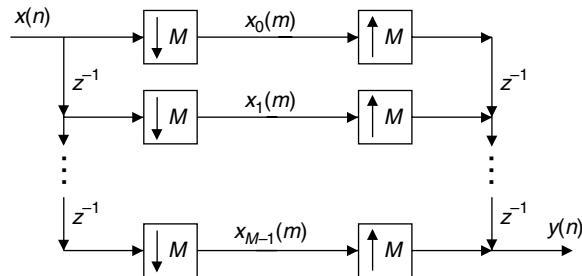
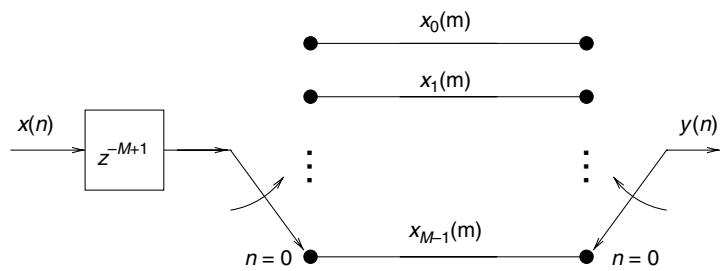


Fig. 9.9. Signal processing in sub-bands.

9.3.2 Perfect reconstruction M -band filter banks

In Figure 9.8b, if $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}$, where \mathbf{I} is the identity matrix, the M -band filter bank becomes the one shown in Figure 9.10 by the commutator models of Figures 8.19 and 8.18b respectively, we arrive at the scheme depicted in Figure 9.11, which is clearly equivalent to a pure delay. Therefore, the condition $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}$ guarantees perfect reconstruction for the M -band filter bank (Vaidyanathan, 1993). It should be noted that if $\mathbf{R}(z)\mathbf{E}(z)$ is equal to a pure delay then one can still consider that perfect reconstruction holds. Therefore, the weaker condition

$$\mathbf{R}(z)\mathbf{E}(z) = z^{-\Delta}\mathbf{I} \quad (9.5)$$

Fig. 9.10. **M-band filter bank when $R(z)E(z) = I$.**Fig. 9.11. **The commutator model of an M -band filter bank when $R(z)E(z) = I$ is equivalent to a pure delay.**

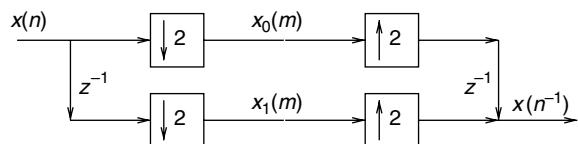
is sufficient for perfect reconstruction. Then, in the general case given by Equation (9.5), the total delay introduced by a perfect reconstruction filter bank is

$$\Delta_{\text{total}} = M\Delta + M - 1, \quad (9.6)$$

where $M\Delta$ is the delay originated from the polyphase matrices product and the term $(M - 1)$ accounts for the delay introduced by the commutator.

Let us now illustrate how a simple perfect reconstruction filter bank can be built. For the two-band case, the structure of Figure 9.12 is equivalent to a unit delay. This implies that the structure of Figure 9.13 is also equivalent to the unit delay, since the inserted matrices in the lower rate processing are inverses of one another.

By expanding the matrix operations, we obtain the realization of the unit delay as depicted in Figure 9.14. Now, by “splitting” the decimators and interpolators and rearranging the gain blocks, we obtain the implementation of Figure 9.15. Since the gains and

Fig. 9.12. **Two-band unit delay.**

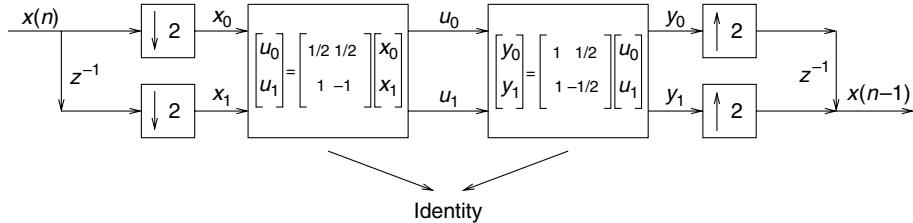


Fig. 9.13. Two-band unit delay, including inverse matrices.

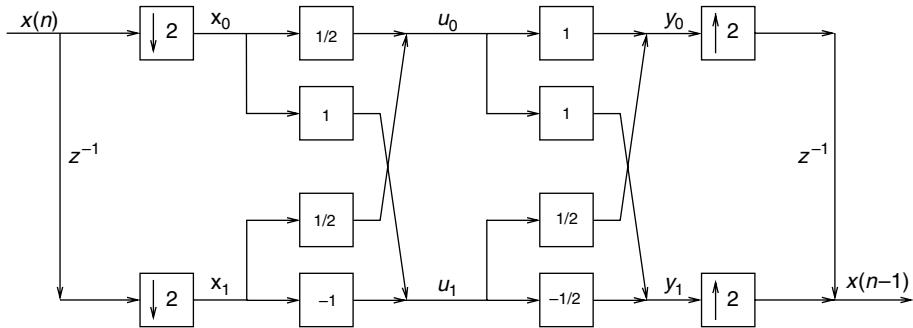


Fig. 9.14. Two-band unit delay, with explicit realization of the matrix products.

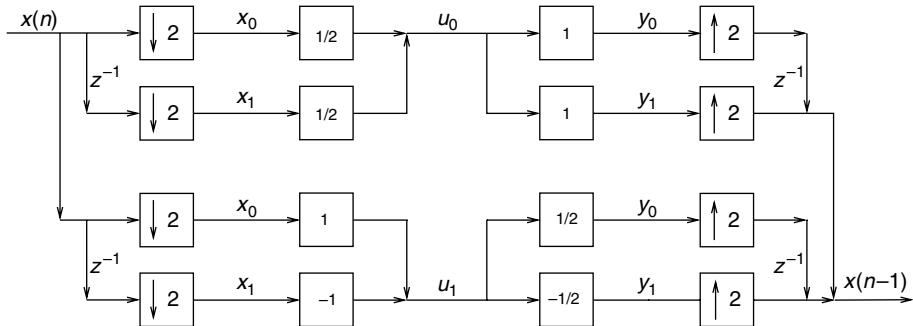


Fig. 9.15. Two-band unit delay, splitting the decimators and interpolators.

interpolators/decimators commute, the structure of Figure 9.15 is equivalent to the one in Figure 9.16.

By “merging” the decimators/interpolators we reach the realization of the unit delay as shown in Figure 9.17. Figure 9.17 is equivalent to the filter bank in Figure 9.18.

Example 9.1 revisits this perfect reconstruction filter bank employing the polyphase decomposition framework. It will be verified in this example that perfect reconstruction has been achieved with analysis and synthesis lowpass/highpass filters which are quite far from being ideal.

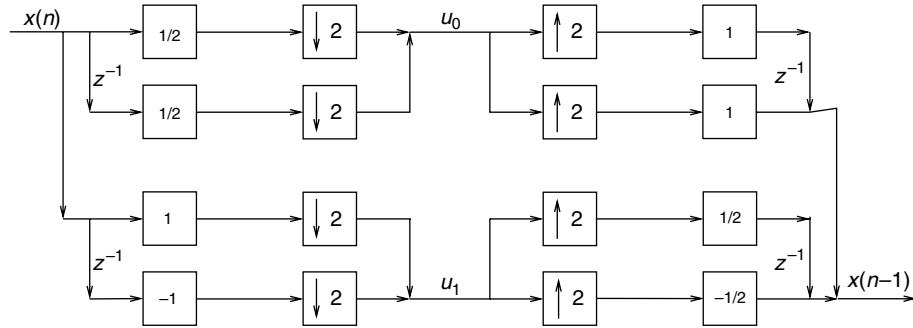


Fig. 9.16. Two-band unit delay, moving the decimators and interpolators.

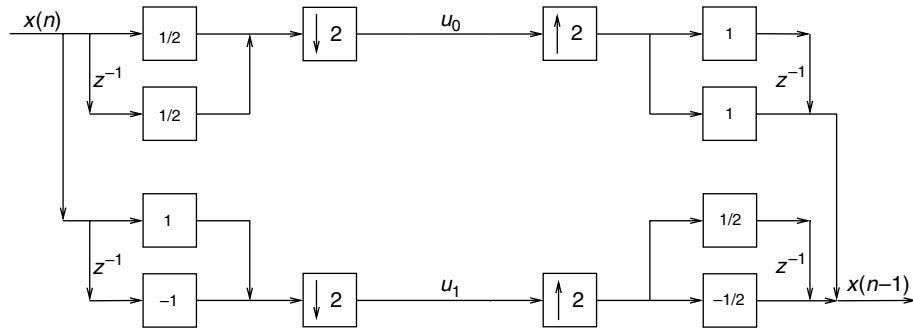


Fig. 9.17. Two-band unit delay, merging the decimators and interpolators.

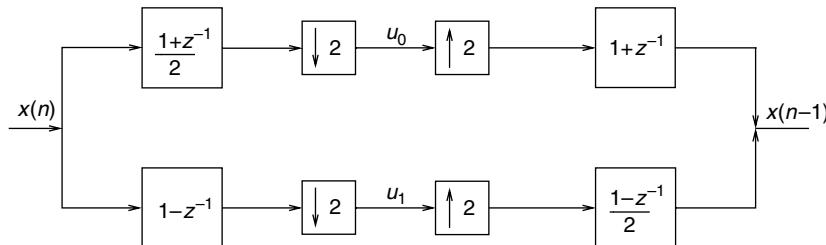


Fig. 9.18. Two-band filter bank with perfect reconstruction.

Example 9.1. Let $M = 2$, and

$$\mathbf{E}(z) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{bmatrix} \quad (9.7)$$

$$\mathbf{R}(z) = \begin{bmatrix} 1 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix}. \quad (9.8)$$

Show that these matrices characterize a perfect reconstruction filter bank, and find the analysis and synthesis filters and their corresponding polyphase components.

Solution

Clearly, $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}$ and the filter bank yields perfect reconstruction. The polyphase components $E_{kj}(z)$ of the analysis filters $H_k(z)$, and $R_{jk}(z)$ of the synthesis filters $G_k(z)$ are then

$$E_{00}(z) = \frac{1}{2}, \quad E_{01}(z) = \frac{1}{2}, \quad E_{10}(z) = 1, \quad E_{11}(z) = -1; \quad (9.9)$$

$$R_{00}(z) = 1, \quad R_{01}(z) = \frac{1}{2}, \quad R_{10}(z) = 1, \quad R_{11}(z) = -\frac{1}{2}. \quad (9.10)$$

From Equations (9.9) and (9.1) we can find $H_k(z)$, and from Equations (9.10) and (9.2) we can find $G_k(z)$. They are

$$H_0(z) = \frac{1}{2}(1 + z^{-1}) \quad (9.11)$$

$$H_1(z) = 1 - z^{-1} \quad (9.12)$$

$$G_0(z) = 1 + z^{-1} \quad (9.13)$$

$$G_1(z) = -\frac{1}{2}(1 - z^{-1}). \quad (9.14)$$

This is known as the Haar filter bank. The normalized magnitude responses of $H_0(z)$ and $H_1(z)$ are depicted in Figure 9.19. From Equations (9.11)–(9.14), we see that the magnitude response of $G_k(z)$ is equal to that of $H_k(z)$ for $k = 0, 1$ except for a gain constant. One can see that perfect reconstruction could be achieved with filters that are far from being ideal. In other words, despite the fact that each sub-band is highly aliased, one can still recover the original signal exactly at the output. \triangle

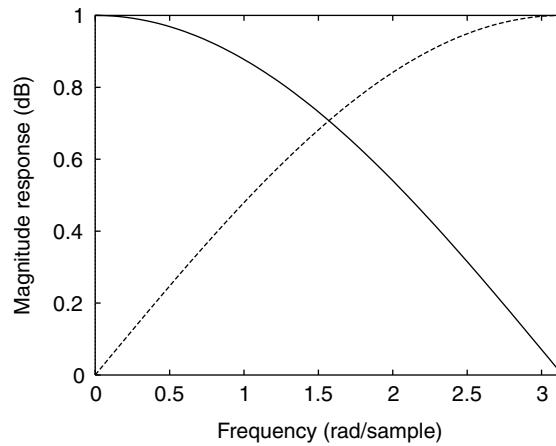


Fig. 9.19.

Magnitude responses of the filters described by Equations (9.11) and (9.12): $H_0(z)$ (solid line); $H_1(z)$ (dashed line).

Example 9.2. Repeat Example 9.1 for the case when

$$\mathbf{E}(z) = \begin{bmatrix} \left(-\frac{1}{8} + \frac{3}{4}z^{-1} - \frac{1}{8}z^{-2}\right) & \left(\frac{1}{4} + \frac{1}{4}z^{-1}\right) \\ \left(\frac{1}{2} + \frac{1}{2}z^{-1}\right) & -1 \end{bmatrix} \quad (9.15)$$

$$\mathbf{R}(z) = \begin{bmatrix} 1 & \left(\frac{1}{4} + \frac{1}{4}z^{-1}\right) \\ \left(\frac{1}{2} + \frac{1}{2}z^{-1}\right) & \left(\frac{1}{8} - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}\right) \end{bmatrix}. \quad (9.16)$$

Solution

Since

$$\mathbf{R}(z)\mathbf{E}(z) = \begin{bmatrix} z^{-1} & 0 \\ 0 & z^{-1} \end{bmatrix} = z^{-1}\mathbf{I} \quad (9.17)$$

then the filter bank has perfect reconstruction. From Equations (9.15) and (9.16), the polyphase components $E_{kj}(z)$ of the analysis filters $H_k(z)$ and the $R_{jk}(z)$ of the synthesis filters $G_k(z)$ are

$$\left. \begin{array}{l} E_{00}(z) = -\frac{1}{8} + \frac{3}{4}z^{-1} - \frac{1}{8}z^{-2} \\ E_{01}(z) = \frac{1}{4} + \frac{1}{4}z^{-1} \\ E_{10}(z) = \frac{1}{2} + \frac{1}{2}z^{-1} \\ E_{11}(z) = -1 \end{array} \right\} \quad (9.18)$$

$$\left. \begin{array}{l} R_{00}(z) = 1 \\ R_{01}(z) = \frac{1}{4} + \frac{1}{4}z^{-1} \\ R_{10}(z) = \frac{1}{2} + \frac{1}{2}z^{-1} \\ R_{11}(z) = \frac{1}{8} - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2} \end{array} \right\}. \quad (9.19)$$

From Equations (9.18) and (9.1) we can find $H_k(z)$, and from Equations (9.19) and (9.2) we can find $G_k(z)$. They are

$$H_0(z) = -\frac{1}{8} + \frac{1}{4}z^{-1} + \frac{3}{4}z^{-2} + \frac{1}{4}z^{-3} - \frac{1}{8}z^{-4} \quad (9.20)$$

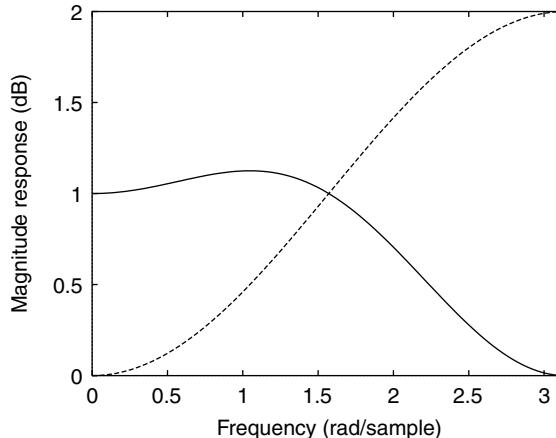


Fig. 9.20. Magnitude responses of the filters described by Equations (9.20) and (9.21): $H_0(z)$ (solid line); $H_1(z)$ (dashed line).

$$H_1(z) = \frac{1}{2} - z^{-1} + \frac{1}{2}z^{-2} \quad (9.21)$$

$$G_0(z) = \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2} \quad (9.22)$$

$$G_1(z) = \frac{1}{8} + \frac{1}{4}z^{-1} - \frac{3}{4}z^{-2} + \frac{1}{4}z^{-3} + \frac{1}{8}z^{-4}. \quad (9.23)$$

The magnitude responses of the analysis filters are depicted in Figure 9.20. \triangle

Example 9.3. If the analysis filters of a perfect reconstruction filter bank are given by

$$\left. \begin{aligned} H_0(z) &= 1 + z^{-1} + \frac{1}{2}z^{-2} \\ H_1(z) &= 1 - z^{-1} + \frac{1}{2}z^{-2} \end{aligned} \right\}, \quad (9.24)$$

determine its synthesis filters.

Solution

From Equation (9.24), we can write the lowpass and highpass analysis filters as

$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 + \frac{1}{2}z^{-2} & 1 \\ 1 + \frac{1}{2}z^{-2} & -1 \end{bmatrix}}_{\mathbf{E}(z^2)} \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \quad (9.25)$$

and then the polyphase analysis matrix is

$$\mathbf{E}(z) = \begin{bmatrix} 1 + \frac{1}{2}z^{-1} & 1 \\ 1 + \frac{1}{2}z^{-1} & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 + \frac{1}{2}z^{-1} & 0 \\ 0 & 1 \end{bmatrix}. \quad (9.26)$$

Therefore, since the filter has perfect reconstruction, we must have that $\mathbf{R}(z) = z^{-\Delta} \mathbf{E}^{-1}(z)$, which gives

$$\mathbf{R}(z) = z^{-\Delta} \begin{bmatrix} \frac{1}{1 + \frac{1}{2}z^{-1}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix} = \frac{z^{-\Delta}}{2} \begin{bmatrix} \frac{1}{1 + \frac{1}{2}z^{-1}} & \frac{1}{1 + \frac{1}{2}z^{-1}} \\ 1 & -1 \end{bmatrix}. \quad (9.27)$$

Then, from Equation (9.4), the synthesis filters are

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \mathbf{R}^T(z) \begin{bmatrix} z^{-1} \\ 1 \end{bmatrix} = \frac{z^{-\Delta}}{2} \begin{bmatrix} \frac{1}{1 + \frac{1}{2}z^{-2}} & 1 \\ \frac{1}{1 + \frac{1}{2}z^{-2}} & -1 \end{bmatrix} \begin{bmatrix} z^{-1} \\ 1 \end{bmatrix}; \quad (9.28)$$

that is:

$$\left. \begin{aligned} G_0(z) &= z^{-\Delta} \frac{1 + z^{-1} + \frac{1}{2}z^{-2}}{2 + z^{-2}} \\ G_1(z) &= z^{-\Delta} \frac{-1 + z^{-1} - \frac{1}{2}z^{-2}}{2 + z^{-2}} \end{aligned} \right\}, \quad (9.29)$$

which correspond to stable IIR filters. Note that in this case the FIR solution is not possible. \triangle

Example 9.4. Assume the analysis filters of a three-band perfect reconstruction filter bank are given by

$$\left. \begin{aligned} H_0(z) &= z^{-2} + 6z^{-1} + 4 \\ H_1(z) &= z^{-1} + 2 \\ H_2(z) &= 1 \end{aligned} \right\}. \quad (9.30)$$

Determine its synthesis filters.

Solution

From Equations (9.1) and (9.3), the polyphase description of the given analysis filters is

$$\begin{bmatrix} H_0(z) \\ H_1(z) \\ H_2(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 4 & 6 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{\mathbf{E}(z^3)} \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \end{bmatrix},$$

and then

$$\mathbf{E}(z^3) = \begin{bmatrix} 4 & 6 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \quad (9.31)$$

We can have perfect reconstruction if $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}$. Thus:

$$\mathbf{R}(z) = \mathbf{E}^{-1}(z) = \begin{bmatrix} 4 & 6 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -2 \\ 1 & -6 & 8 \end{bmatrix}. \quad (9.32)$$

From Equation (9.4), it follows that

$$\begin{bmatrix} G_0(z) \\ G_1(z) \\ G_2(z) \end{bmatrix} = \mathbf{R}^T(z^3) \begin{bmatrix} z^{-2} \\ z^{-1} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -6 \\ 1 & -2 & 8 \end{bmatrix} \begin{bmatrix} z^{-2} \\ z^{-1} \\ 1 \end{bmatrix}. \quad (9.33)$$

Therefore, the transfer functions of the synthesis subfilters are given by

$$\left. \begin{array}{l} G_0(z) = 1 \\ G_1(z) = z^{-1} - 6 \\ G_2(z) = z^{-2} - 2z^{-1} + 8 \end{array} \right\}. \quad (9.34)$$

We note that the synthesis filters are all FIR. This is only possible because the determinant of the polyphase matrix of the analysis filter is proportional to a pure delay (equal to one in this case). \triangle

9.4 Analysis of M -band filter banks

Filter banks with M bands transform the input signal into M signals. This ensemble of M signals can be regarded as a vector signal, whose time sample comprises one sample from each band. Figure 9.21 depicts this vector signal being submitted to some kind of processing by the “signal processing block.” This processing could consist, for example, of quantization, filtering, or other types of signal transformation.

Using this concept, the analysis of the M -band filter bank can be performed in three different, but equivalent ways:

- *Using the polyphase decomposition as given by Equations (9.3) and (9.4):* As seen in Section 9.3, when the polyphase decomposition is used in the analysis and synthesis

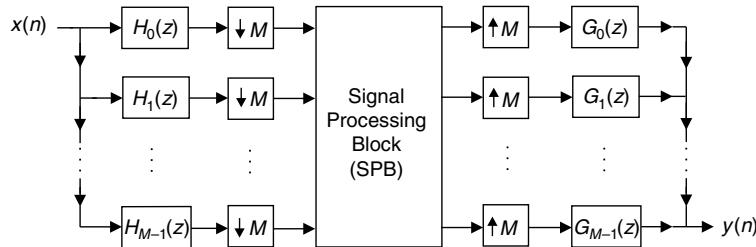


Fig. 9.21. M -band filter bank.

banks, the resulting polyphase matrices are very useful to establish design conditions for perfect reconstruction filter banks.

- *Using the modulation matrix representation:* By representing the sub-band signals in the frequency domain, it is possible to describe the input–output relation of a filter bank. This representation leads to the so-called modulation matrix representation. It is particularly effective in exposing the aliasing effects generated by the decimators. Although this formulation is useful to design alias-free filter banks, it is not the easiest formulation for design purposes.
- *Using time-domain analysis:* It represents the input–output relation of the filter banks in the time domain in terms of the impulse responses of the analysis and synthesis sub-filters. It is very effective in exposing the properties of perfect reconstruction filter banks as defining bases of vector spaces. The analysis operations are seen as signal projections onto bases, while the synthesis operations are seen as signal expansions using bases of the vector space. In this case, properties such as orthogonality and biorthogonality come into play and provide useful insights on filter bank analysis and design.

In the next two subsections we analyze the modulation matrix and time-domain representations of filter banks.

9.4.1 Modulation matrix representation

We now derive the expression for the overall M -band filter bank transfer function assuming the signal processing block as identity. We start by noting that, according to Equation (8.6), the decimated signal $X_d(z)$ is the sum of $X(z^{1/M})$ and its $(M - 1)$ aliased components, $X(z^{1/M} e^{-j(2\pi/M)k})$, for $k = 1, 2, \dots, (M - 1)$; that is:

$$X_d(z) = \frac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} e^{-j(2\pi/M)k}) = \frac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} W_M^k), \quad (9.35)$$

where $W_M = e^{-j2\pi/M}$.

Using the above equation we can express the decimated output of the analysis filters in Figure 9.5, $U_k(z)$, for $k = 0, 1, \dots, (M - 1)$, as

$$\begin{aligned} U_k(z) &= \frac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} W_M^k) H_k(z^{1/M} W_M^k) \\ &= \frac{1}{M} \mathbf{x}_m^T(z^{1/M}) \begin{bmatrix} H_k(z^{1/M}) \\ H_k(z^{1/M} W_M) \\ \vdots \\ H_k(z^{1/M} W_M^{M-1}) \end{bmatrix}, \end{aligned} \quad (9.36)$$

where

$$\mathbf{x}_m(z^{1/M}) = [X(z^{1/M}) \quad X(z^{1/M}W_M) \quad \cdots \quad X(z^{1/M}W_M^{M-1})]^T. \quad (9.37)$$

Therefore, we can define the auxiliary vector

$$\mathbf{U}^T(z) = [U_0(z) \quad U_1(z) \quad \cdots \quad U_{M-1}(z)] = \frac{1}{M} \mathbf{x}_m^T(z^{1/M}) \mathbf{H}_m(z^{1/M}), \quad (9.38)$$

with

$$\mathbf{H}_m(z^{1/M}) = \begin{bmatrix} H_0(z^{1/M}) & H_1(z^{1/M}) & \cdots & H_{M-1}(z^{1/M}) \\ H_0(z^{1/M}W_M) & H_1(z^{1/M}W_M) & \cdots & H_{M-1}(z^{1/M}W_M) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(z^{1/M}W_M^{M-1}) & H_1(z^{1/M}W_M^{M-1}) & \cdots & H_{M-1}(z^{1/M}W_M^{M-1}) \end{bmatrix}. \quad (9.39)$$

Again from Figure 9.5, applying the noble identities, we see that the filter bank output as a function of the sub-bands $U_k(z)$ is given by

$$Y(z) = \sum_{k=0}^{M-1} U_k(z^M) G_k(z) = \mathbf{U}^T(z^M) \mathbf{g}(z), \quad (9.40)$$

where

$$\mathbf{g}(z) = [G_0(z) \quad G_1(z) \quad \cdots \quad G_{M-1}(z)]^T. \quad (9.41)$$

Then, from Equations (9.38) and (9.40), we can express the input–output relation of an M -band filter bank as

$$Y(z) = \frac{1}{M} \mathbf{x}_m^T(z) \mathbf{H}_m(z) \mathbf{g}(z). \quad (9.42)$$

Since $Y(z)$ above is a scalar, we have that $\mathbf{x}_m^T(z) \mathbf{H}_m(z) \mathbf{g}(z) = \mathbf{g}^T(z) \mathbf{H}_m^T(z) \mathbf{x}_m(z)$, and then

$$\begin{aligned} Y(z) &= \frac{1}{M} \mathbf{g}^T(z) \mathbf{H}_m^T(z) \mathbf{x}_m(z) \\ &= \frac{1}{M} [G_0(z) \quad G_1(z) \quad \cdots \quad G_{M-1}(z)] \\ &\quad \times \begin{bmatrix} H_0(z) & H_0(zW_M) & \cdots & H_0(zW_M^{M-1}) \\ H_1(z) & H_1(zW_M) & \cdots & H_1(zW_M^{M-1}) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1}(z) & H_{M-1}(zW_M) & \cdots & H_{M-1}(zW_M^{M-1}) \end{bmatrix} \begin{bmatrix} X(z) \\ X(zW_M) \\ \vdots \\ X(zW_M^{M-1}) \end{bmatrix}. \end{aligned} \quad (9.43)$$

Equations (9.42) and (9.43) are often referred to as the modulation matrix representation of the filter bank.

In Equation (9.43), we have that if

$$\mathbf{g}^T(z)\mathbf{H}_m^T(z) = [B(z) \quad 0 \quad \cdots \quad 0] \quad (9.44)$$

then aliasing is canceled, since

$$Y(z) = \frac{1}{M} [B(z) \quad 0 \quad \cdots \quad 0] \begin{bmatrix} X(z) \\ X(zW_M) \\ \vdots \\ X(zW_M^{M-1}) \end{bmatrix} = \frac{1}{M} B(z)X(z). \quad (9.45)$$

In addition, it can also be inferred that, if $B(z) = Mcz^{-\Delta}$, then the output of the filter bank is just a delayed version of the input scaled by a constant c ; that is, the filter bank has perfect reconstruction.

Example 9.5. Find the perfect reconstruction conditions for all two-band filter banks using the modulation matrix approach.

Solution

For the two-band case, from Equations (9.44) and (9.43), perfect reconstruction requires, since $W_2 = -1$, that

$$[G_0(z) \quad G_1(z)] \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = [2cz^{-\Delta} \quad 0], \quad (9.46)$$

which implies

$$\left. \begin{aligned} H_0(z)G_0(z) + H_1(z)G_1(z) &= 2cz^{-\Delta} \\ H_0(-z)G_0(z) + H_1(-z)G_1(z) &= 0 \end{aligned} \right\}. \quad (9.47)$$

The above equation guarantees the output of the filter bank to be equal to the input delayed by Δ and scaled by a constant c . \triangle

9.4.2 Time-domain analysis

We will carry out the time-domain analysis using the signals in matrix form. Referring to Figures 9.5 and 9.22, we have that the signals $x_k(m)$ at the output of the analysis filters can

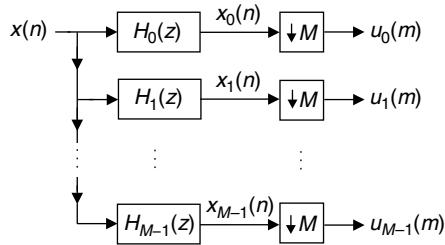


Fig. 9.22. Analysis filter bank.

be expressed as

$$x_k(n) = x(n) * h_k(n) = \sum_{l=-\infty}^{\infty} h_k(n-l)x(l), \quad (9.48)$$

where $h_k(n)$ denotes the impulse response of the k th analysis filter $H_k(z)$, for $k = 0, 1, \dots, (M-1)$.

Since the signals $u_k(m)$ at sub-band k are the signals $x_k(n)$ decimated by a factor of M , from the above equation they are

$$u_k(m) = \sum_{n=-\infty}^{\infty} h_k(mM-n)x(n). \quad (9.49)$$

If from the impulse response of the k th analysis filter bank $h_k(n)$ we define an auxiliary vector corresponding to this impulse response reversed in time and shifted by mM samples as

$$\tilde{h}_k(m) = [\dots \ h_k(mM) \ h_k(mM-1) \ h_k(mM-2) \ \dots \ h_k(mM-n) \ \dots]^T \quad (9.50)$$

such that the n th entry of $\tilde{h}_k^T(m)$ is given by

$$[\tilde{h}_k^T(m)]_n = h_k(mM-n), \quad (9.51)$$

then Equation (9.49) can be written as

$$u_k(m) = \sum_{n=-\infty}^{\infty} [\tilde{h}_k(m)]_n x(n). \quad (9.52)$$

Then, the signal in sub-band k can be expressed as the inner product

$$u_k(m) = \tilde{h}_k^T(m) \mathbf{x}, \quad (9.53)$$

where

$$\mathbf{x} = [\cdots \ x(0) \ x(1) \ x(2) \ \cdots \ x(n) \ \cdots]^T. \quad (9.54)$$

Therefore, the signal in sub-band k in matrix form is

$$\mathbf{u}_k = \begin{bmatrix} \vdots \\ u_k(0) \\ u_k(1) \\ u_k(2) \\ \vdots \\ u_k(m) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \tilde{\mathbf{h}}_k^T(0) \\ \tilde{\mathbf{h}}_k^T(1) \\ \tilde{\mathbf{h}}_k^T(2) \\ \vdots \\ \tilde{\mathbf{h}}_k^T(m) \\ \vdots \end{bmatrix} \mathbf{x} = \mathbf{H}_k \mathbf{x}, \quad (9.55)$$

which can be expanded to

$$\mathbf{u}_k = \underbrace{\begin{bmatrix} \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \cdots & h_k(0) & h_k(-1) & h_k(-2) & \cdots & h_k(-l) & \cdots \\ \cdots & h_k(M) & h_k(M-1) & h_k(M-2) & \cdots & h_k(M-l) & \cdots \\ \cdots & h_k(2M) & h_k(2M-1) & h_k(2M-2) & \cdots & h_k(2M-l) & \cdots \\ \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \cdots & h_k(mM) & h_k(mM-1) & h_k(mM-2) & \cdots & h_k(mM-l) & \cdots \\ \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix}}_{\mathbf{H}_k} \underbrace{\begin{bmatrix} \vdots \\ x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(n) \\ \vdots \end{bmatrix}}_{\mathbf{x}}. \quad (9.56)$$

Now, by defining the vector of outputs of the M sub-bands at sample m as

$$\mathbf{u}(m) = [u_0(m) \ u_1(m) \ \cdots \ u_{M-1}(m)]^T, \quad (9.57)$$

we have from Equation (9.52) that

$$\begin{aligned} \mathbf{u}(m) &= \begin{bmatrix} u_0(m) \\ u_1(m) \\ \vdots \\ u_{M-1}(m) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{h}}_0^T(m) \\ \tilde{\mathbf{h}}_1^T(m) \\ \vdots \\ \tilde{\mathbf{h}}_{M-1}^T(m) \end{bmatrix} \mathbf{x}, \end{aligned} \quad (9.58)$$

or equivalently

$$\mathbf{u}(m) = \underbrace{\begin{bmatrix} \cdots & h_0(mM) & h_0(mM-1) & h_0(mM-2) & \cdots & h_0(mM-l) & \cdots \\ \cdots & h_1(mM) & h_1(mM-1) & h_1(mM-2) & \cdots & h_1(mM-l) & \cdots \\ \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \cdots & h_{M-1}(mM) & h_{M-1}(mM-1) & h_{M-1}(mM-2) & \cdots & h_{M-1}(mM-l) & \cdots \end{bmatrix}}_{\mathbf{H}(m)} \underbrace{\begin{bmatrix} \vdots \\ x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(l) \\ \vdots \end{bmatrix}}_{\mathbf{x}}. \quad (9.59)$$

Therefore, the whole time-domain expression in Equation (9.58) can be written as

$$\underbrace{\begin{bmatrix} \vdots \\ \mathbf{u}(0) \\ \mathbf{u}(1) \\ \mathbf{u}(2) \\ \vdots \\ \mathbf{u}(m) \\ \vdots \end{bmatrix}}_{\mathcal{U}} = \underbrace{\begin{bmatrix} \vdots \\ \mathbf{H}(0) \\ \mathbf{H}(1) \\ \mathbf{H}(2) \\ \vdots \\ \mathbf{H}(m) \\ \vdots \end{bmatrix}}_{\mathcal{H}} \mathbf{x}; \quad (9.60)$$

that is:

$$\mathcal{U} = \mathcal{H}\mathbf{x}. \quad (9.61)$$

Note that, from Equation (9.59), matrix $\mathbf{H}(m)$ consists of matrix $\mathbf{H}(0)$ shifted mM columns to the right. Then matrix \mathcal{H} consists of a concatenation of properly shifted submatrices.

As for the synthesis operation, we refer to Figure 9.23. From Equation (8.21), the signals $y_k(n)$ are

$$y_k(n) = \sum_{m=-\infty}^{\infty} g_k(n - mM) u_k(m). \quad (9.62)$$

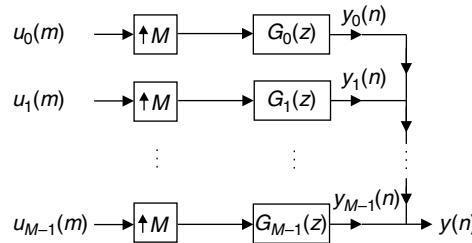


Fig. 9.23.

Synthesis filter bank.

If from the impulse response of the k th synthesis filter bank $g_k(n)$ we define a vector corresponding to this impulse response shifted by mM samples as

$$\mathbf{g}_k(m) = \begin{bmatrix} \vdots \\ g_k(-mM) \\ g_k(1 - mM) \\ g_k(2 - mM) \\ \vdots \\ g_k(n - mM) \\ \vdots \end{bmatrix} \quad (9.63)$$

such that $[\mathbf{g}_k(m)]_n = g_k(n - mM)$, then Equation (9.62) can be written as

$$y_k(n) = \sum_{m=-\infty}^{\infty} [\mathbf{g}_k(m)]_n u_k(m). \quad (9.64)$$

Thus, in matrix form, we have that

$$\begin{aligned} \mathbf{y}_k &= \sum_{m=-\infty}^{\infty} \mathbf{g}_k(m) u_k(m) \\ &= [\dots \mathbf{g}_k(0) \mathbf{g}_k(1) \mathbf{g}_k(2) \dots \mathbf{g}_k(m) \dots] \begin{bmatrix} \vdots \\ u_k(0) \\ u_k(1) \\ u_k(2) \\ \vdots \\ u_k(m) \\ \vdots \end{bmatrix}. \end{aligned} \quad (9.65)$$

Using the definition of the matrix \mathcal{U} in Equation (9.60), the above equation can be rewritten as

$$\mathbf{y}_k = [\dots \ 0 \ \dots \ \mathbf{g}_k(0) \ \dots \ 0 \ \dots \ \mathbf{g}_k(1) \ \dots \ 0 \ \dots \ \mathbf{g}_k(m) \ \dots \ 0 \ \dots] \begin{bmatrix} \vdots \\ u_0(0) \\ u_1(0) \\ \vdots \\ u_k(0) \\ \vdots \\ u_{M-1}(0) \\ u_0(1) \\ u_1(1) \\ \vdots \\ u_k(1) \\ \vdots \\ u_{M-1}(1) \\ \vdots \\ u_0(m) \\ u_1(m) \\ \vdots \\ u_k(m) \\ \vdots \\ u_{M-1}(m) \\ \vdots \end{bmatrix} . \quad (9.66)$$

\mathcal{U}

From Figure 9.23, since

$$y(n) = \sum_{k=0}^{M-1} y_k(n), \quad (9.67)$$

in matrix form we have that

$$\mathbf{y} = \sum_{k=0}^{M-1} \mathbf{y}_k. \quad (9.68)$$

The above equation together with Equation (9.66) imply that

$$\mathbf{y} = \mathcal{G}\mathcal{U}, \quad (9.69)$$

where

$$\begin{aligned} \mathcal{G} &= [\cdots \mathbf{g}_0(0) \cdots \mathbf{g}_{M-1}(0) \mathbf{g}_0(1) \cdots \mathbf{g}_{M-1}(1) \cdots \mathbf{g}_0(m) \cdots \mathbf{g}_{M-1}(m) \cdots] \\ &= \begin{bmatrix} \vdots & \vdots \\ \cdots g_0(0) & \cdots g_{M-1}(0) & g_0(-M) & \cdots g_{M-1}(-M) & \cdots g_0(-mM) & \cdots g_{M-1}(-mM) & \cdots \\ \cdots g_0(1) & \cdots g_{M-1}(1) & g_0(1-M) & \cdots g_{M-1}(1-M) & \cdots g_0(1-mM) & \cdots g_{M-1}(1-mM) & \cdots \\ \cdots g_0(2) & \cdots g_{M-1}(2) & g_0(2-M) & \cdots g_{M-1}(2-M) & \cdots g_0(2-mM) & \cdots g_{M-1}(2-mM) & \cdots \\ \cdots \vdots & \vdots \\ \cdots g_0(n) & \cdots g_{M-1}(n) & g_0(n-M) & \cdots g_{M-1}(n-M) & \cdots g_0(n-mM) & \cdots g_{M-1}(n-mM) & \cdots \\ \cdots \vdots & \vdots \end{bmatrix}. \end{aligned} \quad (9.70)$$

The structure of \mathcal{G} becomes more evident if we define

$$\begin{aligned} \mathbf{G}(m) &= [\mathbf{g}_0(m) \quad \mathbf{g}_1(m) \quad \cdots \quad \mathbf{g}_{M-1}(m)] \\ &= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ g_0(-mM) & g_1(-mM) & \cdots & g_{M-1}(-mM) & \vdots \\ g_0(1-mM) & g_1(1-mM) & \cdots & g_{M-1}(1-mM) & \vdots \\ g_0(2-mM) & g_1(2-mM) & \cdots & g_{M-1}(2-mM) & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_0(n-mM) & g_1(n-mM) & \cdots & g_{M-1}(n-mM) & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \end{aligned} \quad (9.71)$$

Using this definition, \mathcal{G} in Equation (9.70) can be expressed as

$$\mathcal{G} = [\cdots \mathbf{G}(0) \quad \mathbf{G}(1) \quad \mathbf{G}(2) \quad \cdots \quad \mathbf{G}(m) \quad \cdots], \quad (9.72)$$

and thus Equation (9.69) can be rewritten as

$$\mathbf{y} = \underbrace{[\cdots \mathbf{G}(0) \quad \mathbf{G}(1) \quad \mathbf{G}(2) \quad \cdots \quad \mathbf{G}(m) \quad \cdots]}_{\mathcal{G}} \begin{bmatrix} \vdots \\ \mathbf{u}(0) \\ \mathbf{u}(1) \\ \mathbf{u}(2) \\ \vdots \\ \mathbf{u}(m) \\ \vdots \end{bmatrix} \underbrace{\begin{bmatrix} \vdots \\ \mathbf{u}(0) \\ \mathbf{u}(1) \\ \mathbf{u}(2) \\ \vdots \\ \mathbf{u}(m) \\ \vdots \end{bmatrix}}_{\mathcal{U}}. \quad (9.73)$$

Note, from Equation (9.71), that matrix $\mathbf{G}(m)$ consists of matrix $\mathbf{G}(0)$ shifted mM rows up. Then, similar to matrix \mathcal{H} in Equation (9.60), from Equation (9.72), matrix \mathcal{G} consists of a concatenation of properly shifted submatrices.

In order to summarize what we have seen so far, we can take Equations (9.58), (9.61), (9.69), and (9.73) and express the analysis and synthesis operations in a filter bank as

$$\left. \begin{aligned} \mathbf{u}(m) &= \mathbf{H}(m)\mathbf{x}, \quad \text{for } m = \dots, 0, 1, 2, \dots \\ \mathbf{y} &= \sum_{m=-\infty}^{\infty} \mathbf{G}(m)\mathbf{u}(m) \end{aligned} \right\} \quad (9.74)$$

or

$$\left. \begin{aligned} \mathcal{U} &= \mathcal{H}\mathbf{x} \\ \mathbf{y} &= \mathcal{G}\mathcal{U} \end{aligned} \right\}. \quad (9.75)$$

If the filter bank has perfect reconstruction and zero delay, then we have that $\mathbf{x} = \mathbf{y}$. Therefore, from Equation (9.75), we have that \mathcal{H} and \mathcal{G} must satisfy the following constraints:

$$\mathcal{G}\mathcal{H} = \mathcal{H}\mathcal{G} = \mathbf{I}. \quad (9.76)$$

Note that if the filter bank has delay equal to Δ , then $\mathcal{G}\mathcal{H}$ corresponds to a delay of Δ and $\mathcal{H}\mathcal{G}$ corresponds to an advance of Δ .

Example 9.6. For the two-band perfect reconstruction filter bank from Example 9.2 (Equations (9.20), (9.21), (9.22), and (9.23)), we have that

$$\left. \begin{aligned} H_0(z) &= -\frac{1}{8} + \frac{1}{4}z^{-1} + \frac{3}{4}z^{-2} + \frac{1}{4}z^{-3} - \frac{1}{8}z^{-4} \\ H_1(z) &= \frac{1}{2} - z^{-1} + \frac{1}{2}z^{-2} \end{aligned} \right\} \quad (9.77)$$

$$\left. \begin{aligned} G_0(z) &= \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2} \\ G_1(z) &= \frac{1}{8} + \frac{1}{4}z^{-1} - \frac{3}{4}z^{-2} + \frac{1}{4}z^{-3} + \frac{1}{8}z^{-4} \end{aligned} \right\} \quad (9.78)$$

describe the matrices \mathcal{H} and \mathcal{G} .

Solution

The impulse responses of the analysis and synthesis and filters are (only the nonzero samples are shown)

$$\left. \begin{aligned} h_0(0) &= -\frac{1}{8}, & h_0(1) &= \frac{1}{4}, & h_0(2) &= \frac{3}{4}, & h_0(3) &= \frac{1}{4}, & h_0(4) &= -\frac{1}{8} \\ h_1(0) &= \frac{1}{2}, & h_1(1) &= -1, & h_1(2) &= \frac{1}{2} \end{aligned} \right\} \quad (9.79)$$

$$\left. \begin{aligned} g_0(0) &= \frac{1}{2}, & g_0(1) &= 1, & g_0(2) &= \frac{1}{2} \\ g_1(0) &= \frac{1}{8}, & g_1(1) &= \frac{1}{4}, & g_1(2) &= -\frac{3}{4}, & g_1(3) &= \frac{1}{4}, & g_1(4) &= \frac{1}{8} \end{aligned} \right\}. \quad (9.80)$$

Since the number of bands $M = 2$, from Equations (9.58) and (9.60) we have that matrix \mathcal{H} is

and from Equations (9.71) and (9.72) we have that the matrix \mathcal{G} is

$$= \begin{bmatrix} \ddots & \vdots & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{8} & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{4} & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 1 & \frac{1}{4} & 0 & \frac{1}{4} & \cdots \\ \cdots & 0 & 0 & \frac{1}{2} & \frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & 0 & \frac{1}{8} & \cdots \\ \cdots & 0 & 0 & 1 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \cdots \\ \cdots & \frac{1}{2} & \frac{1}{8} & \frac{1}{2} & -\frac{3}{4} & 0 & \frac{1}{8} & 0 & 0 & \cdots \\ \cdots & 1 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & \cdots \\ \cdots & \vdots & \ddots \end{bmatrix}. \quad (9.82)$$

The readers are encouraged to verify by themselves that $\mathcal{H}\mathcal{G} = \mathcal{G}\mathcal{H} = \mathbf{I}$. \triangle

9.4.3 Orthogonality and biorthogonality in filter banks

In Section 9.4.2, Equation (9.76), we have seen that the perfect reconstruction condition requires that

$$\mathcal{H}\mathcal{G} = \mathbf{I}. \quad (9.83)$$

Using the expressions for \mathcal{H} and \mathcal{G} in Equations (9.60) and (9.72), the above equation becomes

$$\begin{bmatrix} \vdots \\ \mathbf{H}(r) \\ \mathbf{H}(r+1) \\ \vdots \\ \mathbf{H}(r+m) \\ \vdots \end{bmatrix} \begin{bmatrix} \cdots & \mathbf{G}(0) & \mathbf{G}(1) & \cdots & \mathbf{G}(m) & \cdots \end{bmatrix} = \mathbf{I}. \quad (9.84)$$

Note that r is an integer value that is used to emphasize the fact that we are not specifying either the number of rows or the index of the first row of the matrix \mathcal{H} . This gives

$$\begin{bmatrix} \ddots & \vdots & \vdots & \cdots & \vdots & \cdots \\ \cdots & \mathbf{H}(r)\mathbf{G}(0) & \mathbf{H}(r)\mathbf{G}(1) & \cdots & \mathbf{H}(r)\mathbf{G}(m) & \cdots \\ \cdots & \mathbf{H}(r+1)\mathbf{G}(0) & \mathbf{H}(r+1)\mathbf{G}(1) & \cdots & \mathbf{H}(r+1)\mathbf{G}(m) & \cdots \\ \cdots & \vdots & \vdots & \ddots & \vdots & \cdots \\ \cdots & \mathbf{H}(r+m)\mathbf{G}(0) & \mathbf{H}(r+m)\mathbf{G}(1) & \cdots & \mathbf{H}(r+m)\mathbf{G}(m) & \cdots \\ \cdots & \vdots & \vdots & \cdots & \vdots & \ddots \end{bmatrix} = \mathbf{I}. \quad (9.85)$$

Since, from Equations (9.58) and (9.71), $\mathbf{H}(k)$ has M rows and $\mathbf{G}(l)$ has M columns, the above equation implies that there must be an integer r such that

$$\mathbf{H}(r+k)\mathbf{G}(l) = \delta(k-l)\mathbf{I}_M. \quad (9.86)$$

Now, expressing $\mathbf{H}(r+k)$ as a function of its rows $\tilde{\mathbf{h}}_i^T(r+k)$ (Equation (9.58)) and $\mathbf{G}(l)$ as a function of its columns $\mathbf{g}_j(l)$ (Equation (9.71)) we have that Equation (9.86) becomes

$$\begin{aligned} \mathbf{H}(r+k)\mathbf{G}(l) &= \begin{bmatrix} \tilde{\mathbf{h}}_0^T(r+k) \\ \tilde{\mathbf{h}}_1^T(r+k) \\ \vdots \\ \tilde{\mathbf{h}}_{M-1}^T(r+k) \end{bmatrix} \begin{bmatrix} \mathbf{g}_0(l) & \mathbf{g}_1(l) & \cdots & \mathbf{g}_{M-1}(l) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{h}}_0^T(r+k)\mathbf{g}_0(l) & \tilde{\mathbf{h}}_0^T(r+k)\mathbf{g}_1(l) & \cdots & \tilde{\mathbf{h}}_0^T(r+k)\mathbf{g}_{M-1}(l) \\ \tilde{\mathbf{h}}_1^T(r+k)\mathbf{g}_0(l) & \tilde{\mathbf{h}}_1^T(r+k)\mathbf{g}_1(l) & \cdots & \tilde{\mathbf{h}}_1^T(r+k)\mathbf{g}_{M-1}(l) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{h}}_{M-1}^T(r+k)\mathbf{g}_0(l) & \tilde{\mathbf{h}}_{M-1}^T(r+k)\mathbf{g}_1(l) & \cdots & \tilde{\mathbf{h}}_{M-1}^T(r+k)\mathbf{g}_{M-1}(l) \end{bmatrix} \\ &= \delta(k-l)\mathbf{I}_M \end{aligned} \quad (9.87)$$

and Equation (9.87) is equivalent to

$$\tilde{\mathbf{h}}_i^T(r+k)\mathbf{g}_j(l) = \delta(k-l)\delta(i-j). \quad (9.88)$$

This means that for $k \neq l$ or $i \neq j$ the vectors $\tilde{\mathbf{h}}_i^T(r+k)$ and $\mathbf{g}_j^*(l)$ are orthogonal (see Chapter 3, Equation (3.190) for a definition of orthogonality). If $k = l$ and $i = j$ then their inner product should be 1. Since $\tilde{\mathbf{h}}_i^T(k)$ are the rows of \mathcal{H} and $\mathbf{g}_j(l)$ are the columns of \mathcal{G} , then perfect reconstruction implies that the rows of \mathcal{H} are orthogonal to the columns of \mathcal{G}^* , except the cases where the $(r+m)$ th row of \mathcal{H} should not be orthogonal to the m th column of \mathcal{G}^* , for all m . One often refers to the rows of \mathcal{H} and columns of \mathcal{G}^* as being biorthogonal.

In the case when $\mathcal{H} = \mathcal{G}^{*\top}$, there must be an integer r such that $\tilde{\mathbf{h}}_i(r+m) = \mathbf{g}_i^*(m)$, which implies that, for $i = 0, 1, \dots, (M-1)$ and $m \in \mathbb{Z}$, there is an integer s such that

$$h_i(rM + mM - n - s) = g_i^*(n - mM), \quad (9.89)$$

where the integer s is used to emphasize the fact that we are not specifying either the number of columns or the index of the first column of the matrix \mathcal{H} . This gives

$$h_i(rM - n - s) = g_i^*(n). \quad (9.90)$$

Then Equation (9.88) becomes

$$\mathbf{g}_i^{*\top}(k)\mathbf{g}_j(l) = \delta(k-l)\delta(i-j), \quad (9.91)$$

which means that the columns of \mathcal{G} (or the rows of \mathcal{H}) are orthogonal to each other. In this case, the filter bank is said to be orthogonal, and the perfect reconstruction condition is

$$\mathcal{H}\mathcal{H}^{*T} = \mathcal{H}^{*T}\mathcal{H} = \mathbf{I} \quad (9.92)$$

or

$$\mathcal{G}\mathcal{G}^{*T} = \mathcal{G}^{*T}\mathcal{G} = \mathbf{I} \quad (9.93)$$

and the pair in Equation (9.75) becomes

$$\left. \begin{array}{l} \mathcal{U} = \mathcal{H}\mathbf{x} \\ \mathbf{x} = \mathcal{H}^{*T}\mathcal{U} \end{array} \right\}. \quad (9.94)$$

Comparing the pair of equations above with Equations (3.180) and (3.181) in Chapter 3, we see that an orthogonal perfect reconstruction filter bank can be regarded as a unitary transform that maps a signal \mathbf{x} into the transform coefficients \mathcal{U} .

The interpretation of an orthogonal filter bank as being equivalent to a unitary transform brings about an interesting insight on biorthogonal filter banks. Using the notations of Equations (3.196) and (3.197) we can say that if

$$\mathcal{U} = \mathcal{H}\mathbf{x} \quad (9.95)$$

then the k th element of \mathcal{U} can be regarded as the inner product between the complex conjugate of the k th row of \mathcal{H} and \mathbf{x} . Likewise, from Equation (3.198), we can say that if

$$\mathbf{x} = \mathcal{G}\mathcal{U} \quad (9.96)$$

then vector \mathbf{x} can be regarded as a linear combination of the columns of \mathcal{G} , where the weight of the k th column is the k th element of \mathcal{U} . Therefore, in a biorthogonal filter bank, the analysis filters project the input signal on the rows of \mathcal{H}^* . The synthesis filter bank takes these projections and uses them to weight the columns of \mathcal{G} in order to recover the input signal.

This interpretation of orthogonality and biorthogonality is illustrated in Figure 9.24 for the two-dimensional case. Figure 9.24a shows the case where \mathbf{v}_1 and \mathbf{v}_2 are unit-norm, orthogonal vectors. One can compute the coordinates (transform) of vector \mathbf{x} by projecting it on \mathbf{v}_1 and \mathbf{v}_2 . It can be recovered by the vector addition of the two projections; that is:

$$\begin{aligned} \mathbf{x} &= \|\mathbf{x}\| \cos(\theta) \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} + \|\mathbf{x}\| \sin(\theta) \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} \\ &= \left\langle \mathbf{x}, \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} \right\rangle \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} + \left\langle \mathbf{x}, \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} \right\rangle \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}. \end{aligned} \quad (9.97)$$

Figure 9.24b shows the biorthogonal cases, where one wants to express a vector \mathbf{x} as a linear combination of two vectors \mathbf{v}_1 and \mathbf{v}_2 that are not orthogonal. This can be made by constructing the parallelogram with sides parallel to \mathbf{v}_1 and \mathbf{v}_2 . But a side parallel to \mathbf{v}_1 is

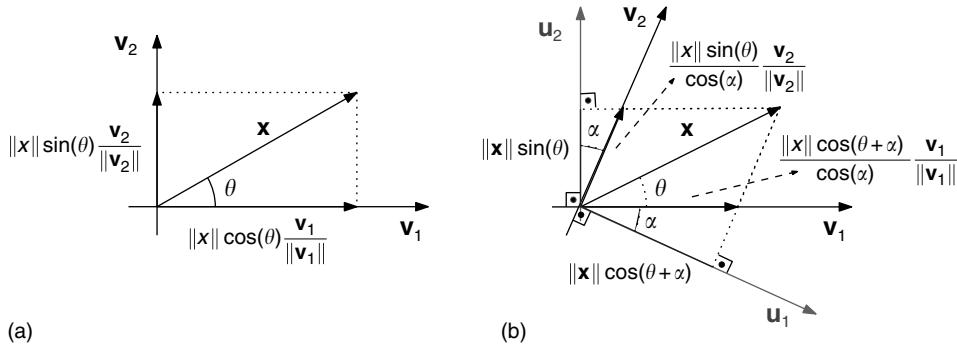


Fig. 9.24. Orthogonality and biorthogonality in two dimensions.

orthogonal to a vector \mathbf{u}_2 that is also orthogonal to \mathbf{v}_1 , and a side parallel to \mathbf{v}_2 is orthogonal to a vector \mathbf{u}_1 that is also orthogonal to \mathbf{v}_2 . Then, with the help of Figure 9.24b, we can see that, in order to find the length of the parallelogram side with the same direction as \mathbf{v}_1 , we project it on \mathbf{u}_1 and divide it by the cosine of the angle between \mathbf{v}_1 and \mathbf{u}_1 . Similarly, in order to find the length of the parallelogram side with the same direction as \mathbf{v}_2 , we project it on \mathbf{u}_2 and divide it by the cosine of the angle between \mathbf{v}_2 and \mathbf{u}_2 . That is, the analysis operation is carried out by the projection on \mathbf{u}_1 and \mathbf{u}_2 , while the synthesis by the linear combination of vectors \mathbf{v}_1 and \mathbf{v}_2 , where \mathbf{u}_1 is orthogonal to \mathbf{v}_2 and \mathbf{u}_2 is orthogonal to \mathbf{v}_1 . Mathematically, the above operations are expressed as

$$\mathbf{x} = \frac{\|\mathbf{x}\| \cos(\theta + \alpha)}{\cos(\alpha)} \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} + \frac{\|\mathbf{x}\| \sin(\theta)}{\cos(\alpha)} \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}. \quad (9.98)$$

Since

$$\left. \begin{aligned} \cos(\alpha) &= \left\langle \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \right\rangle = \left\langle \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}, \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \right\rangle \\ \|\mathbf{x}\| \cos(\alpha + \theta) &= \left\langle \mathbf{x}, \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \right\rangle \\ \|\mathbf{x}\| \sin(\theta) &= \left\langle \mathbf{x}, \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \right\rangle \end{aligned} \right\}, \quad (9.99)$$

Equation (9.98) becomes

$$\begin{aligned} \mathbf{x} &= \frac{\left\langle \mathbf{x}, \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \right\rangle}{\left\langle \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \right\rangle} \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} + \frac{\left\langle \mathbf{x}, \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \right\rangle}{\left\langle \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}, \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \right\rangle} \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} \\ &= \frac{\langle \mathbf{x}, \mathbf{u}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{u}_1 \rangle} \mathbf{v}_1 + \frac{\langle \mathbf{x}, \mathbf{u}_2 \rangle}{\langle \mathbf{v}_2, \mathbf{u}_2 \rangle} \mathbf{v}_2. \end{aligned} \quad (9.100)$$

In the above equation it can be clearly seen that the coordinates in the nonorthogonal basis composed by \mathbf{v}_1 and \mathbf{v}_2 are given by $\langle \mathbf{x}, \mathbf{u}_1 \rangle / \langle \mathbf{v}_1, \mathbf{u}_1 \rangle$ and $\langle \mathbf{x}, \mathbf{u}_2 \rangle / \langle \mathbf{v}_2, \mathbf{u}_2 \rangle$, where \mathbf{u}_1 is orthogonal to \mathbf{v}_2 and \mathbf{u}_2 is orthogonal to \mathbf{v}_1 .

9.4.3.1 Orthogonality in the z transform domain

For an orthogonal filter bank we have that, from Equation (9.90), there are integers r and s such that the analysis and synthesis filter banks satisfy

$$g_i(n) = h_i^*(rM - n - s), \quad \text{for } i = 0, 1, \dots, (M-1). \quad (9.101)$$

In the z transform domain, for $i = 0, 1, \dots, (M-1)$, this implies that

$$\begin{aligned} G_i(z) &= \sum_{n=-\infty}^{\infty} g_i(n)z^{-n} \\ &= \sum_{n=-\infty}^{\infty} h_i^*(rM - n - s)z^{-n} \\ &= \sum_{t=-\infty}^{\infty} h_i^*(t)z^{t+s-rM} \\ &= z^{-rM+s} \sum_{t=-\infty}^{\infty} h_i^*(t)z^t \\ &= z^{-rM+s} \left(\sum_{t=-\infty}^{\infty} h_i(t)(z^*)^t \right)^* \\ &= z^{-rM+s} H_i^*((z^*)^{-1}). \end{aligned} \quad (9.102)$$

Very often the filters have real coefficients. In this case, orthogonality of the filter bank demands that there must be integers r and s such that

$$G_i(z) = z^{-rM+s} H_i(z^{-1}), \quad \text{for } i = 0, 1, \dots, (M-1). \quad (9.103)$$

Returning to the general complex case, Equation (9.102) can be expressed, in matrix form, as

$$\begin{bmatrix} G_0(z) \\ G_1(z) \\ \vdots \\ G_{M-1}(z) \end{bmatrix} = z^{-rM+s} \begin{bmatrix} H_0^*((z^*)^{-1}) \\ H_1^*((z^*)^{-1}) \\ \vdots \\ H_{M-1}^*((z^*)^{-1}) \end{bmatrix}. \quad (9.104)$$

Substituting in the equation above the matrix forms of the polyphase decompositions in Equations (9.3) and (9.4), we have that

$$\mathbf{R}^T(z^M) \begin{bmatrix} z^{-(M-1)} \\ z^{-(M-2)} \\ \vdots \\ 1 \end{bmatrix} = z^{-rM+s} \left\{ \mathbf{E}((z^*)^{-M}) \begin{bmatrix} 1 \\ z^* \\ \vdots \\ (z^*)^{(M-1)} \end{bmatrix} \right\}^*$$

$$\begin{aligned}
&= z^{-rM+s} \mathbf{E}^*((z^*)^{-M}) \begin{bmatrix} 1 \\ z \\ \vdots \\ z^{M-1} \end{bmatrix} \\
&= z^{(-rM+s+M-1)} \mathbf{E}^*((z^*)^{-M}) \begin{bmatrix} z^{-(M-1)} \\ z^{-(M-2)} \\ \vdots \\ 1 \end{bmatrix}. \tag{9.105}
\end{aligned}$$

Therefore, for orthogonal filters banks, we conclude from the above equation that

$$\mathbf{R}^T(z^M) = z^{-rM+s+M-1} \mathbf{E}^*((z^*)^{-M}). \tag{9.106}$$

If we choose a matrix \mathcal{H} such that $s = 1$, Equation (9.106) becomes

$$\mathbf{R}^T(z^M) = z^{-rM+M} \mathbf{E}^*((z^*)^{-M}) \tag{9.107}$$

and thus we can write

$$\mathbf{R}^T(z) = z^{-r+1} \mathbf{E}^*((z^*)^{-1}). \tag{9.108}$$

Since the perfect reconstruction condition is $\mathbf{R}(z)\mathbf{E}(z) = z^{-\Delta}\mathbf{I}_M$, we have that

$$\mathbf{E}^{*T}((z^*)^{-1})\mathbf{E}(z) = z^{r-1-\Delta}\mathbf{I}_M. \tag{9.109}$$

By choosing \mathcal{H} such that $r = \Delta + 1$, we can see that a sufficient condition for the perfect reconstruction filter bank to be orthogonal is

$$\mathbf{E}^{*T}((z^*)^{-1})\mathbf{E}(z) = \mathbf{I}_M. \tag{9.110}$$

A matrix $\mathbf{E}(z)$ satisfying the above condition is referred to as a paraunitary matrix. Therefore, an orthogonal perfect reconstruction filter bank is also referred to as a paraunitary filter bank (Vaidyanathan, 1993). If the filters have real coefficients, Equation (9.110) simplifies to

$$\mathbf{E}^T(z^{-1})\mathbf{E}(z) = \mathbf{I}_M. \tag{9.111}$$

9.4.4 Transmultiplexers

If two identical zero-delay M -channel perfect reconstruction filter banks as in Figure 9.5 are cascaded, then we have that the signal corresponding to $u_k(m)$ in one filter bank is

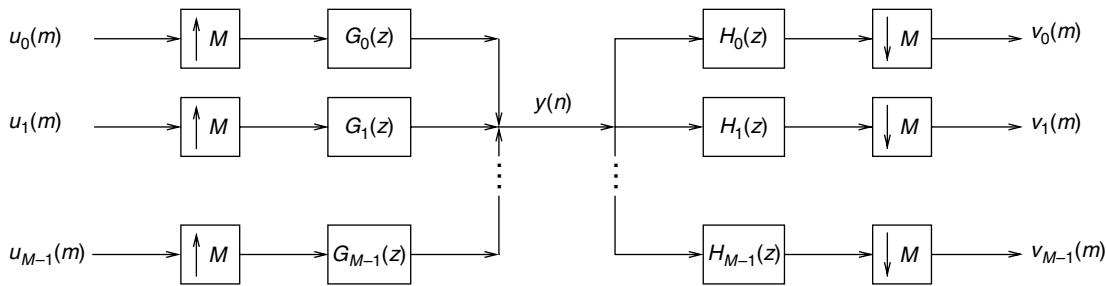


Fig. 9.25. M -band transmultiplexer.

identical to the corresponding signal in the other filter bank, for each $k = 0, 1, \dots, (M - 1)$. Therefore, with the same filters as in Figure 9.5, one can build a perfect reconstruction transmultiplexer as in Figure 9.25, which can combine the M signals $u_k(m)$ into one single signal $y(n)$, and then recover the signals $v_k(m)$ that are identical to $u_k(m)$ (Vaidyanathan, 1993). One important application of such transmultiplexers is the multiplexing of M signals so that they can be easily recovered without any cross-interference. What is very interesting in this case is that the filters do not need to be at all selective for this kind of transmultiplexer to work.² This is the basis of generalized interpretations of digital multiple access systems, such as TDMA, FDMA, and CDMA (Hettling *et al.*, 1999, Chapter 1).

9.5 General two-band perfect reconstruction filter banks

The general two-band case is shown in Figure 9.26.

Representing the filters $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$ in terms of their polyphase components, using Equations (9.1) and (9.2), we then have that

$$H_0(z) = E_{00}(z^2) + z^{-1}E_{01}(z^2) \quad (9.112)$$

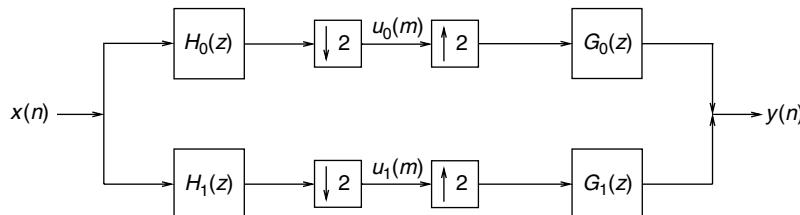


Fig. 9.26. Two-band filter bank.

² If the filter banks are not zero-delay, then this kind of transmultiplexer still works. It can be shown that the signals $v_k(m)$ are equal to delayed versions of $u_k(m)$, provided that a suitable delay is introduced between the filter banks.

$$H_1(z) = E_{10}(z^2) + z^{-1}E_{11}(z^2) \quad (9.113)$$

$$G_0(z) = z^{-1}R_{00}(z^2) + R_{10}(z^2) \quad (9.114)$$

$$G_1(z) = z^{-1}R_{01}(z^2) + R_{11}(z^2). \quad (9.115)$$

The matrices $\mathbf{E}(z)$ and $\mathbf{R}(z)$ in Figure 9.8b are then

$$\mathbf{E}(z) = \begin{bmatrix} E_{00}(z) & E_{01}(z) \\ E_{10}(z) & E_{11}(z) \end{bmatrix} \quad (9.116)$$

$$\mathbf{R}(z) = \begin{bmatrix} R_{00}(z) & R_{01}(z) \\ R_{10}(z) & R_{11}(z) \end{bmatrix}. \quad (9.117)$$

If $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}$, we have perfect reconstruction, as represented in Figures 9.10 and 9.11. In fact, from Figure 9.11, we see that the output signal $y(n)$ will be equal to $x(n)$ delayed by $(M - 1)$ samples, which for a two-band filter bank is equal to just one sample. In the general case, we have $\mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}z^{-\Delta}$, which makes the output signal of the two-band filter bank $y(n)$ equal to $x(n)$ delayed by $(2\Delta + 1)$ samples, as given by Equation (9.6). Therefore, the two-band filter bank will be equivalent to a delay of $(2\Delta + 1)$ samples if

$$\mathbf{R}(z) = z^{-\Delta}\mathbf{E}^{-1}(z). \quad (9.118)$$

From Equations (9.116) and (9.117), this implies that

$$\begin{bmatrix} R_{00}(z) & R_{01}(z) \\ R_{10}(z) & R_{11}(z) \end{bmatrix} = \frac{z^{-\Delta}}{E_{00}(z)E_{11}(z) - E_{01}(z)E_{10}(z)} \begin{bmatrix} E_{11}(z) & -E_{01}(z) \\ -E_{10}(z) & E_{00}(z) \end{bmatrix}. \quad (9.119)$$

This result is sufficient for IIR filter bank design, as long as stability constraints are taken into consideration. However, if we want the filters to be FIR, as is often the case, then the term in the denominator must be proportional to a pure delay; that is:

$$E_{00}(z)E_{11}(z) - E_{01}(z)E_{10}(z) = cz^{-l}. \quad (9.120)$$

From Equations (9.112)–(9.115), we can express the polyphase components in terms of the filters $H_k(z)$ and $G_k(z)$ as

$$\begin{aligned} E_{00}(z^2) &= \frac{H_0(z) + H_0(-z)}{2}, & E_{01}(z^2) &= \frac{H_0(z) - H_0(-z)}{2z^{-1}}, \\ E_{10}(z^2) &= \frac{H_1(z) + H_1(-z)}{2}, & E_{11}(z^2) &= \frac{H_1(z) - H_1(-z)}{2z^{-1}} \end{aligned} \quad (9.121)$$

$$\begin{aligned} R_{00}(z^2) &= \frac{G_0(z) - G_0(-z)}{2z^{-1}}, & R_{10}(z^2) &= \frac{G_0(z) + G_0(-z)}{2}, \\ R_{01}(z^2) &= \frac{G_1(z) - G_1(-z)}{2z^{-1}}, & R_{11}(z^2) &= \frac{G_1(z) + G_1(-z)}{2}. \end{aligned} \quad (9.122)$$

Substituting Equation (9.121) into Equation (9.120), we have that

$$H_0(-z)H_1(z) - H_0(z)H_1(-z) = 2cz^{-2l-1}. \quad (9.123)$$

Now, substituting Equation (9.120) into Equation (9.119), and computing the $G_k(z)$ from Equations (9.114) and (9.115), we arrive at

$$G_0(z) = -\frac{z^{2(l-\Delta)}}{c}H_1(-z) \quad (9.124)$$

$$G_1(z) = \frac{z^{2(l-\Delta)}}{c}H_0(-z). \quad (9.125)$$

The reader can verify that, whatever is the value of l , the overall filter bank transfer function consists of a delay of value $\Delta_{\text{total}} = 2\Delta + 1$.

Equations (9.123)–(9.125) suggest a possible way to design two-band perfect reconstruction filter banks. The design procedure is as follows (Vetterli and Kovačević, 1995):

- (i) Find a polynomial $P(z)$ such that $P(-z) - P(z) = 2cz^{-2l-1}$.
- (ii) Factorize $P(z)$ into two factors, $H_0(z)$ and $H_1(-z)$. Care must be taken to guarantee that $H_0(z)$ and $H_1(-z)$ are lowpass filters.
- (iii) Find $G_0(z)$ and $G_1(z)$ using Equations (9.124) and (9.125).

Some important points should be noted in this case:

- If the delay Δ , is zero, then some of the filters are certainly noncausal: for negative l , either $H_0(z)$ or $H_1(z)$ must be noncausal (see Equation (9.123)); for positive l , either $G_0(z)$ or $G_1(z)$ must be noncausal. Therefore, a causal perfect reconstruction filter bank will necessarily have nonzero delay.
- The magnitude responses $|G_0(e^{j\omega})|$ and $|H_1(e^{j\omega})|$ are mirror images of each other around $\omega = \pi/2$ (Equation (9.124)). The same happens to $|H_0(e^{j\omega})|$ and $|G_1(e^{j\omega})|$ (see Equation (9.125)).

In addition, if one wants the filter bank to be composed of linear-phase filters, it suffices to find a linear-phase product filter $P(z)$, and make linear-phase factorizations of it (one should remember that if z_0 is a zero of a linear phase FIR filter, then z_0^{-1} is also a zero). In this case we have some additional constraints on the filters. As we have seen above:

$$P(z) - P(-z) = 2cz^{-2l-1}. \quad (9.126)$$

This implies that all the odd-power terms of polynomial $P(z)$ except z^{-2l-1} are null. In addition, a linear-phase $P(z)$ should be either symmetric or antisymmetric (see Section 4.2.3), with the central term being cz^{-2l-1} . Therefore, if $P(z)$ has more than two terms, and considering that the first term of $P(z)$ is az^0 , then its last term should be $\pm az^{-4l-2}$. Then, its order is $(4l + 2)$. Taking this into consideration, we have two cases:

- (a) *Both filters have even order.* In this case, $H_0(z)$ and $H_1(-z)$ should be either Type I or Type III filters (see Section 4.2.3). However, since a Type III filter should have zeros

at $\omega = 0$ and $\omega = \pi$, and $P(z)$ should be a lowpass filter, then both $H_0(z)$ and $H_1(-z)$ are Type I filters, and so is $H_1(z)$ (see Exercise 4.3). In addition, since the sum of their orders is $(4l + 2)$, the order of one is a multiple of 4 and the order of the other is not (that is, their orders differ by an odd multiple of 2).

- (b) *Both filters have odd order.* In this case, $H_0(z)$ and $H_1(-z)$ should be either Type II or Type IV filters. However, since a Type IV filter should have a zero at $\omega = 0$, and $P(z)$ should be a lowpass filter, then both $H_0(z)$ and $H_1(-z)$ should be Type II filters, and $H_1(z)$ is Type IV (see Exercise 4.3). In addition, if the orders of $H_0(z)$ and $H_1(z)$ are $(2k_0 + 1)$ and $(2k_1 + 1)$ respectively, then, since the sum of their orders is equal to $(4l + 2)$, we have that

$$(2k_0 + 1) + (2k_1 + 1) = 4l + 2 \Rightarrow k_0 = 2l - k_1 \quad (9.127)$$

and then the difference of their orders is

$$\begin{aligned} (2k_0 + 1) - (2k_1 + 1) &= 4l - 2k_1 + 1 - 2k_1 - 1 \\ &= 4l - 2k_1 - 2k_1 \\ &= 4(l - k_1), \end{aligned} \quad (9.128)$$

which is a multiple of 4.

In the case that $P(z)$ has only two terms, since it is linear phase, we have that Equation (9.126) implies that

$$P(z) = cz^{-2l-1} \pm cz^{-2r} = cz^{-2r}(z^{-2l+2r-1} \pm 1). \quad (9.129)$$

Thus, $P(z)$ has odd order equal to $|2l - 2r + 1|$ and all its zeros on the unity circle. We then have two cases:

- (a) $H_0(z)$ has even order and $H_1(z)$ has odd order. Then, $P(z)$ being a lowpass filter implies that $H_0(z)$ is Type I and $H_1(-z)$ is Type II, and therefore $H_1(z)$ is Type IV (see Exercise 4.3).
- (b) $H_0(z)$ has odd order and $H_1(z)$ has even order. Then, in this case $H_0(z)$ is Type II and $H_1(-z)$ is Type I, and therefore $H_1(z)$ is also Type I (see Exercise 4.3).

This last case is of little practical interest, since the resulting $H_0(z)$ and $H_1(z)$ tend to have poor frequency selectivity.

Example 9.7. A product filter $P(z)$ satisfying $P(z) - P(-z) = 2z^{-2l-1}$ is

$$P(z) = \frac{1}{16}(-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6}) = \frac{1}{16}(1 + z^{-1})^4(-1 + 4z^{-1} - z^{-2}). \quad (9.130)$$

Find two possible factorizations of $P(z)$ and plot the magnitude responses of their corresponding analysis filters.

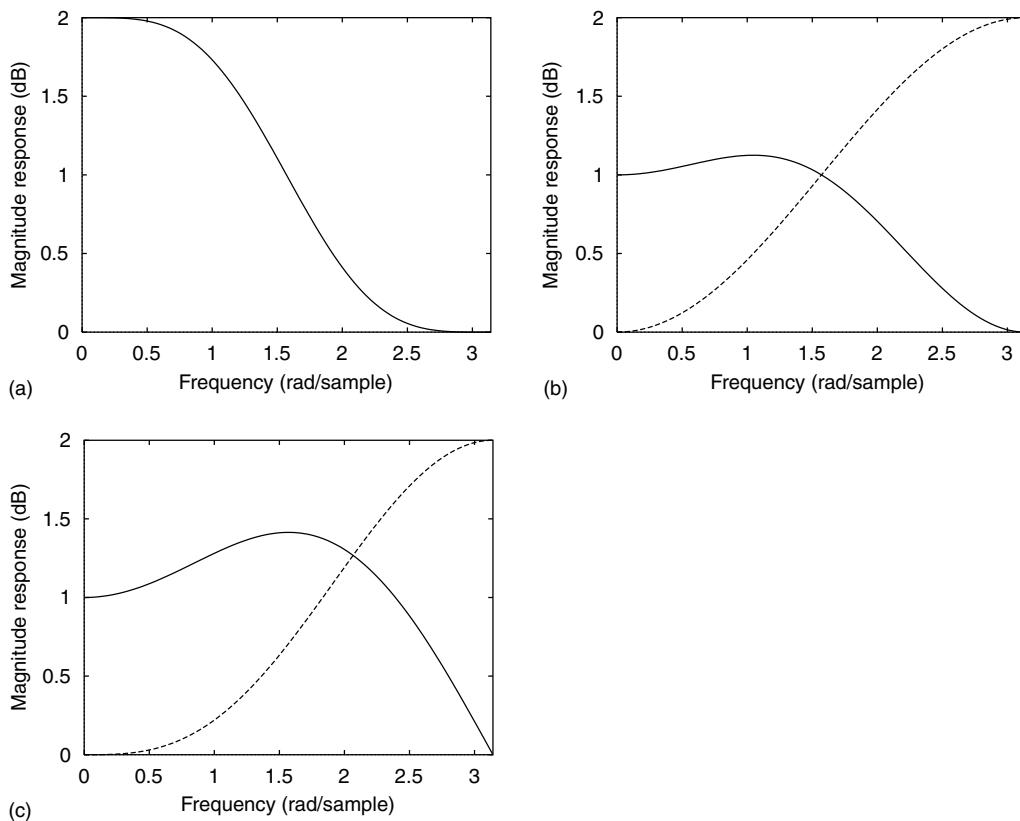


Fig. 9.27. Magnitude responses: (a) $P(z)$ from Equation (9.130); (b) $H_0(z)$ (solid line) and $H_1(z)$ (dashed line) from the factorizations in Equations (9.131) and (9.132); (c) $H_0(z)$ (solid line) and $H_1(z)$ (dashed line) from the factorizations in Equations (9.135) and (9.136).

Solution

We can see from the magnitude response in Figure 9.27a that $P(z)$ is a lowpass filter.

One possible factorization of $P(z)$ results in the following filter bank, which is the same as the one in Example 9.2. This filter bank is the popular symmetric short-kernel filter bank (Le Gall & Tabatabai, 1988; Vetterli and Kovačević, 1995):

$$H_0(z) = \frac{1}{8}(-1 + 2z^{-1} + 6z^{-2} + 2z^{-3} - z^{-4}) \quad (9.131)$$

$$H_1(z) = \frac{1}{2}(1 - 2z^{-1} + z^{-2}) \quad (9.132)$$

$$G_0(z) = \frac{1}{2}(1 + 2z^{-1} + z^{-2}) \quad (9.133)$$

$$G_1(z) = \frac{1}{8}(1 + 2z^{-1} - 6z^{-2} + 2z^{-3} + z^{-4}). \quad (9.134)$$

The magnitude responses of the analysis filters are depicted in Figure 9.27b.

Another possible factorization is as follows:

$$H_0(z) = \frac{1}{4}(-1 + 3z^{-1} + 3z^{-2} - z^{-3}) \quad (9.135)$$

$$H_1(z) = \frac{1}{4}(1 - 3z^{-1} + 3z^{-2} - z^{-3}) \quad (9.136)$$

$$G_0(z) = \frac{1}{4}(1 + 3z^{-1} + 3z^{-2} + z^{-3}) \quad (9.137)$$

$$G_1(z) = \frac{1}{4}(1 + 3z^{-1} - 3z^{-2} - z^{-3}). \quad (9.138)$$

The corresponding magnitude responses of the analysis filters are depicted in Figure 9.27c.

△

In the two following sections, we examine two particular cases of two-band filter bank design which have become popular in the technical literature.

9.6 QMF filter banks

One of the earliest proposed approaches for the design of two-band FIR filter banks is the so-called quadrature mirror filter (QMF) bank (Crosier *et al.*, 1976), where the analysis highpass filter is designed to alternate the signs of the impulse-response samples of the lowpass filter; that is:

$$H_1(z) = H_0(-z). \quad (9.139)$$

Please note that it is assumed the filters have real coefficients. For this choice of the analysis filter bank, the magnitude response of the highpass filter, $|H_1(e^{j\omega})|$, is the mirror image of the lowpass filter magnitude response, $|H_0(e^{j\omega})|$, with respect to the quadrature frequency $\pi/2$. Hence the QMF nomenclature.

QMF filter banks are designed to structurally avoid aliasing while keeping the constraint in Equation (9.139). Filter $H_0(z)$ is then designed so that the filter bank is close enough to achieving perfect reconstruction. Its design equations can be derived by starting from the modulation matrix representation in Section 9.4.1, Equation (9.42), for $M = 2$ bands:

$$Y(z) = \frac{1}{2} \begin{bmatrix} X(z) & X(-z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix} \begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix}. \quad (9.140)$$

The aliasing effect is represented by the terms containing $X(-z)$. A possible solution to avoid aliasing is to choose the synthesis filters such that

$$G_0(z) = H_1(-z) \quad (9.141)$$

$$G_1(z) = -H_0(-z). \quad (9.142)$$

This choice keeps the filters $G_0(z)$ and $G_1(z)$ as lowpass and highpass filters, respectively, as desired. Also, the aliasing is now canceled by the synthesis filters, instead of being totally avoided by the analysis filters, relieving the specifications of the latter filters (see Section 9.2.3).

The overall transfer function of the filter bank, after the aliasing component is eliminated, is given by

$$H(z) = \frac{1}{2}(H_0(z)G_0(z) + H_1(z)G_1(z)) = \frac{1}{2}(H_0(z)H_1(-z) - H_1(z)H_0(-z)), \quad (9.143)$$

where in the second equality we employed the aliasing elimination constraint of Equations (9.141) and (9.142).

In the original QMF design, the aliasing elimination condition is combined with the alternating-sign choice for the highpass filter of Equation (9.139). In such a case, the overall transfer function is given by

$$H(z) = \frac{1}{2}(H_0^2(z) - H_0^2(-z)). \quad (9.144)$$

The expression above can be rewritten in a more convenient form by employing the polyphase decomposition of the lowpass filter $H_0(z) = E_{00}(z^2) + z^{-1}E_{01}(z^2)$ as follows:

$$\begin{aligned} H(z) &= \frac{1}{2}(H_0(z) + H_0(-z))(H_0(z) - H_0(-z)) \\ &= 2z^{-1}E_{00}(z^2)E_{01}(z^2). \end{aligned} \quad (9.145)$$

As pointed out above, the QMF design approach of two-band filter banks consists of designing the lowpass filter $H_0(z)$. The above equation also shows that perfect reconstruction is achievable only if the polyphase components of the lowpass filter (that is, $E_{00}(z)$ and $E_{01}(z)$) are simple delays. This constraint limits the selectivity of the generated filters.

Therefore, for QMF design we usually adopt an approximate solution by choosing $H_0(z)$ to be an N th-order FIR linear-phase lowpass filter. This eliminates any phase distortion of the overall transfer function $H(z)$, which, in this case, will also have linear phase. For either a Type I or Type II N th-order filter (see Section 4.2.3, Equation (4.16)), the filter bank transfer function in Equation (9.144) can then be written as

$$\begin{aligned} H(e^{j\omega}) &= \frac{1}{2} \left(B_0^2(\omega) e^{-j\omega N} - B_0^2(\omega - \pi) e^{-j(\omega - \pi)N} \right) \\ &= \frac{e^{-j\omega N}}{2} \left(|H_0(e^{j\omega})|^2 - e^{j\pi N} |H_0(e^{j(\omega - \pi)})|^2 \right) \\ &= \frac{e^{-j\omega N}}{2} \left(|H_0(e^{j\omega})|^2 - (-1)^N |H_0(e^{j(\omega - \pi)})|^2 \right). \end{aligned} \quad (9.146)$$

From the above equation, we see that, for N even, $H(e^{j\pi/2}) = 0$, which is undesirable. Therefore, for QMF filters the filter order must be odd. In this case, Equation (9.146)

becomes

$$\begin{aligned} H(e^{j\omega}) &= \frac{e^{-j\omega N}}{2} \left(|H_0(e^{j\omega})|^2 + |H_0(e^{j(\omega-\pi)})|^2 \right) \\ &= \frac{e^{-j\omega N}}{2} \left(|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 \right). \end{aligned} \quad (9.147)$$

The design procedure goes on to minimize the following objective function using an optimization algorithm:

$$\xi = \delta \int_{\omega_s}^{\pi} |H_0(e^{j\omega})|^2 d\omega + (1 - \delta) \int_0^{\pi} \left| H(e^{j\omega}) - \frac{e^{-j\omega N}}{2} \right|^2 d\omega, \quad (9.148)$$

where ω_s is the stopband edge, usually chosen slightly above 0.5π . The parameter $0 < \delta < 1$ provides a trade-off between the stopband attenuation of the lowpass filter and the amplitude distortion of the filter bank. Although this objective function has local minima, a good starting point for the coefficients of the lowpass filter and an adequate nonlinear optimization algorithm lead to good results; that is, filter banks with low-amplitude distortions and good selectivity of the filters. Usually, a simple window-based design provides a good starting point for the lowpass filter. Overall, the simplicity of the QMF design makes it widely used in practice. Figure 9.28a depicts the magnitude responses of the analysis filters of a QMF design of order $N = 15$ and Figure 9.28b depicts the amplitude response of the complete filter bank. Johnston (1980) was among the first to provide QMF coefficients for several designs. Owing to his pioneering work, such QMF filters are usually termed Johnston filter banks.

The name QMF filter banks is also used to denote M -band maximally decimated filter banks. For M -band QMF filter banks there are two design approaches that are widely used, namely the perfect reconstruction QMF filter banks and the pseudo-QMF filter banks. The

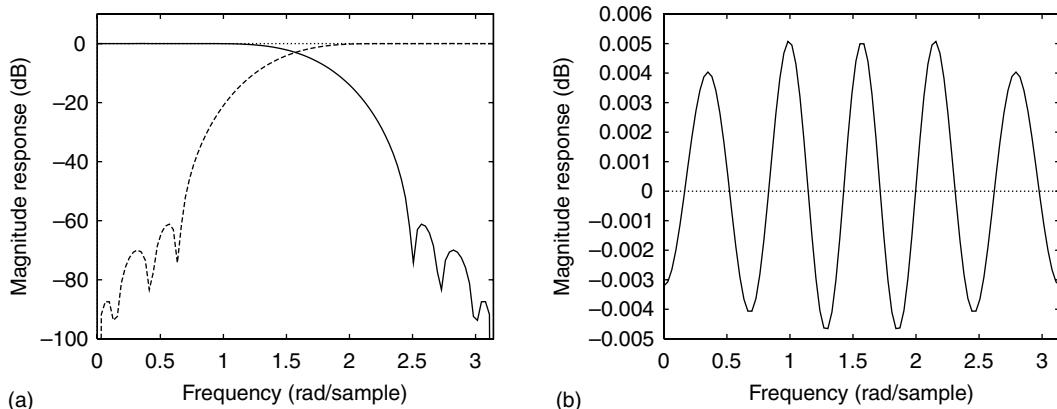


Fig. 9.28. QMF design of order $N = 15$: (a) magnitude responses of the analysis filters (solid line: $H_0(z)$; dashed line: $H_1(z)$); (b) overall magnitude response.

perfect reconstruction QMF designs require the use of sophisticated nonlinear optimization programs because the objective function is a very nonlinear function of the filter parameters (Vaidyanathan, 1993). In particular, for a large number of sub-bands, the number of parameters becomes excessively large. Meanwhile, the pseudo-QMF designs consist of designing a prototype filter, with the sub-filters of the analysis bank being obtained by the modulation of the prototype. As a consequence, the pseudo-QMF filter has a very efficient design procedure. The pseudo-QMF filter banks are also known as cosine-modulated filter banks, since they are designed by applying cosine modulation to a lowpass prototype filter. There are cosine-modulated filter banks which achieve perfect reconstruction (Koilpillai & Vaidyanathan, 1992; Nguyen & Koilpillai, 1996), and the procedure for their design is given in Section 9.9.

9.7 CQF filter banks

QMF design, where the highpass filter is determined from the lowpass prototype by alternating the signs of its impulse response, is quite simple. However, the possibility of getting perfect reconstruction is lost except in a few trivial cases. In Smith & Barnwell (1986), it was disclosed that, by time reversing the impulse response and alternating the signs of the lowpass filter, one can design perfect reconstruction filter banks with more selective sub-filters. The resulting filters became known as the conjugate quadrature filter (CQF) banks.

By referring to Section 9.4.3, the CQF filter bank is an example of an orthogonal filter bank. Its design can then be deduced from the orthogonality condition in Equation (9.103), which, for the two-band case, is

$$G_i(z) = z^{-2r+s} H_i(z^{-1}), \quad \text{for } i = 0, 1. \quad (9.149)$$

The perfect reconstruction conditions for the two-band case are given by Equations (9.124) and (9.125):

$$G_0(z) = -\frac{z^{2(l-\Delta)}}{c} H_1(-z) \quad (9.150)$$

$$G_1(z) = \frac{z^{2(l-\Delta)}}{c} H_0(-z), \quad (9.151)$$

where $2\Delta + 1$ is the total delay of the filter bank and l can be any integer. Using Equation (9.149) these conditions become

$$z^{-2r+s} H_0(z^{-1}) = -\frac{z^{2(l-\Delta)}}{c} H_1(-z) \quad (9.152)$$

$$z^{-2r+s} H_1(z^{-1}) = \frac{z^{2(l-\Delta)}}{c} H_0(-z). \quad (9.153)$$

From Equation (9.152) we have that

$$(-z)^{2r-s}H_0(-z) = -\frac{(-z)^{-2(l-\Delta)}}{c}H_1(z^{-1}) \quad (9.154)$$

and then

$$H_1(z^{-1}) = -c(-z)^{[2(l-\Delta)+2r-s]}H_0(-z). \quad (9.155)$$

By replacing $H_1(z^{-1})$ from Equation (9.155) in Equation (9.153) we get

$$z^{-2r+s} \left[-c(-z)^{[2(l-\Delta)+2r-s]}H_0(-z) \right] = \frac{z^{2(l-\Delta)}}{c}H_0(-z), \quad (9.156)$$

or equivalently

$$-c^2(-1)^{[2(l-\Delta)+2r-s]} \left[z^{2(l-\Delta)}H_0(-z) \right] = z^{2(l-\Delta)}H_0(-z) \quad (9.157)$$

and then

$$c^2 = -(-1)^{[2(l-\Delta)+2r-s]} = -(-1)^s. \quad (9.158)$$

Therefore, in order for the two-band perfect reconstruction filter bank to be orthogonal, we must have s odd and $c = \pm 1$. Then, since for s odd, $[2(l - \Delta) + 2r - s]$ is also odd, Equation (9.155) implies that

$$H_1(z) = cz^{-[2(l-\Delta)+2r-s]}H_0(-z^{-1}). \quad (9.159)$$

In the CQF design one usually makes the following design choices:

- $c = -1$;
- the order of the lowpass filter is the odd number

$$N = 2(l - \Delta) + 2r - s. \quad (9.160)$$

Note that, as seen in Section 9.4.3, Equations (9.103)–(9.110), in order for orthogonality to imply the paraunitarity of matrix $\mathbf{E}(z)$, we must have $s = 1$. On the other hand, as can be deduced from Equation (9.158), in the CQF bank it is sufficient that s be odd. However, from Equation (9.160) we see that, for any odd value of N , we can find a value of r such that $s = 1$. Therefore, we have that for CQF filter banks the polyphase matrix $\mathbf{E}(z)$ is always paraunitary. This is an important property for CQF filter banks.

With these design choices, we have that the analysis highpass filter is given by

$$H_1(z) = -z^{-N}H_0(-z^{-1}). \quad (9.161)$$

Since perfect reconstruction also requires that Equation (9.123) be valid, we must have

$$\begin{aligned} -2z^{-2l-1} &= H_0(-z)H_1(z) - H_0(z)H_1(-z) \\ &= H_0(-z)(-z^{-N})H_0(-z^{-1}) - H_0(z)[-(-z)^{-N}]H_0(z^{-1}) \\ &= -z^{-N} \left[H_0(-z)H_0(-z^{-1}) + H_0(z)H_0(z^{-1}) \right]. \end{aligned} \quad (9.162)$$

By defining

$$P(z) = H_0(z)H_0(z^{-1}) \quad (9.163)$$

the perfect reconstruction condition in Equation (9.162) becomes

$$P(z) + P(-z) = 2z^{N-2l-1}. \quad (9.164)$$

From the definition of $P(z)$ in Equation (9.163), we see that $P(z) = P(z^{-1})$. Therefore, from Equation (9.164) we have

$$\begin{aligned} 2z^{N-2l-1} &= P(z) + P(-z) \\ &= P(z^{-1}) + P(-z^{-1}) \\ &= 2z^{-N+2l+1}, \end{aligned} \quad (9.165)$$

which implies that we should choose l such that $N = 2l + 1$. This condition makes the perfect reconstruction condition to be

$$P(z) + P(-z) = 2. \quad (9.166)$$

In addition, the synthesis filters from Equations (9.150) and (9.151) become

$$G_0(z) = z^{[2(l-\Delta)-N]}H_0(z^{-1}) = z^{-(2\Delta+1)}H_0(z^{-1}) \quad (9.167)$$

$$G_1(z) = -z^{2(l-\Delta)}H_0(-z) = -z^{(N-2\Delta-1)}H_0(-z). \quad (9.168)$$

If $p(n)$ is the inverse z transform of $P(z)$, then Equation (9.166) is equivalent to

$$p(n)[1 + (-1)^n] = 2\delta(n). \quad (9.169)$$

From Equations (9.161), (9.163), (9.166), (9.167), and (9.168) a design procedure for CQF filter banks consists of the following steps:

- (i) Noting that $p(n) = 0$ for n even, except for $n = 0$, we start by designing a half-band filter, namely a filter such that $(\omega_p + \omega_r)/2 = \pi/2$, with order $2N$ and the same ripple δ_{hb} in the passband and stopband, see Figure 5.10. The resulting half-band filter will have null samples on its impulse response for every even n except for $n = 0$. Such a

filter can be designed, for example, using the standard Chebyshev approach for FIR filters (see Section 5.6.2) as follows:

1. Design a zero-delay Hilbert transformer $H_h(z)$ with order $2N$. Since its order is even it must be a Type III FIR filter (see Section 4.2.3). Its ripple must be smaller than $\pm\delta_{hb}/2$ in its passband and its transition bandwidth around $\omega = 0$ and $\omega = \pi$ should be ω_r .
2. From the Hilbert transformer, create the filter $P(z)$ such that (see Exercise 9.14)

$$P(z) = 1 + \frac{\delta_{hb}}{2} - jH_h(-jz). \quad (9.170)$$

Note that the term $-jH_h(-jz)$ corresponds to a zero-delay filter with gain equal to $(1 \pm \delta_{hb}/2)$, for $0 \leq |\omega| \leq \pi/2$, and gain equal to $-(1 \pm \delta_{hb}/2)$, for $\pi/2 \leq |\omega| \leq \pi$. By summing $(1 + \delta_{hb}/2)$ to its frequency response we obtain a zero-delay lowpass filter with gain 2 and ripple δ_{hb} in the passband and a nonnegative stopband gain ranging from zero to δ_{hb} . This is necessary since, from Equation (9.163), $P(e^{j\omega})$ must be nonnegative for all ω , as it corresponds to the modulus squared of $H_0(e^{j\omega})$. As a rule of thumb, to simplify the design procedure, the stopband attenuation in decibels of the half-band filter should be at least twice the desired stopband attenuation plus 6 dB (Vaidyanathan, 1993).

- (ii) The usual approach is to decompose $P(z) = H_0(z)H_0(z^{-1})$ such that $H_0(z)$ has either near linear phase or has minimum phase. In order to obtain near linear phase, one can select the zeros of $H_0(z)$ to be alternately from inside and outside the unit circle as the frequency is increased. Minimum phase is obtained when all zeros are either inside or on the unit circle of the z plane.

If we wanted the filter bank to be composed of linear-phase filters, we would have to find a linear-phase product filter $P(z)$ and make linear-phase factorizations of it, as seen in Section 9.5. However, the only linear-phase two-band filter banks that satisfy Equation (9.169) are those composed of trivial linear-phase filters such as the ones described by Equations (9.11)–(9.14). Therefore, there is no point in looking for linear-phase factorizations of $P(z)$. This is why in step (ii) above the usual approach is to look for either minimum phase or near linear-phase factorizations of $P(z)$.

Example 9.8. Design a perfect reconstruction filter bank for which the filter playing the role of the lowpass analysis filter is given by:

$$H(z) = -z^{-3} + z^{-2} + z^{-1} + 1. \quad (9.171)$$

Solution

In this example there is no particular constraint imposed on the highpass analysis filters, so that we can look for a CQF design.

The CQF design is only possible if Equations (9.163) and (9.166) are valid. Since

$$\begin{aligned} H(z)H(z^{-1}) &= (-z^{-3} + z^{-2} + z^{-1} + 1)(-z^3 + z^2 + z + 1) \\ &= -z^3 + z + 4 + z^{-1} - z^{-3} \end{aligned} \quad (9.172)$$

we have that

$$H(z)H(z^{-1}) + H(-z)H(-z^{-1}) = 8. \quad (9.173)$$

Therefore, if we choose

$$H_0(z) = \frac{1}{2}H(z) \quad (9.174)$$

we have that $H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 2$, and a CQF design with $H_0(z)$ equal to its lowpass analysis filter is possible. For this choice of $H_0(z)$, the order of the CQF bank is $N = 3$. Therefore, supposing an overall delay of $2\Delta + 1 = 3$ samples, from Equations (9.161), (9.167), and (9.168) the filter bank becomes

$$\left. \begin{array}{ll} H_0(z) = \frac{1}{2}H(z) & = \frac{1}{2}(-z^{-3} + z^{-2} + z^{-1} + 1) \\ H_1(z) = -z^{-3}H_0(-z^{-1}) & = \frac{1}{2}(-z^{-3} + z^{-2} - z^{-1} - 1) \\ G_0(z) = z^{-3}H_0(z^{-1}) & = \frac{1}{2}(z^{-3} + z^{-2} + z^{-1} - 1) \\ G_1(z) = -H_0(-z) & = \frac{1}{2}(-z^{-3} - z^{-2} + z^{-1} - 1) \end{array} \right\}. \quad (9.175)$$

An alternative way of designing a CQF filter bank is using the fact that the analysis polyphase matrix is paraunitary. Using $H_0(z)$ and $H_1(z)$ from Equation (9.175), from Equation (9.121) we see that

$$\left. \begin{array}{l} E_{00}(z^2) = \frac{1}{2}(z^{-2} + 1) \\ E_{01}(z^2) = \frac{1}{2}(-z^{-2} + 1) \\ E_{10}(z^2) = \frac{1}{2}(z^{-2} - 1) \\ E_{11}(z^2) = \frac{1}{2}(-z^{-2} - 1) \end{array} \right\} \quad (9.176)$$

and, therefore, the analysis polyphase matrix $\mathbf{E}(z)$ is

$$\mathbf{E}(z) = \frac{1}{2} \begin{bmatrix} 1 + z^{-1} & 1 - z^{-1} \\ -1 + z^{-1} & -1 - z^{-1} \end{bmatrix}, \quad (9.177)$$

such that

$$\mathbf{E}^{*\top}((z^*)^{-1}) = \frac{1}{2} \begin{bmatrix} 1 + z & -1 + z \\ 1 - z & -1 - z \end{bmatrix}. \quad (9.178)$$

Therefore:

$$\begin{aligned}\mathbf{E}^{*\top}((z^*)^{-1})\mathbf{E}(z) &= \frac{1}{4} \begin{bmatrix} 1+z & -1+z \\ 1-z & -1-z \end{bmatrix} \begin{bmatrix} 1+z^{-1} & 1-z^{-1} \\ -1+z^{-1} & -1-z^{-1} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\end{aligned}\quad (9.179)$$

and the matrix $\mathbf{E}(z)$ is paraunitary, which implies that it is possible to have a CQF filter with the given analysis filters. From Equation (9.118):

$$\mathbf{R}(z) = z^{-\Delta}\mathbf{E}^{-1}(z), \quad (9.180)$$

where $\Delta = 1$, since the overall delay ($2\Delta + 1$) in this example was chosen to be equal to the filter order $N = 3$.

Thus, the paraunitarity of $\mathbf{E}(z)$ implies that the above equation is

$$\mathbf{R}(z) = z^{-1}\mathbf{E}^{*\top}((z^*)^{-1}) = \frac{1}{2} \begin{bmatrix} z^{-1}+1 & -z^{-1}+1 \\ z^{-1}-1 & -z^{-1}-1 \end{bmatrix}, \quad (9.181)$$

which implies that the synthesis filters are

$$\begin{aligned}\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} &= \mathbf{R}^T(z^2) \begin{bmatrix} z^{-1} \\ 1 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} z^{-2}+1 & z^{-2}-1 \\ -z^{-2}+1 & -z^{-2}-1 \end{bmatrix} \begin{bmatrix} z^{-1} \\ 1 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} z^{-3}+z^{-2}+z^{-1}-1 \\ -z^{-3}-z^{-2}+z^{-1}-1 \end{bmatrix},\end{aligned}\quad (9.182)$$

which are the same ones as in Equation (9.175). \triangle

9.8 Block transforms

Perhaps the most popular example of M -band perfect reconstruction filter banks is given by the block transforms (see discrete transforms in Chapter 3). For instance, the DCT does essentially the same job as a filter bank: it divides a signal into several frequency components. The main difference is that, given a length- N signal, the DCT divides it into N frequency channels, whereas a filter bank divides it into M channels, with $M < N$. However, in many applications one wants to divide a length- N signal into J blocks, each having length M , and separately apply the transform to each one. This is done, for example, in the MPEG2 standard employed in digital video transmission (Le Gall, 1992; Whitaker, 1999). In it, instead of transmitting the individual pixels of a video sequence, each frame is first divided into 8×8 blocks. Then a two-dimensional DCT is applied to each block,

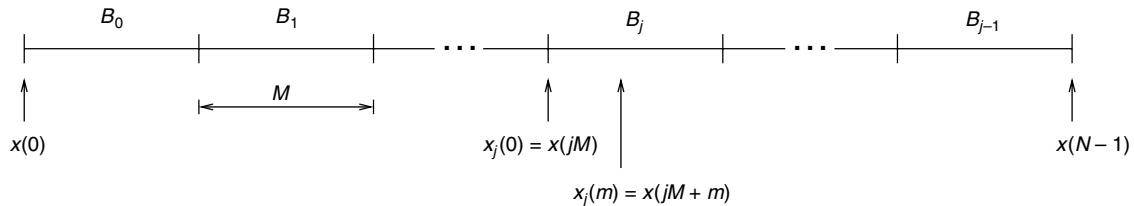


Fig. 9.29. Division of a length- N signal into J nonoverlapping blocks of length M .

and the DCT coefficients are transmitted instead. In this section, we show that these block transforms are equivalent to M -band filter banks.

Consider a signal $x(n)$, for $n = 0, 1, \dots, (N - 1)$, divided into J blocks B_j , with $j = 0, 1, \dots, (J - 1)$, each of size M . Block B_j then consists of the signal $x_j(m)$, given by

$$x_j(m) = x(jM + m) \quad (9.183)$$

for $j = 0, 1, \dots, (J - 1)$ and $m = 0, 1, \dots, (M - 1)$. This is depicted in Figure 9.29.

Suppose that $y_j(k)$, for $k = 0, 1, \dots, (M - 1)$ and $j = 0, 1, \dots, (J - 1)$, is the transform of a block $x_j(m)$, where the direct and inverse transforms are described by

$$y_j(k) = \sum_{m=0}^{M-1} c_k(m)x_j(m) \quad (9.184)$$

and

$$x_j(m) = \sum_{k=0}^{M-1} c_k^*(m)y_j(k) \quad (9.185)$$

respectively, where $c_k^*(m)$, for $m = 0, 1, \dots, (M - 1)$, is the k th basis function of the transform, or, alternatively, the k th row of the transform matrix. We can then regroup the transforms of the blocks sequentially according to k ; that is, group all the $y_j(0)$, all the $y_j(1)$, and so on. This is equivalent to creating M signals $u_k(j) = y_j(k)$, for $j = 0, 1, \dots, (J - 1)$ and $k = 0, 1, \dots, (M - 1)$. Since $x_j(m) = x(Mj + m)$, from Equations (9.184) and (9.185), we have that

$$u_k(j) = \sum_{m=0}^{M-1} c_k(m)x(Mj + m) \quad (9.186)$$

$$x(Mj + m) = \sum_{k=0}^{M-1} c_k^*(m)u_k(j). \quad (9.187)$$

From Equation (9.186), we can interpret $u_k(j)$ as the convolution, sampled at the points Mj , of $x(n)$ with $c_k(-n)$. This is the same as filtering $x(n)$ with a filter of impulse response $c_k(-n)$ and decimating its output by M .

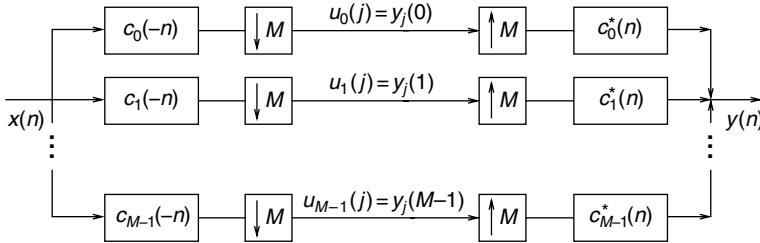


Fig. 9.30.

Interpretation of direct and inverse block transforms as a perfect reconstruction filter bank.

Likewise, if we define $u_k^{(i)}(j)$ as the signal $u_k(j)$ interpolated by M , then we have that $u_k^{(i)}(Mj + n) = 0$, for $n = 1, 2, \dots, (M - 1)$ (see Equation (8.17)). This implies that

$$c_k^*(m)u_k(j) = \sum_{n=0}^{M-1} c_k^*(m-n)u_k^{(i)}(Mj+n). \quad (9.188)$$

Such an expression can be interpreted as $c_k^*(m)u_k(j)$ being the result of interpolating $u_k(j)$ by M , and filtering it with a filter of impulse response $c_k^*(m)$.

Substituting Equation (9.188) in Equation (9.187), we arrive at the following expression:

$$x(Mj + m) = \sum_{k=0}^{M-1} \sum_{n=0}^{M-1} c_k^*(m-n)u_k^{(i)}(Mj+n). \quad (9.189)$$

Therefore, from Equations (9.186) and (9.189), the direct and inverse block-transform operations can be interpreted as in Figure 9.30. Comparing it with Figure 9.5, we can see that a block transform is equivalent to a perfect reconstruction filter bank having the impulse responses of band k analysis and synthesis filters equal to $c_k(-m)$ and $c_k^*(m)$ respectively.

Example 9.9. As seen in Chapter 3, the coefficients of the length- M DCT are given by

$$c_k(m) = \alpha(k) \cos\left[\frac{\pi(2m+1)k}{2M}\right], \quad \text{for } m = 0, 1, \dots, (M-1), \quad (9.190)$$

where $\alpha(0) = \sqrt{1/M}$ and $\alpha(k) = \sqrt{2/M}$, for $k = 1, 2, \dots, (M-1)$. Plot the impulse response and the magnitude response of the analysis filters corresponding to the length-10 DCT. In addition, determine whether the filter bank has linear phase or not.

Solution

As can be seen from Figure 9.30, the impulse responses $h_k(n)$ of the analysis filters for the length-10 DCT filter bank are given by

$$h_k(n) = c_k(-n) = \alpha(k) \cos\left[\frac{\pi(1-2n)k}{20}\right], \quad \text{for } k, n = 0, 1, \dots, 9. \quad (9.191)$$

These impulse responses are depicted in Figure 9.31 for each band k , and the corresponding magnitude responses are depicted in Figure 9.32. Also, from Equation (9.191), we can see

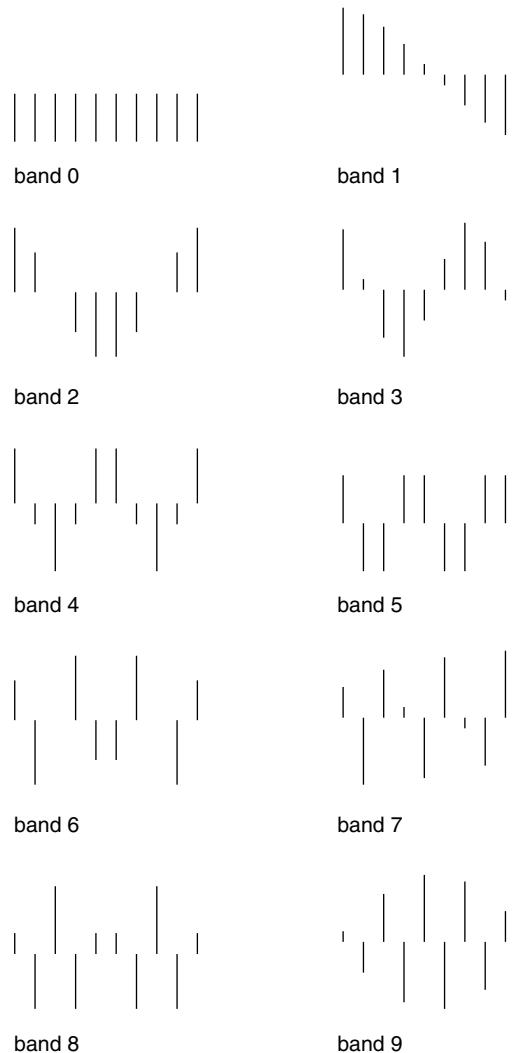


Fig. 9.31. Impulse responses of the filters of the 10-band DCT.

that $c_k(m) = (-1)^k c_k(9 - m)$ and, therefore, the filter bank has linear phase. This also implies that the magnitude responses of the analysis and synthesis filters are the same. \triangle

An alternative way to describe the block transforms as filter banks is to redraw them as depicted in Figure 9.33. Note that in this case the M -sample block slides along the signal. However, since the transform is computed for nonoverlapping blocks, then the outputs of the analysis bank of Figure 9.33 should be decimated by M . This yields the representation of the direct and inverse transforms as the causal filter bank in Figure 9.34 (note that there is a delay of $(M - 1)$ samples in this causal implementation of the transform).

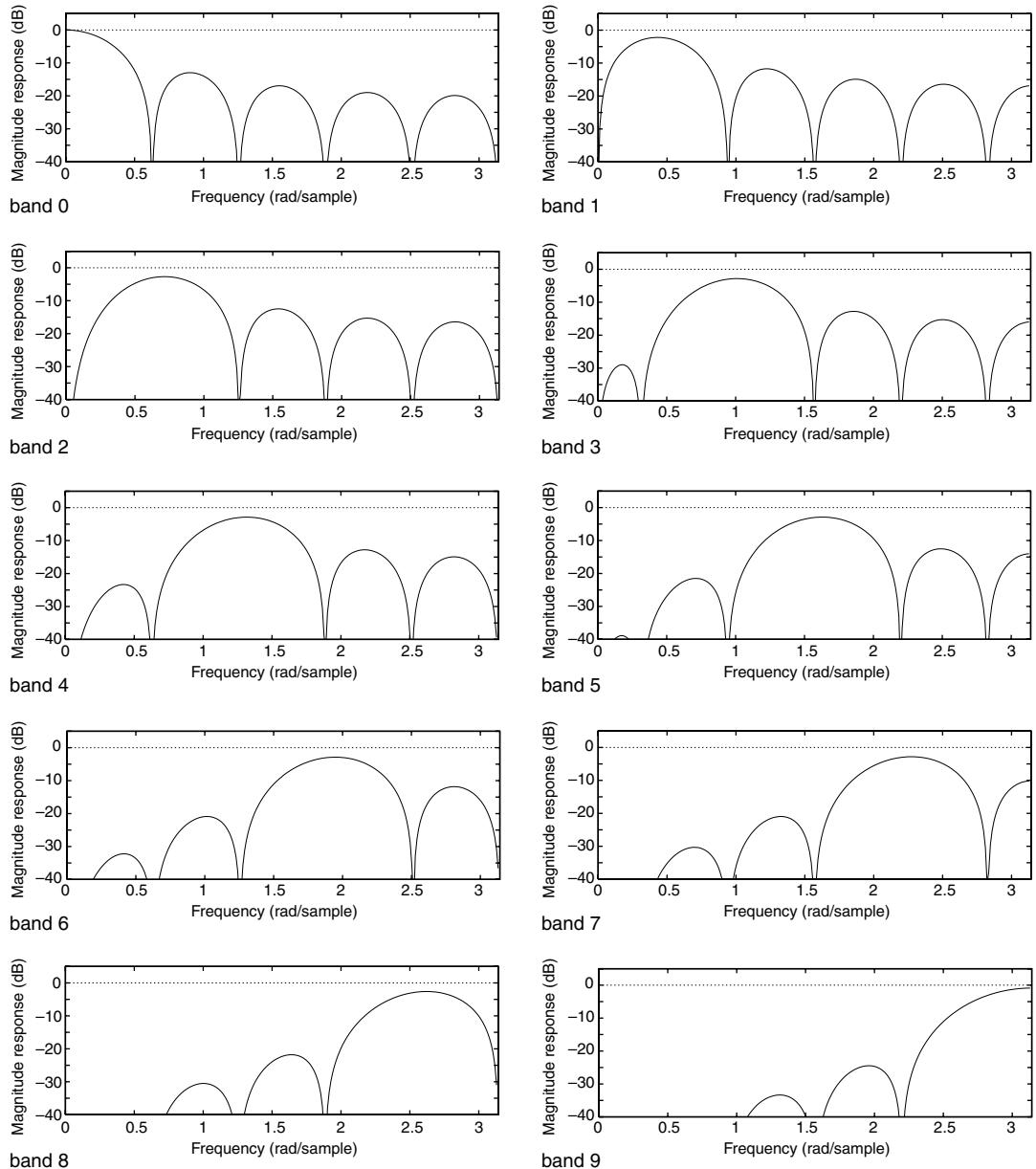


Fig. 9.32. Magnitude responses of the filters of the 10-band DCT.

By referring to Figure 9.8b, we see that the transform matrix \mathbf{T} corresponds to the polyphase matrices as follows:

$$\left. \begin{aligned} \mathbf{E}(z^M) &= \mathbf{T} \\ \mathbf{R}(z^M) &= \mathbf{T}^{*\top} \end{aligned} \right\}. \quad (9.192)$$

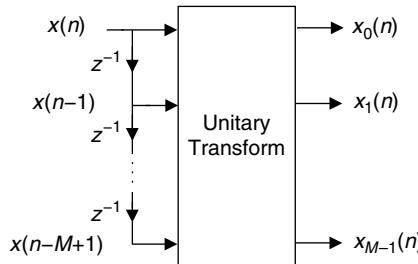


Fig. 9.33. Unitary transform analysis filter bank.

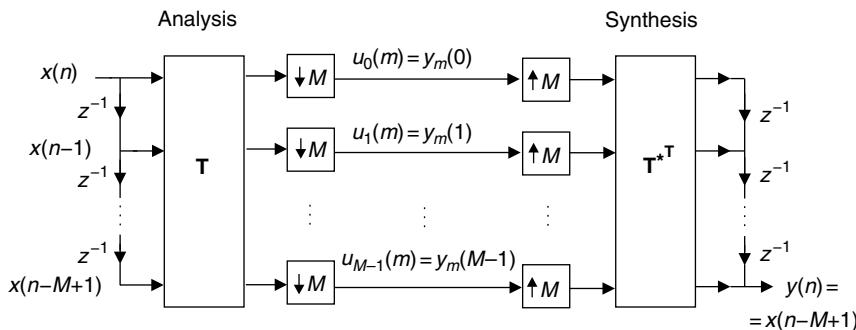


Fig. 9.34. Unitary transform filter bank.

Therefore, we conclude that the polyphase components of the filter bank corresponding to a signal transformation are constant. In addition, note that the transform matrix \mathbf{T} is such that $\mathbf{T}_{km} = c_k(M - 1 - m)$. This is so because, in the filter-bank configuration seen in Figure 9.34, the block is presented to \mathbf{T} in reversed order, compared with in a standard signal transformation operation. Then, we have that

$$\left. \begin{aligned} E_{km}(z^M) &= c_k(M - 1 - m) \\ R_{mk}(z^M) &= c_k^*(M - 1 - m) \end{aligned} \right\}. \quad (9.193)$$

Thus, from Equations (9.1) and (9.2), we conclude that

$$\left. \begin{aligned} h_k(m) &= c_k(M - 1 - m) \\ g_k(m) &= c_k^*(m) \end{aligned} \right\}. \quad (9.194)$$

Note that, in Equation (9.194), the analysis bank is delayed by \$(M - 1)\$ samples in comparison with the one shown in Figure 9.30. This explains the delay of \$(M - 1)\$ samples in Figure 9.34.

9.9 Cosine-modulated filter banks

The cosine-modulated filter banks are an attractive choice for the design and implementation of filter banks with a large number of sub-bands. Their main features are:

- Simple design procedure, consisting of generating a lowpass prototype whose impulse response satisfies some constraints required to achieve perfect reconstruction.
- Low cost of implementation measured in terms of multiplication count, since the resulting analysis and synthesis filter banks rely on a type of DCT, which is amenable to fast implementation and can share the prototype implementation cost with each sub-filter.

In the cosine-modulated filter bank design, we begin by finding a linear-phase prototype lowpass filter $H(z)$ of order N , with passband edge $\omega_p = [(\pi/2M) - \rho]$ and stopband edge $\omega_r = [(\pi/2M) + \rho]$, where 2ρ is the width of the transition band. For convenience, we assume that the length $N + 1$ is an even multiple of the number M of sub-bands; that is, $N = (2LM - 1)$. Although the actual length of the prototype can be arbitrary, this assumption greatly simplifies our analysis.³

Given the prototype filter, we can generate cosine-modulated versions of it in order to obtain the analysis and synthesis filter banks as follows:

$$h_m(n) = 2h(n) \cos\left[(2m+1)\frac{\pi}{2M}\left(n - \frac{N}{2}\right) + (-1)^m \frac{\pi}{4}\right] \quad (9.195)$$

$$g_m(n) = 2h(n) \cos\left[(2m+1)\frac{\pi}{2M}\left(n - \frac{N}{2}\right) - (-1)^m \frac{\pi}{4}\right] \quad (9.196)$$

for $n = 0, 1, \dots, N$ and $m = 0, 1, \dots, (M - 1)$, with $N = (2LM - 1)$. We should note that in Equation (9.195) the term that multiplies $h(n)$ is related to the (m, n) entry $c_{m,n}$ of an $(M \times 2LM)$ DCT-type matrix \mathbf{C} (Sorensen & Burrus, 1993), given by

$$c_{m,n} = 2 \cos\left[(2m+1)\frac{\pi}{2M}\left(n - \frac{N}{2}\right) + (-1)^m \frac{\pi}{4}\right]. \quad (9.197)$$

The prototype filter can be decomposed into $2M$ polyphase components as follows:

$$H(z) = \sum_{l=0}^{L-1} \sum_{j=0}^{2M-1} h(2lM+j)z^{-(2lM+j)} = \sum_{j=0}^{2M-1} z^{-j} E_j(z^{2M}), \quad (9.198)$$

³ It is also worth mentioning that the prototype filter does not need to be linear phase. In fact, we describe here only a particular type of cosine-modulated filter bank; many others exist.

where $E_j(z) = \sum_{l=0}^{L-1} h(2lM+j)z^{-l}$ are the polyphase components of the filter $H(z)$. With this formulation, the analysis filter bank can be described as

$$\begin{aligned} H_m(z) &= \sum_{n=0}^N h_m(n)z^{-n} \\ &= \sum_{n=0}^{2LM-1} c_{m,n}h(n)z^{-n} \\ &= \sum_{l=0}^{L-1} \sum_{j=0}^{2M-1} c_{m,2lM+j}h(2lM+j)z^{-(2lM+j)}. \end{aligned} \quad (9.199)$$

This expression can be further simplified, if we explore the following property:

$$\begin{aligned} \cos \left\{ (2m+1) \frac{\pi}{2M} \left[(n+2kM) - \frac{N}{2} \right] + \phi \right\} \\ = (-1)^k \cos \left[(2m+1) \frac{\pi}{2M} \left(n - \frac{N}{2} \right) + \phi \right], \end{aligned} \quad (9.200)$$

which leads to

$$c_{m,n+2kM} = (-1)^k c_{m,n} \quad (9.201)$$

and, therefore, substituting j for n , and l for k , we get

$$c_{m,j+2lM} = (-1)^l c_{m,j}. \quad (9.202)$$

With this relation, and after some manipulation, we can rewrite Equation (9.199) as

$$H_m(z) = \sum_{j=0}^{2M-1} c_{m,j} z^{-j} \sum_{l=0}^{L-1} (-1)^l h(2lM+j) z^{-2lM} = \sum_{j=0}^{2M-1} c_{m,j} z^{-j} E_j(-z^{2M}), \quad (9.203)$$

which can be rewritten in a compact form as

$$\mathbf{e}(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = [\mathbf{C}_1 \quad \mathbf{C}_2] \begin{bmatrix} E_0(-z^{2M}) \\ z^{-1}E_1(-z^{2M}) \\ \vdots \\ z^{-(2M-1)}E_{2M-1}(-z^{2M}) \end{bmatrix}, \quad (9.204)$$

where \mathbf{C}_1 and \mathbf{C}_2 are $M \times M$ matrices whose (m,j) elements are $c_{m,j}$ and $c_{m,j+M}$ respectively, for $m, j = 0, 1, \dots, (M-1)$.

Now, defining $\mathbf{d}(z) = [1 \ z^{-1} \ \dots \ z^{-M+1}]^T$, Equation (9.204) can be expressed in a convenient form as

$$\begin{aligned}
\mathbf{e}(z) &= [\mathbf{C}_1 \ \mathbf{C}_2] \begin{bmatrix} E_0(-z^{2M}) & \mathbf{0} \\ E_1(-z^{2M}) & \cdot \\ \cdot & \cdot \\ \mathbf{0} & E_{2M-1}(-z^{2M}) \end{bmatrix} \begin{bmatrix} \mathbf{d}(z) \\ z^{-M}\mathbf{d}(z) \end{bmatrix} \\
&= \left\{ \mathbf{C}_1 \begin{bmatrix} E_0(-z^{2M}) & \mathbf{0} \\ E_1(-z^{2M}) & \cdot \\ \cdot & \cdot \\ \mathbf{0} & E_{M-1}(-z^{2M}) \end{bmatrix} + z^{-M} \mathbf{C}_2 \begin{bmatrix} E_M(-z^{2M}) & \mathbf{0} \\ E_{M+1}(-z^{2M}) & \cdot \\ \cdot & \cdot \\ \mathbf{0} & E_{2M-1}(-z^{2M}) \end{bmatrix} \right\} \mathbf{d}(z) \\
&= \mathbf{E}(z^M)\mathbf{d}(z), \tag{9.205}
\end{aligned}$$

where $\mathbf{E}(z)$ is the polyphase matrix as in Equation (9.3).

To achieve perfect reconstruction in a filter bank with M bands, we should have $\mathbf{E}(z)\mathbf{R}(z) = \mathbf{R}(z)\mathbf{E}(z) = \mathbf{I}z^{-\Delta}$. However, it is well known (see Vaidyanathan (1993)) that the polyphase matrix of the analysis filter bank can be designed to be paraunitary or lossless; that is, $\mathbf{E}^T(z^{-1})\mathbf{E}(z) = \mathbf{I}$, where \mathbf{I} is an identity matrix of dimension M , supposing that the filter-bank coefficients are real. In this case, the synthesis filters can be easily obtained from the analysis filter bank using either Equation (9.196) or

$$\mathbf{R}(z) = z^{-\Delta}\mathbf{E}^{-1}(z) = z^{-\Delta}\mathbf{E}^T(z^{-1}). \tag{9.206}$$

It only remains to show how the prototype filter can be constrained so that the polyphase matrix of the analysis filter bank becomes paraunitary. The desired result is the following.

Property 9.1. The polyphase matrix of the analysis filter bank becomes paraunitary, for a real coefficient prototype filter, if and only if

$$E_j(z^{-1})E_j(z) + E_{j+M}(z^{-1})E_{j+M}(z) = \frac{1}{2M} \tag{9.207}$$

for $j = 0, 1, \dots, (M-1)$. If the prototype filter has linear phase, then these constraints can be reduced by half, because they are required only for $j = 0, 1, \dots, (M-1)/2$, in case M is odd, and for $j = 0, 1, \dots, (M/2)-1$, in case M is even.

Outline

In order to prove the desired result, we need the following properties related to the matrices \mathbf{C}_1 and \mathbf{C}_2 :

$$\mathbf{C}_1^T \mathbf{C}_1 = 2M[\mathbf{I} + (-1)^{L-1} \mathbf{J}] \quad (9.208)$$

$$\mathbf{C}_2^T \mathbf{C}_2 = 2M[\mathbf{I} - (-1)^{L-1} \mathbf{J}] \quad (9.209)$$

$$\mathbf{C}_1^T \mathbf{C}_2 = \mathbf{C}_2^T \mathbf{C}_1 = \mathbf{0}, \quad (9.210)$$

where \mathbf{I} is the identity matrix, \mathbf{J} is the reverse identity matrix, and $\mathbf{0}$ is a matrix with all elements equal to zero. All these matrices are square with order M . These results are widely discussed in the literature, and can be found, for instance, in Vaidyanathan (1993).

With Equations (9.208)–(9.210), it is straightforward to show that

$$\begin{aligned} & \mathbf{E}^T(z^{-1}) \mathbf{E}(z) \\ &= \left[\begin{array}{cc} E_0(-z^{-2}) & \mathbf{0} \\ E_1(-z^{-2}) & \\ \ddots & \\ \mathbf{0} & E_{M-1}(-z^{-2}) \end{array} \right] \mathbf{C}_1^T \mathbf{C}_1 \left[\begin{array}{cc} E_0(-z^2) & \mathbf{0} \\ E_1(-z^2) & \\ \ddots & \\ \mathbf{0} & E_{M-1}(-z^2) \end{array} \right] \\ &+ \left[\begin{array}{cc} E_M(-z^{-2}) & \mathbf{0} \\ E_{M+1}(-z^{-2}) & \\ \ddots & \\ \mathbf{0} & E_{2M-1}(-z^{-2}) \end{array} \right] \mathbf{C}_2^T \mathbf{C}_2 \left[\begin{array}{cc} E_M(-z^2) & \mathbf{0} \\ E_{M+1}(-z^2) & \\ \ddots & \\ \mathbf{0} & E_{2M-1}(-z^2) \end{array} \right]. \end{aligned} \quad (9.211)$$

Since the prototype is a linear-phase filter, after some manipulation it can be shown that

$$\begin{aligned} & \left[\begin{array}{cc} E_0(-z^{-2}) & \mathbf{0} \\ E_1(-z^{-2}) & \\ \ddots & \\ \mathbf{0} & E_{M-1}(-z^{-2}) \end{array} \right] \mathbf{J} \left[\begin{array}{cc} E_0(-z^2) & \mathbf{0} \\ E_1(-z^2) & \\ \ddots & \\ \mathbf{0} & E_{M-1}(-z^2) \end{array} \right] \\ &= \left[\begin{array}{cc} E_M(-z^{-2}) & \mathbf{0} \\ E_{M+1}(-z^{-2}) & \\ \ddots & \\ \mathbf{0} & E_{2M-1}(-z^{-2}) \end{array} \right] \mathbf{J} \left[\begin{array}{cc} E_M(-z^2) & \mathbf{0} \\ E_{M+1}(-z^2) & \\ \ddots & \\ \mathbf{0} & E_{2M-1}(-z^2) \end{array} \right] \end{aligned} \quad (9.212)$$

This result allows some simplification in Equation (9.211), after we apply the expressions for $\mathbf{C}_1^T \mathbf{C}_1$ and $\mathbf{C}_2^T \mathbf{C}_2$ given in Equations (9.208) and (9.209), yielding

$$\begin{aligned} \mathbf{E}^T(z^{-1})\mathbf{E}(z) &= 2M \left\{ \begin{bmatrix} E_0(-z^{-2}) & \mathbf{0} \\ E_1(-z^{-2}) & \ddots \\ \vdots & \ddots \\ \mathbf{0} & E_{M-1}(-z^{-2}) \end{bmatrix} \begin{bmatrix} E_0(-z^2) & \mathbf{0} \\ E_1(-z^2) & \ddots \\ \vdots & \ddots \\ \mathbf{0} & E_{M-1}(-z^2) \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} E_M(-z^{-2}) & \mathbf{0} \\ E_{M+1}(-z^{-2}) & \ddots \\ \vdots & \ddots \\ \mathbf{0} & E_{2M-1}(-z^{-2}) \end{bmatrix} \begin{bmatrix} E_M(-z^2) & \mathbf{0} \\ E_{M+1}(-z^2) & \ddots \\ \vdots & \ddots \\ \mathbf{0} & E_{2M-1}(-z^2) \end{bmatrix} \right\}. \end{aligned} \quad (9.213)$$

If the matrix above is equal to the identity matrix, then we achieve perfect reconstruction. This is equivalent to requiring that polyphase components of the prototype filter are pairwise power complementary, which is exactly the result of Equation (9.207). In fact, this last property can be explored to further reduce the computational complexity of these types of filter banks, by implementing the power complementary pairs with lattice realizations, which are structures especially suited for this task, as described in Vaidyanathan (1993).

Equation (9.204) suggests the structure of Figure 9.35 for the implementation of the cosine-modulated filter bank.

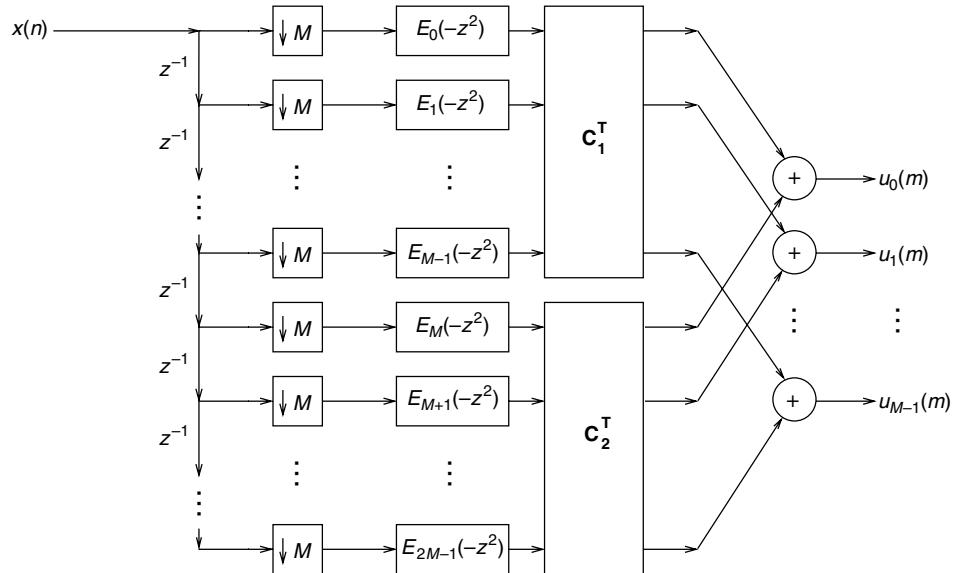


Fig. 9.35. Cosine-modulated filter bank.

This structure can be implemented using the DCT-IV, which is represented by the \mathbf{C}_M^{IV} matrix as described in Section 3.6.4 (Sorensen & Burrus, 1993), by noting that the matrices \mathbf{C}_1 and \mathbf{C}_2 can be expressed as follows (Vaidyanathan, 1993):

$$\mathbf{C}_1 = \sqrt{M}(-1)^{L/2}\mathbf{C}_M^{\text{IV}}(\mathbf{I} - \mathbf{J}) \quad (9.214)$$

$$\mathbf{C}_2 = -\sqrt{M}(-1)^{L/2}\mathbf{C}_M^{\text{IV}}(\mathbf{I} + \mathbf{J}) \quad (9.215)$$

for L even and

$$\mathbf{C}_1 = \sqrt{M}(-1)^{(L-1)/2}\mathbf{C}_M^{\text{IV}}(\mathbf{I} + \mathbf{J}) \quad (9.216)$$

$$\mathbf{C}_2 = \sqrt{M}(-1)^{(L-1)/2}\mathbf{C}_M^{\text{IV}}(\mathbf{I} - \mathbf{J}) \quad (9.217)$$

for L odd, where, from Equation (3.219) and Table 3.1,

$$\{\mathbf{C}_M^{\text{IV}}\}_{m,n} = \sqrt{\frac{2}{M}} \cos \left[(2m+1) \left(n + \frac{1}{2} \right) \frac{\pi}{2M} \right] \quad (9.218)$$

Equations (9.214)–(9.217) can be put in the following form:

$$\mathbf{C}_1 = \sqrt{M}(-1)^{\lfloor L/2 \rfloor} \mathbf{C}_M^{\text{IV}}[\mathbf{I} - (-1)^L \mathbf{J}] \quad (9.219)$$

$$\mathbf{C}_2 = -\sqrt{M}(-1)^{\lfloor L/2 \rfloor} \mathbf{C}_M^{\text{IV}}[(-1)^L \mathbf{I} + \mathbf{J}], \quad (9.220)$$

where $\lfloor x \rfloor$ represents the largest integer smaller than or equal to x .

From Equation (9.204) and the above equations, the structure in Figure 9.36 immediately follows. One of the main advantages of such a structure is that it can benefit from the fast implementation algorithms for the DCT-IV.

9.9.1 The optimization problem in the design of cosine-modulated filter banks

The procedure to design the prototype filter requires the definition of an appropriate objective function imposing not only the frequency selectivity shape but also the perfect reconstruction constraints. Usually the starting point is to establish an objective function related to the magnitude response of the desired filter that should be minimized in the least squares or Chebyshev (minimax) sense. Since the prototype filter of a cosine-modulated filter bank requires the design of a lowpass filter, the overall objective function should include a term defined as

$$\min_{\mathbf{h}} \{E_p(\omega)\} = \min_{\mathbf{h}} \left\{ \int_{\Omega_r}^{\pi} |H(e^{j\omega})|^p d\omega \right\}, \quad (9.221)$$

where $H(e^{j\omega})$ is the frequency response of the prototype filter, \mathbf{h} is the prototype filter coefficient vector, Ω_r is the stopband frequency edge, and usually $p = 2$ or $p = \infty$.

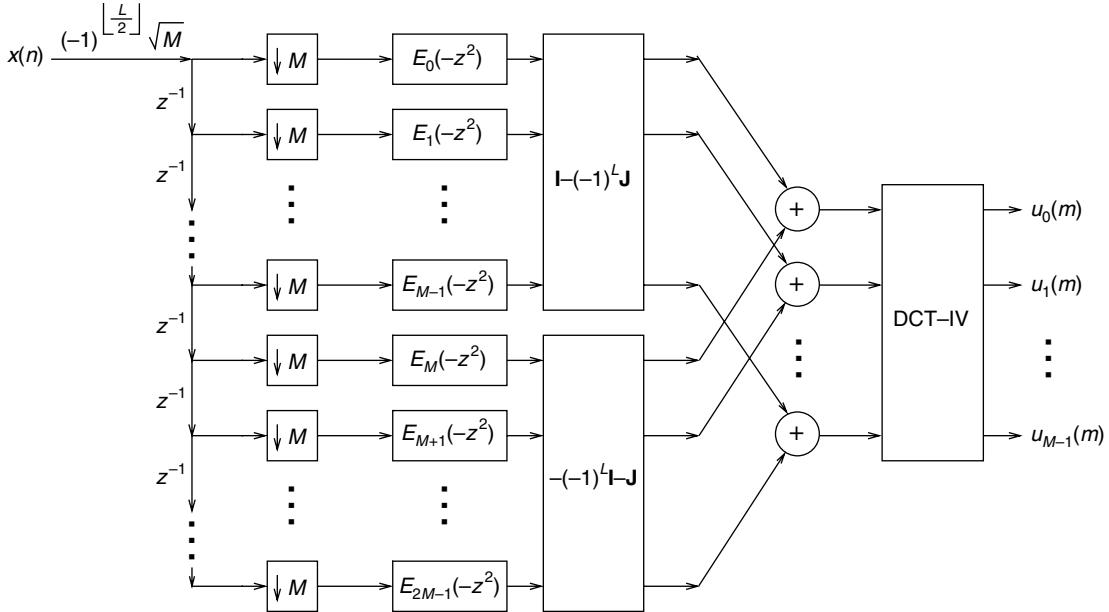


Fig. 9.36. Implementation of the cosine-modulated filter bank using the DCT-IV.

Solutions for this optimization problem can be found in Chapter 5. Unfortunately, these standard FIR filter design algorithms cannot be employed to solve the prototype filter design due to the required nonlinear constraints to enforce perfect reconstruction as given in Equation (9.207). In such cases, the prototype filter coefficients should be designed by minimizing a modified objective function $\hat{E}_p(\omega)$, that combines the original objective function $E_p(\omega)$ with a weighted set of constraints, described by

$$\hat{E}_p(\omega) = E_p(\omega) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{h}), \quad (9.222)$$

where $\boldsymbol{\lambda}$ is the vector of constraint weights and $\mathbf{c}(\mathbf{h})$ is the vector enforcing the constraints. The vectors are defined as

$$\boldsymbol{\lambda} = [\lambda_0 \lambda_1 \cdots \lambda_{N_c-1}]^T \quad (9.223)$$

$$\mathbf{c}(\mathbf{h}) = [c_0(\mathbf{h}) c_1(\mathbf{h}) \cdots c_{N_c-1}(\mathbf{h})]^T, \quad (9.224)$$

where N_c is the total number of constraints.

For example the constraints of Equation (9.207) can be described in the time domain using a MATLAB notation as follows:

$$\text{conv}(\hat{\mathbf{e}}_j, \hat{\mathbf{e}}_j \mathbf{J}) + \text{conv}(\hat{\mathbf{e}}_{j+M}, \hat{\mathbf{e}}_{j+M} \mathbf{J}) - \frac{1}{2M} dt(i) = 0 \quad (9.225)$$

for $j = 0, 1, \dots, (M-1)$, where $\hat{\mathbf{e}}_j$ is a row vector containing the $L = (N+1)/2M$ coefficients of the polyphase component $E_j(z)$, as defined in Equation (9.198), \mathbf{J} is the

antidiagonal matrix, and $\text{dt}(i)$ in the present discussion represents a sequence in MATLAB with $(L - 1)$ zeros followed by a unit impulse at $i = 0$ and $(L - 1)$ additional zeros; that is:

$$\text{dt}(i) = [\text{zeros}(L - 1, 1); 1; \text{zeros}(L - 1, 1)]. \quad (9.226)$$

This set of constraints should be incorporated in the optimization process in the appropriate manner.

This optimization problem can be solved using quadratic programming (QP) algorithms (Antoniou and Lu, 2007), which may require information on the first and second derivatives of $E_p(\omega)$, if possible, to simplify its implementation and speed the convergence. The constraint weights should be chosen beforehand when using a QP optimization method. Another solution is to employ a sequential quadratic programming (SQP) algorithm, which optimally sets the weights of the constraints based on the method of Lagrange multipliers with the Kuhn–Tucker conditions; see Antoniou and Lu (2007). The minimization problem can also be solved utilizing the technique described in Section 6.5.2. The minimization of $\hat{E}_p(\omega)$ in Equation (9.222), using the WLS–Chebyshev method, was successfully proposed by Furtado *et al.* (2005a), leading to a rather simpler design method. Other efficient methods for designing cosine-modulated filter banks are available, such as the one in Lu *et al.* (2004) based on the cone programming approach (see Antoniou and Lu (2007)), and the method proposed by Kha *et al.* (2009) based on convex optimization (see Boyd and Vandenberghe (2004)). Indeed, the design of highly sophisticated cosine-modulated filter banks can be achieved by adapting and applying the frequency response masking approach of Section 12.7.3 to the prototype filter design, in order to reduce the number of distinct parameters; see Diniz *et al.* (2004) and Furtado *et al.* (2003, 2005a,b).

Example 9.10. Design a filter bank with $M = 10$ sub-bands using the cosine-modulated method with $L = 3$.

Solution

For the prototype linear-phase design, we employ the least-squares design method described in Chapter 5, using as the objective the minimization of the filter stopband energy. This has been obtained by sampling the error function only over the frequencies $\omega_i > \omega_p$. The perfect reconstruction restrictions in Equations (9.207) and (9.225) have been dealt with using the method proposed in Furtado, *et al.* (2005a).

The length of the resulting prototype filter is $(N + 1) = 2LM = 60$, and the minimum stopband attenuation obtained was $A_r \approx 40$ dB, as shown in Figure 9.37. Its impulse response is depicted in Figure 9.38 and the values of the filter coefficients are listed in Table 9.1. Since the prototype filter has linear phase, only half of its coefficients are shown.

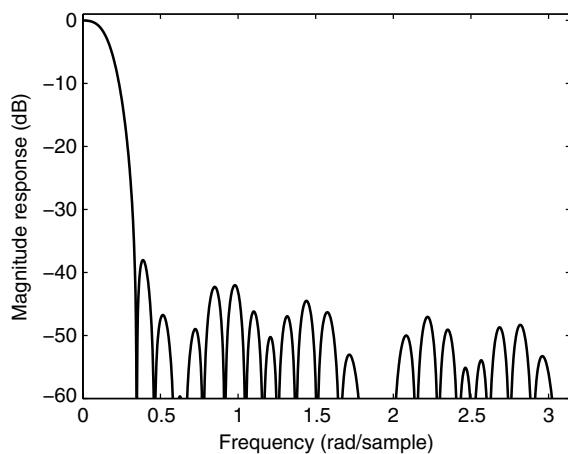
As an illustration, let us verify a subset of the constraints of Equation (9.225) for the case $j = 0$, where

$$\begin{aligned} \hat{\mathbf{e}}_0 &= [h(0) \ h(20) \ h(40)] \\ &= [-8.1483\text{E}-04 \ 2.5850\text{E}-02 \ 2.0509\text{E}-02] \end{aligned} \quad (9.227)$$

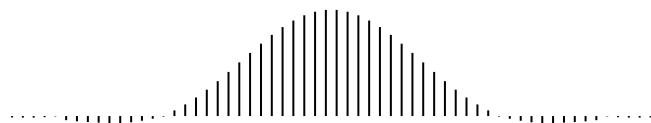
Table 9.1.

Prototype filter coefficients $h(0)$ to $h(29)$ for cosine-modulated filter bank.

$h(0) = -8.1483E-04$	$h(10) = -4.0655E-03$	$h(20) = 2.5850E-02$
$h(1) = -9.0356E-04$	$h(11) = -3.6256E-03$	$h(21) = 3.1419E-02$
$h(2) = -8.8926E-04$	$h(12) = -2.7854E-03$	$h(22) = 3.7012E-02$
$h(3) = -7.0418E-04$	$h(13) = -1.5731E-03$	$h(23) = 4.2454E-02$
$h(4) = -2.5612E-04$	$h(14) = -3.4745E-04$	$h(24) = 4.7570E-02$
$h(5) = -2.4480E-03$	$h(15) = 3.3209E-03$	$h(25) = 5.2153E-02$
$h(6) = -3.0413E-03$	$h(16) = 6.7943E-03$	$h(26) = 5.6030E-02$
$h(7) = -3.4742E-03$	$h(17) = 1.0882E-02$	$h(27) = 5.9085E-02$
$h(8) = -3.8572E-03$	$h(18) = 1.5477E-02$	$h(28) = 6.1192E-02$
$h(9) = -4.1105E-03$	$h(19) = 2.0509E-02$	$h(29) = 6.2266E-02$

**Fig. 9.37.**

Magnitude response of the prototype filter for cosine-modulated filter bank.

**Fig. 9.38.**

Impulse response of the prototype filter.

$$\begin{aligned}\hat{\mathbf{e}}_{10} &= [h(10) \ h(30) \ h(50)] \\ &= [-4.0655E-03 \ 6.2266E-02 \ -4.1105E-03],\end{aligned}\quad (9.228)$$

which yield

$$\begin{aligned}\text{conv}(\hat{\mathbf{e}}_0, \hat{\mathbf{e}}_0 \mathbf{J}) + \text{conv}(\hat{\mathbf{e}}_{10}, \hat{\mathbf{e}}_{10} \mathbf{J}) &= [0 \ 0 \ 0.05 \ 0 \ 0]^T \\ &= \frac{1}{2M} dt(i) = 0.05 dt(i),\end{aligned}\quad (9.229)$$

which is the expected result.

By applying cosine modulation to the prototype filter using Equations (9.195) and (9.196), we obtain the coefficients of the analysis and synthesis filters belonging to the filter bank. The impulse responses of the resulting analysis filters, each of length 60, for the bank with $M = 10$ sub-bands, are shown in Figure 9.39 and their normalized magnitude responses in Figure 9.40. \triangle

The filter banks discussed in this section have the disadvantage of having the nonlinear-phase characteristic of the analysis filters, despite the prototype filter being linear phase. This problem is solved with the filter banks described in the next section.

9.10 Lapped transforms

The lapped orthogonal transforms (LOTs) were originally proposed to be block transforms whose basis functions extended beyond the block boundaries; that is, the basis functions from neighboring blocks overlapped with each other. Their main goal was to reduce blocking effects, usually present under quantization of the block transform coefficients, as is the case of the DCT. Blocking effects are discontinuities that appear across the block boundaries. They occur because each block is transformed and quantized independently of the others, and this type of distortion is particularly annoying in images (Malvar, 1992). If the transform basis functions are allowed to overlap, the blocking effects are greatly reduced.

Since block transforms are equivalent to M -band perfect reconstruction filter banks, one idea would be to replace the DCT by a cosine-modulated filter bank. However, the analysis filters of the cosine-modulated filter banks discussed in the previous section have nonlinear phase, an undesirable feature in applications such as image coding. LOT-based filter banks are very attractive because they lead to linear-phase analysis filters and have fast implementation. The LOT-based filter banks are members of the family of paraunitary FIR perfect reconstruction filter banks with linear phase. Although there are a number of possible designs for linear-phase filter banks with perfect reconstruction, the LOT-based design is simple to derive and to implement. The term LOT applies to the cases where the analysis filters have length $2M$. Generalizations of the LOT to longer analysis and synthesis filters (length LM) are available. They are known as the extended lapped transforms (ELTs) proposed by Malvar (1992) and the generalized LOT (GenLOT) proposed by De Queiroz *et al.* (1996). The ELT is a cosine-modulated filter bank and does not produce linear-phase analysis filters. The GenLOT is a good choice when long analysis filters with high selectivity are required together with linear phase.

In this section we begin by briefly discussing the LOT filter bank, where the analysis and synthesis filter banks have lengths $2M$.

Once again, similar to the cosine-modulated filter banks, the LOT analysis filters are given by

$$\mathbf{e}(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{C}}_1 & \hat{\mathbf{C}}_2 \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(2M-1)} \end{bmatrix}, \quad (9.230)$$

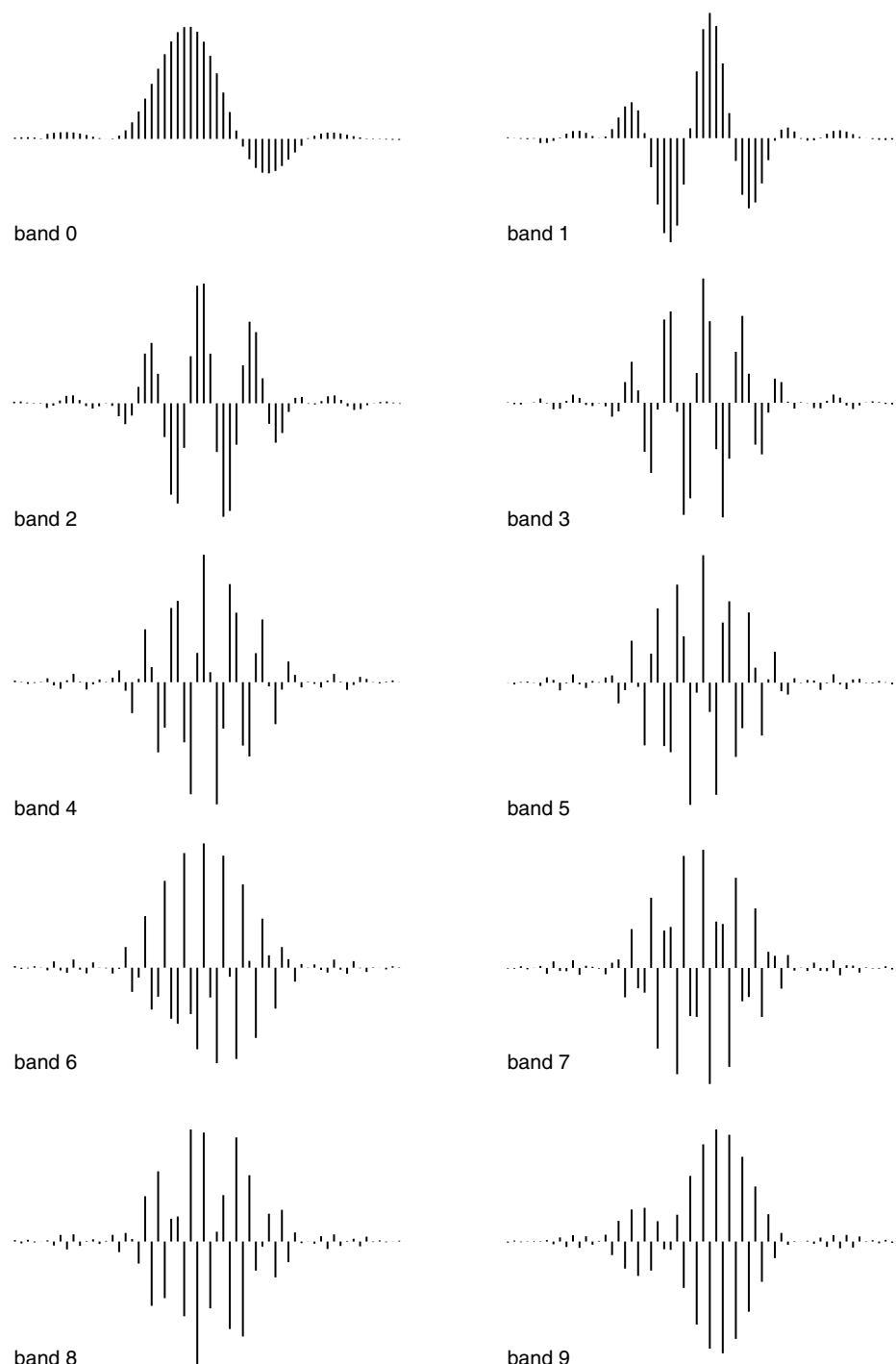


Fig. 9.39. Impulse responses of the analysis filters of a 10-band cosine-modulated filter bank of length 60.

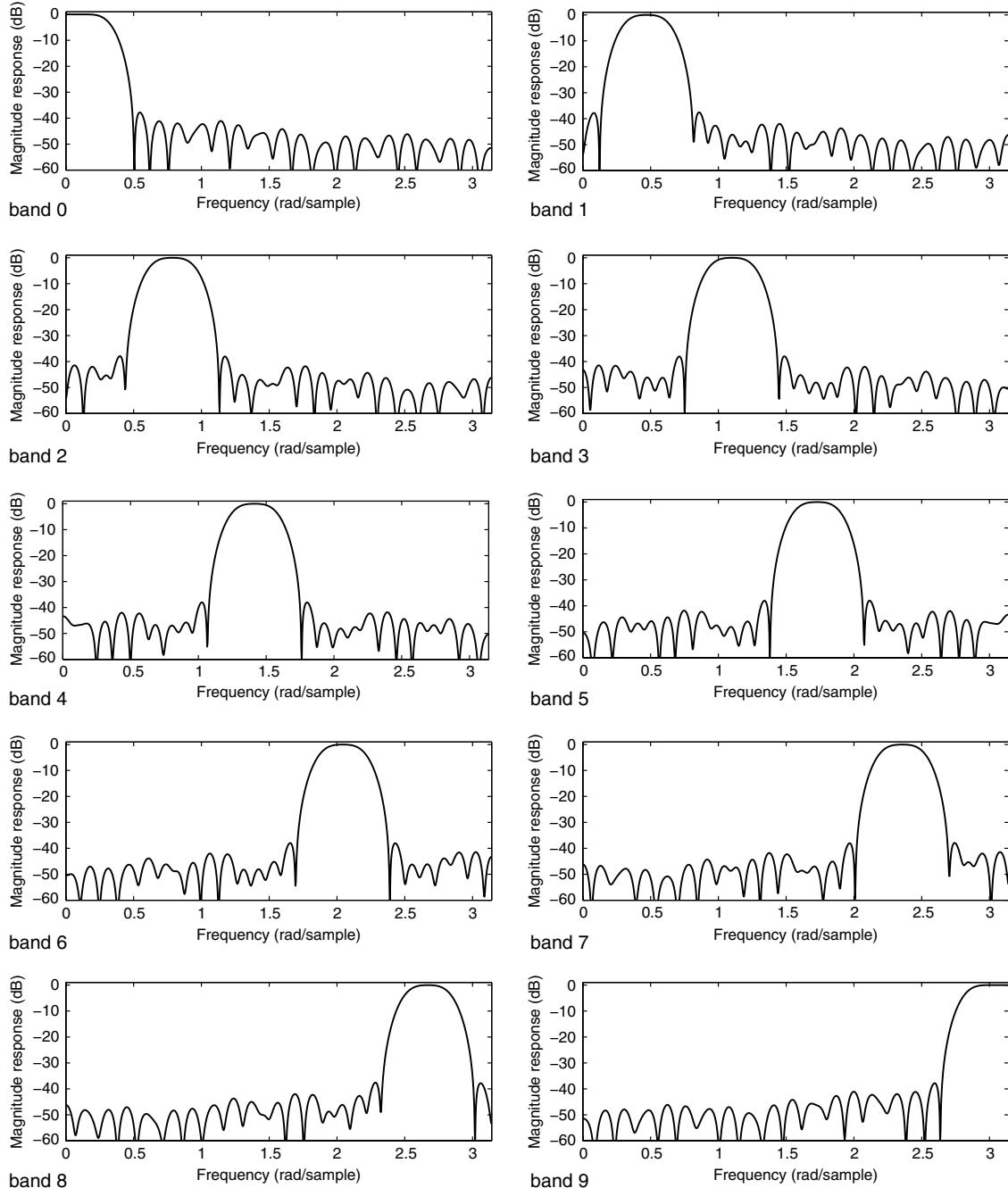


Fig. 9.40. Magnitude responses of the analysis filters of a 10-band cosine-modulated filter bank of length 60.

or, as in Equation (9.205), by

$$\mathbf{e}(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = (\hat{\mathbf{C}}_1 + z^{-M}\hat{\mathbf{C}}_2)\mathbf{d}(z) = \mathbf{E}(z^M)\mathbf{d}(z), \quad (9.231)$$

where $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{C}}_2$ are $M \times M$ DCT-type matrices and $\mathbf{E}(z)$ is the polyphase matrix of the analysis filter bank. Note that if $\hat{\mathbf{C}}_1$ is applied to a length- M block, then due to the term z^{-M} , matrix $\hat{\mathbf{C}}_2$ is applied to the previous length- M block. In other words, it is as if two adjacent blocks are needed to compute a transform of a given block; that is, there is an overlap between blocks in the computation of this transform. The perfect reconstruction condition with paraunitary polyphase matrices is generated if

$$\mathbf{R}(z) = z^{-\Delta}\mathbf{E}^{-1}(z) = z^{-\Delta}\mathbf{E}^T(z^{-1}). \quad (9.232)$$

We then have the result that follows.

Property 9.2. The polyphase matrix of the analysis filter bank becomes paraunitary, for a real coefficient prototype filter, if the following conditions are satisfied:

$$\hat{\mathbf{C}}_1\hat{\mathbf{C}}_1^T + \hat{\mathbf{C}}_2\hat{\mathbf{C}}_2^T = \mathbf{I} \quad (9.233)$$

$$\hat{\mathbf{C}}_1\hat{\mathbf{C}}_2^T = \hat{\mathbf{C}}_2\hat{\mathbf{C}}_1^T = \mathbf{0}. \quad (9.234)$$

Proof From Equation (9.231), we have that

$$\mathbf{E}(z) = \hat{\mathbf{C}}_1 + z^{-1}\hat{\mathbf{C}}_2. \quad (9.235)$$

Perfect reconstruction requires that $\mathbf{E}(z)\mathbf{E}^T(z^{-1}) = \mathbf{I}$. Therefore:

$$(\hat{\mathbf{C}}_1 + z^{-1}\hat{\mathbf{C}}_2)(\hat{\mathbf{C}}_1^T + z\hat{\mathbf{C}}_2^T) = \hat{\mathbf{C}}_1\hat{\mathbf{C}}_1^T + \hat{\mathbf{C}}_2\hat{\mathbf{C}}_2^T + z\hat{\mathbf{C}}_1\hat{\mathbf{C}}_2^T + z^{-1}\hat{\mathbf{C}}_2\hat{\mathbf{C}}_1^T = \mathbf{I} \quad (9.236)$$

and then Equations (9.233) and (9.234) immediately follow. \square

Equation (9.233) implies that the rows of $\hat{\mathbf{C}} = [\hat{\mathbf{C}}_1 \quad \hat{\mathbf{C}}_2]$ are orthogonal. Meanwhile, Equation (9.234) implies that the rows of $\hat{\mathbf{C}}_1$ are orthogonal to the rows of $\hat{\mathbf{C}}_2$, which is the same as saying that the overlapping tails of the basis functions of the LOT are orthogonal.

A simple construction for the matrices above, based on the DCT, and leading to linear-phase filters, results from choosing

$$\hat{\mathbf{C}}_1 = \frac{1}{2} \begin{bmatrix} \mathbf{C}_e - \mathbf{C}_o \\ \mathbf{C}_e + \mathbf{C}_o \end{bmatrix} \quad (9.237)$$

$$\hat{\mathbf{C}}_2 = \frac{1}{2} \begin{bmatrix} (\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \\ -(\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathbf{C}_e + \mathbf{C}_o \\ -\mathbf{C}_e - \mathbf{C}_o \end{bmatrix}, \quad (9.238)$$

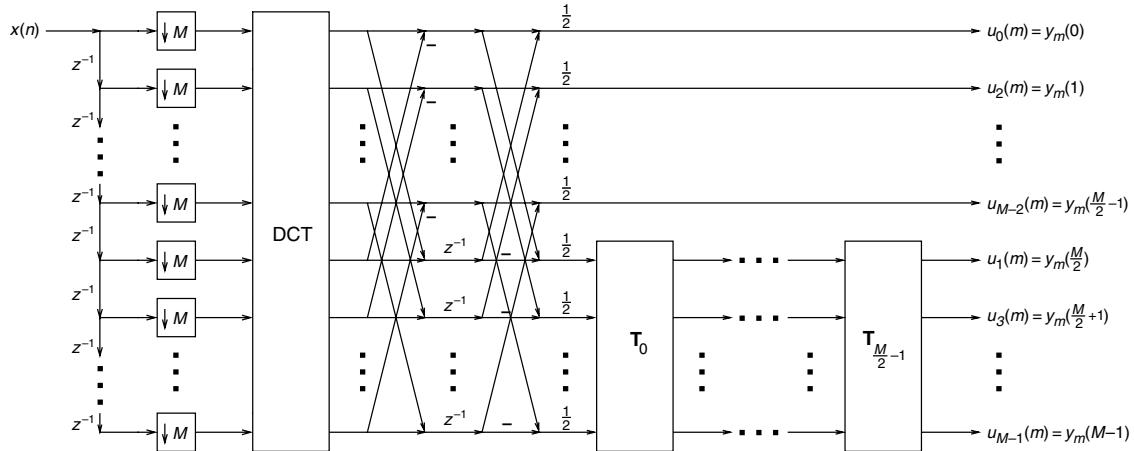


Fig. 9.41. Implementation of the lapped orthogonal transform.

where \mathbf{C}_e and \mathbf{C}_o are $\frac{M}{2} \times M$ matrices consisting of the even and odd DCT bases of length M respectively (see Equation (9.190)). Observe that $\mathbf{C}_e\mathbf{J} = \mathbf{C}_e$ and $\mathbf{C}_o\mathbf{J} = -\mathbf{C}_o$, because the even basis functions are symmetric and the odd ones are antisymmetric. The reader can easily verify that the above choice satisfies the relations (9.233) and (9.234). With this, we can build an initial LOT whose polyphase matrix, as in Equation (9.235), is given by

$$\mathbf{E}(z) = \frac{1}{2} \begin{bmatrix} \mathbf{C}_e - \mathbf{C}_o + z^{-1}(\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \\ \mathbf{C}_e - \mathbf{C}_o - z^{-1}(\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathbf{I} & z^{-1}\mathbf{I} \\ \mathbf{I} & -z^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{I} \\ \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{C}_e \\ \mathbf{C}_o \end{bmatrix}. \quad (9.239)$$

This expression suggests the structure of Figure 9.41 for the implementation of the LOT filter bank. This structure consists of the implementation of the polyphase components of the prototype filter using a DCT-based matrix, followed by several orthogonal matrices $\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_{(M/2)-2}$, which are discussed next. Actually, we can pre-multiply the right-hand term in Equation (9.239) by an orthogonal matrix \mathbf{L}_1 , and still keep the perfect reconstruction conditions.⁴ The polyphase matrix is then given by

$$\mathbf{E}(z) = \frac{1}{2} \mathbf{L}_1 \begin{bmatrix} \mathbf{I} & z^{-1}\mathbf{I} \\ \mathbf{I} & -z^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{I} \\ \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{C}_e \\ \mathbf{C}_o \end{bmatrix}. \quad (9.240)$$

The basic construction of the LOT presented above is equivalent to the one proposed by Malvar (1992) that utilizes a block transform formulation to generate lapped transforms. Note that, in the block transform formulation, Equation (9.230) is equivalent to a block transform whose transform matrix \mathbf{L}_{LOT} is equal to $[\hat{\mathbf{C}}_1 \hat{\mathbf{C}}_2]$. Referring to Figure 9.29, it is important to point out that, since the transform matrix \mathbf{L}_{LOT} has dimensions $M \times 2M$, in order to compute the transform coefficients $y_j(k)$ of a block B_j one needs the samples

⁴ It can easily be seen that Equations (9.233) and (9.234) remain valid, if $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{C}}_2$ are pre-multiplied by an orthogonal matrix. However, \mathbf{L}_1 should not destroy the symmetry or the antisymmetry of the analysis filter's impulse response.

$x_j(m)$ and $x_{j+1}(m)$ of the blocks B_j and B_{j+1} respectively. The term “lapped transform” comes from the fact that the block B_{j+1} is needed in the computation of both $y_j(k)$ and $y_{j+1}(k)$; that is, the transforms of the two blocks overlap. This is expressed formally by the following equation:

$$\begin{aligned} \begin{bmatrix} y_j(0) \\ \vdots \\ y_j(M-1) \end{bmatrix} &= \mathbf{L}_{\text{LOT}} \begin{bmatrix} x_j(0) \\ \vdots \\ x_j(M-1) \\ x_{j+1}(0) \\ \vdots \\ x_{j+1}(M-1) \end{bmatrix} \\ &= [\hat{\mathbf{C}}_1 \quad \hat{\mathbf{C}}_2] \begin{bmatrix} x_j(0) \\ \vdots \\ x_j(M-1) \\ x_{j+1}(0) \\ \vdots \\ x_{j+1}(M-1) \end{bmatrix} \\ &= \hat{\mathbf{C}}_1 \begin{bmatrix} x_j(0) \\ \vdots \\ x_j(M-1) \end{bmatrix} + \hat{\mathbf{C}}_2 \begin{bmatrix} x_{j+1}(0) \\ \vdots \\ x_{j+1}(M-1) \end{bmatrix}. \end{aligned} \quad (9.241)$$

Malvar’s formulation for the LOT differs somewhat from that in Equations (9.230)–(9.240). In fact, Malvar starts with an orthogonal matrix based on the DCT having the following form:

$$\mathbf{L}_0 = \frac{1}{2} \begin{bmatrix} \mathbf{C}_e - \mathbf{C}_o & (\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \\ \mathbf{C}_e - \mathbf{C}_o & -(\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \end{bmatrix}. \quad (9.242)$$

The choice is not at random. First, it satisfies the conditions of Equations (9.233) and (9.234). Also, the first half of the basis functions are symmetric, whereas the second half are antisymmetric, thus keeping the phase linear, as desired. The choice of \mathbf{L}_0 based on the DCT is the key to generating a fast implementation algorithm. Starting with \mathbf{L}_0 , we can generate a family of more selective analysis filters in the following form:

$$\mathbf{L}_{\text{LOT}} = [\hat{\mathbf{C}}_1 \quad \hat{\mathbf{C}}_2] = \mathbf{L}_1 \mathbf{L}_0, \quad (9.243)$$

where the matrix \mathbf{L}_1 should be orthogonal and should also be amenable to fast implementation. Usually, the matrix \mathbf{L}_1 is of the form

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 \end{bmatrix}, \quad (9.244)$$

where \mathbf{L}_2 is a square matrix of dimension $M/2$ consisting of a set of plane rotations. More specifically:

$$\mathbf{L}_2 = \mathbf{T}_{(M/2)-2} \cdots \mathbf{T}_1 \mathbf{T}_0, \quad (9.245)$$

where

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{I}_i & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}(\theta_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{(M/2)-2-i} \end{bmatrix} \quad (9.246)$$

and

$$\mathbf{Y}(\theta_i) = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix}. \quad (9.247)$$

The rotation angles θ_i are submitted to optimization aimed at maximizing the coding gain, when using the filter bank in sub-band coders, or at improving the selectivity of the analysis and synthesis filters (Malvar, 1992). Since in the multiplication by \mathbf{T}_i only the samples i and $i + 1$ are modified, a flowgraph that implements it is as depicted in Figure 9.42.

By referring to Figure 9.41 and Equation (9.244), one can see that, in Equation (9.243), the rows of matrix \mathbf{L}_{LOT} are not organized in frequency order – in fact, the first $M/2$ rows correspond to the even bands and the last $M/2$ rows correspond to the odd bands.

A simple fast LOT implementation not allowing any type of optimization consists of, instead of implementing \mathbf{L}_2 as a cascade of rotations \mathbf{T}_i , implementing \mathbf{L}_2 as a cascade of square matrices of dimension $\hat{M} = M/2$ comprised of DCTs Type II and Type IV (Malvar, 1992) (see Section 3.6.4), the elements of which, from Equation (3.219) and Table 3.1, are

$$c_{l,n} = \alpha_{\hat{M}}(l) \sqrt{\frac{2}{\hat{M}}} \cos \left[(2l+1) \frac{\pi}{2\hat{M}} (n) \right] \quad (9.248)$$

and

$$c_{l,n} = \sqrt{\frac{2}{\hat{M}}} \cos \left[(2l+1) \frac{\pi}{2\hat{M}} \left(n + \frac{1}{2} \right) \right] \quad (9.249)$$

respectively, where $\alpha_{\hat{M}}(l) = 1/\sqrt{2}$, for $l = 0$, or $l = \hat{M}$, and $\alpha_{\hat{M}}(l) = 1$ otherwise. The implementation is fast because the DCTs of Types II and IV have fast algorithms.

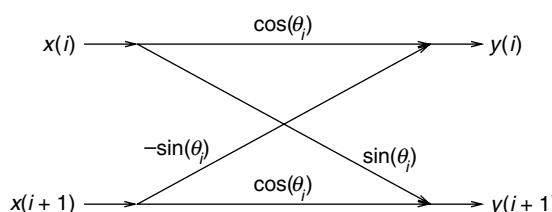


Fig. 9.42. Implementation of multiplication by \mathbf{T}_i .

Example 9.11. Design a filter bank with $M = 10$ sub-bands using the fast LOT.

Solution

The length of the analysis and synthesis filters should be $(N + 1) = 2M = 20$. Therefore, we use as a basis for this design the DCT matrix of order $M = 10$, whose basis functions are given by Equation (9.190) in Example 9.9 as

$$c_k(m) = \alpha(k) \cos\left[\frac{\pi(2m+1)k}{20}\right], \quad (9.250)$$

where $\alpha(0) = \sqrt{1/20}$ and $\alpha(k) = \sqrt{1/10}$, for $k = 1, 2, \dots, 9$.

Therefore, the $M/2 \times M$ matrices \mathbf{C}_e and \mathbf{C}_o are

$$\mathbf{C}_e = \begin{bmatrix} c_0(0) & c_0(1) & c_0(2) & c_0(3) & c_0(4) & c_0(5) & c_0(6) & c_0(7) & c_0(8) & c_0(9) \\ c_2(0) & c_2(1) & c_2(2) & c_2(3) & c_2(4) & c_2(5) & c_2(6) & c_2(7) & c_2(8) & c_2(9) \\ c_4(0) & c_4(1) & c_4(2) & c_4(3) & c_4(4) & c_4(5) & c_4(6) & c_4(7) & c_4(8) & c_4(9) \\ c_6(0) & c_6(1) & c_6(2) & c_6(3) & c_6(4) & c_6(5) & c_6(6) & c_6(7) & c_6(8) & c_6(9) \\ c_8(0) & c_8(1) & c_8(2) & c_8(3) & c_8(4) & c_8(5) & c_8(6) & c_8(7) & c_8(8) & c_8(9) \end{bmatrix} \quad (9.251)$$

$$\mathbf{C}_o = \begin{bmatrix} c_1(0) & c_1(1) & c_1(2) & c_1(3) & c_1(4) & c_1(5) & c_1(6) & c_1(7) & c_1(8) & c_1(9) \\ c_3(0) & c_3(1) & c_3(2) & c_3(3) & c_3(4) & c_3(5) & c_3(6) & c_3(7) & c_3(8) & c_3(9) \\ c_5(0) & c_5(1) & c_5(2) & c_5(3) & c_5(4) & c_5(5) & c_5(6) & c_5(7) & c_5(8) & c_5(9) \\ c_7(0) & c_7(1) & c_7(2) & c_7(3) & c_7(4) & c_7(5) & c_7(6) & c_7(7) & c_7(8) & c_7(9) \\ c_9(0) & c_9(1) & c_9(2) & c_9(3) & c_9(4) & c_9(5) & c_9(6) & c_9(7) & c_9(8) & c_9(9) \end{bmatrix} \quad (9.252)$$

and the matrix \mathbf{L}_0 can be built using Equation (9.242). For the fast LOT design, we use a factorable \mathbf{L}_1 matrix composed of a cascade of a \mathbf{C}^{II} transform and a transposed \mathbf{C}^{IV} transform on its lower part, to allow fast implementation; that is:

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{II} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{IV^T} \end{bmatrix}, \quad (9.253)$$

where \mathbf{C}^{II} and \mathbf{C}^{IV} are square matrices of dimension $M/2 = 5$, defined in Equation (3.219) and Table 3.1.

The impulse responses of the resulting analysis filters, given by Equation (9.230), with $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{C}}_2$ as defined in Equations (9.237) and (9.238) respectively, are shown in Figure 9.43. The coefficients of the analysis filters are given by the rows of \mathbf{L}_{LOT} , which is defined in Equation (9.243). The coefficients of the first analysis filter are listed in Table 9.2. Again, because of the linear-phase property, only half of the coefficients are shown.

Figure 9.44 depicts the normalized magnitude responses of the analysis filters for the fast LOT filter bank. The impulse responses and magnitude responses of the analysis filters are shown in Figures 9.43 and 9.44 in an increasing position of the magnitude response peak.

△

Table 9.2.

Fast LOT analysis filter coefficients $h(0)$ to $h(9)$: band 0.

$h(0) = -6.2740\text{E}-02$	$h(4) = 1.2313\text{E}-01$	$h(7) = 3.1623\text{E}-01$
$h(1) = -4.1121\text{E}-02$	$h(5) = 1.9309\text{E}-01$	$h(8) = 3.5735\text{E}-01$
$h(2) = -2.7756\text{E}-17$	$h(6) = 2.5963\text{E}-01$	$h(9) = 3.7897\text{E}-01$
$h(3) = 5.6599\text{E}-02$		

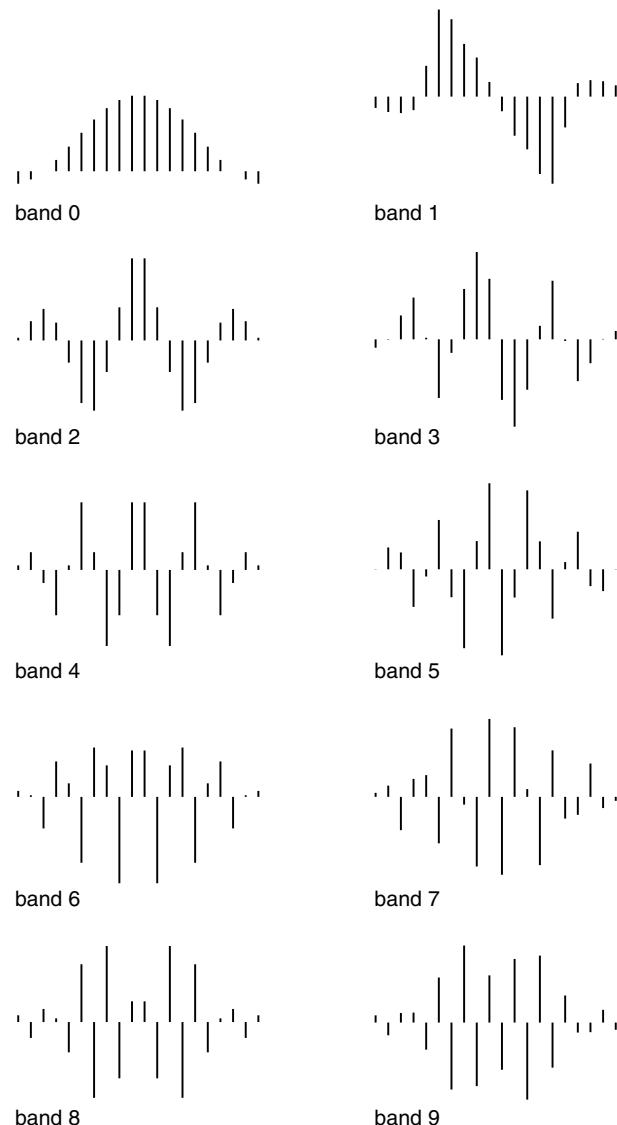


Fig. 9.43.

Impulse responses of the filters of a 10-band fast LOT.

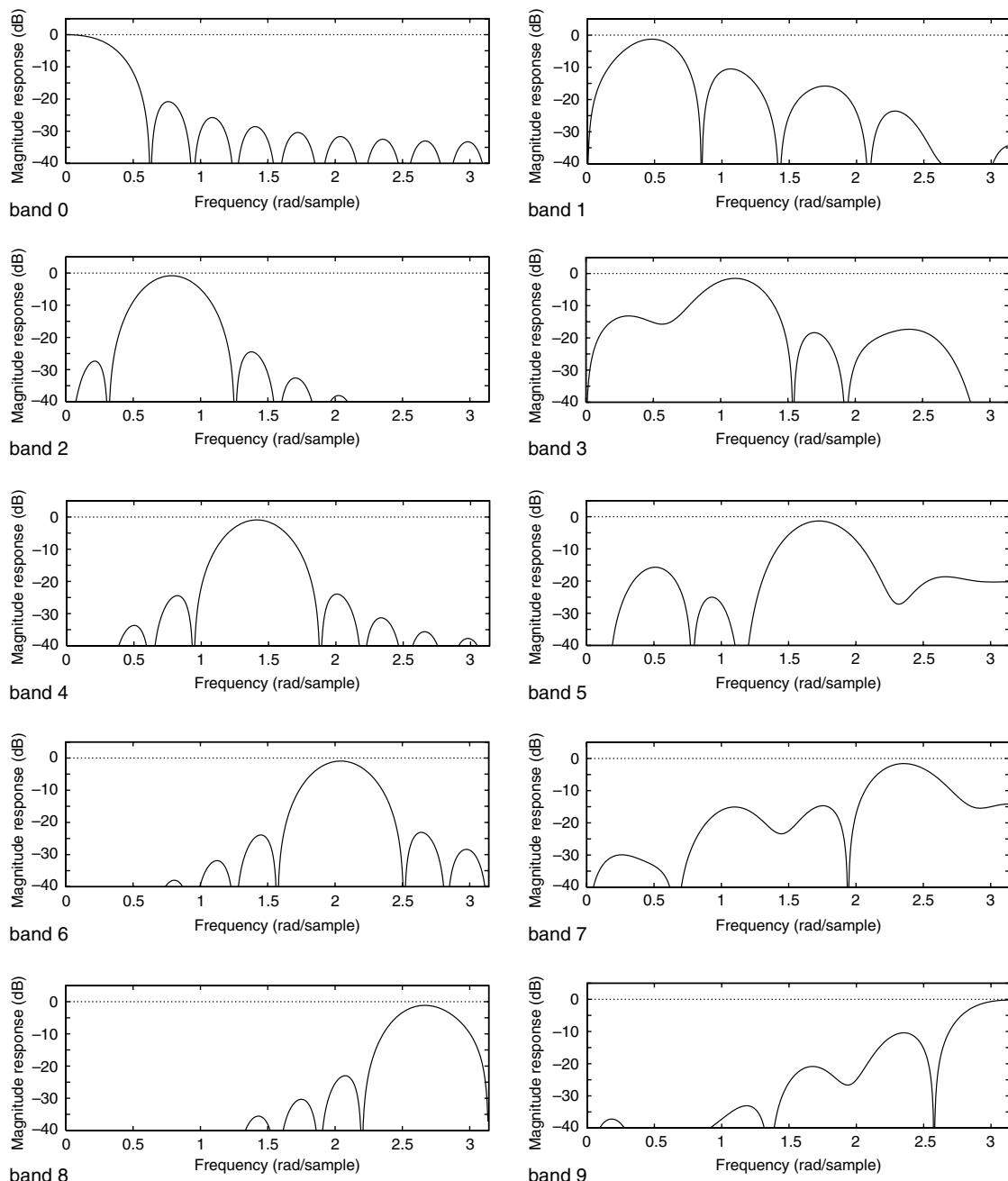


Fig. 9.44. Magnitude responses of the filters of a 10-band fast LOT.

9.10.1 Fast algorithms and biorthogonal LOT

We now present a general construction of a fast algorithm for the LOT. We start by defining two matrices $\hat{\mathbf{C}}_3$ and $\hat{\mathbf{C}}_4$ such that

$$\hat{\mathbf{C}}_1 = \hat{\mathbf{C}}_3 \hat{\mathbf{C}}_4 \quad (9.254)$$

$$\hat{\mathbf{C}}_2 = (\mathbf{I} - \hat{\mathbf{C}}_3) \hat{\mathbf{C}}_4. \quad (9.255)$$

With the above relations, it is straightforward to show using Equation (9.235) that the polyphase components of the analysis filter can be written as

$$\mathbf{E}(z) = [\hat{\mathbf{C}}_3 + z^{-1}(\mathbf{I} - \hat{\mathbf{C}}_3)]\hat{\mathbf{C}}_4. \quad (9.256)$$

The initial solution for the LOT matrix discussed previously can be analyzed in the light of this general formulation. Equation (9.256) suggests the implementation in Figure 9.45. After a few manipulations, the matrices of the polyphase description above corresponding to the LOT matrix of Equation (9.242) are given by

$$\hat{\mathbf{C}}_3 = \frac{1}{2} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{I} \end{bmatrix} \quad (9.257)$$

$$\hat{\mathbf{C}}_4 = \frac{1}{2} \begin{bmatrix} \mathbf{C}_e - \mathbf{C}_o + (\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \\ \mathbf{C}_e - \mathbf{C}_o - (\mathbf{C}_e - \mathbf{C}_o)\mathbf{J} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_e \\ -\mathbf{C}_o \end{bmatrix}. \quad (9.258)$$

One can see that, in the LOT given by Equations (9.257) and (9.258), matrices $\hat{\mathbf{C}}_3$ and $\hat{\mathbf{C}}_4$ have fast implementation algorithms, and Figure 9.45 leads to Figure 9.41. In fact, as long as there are fast implementation algorithms for $\hat{\mathbf{C}}_3$ and $\hat{\mathbf{C}}_4$, the LOT will have a fast algorithm.

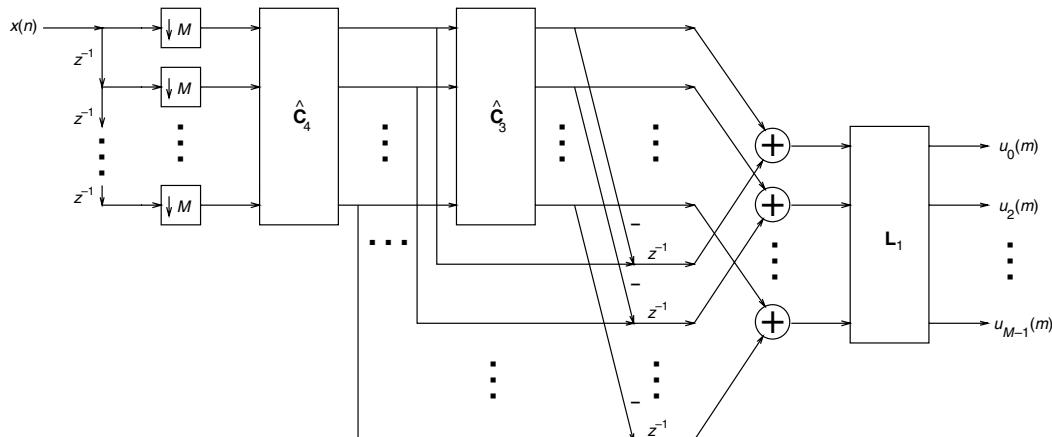


Fig. 9.45. Implementation of the LOT according to the general formulation in Equation (9.256).

Biorthogonal lapped transforms can be constructed using the formulation of Equation (9.256), if $\hat{\mathbf{C}}_3$ is chosen such that $\hat{\mathbf{C}}_3\hat{\mathbf{C}}_3 = \hat{\mathbf{C}}_3$, matrix $\hat{\mathbf{C}}_4$ is nonsingular and nonorthogonal, and the polyphase matrix $\mathbf{R}(z)$ is such that

$$\mathbf{R}(z) = \hat{\mathbf{C}}_4^{-1}[z^{-1}\hat{\mathbf{C}}_3 + (\mathbf{I} - \hat{\mathbf{C}}_3)]. \quad (9.259)$$

Example 9.12. Show the two-band lapped-transform structure that realizes the filter bank with lowpass analysis filter

$$H_0(z) = -z^{-3} + 3z^{-2} + 3z^{-1} - 1, \quad (9.260)$$

specifying the value of each coefficient in that structure. Determine the corresponding highpass analysis filter.

Solution

Since $M = 2$, the DCT matrix has the following form:

$$\begin{bmatrix} \mathbf{C}_e \\ \mathbf{C}_o \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ \cos(\pi/4) & \cos(3\pi/4) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad (9.261)$$

such that

$$\mathbf{C}_e = [1/\sqrt{2} \ 1/\sqrt{2}]; \quad \mathbf{C}_o = [1/\sqrt{2} \ -1/\sqrt{2}]. \quad (9.262)$$

Since the lapped-transform filter bank has two bands and the desired structure must have linear phase, then there is no nontrivial orthogonal solution for this particular case; that is, $\mathbf{E}^T(z^{-1}) \neq \mathbf{E}^{-1}(z)$.

One option is to look for biorthogonal solutions as given by Equations (9.256) and (9.259), whereby $\hat{\mathbf{C}}_4$ should be chosen as a general nonorthogonal 2×2 matrix. Another option is to use an orthogonal $\hat{\mathbf{C}}_4$ and use a nonorthogonal matrix \mathbf{P} in the input of Figure 9.45 in order to achieve perfect reconstruction. In this case, we can choose $\hat{\mathbf{C}}_3$ and $\hat{\mathbf{C}}_4$ according to Equations (9.257) and (9.258) respectively; that is:

$$\hat{\mathbf{C}}_3 = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (9.263)$$

$$\hat{\mathbf{C}}_4 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \quad (9.264)$$

such that

$$\hat{\mathbf{C}}_3\hat{\mathbf{C}}_3 = \frac{1}{4} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \hat{\mathbf{C}}_3. \quad (9.265)$$

In order to achieve a biorthogonal solution, matrix

$$\hat{\mathbf{P}} = \begin{bmatrix} \hat{p}_{00} & \hat{p}_{01} \\ \hat{p}_{10} & \hat{p}_{11} \end{bmatrix} \quad (9.266)$$

is placed before $\hat{\mathbf{C}}_4$ in the block diagram of the analysis filter, post-multiplying $\hat{\mathbf{C}}_4$ as follows:

$$\begin{aligned}
 \mathbf{E}(z) &= \left[\hat{\mathbf{C}}_3 + z^{-1}(\mathbf{I} - \hat{\mathbf{C}}_3) \right] \hat{\mathbf{C}}_4 \hat{\mathbf{P}} \\
 &= \left(\frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + z^{-1} \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right) \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \hat{p}_{00} & \hat{p}_{01} \\ \hat{p}_{10} & \hat{p}_{11} \end{bmatrix} \right) \\
 &= \frac{1}{2\sqrt{2}} \left(\begin{bmatrix} 0 & 2 \\ 0 & 2 \end{bmatrix} + z^{-1} \begin{bmatrix} 2 & 0 \\ -2 & 0 \end{bmatrix} \right) \begin{bmatrix} \hat{p}_{00} & \hat{p}_{01} \\ \hat{p}_{10} & \hat{p}_{11} \end{bmatrix} \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} z^{-1} & 1 \\ -z^{-1} & 1 \end{bmatrix} \begin{bmatrix} \hat{p}_{00} & \hat{p}_{01} \\ \hat{p}_{10} & \hat{p}_{11} \end{bmatrix} \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} \hat{p}_{00}z^{-1} + \hat{p}_{10} & \hat{p}_{01}z^{-1} + \hat{p}_{11} \\ -\hat{p}_{00}z^{-1} + \hat{p}_{10} & -\hat{p}_{01}z^{-1} + \hat{p}_{11} \end{bmatrix}. \tag{9.267}
 \end{aligned}$$

Since the polyphase components of $H_0(z)$ should be $E_{00}(z) = -1 + 3z^{-1}$ and $E_{01}(z) = 3 - z^{-1}$, the above equation implies that

$$\frac{1}{\sqrt{2}} \begin{bmatrix} \hat{p}_{00}z^{-1} + \hat{p}_{10} & \hat{p}_{01}z^{-1} + \hat{p}_{11} \\ -\hat{p}_{00}z^{-1} + \hat{p}_{10} & -\hat{p}_{01}z^{-1} + \hat{p}_{11} \end{bmatrix} = \begin{bmatrix} -1 + 3z^{-1} & 3 - z^{-1} \\ E_{10}(z) & E_{11}(z) \end{bmatrix}, \tag{9.268}$$

so that, by choosing $\hat{p}_{00} = 3\sqrt{2} = \hat{p}_{11}$ and $\hat{p}_{01} = -\sqrt{2} = \hat{p}_{10}$, it follows that $E_{10}(z) = -1 - 3z^{-1}$ and $E_{11}(z) = 3 + z^{-1}$. As a result, the transfer function of the highpass analysis filter is given by

$$H_1(z) = -1 + 3z^{-1} - 3z^{-2} + z^{-3}. \tag{9.269}$$

The resulting structure is depicted in Figure 9.46. \triangle

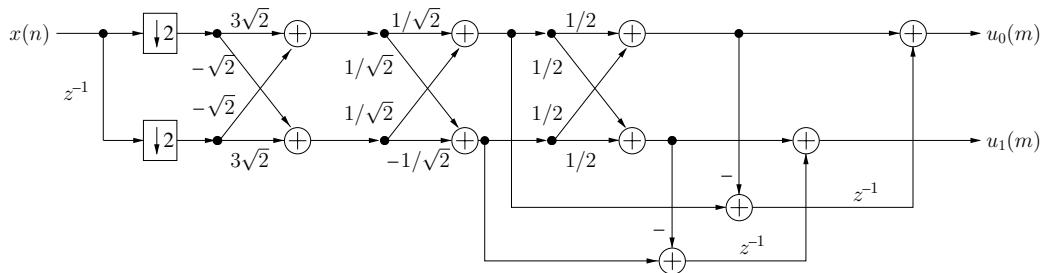


Fig. 9.46. Lapped transform implementation.

9.10.2 Generalized LOT

The formulation described in the previous subsection can be extended to obtain generalized lapped transforms in their orthogonal and biorthogonal forms. These transforms allow the overlapping of multiple blocks of length M . The approach described here follows closely the approach of De Queiroz *et al.* (1996).

The GenLOTs can also be constructed if the polyphase matrices are designed as follows:

$$\mathbf{E}(z) = \prod_{j=L}^1 \left\{ \begin{bmatrix} \mathbf{L}_{3,j} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{2,j} \end{bmatrix} [\hat{\mathbf{C}}_{3,j} + z^{-1}(\mathbf{I} - \hat{\mathbf{C}}_{3,j})] \right\} \hat{\mathbf{C}}_4 \quad (9.270)$$

$$\mathbf{R}(z) = \hat{\mathbf{C}}_4^{-1} \prod_{j=1}^L \left\{ [z^{-1}\hat{\mathbf{C}}_{3,j} + (\mathbf{I} - \hat{\mathbf{C}}_{3,j})] \begin{bmatrix} \mathbf{L}_{3,j}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{2,j}^{-1} \end{bmatrix} \right\}. \quad (9.271)$$

Biorthogonal GenLOTs are obtained if $\hat{\mathbf{C}}_{3,j}$ is chosen such that $\hat{\mathbf{C}}_{3,j}\hat{\mathbf{C}}_{3,j}^\top = \hat{\mathbf{C}}_{3,j}$. Perfect reconstruction conditions follow directly from the definitions in Equations (9.270) and (9.271).

In order to obtain an orthogonal GenLOT it is further required that $\hat{\mathbf{C}}_4^\top \hat{\mathbf{C}}_4 = \mathbf{I}$, $\mathbf{L}_{3,j}^\top \mathbf{L}_{3,j} = \mathbf{I}$, and $\mathbf{L}_{2,j}^\top \mathbf{L}_{2,j} = \mathbf{I}$. If we choose $\hat{\mathbf{C}}_3$ and $\hat{\mathbf{C}}_4$ as in Equations (9.257) and (9.258), then we have a valid GenLOT possessing a fast algorithm of the following form:

$$\begin{aligned} \mathbf{E}(z) &= \prod_{j=L}^1 \left(\frac{1}{2} \begin{bmatrix} \mathbf{L}_{3,j} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{2,j} \end{bmatrix} \begin{bmatrix} \mathbf{I} + z^{-1}\mathbf{I} & \mathbf{I} - z^{-1}\mathbf{I} \\ \mathbf{I} - z^{-1}\mathbf{I} & \mathbf{I} + z^{-1}\mathbf{I} \end{bmatrix} \right) \hat{\mathbf{C}}_4 \\ &= \prod_{j=L}^1 \left(\frac{1}{2} \begin{bmatrix} \mathbf{L}_{3,j} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{2,j} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & z^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} \right) \hat{\mathbf{C}}_4 \\ &= \prod_{j=L}^1 (\mathbf{K}_j) \hat{\mathbf{C}}_4. \end{aligned} \quad (9.272)$$

If, in order to have a fast algorithm, we fix $\hat{\mathbf{C}}_4$ as in Equation (9.258), then the degrees of freedom to design the filter bank are the choices of matrices $\mathbf{L}_{3,j}$ and $\mathbf{L}_{2,j}$, which are constrained to real and orthogonal matrices, to allow a valid GenLOT. The effectiveness in terms of computational complexity is highly dependent on how fast we can compute these matrices.

Equation (9.272) suggests the structure of Figure 9.47 for the implementation of the GenLOT filter bank. This structure consists of the implementation of the matrix \mathbf{L}_0 in cascade with a set of similar building blocks denoted \mathbf{K}_j . The structure for the implementation of each \mathbf{K}_j is depicted in Figure 9.48.

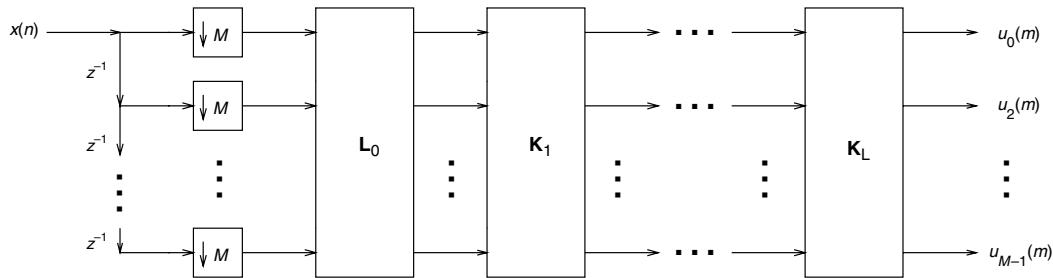


Fig. 9.47. Implementation of the GenLOT.

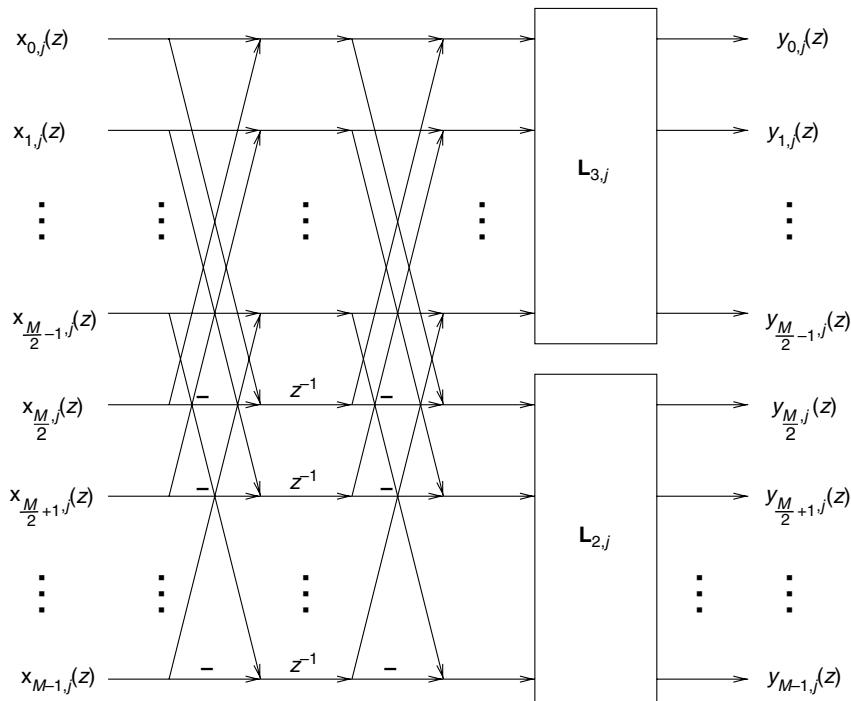


Fig. 9.48. Implementation of the building blocks K_j of the GenLOT.

In the formulation presented here, the subfilter lengths are constrained to be multiples of the number of sub-bands. A more general formulation for the design of sub-filters with arbitrary lengths is proposed in Tran *et al.* (2000).

The impulse and normalized magnitude responses of an orthogonal length-40 GenLOT with $M = 10$ sub-bands are shown in Figures 9.49 and 9.50 respectively. Table 9.3 lists the coefficients of the first analysis filter. Again, because of the linear-phase property, only half of the coefficients are shown.

Table 9.3.GenLOT analysis filter coefficients $h(0)$ to $h(19)$: band 0.

$h(0) = -0.000\,734$	$h(7) = 0.006\,070$	$h(14) = -0.096\,310$
$h(1) = -0.001\,258$	$h(8) = 0.004\,306$	$h(15) = -0.191\,556$
$h(2) = 0.000\,765$	$h(9) = 0.003\,363$	$h(16) = -0.268\,528$
$h(3) = 0.000\,017$	$h(10) = 0.053\,932$	$h(17) = -0.318\,912$
$h(4) = 0.000\,543$	$h(11) = 0.030\,540$	$h(18) = -0.354\,847$
$h(5) = 0.005\,950$	$h(12) = -0.013\,838$	$h(19) = -0.383\,546$
$h(6) = 0.000\,566$	$h(13) = -0.055\,293$	

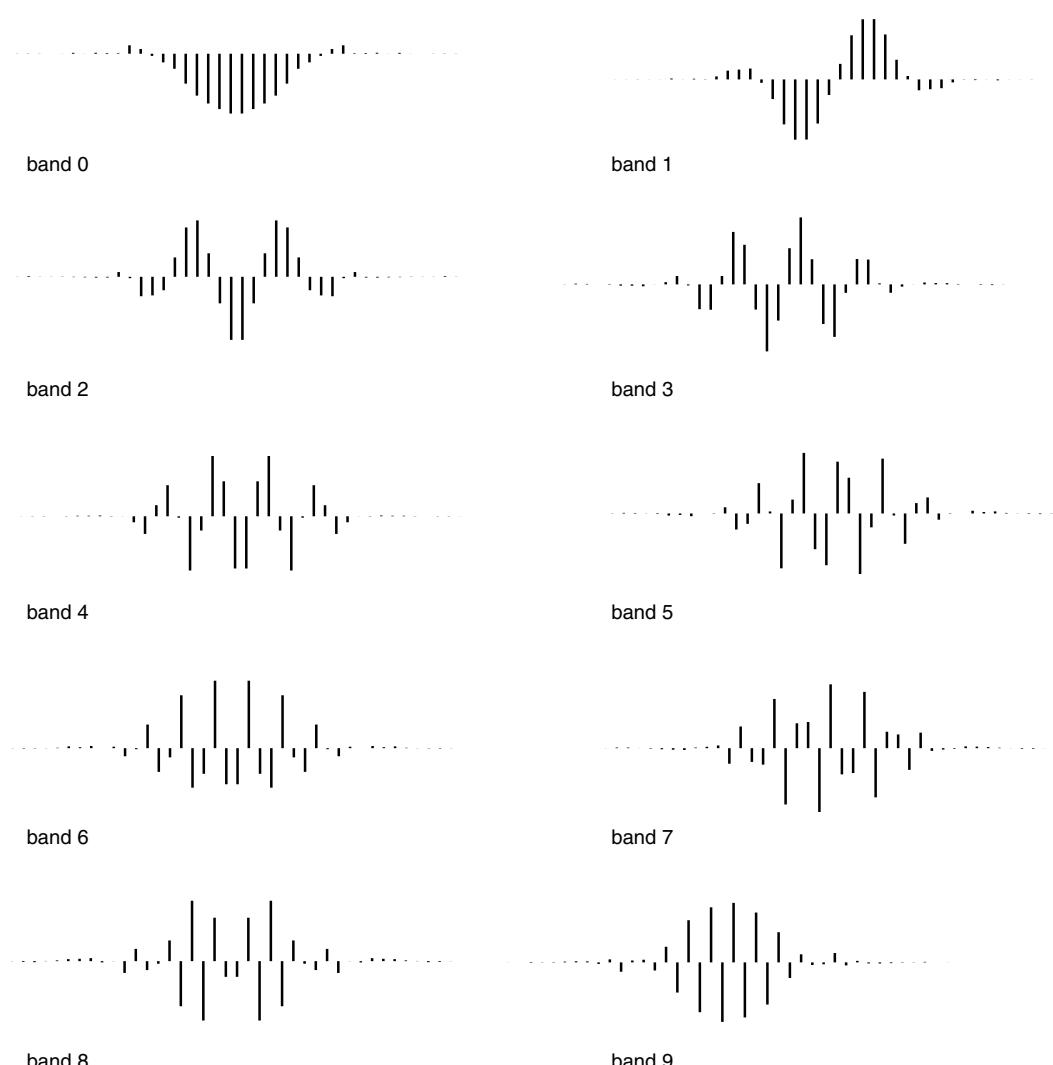


Fig. 9.49.

Impulse responses of the analysis filters of the 10-band GenLOT of length 40.

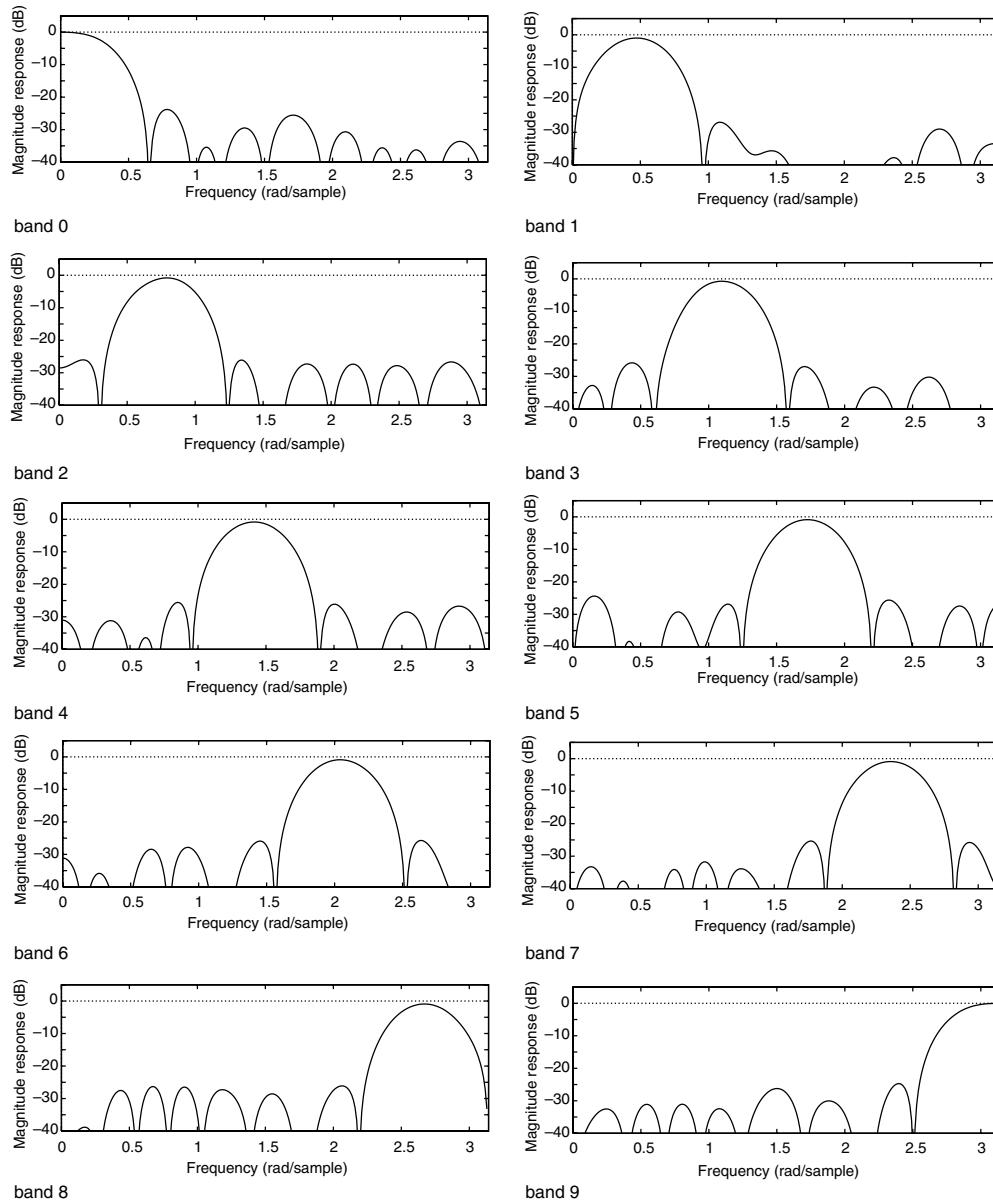


Fig. 9.50. Magnitude responses of the analysis filters of the 10-band GenLOT of length 40.

Figures 9.51–9.54 characterize a length-32 linear-phase lapped biorthogonal transform with $M = 8$ sub-bands. In this case, the coefficients of the first analysis and synthesis filters are listed in Table 9.4. Figures 9.51 and 9.52 characterize the analysis filters in the time and frequency domains respectively. Similar descriptions for the synthesis filters are shown in Figures 9.53 and 9.54.

Table 9.4.

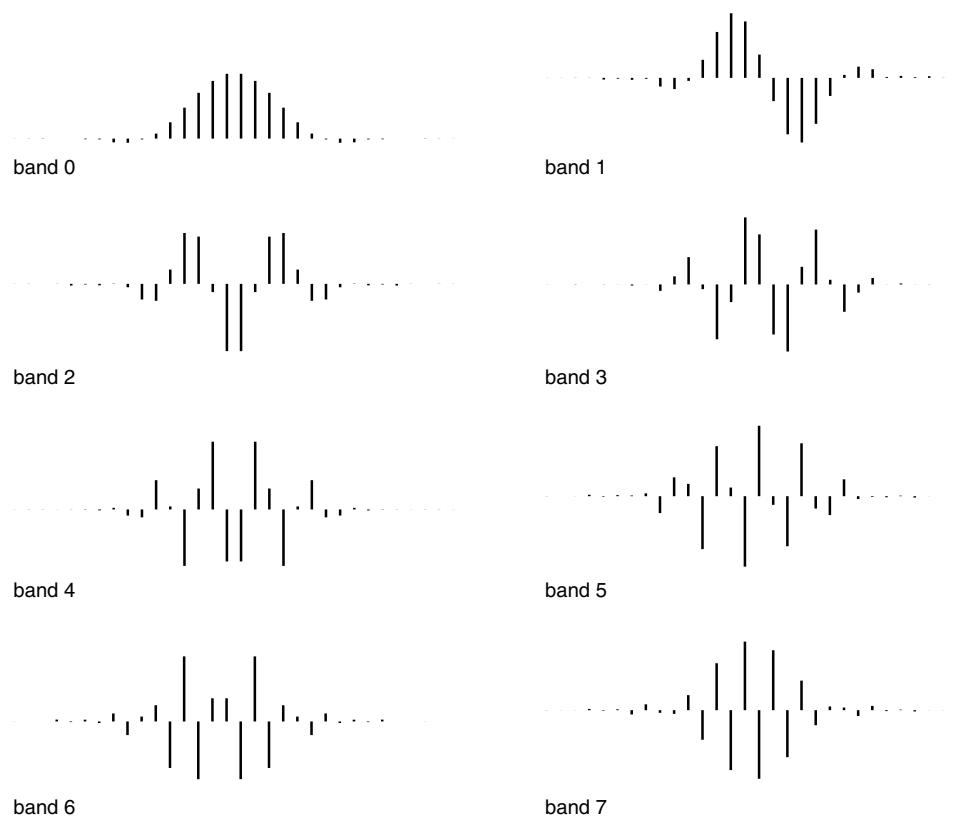
Biorthogonal lapped transform analysis and synthesis filter coefficients $h(0)$ to $h(15)$: band 0.

Analysis filter

$h(0) = 0.000\,541$	$h(6) = -0.005\,038$	$h(11) = 0.104\,544$
$h(1) = -0.000\,688$	$h(7) = -0.023\,957$	$h(12) = 0.198\,344$
$h(2) = -0.001\,211$	$h(8) = -0.026\,804$	$h(13) = 0.293\,486$
$h(3) = -0.000\,078$	$h(9) = -0.005\,171$	$h(14) = 0.369\,851$
$h(4) = -0.000\,030$	$h(10) = 0.032\,177$	$h(15) = 0.415\,475$
$h(5) = -0.003\,864$		

Synthesis filter

$h(0) = 0.000\,329$	$h(6) = -0.011\,460$	$h(11) = 0.125\,759$
$h(1) = -0.000\,320$	$h(7) = -0.027\,239$	$h(12) = 0.255\,542$
$h(2) = -0.001\,866$	$h(8) = -0.036\,142$	$h(13) = 0.359\,832$
$h(3) = -0.000\,146$	$h(9) = -0.023\,722$	$h(14) = 0.408\,991$
$h(4) = -0.004\,627$	$h(10) = 0.019\,328$	$h(15) = 0.425\,663$
$h(5) = -0.004\,590$		

**Fig. 9.51.**

Impulse responses of the analysis filters of the eight-band lapped biorthogonal transform filter bank of length 32.

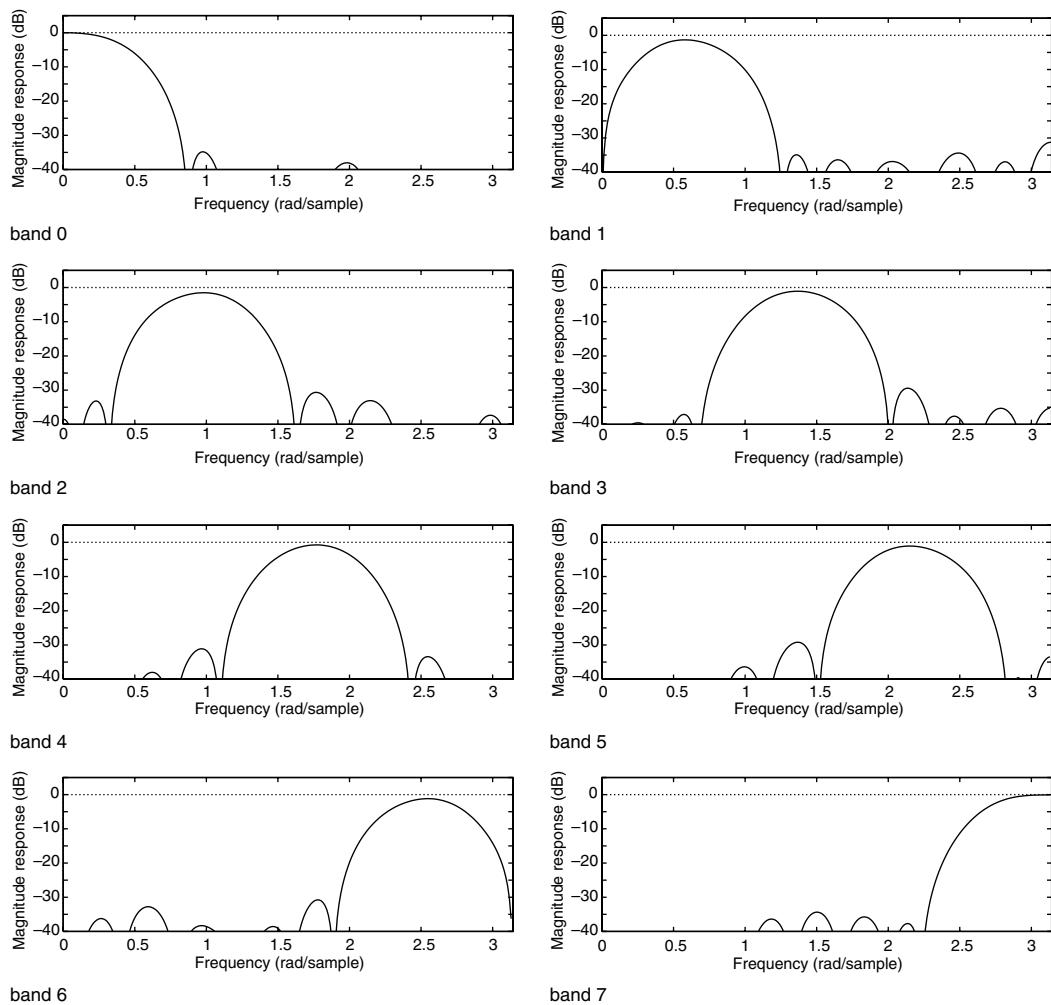


Fig. 9.52. Magnitude responses of the analysis filters of the eight-band lapped biorthogonal transform filter bank of length 32.

9.11 Do-it-yourself: filter banks

Experiment 9.1

Here we will design a third-order CQF filter bank. We start by designing the product filter $P(z)$ using a Hilbert transformer as in Equation (9.170):

$$P(z) = \frac{1}{2} \left(1 + \frac{\delta_{hb}}{2} - jH_h(-jz) \right). \quad (9.273)$$

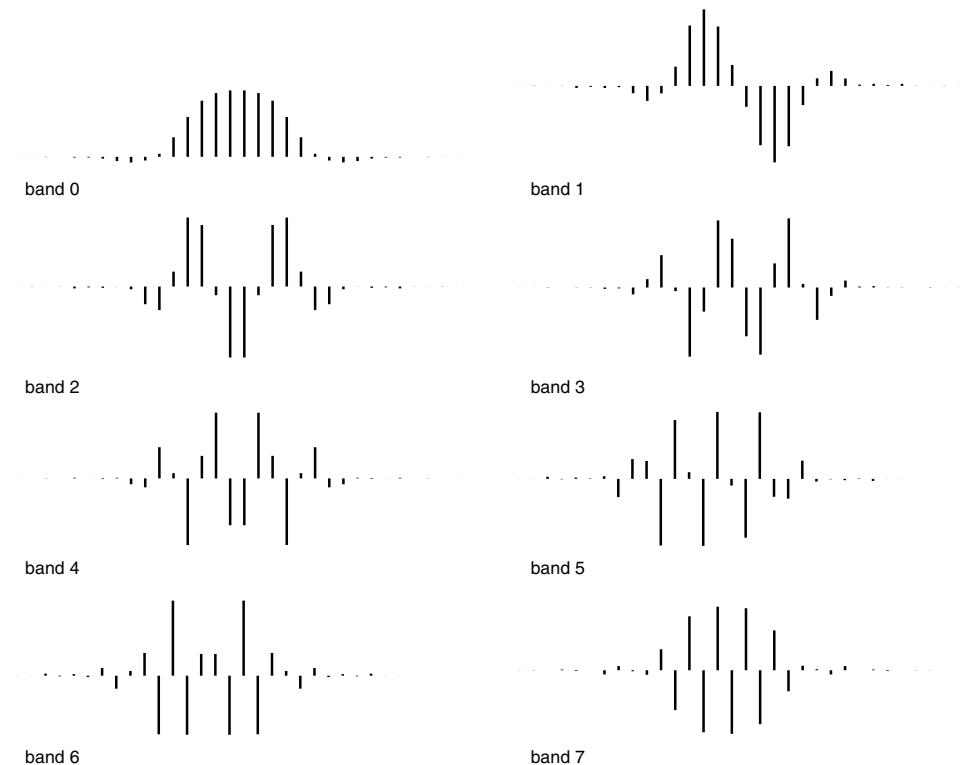


Fig. 9.53. Impulse responses of the synthesis filters of the eight-band lapped biorthogonal transform filter bank of length 32.

From Equation (5.18), the impulse response of an ideal Hilbert transformer is

$$h(n) = \begin{cases} 0, & \text{for } n = 0 \\ \frac{1}{\pi n} [1 - (-1)^n], & \text{for } n \neq 0. \end{cases} \quad (9.274)$$

Since the CQF filter bank should have order 3, then the product filter $P(z)$ should have order 6, and so should the Hilbert transformer. Applying a rectangular window to the ideal impulse response, the z transform of a sixth-order Hilbert transformer is

$$H_h(z) = \frac{2}{\pi} \left(-\frac{1}{3} z^3 - z + z^{-1} + \frac{1}{3} z^{-3} \right). \quad (9.275)$$

We can plot its magnitude response using the MATLAB commands

```
hh = (2/pi)*[-1/3 0 -1 0 1 0 1/3];
[Hh,w] = freqz(hh);
plot(w,abs(Hh));
```

which yield Figure 9.55.

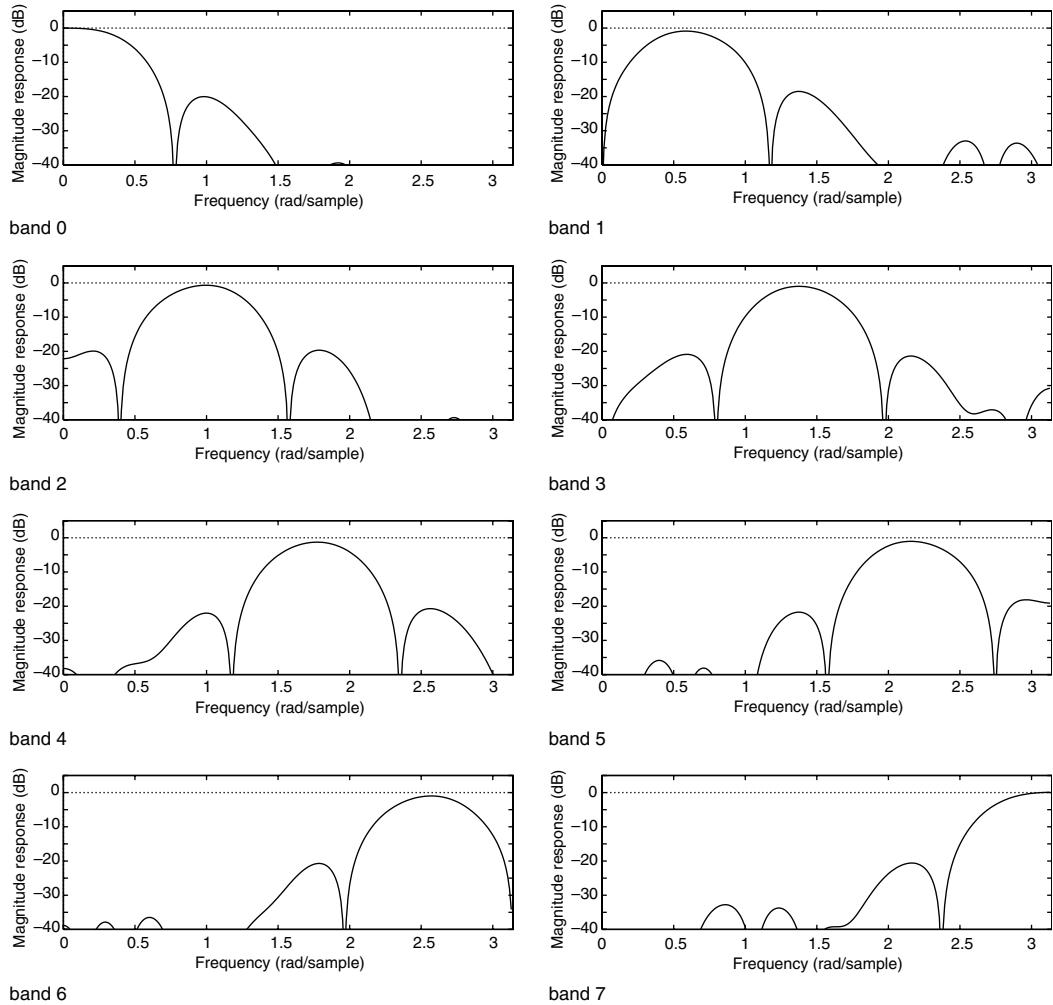


Fig. 9.54. Magnitude responses of the synthesis filters of the eight-band lapped biorthogonal transform filter bank of length 32.

We observe from Figure 9.55 that the magnitude response of the Hilbert transformer has two maxima. We can compute them in MATLAB using the command

```
maxHh = max(abs(Hh));
freqmax = w(find(abs(Hh)==maxHh))/pi;
```

which gives for the maxima the value $\text{maxHh} = 1.2004$ at the frequencies $\pi/4$ and $3\pi/4$. Alternatively, we can compute them analytically from $H_h(e^{j\omega})$ as

$$H_h(e^{j\omega}) = \frac{2}{\pi} \left(-\frac{1}{3} e^{j3\omega} - e^{j\omega} + e^{-j\omega} + \frac{1}{3} e^{-j3\omega} \right) = -\frac{4j}{\pi} \left(\frac{1}{3} \sin 3\omega + \sin \omega \right). \quad (9.276)$$

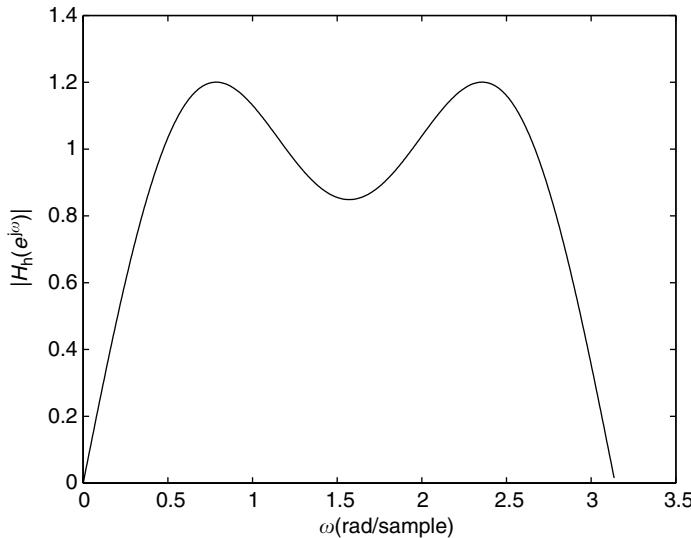


Fig. 9.55. Magnitude response of the order 6 Hilbert transformer from Experiment 9.1.

At the maxima, we should have

$$\begin{aligned}
 \frac{dH_h(e^{j\omega})}{d\omega} &= -\frac{4j}{\pi} (\cos 3\omega + \cos \omega) \\
 &= -\frac{4j}{\pi} (4\cos^3 \omega - 3\cos \omega + \cos \omega) \\
 &= -\frac{4j}{\pi} \cos \omega (4\cos^2 \omega - 2) \\
 &= 0
 \end{aligned} \tag{9.277}$$

and then

$$\begin{cases} \cos \omega = 0 \\ \cos \omega = \pm \frac{\sqrt{2}}{2} \end{cases} \Rightarrow \begin{cases} \omega = \frac{\pi}{2} + k\pi, & k \in \mathbb{Z} \\ \omega = \pm \frac{\pi}{4} + k\pi, & k \in \mathbb{Z}. \end{cases} \tag{9.278}$$

Since

$$\left. \begin{aligned} H_h(e^{j\pi/2}) &= -\frac{4j}{\pi} \left(\frac{1}{3} \sin \frac{3\pi}{2} + \sin \frac{\pi}{2} \right) = -\frac{8j}{3\pi} \\ H_h(e^{j\pi/4}) &= -\frac{4j}{\pi} \left(\frac{1}{3} \sin \frac{3\pi}{4} + \sin \frac{\pi}{4} \right) = -\frac{8\sqrt{2}j}{3\pi} \\ H_h(e^{j3\pi/4}) &= -\frac{4j}{\pi} \left(\frac{1}{3} \sin \frac{9\pi}{4} + \sin \frac{3\pi}{4} \right) = -\frac{8\sqrt{2}j}{3\pi} \end{aligned} \right\} \tag{9.279}$$

we thus confirm that the maxima of $|H_h(e^{j\omega})|$ occur at $\omega = \pi/4$ and $\omega = 3\pi/4$, having the value of $8\sqrt{2}/3\pi \approx 1.2004$.

Therefore, since $P(e^{j\omega})$ should be nonnegative, we have that the value of $\delta_{hb}/2$ in Equation (9.273) should be such that

$$1 + \frac{\delta_{hb}}{2} = \frac{8\sqrt{2}}{3\pi}. \quad (9.280)$$

With this value, the product filter $P(z)$ becomes

$$\begin{aligned} P(z) &= \frac{1}{2} \left(\frac{8\sqrt{2}}{3\pi} - jH_h(-jz) \right) \\ &= \frac{4\sqrt{2}}{3\pi} - j\frac{1}{\pi} \left[-\frac{1}{3}(-jz)^3 - (-jz) + (-jz)^{-1} + \frac{1}{3}(-jz)^{-3} \right] \\ &= \frac{1}{3\pi} \left(-z^3 + 3z + 4\sqrt{2} + 3z^{-1} - z^{-3} \right). \end{aligned} \quad (9.281)$$

However, for this product filter $P(z)$, we have that

$$P(z) + P(-z) = \frac{8\sqrt{2}}{3\pi} \neq 2. \quad (9.282)$$

Therefore, since for the CQF filter bank we must have, from Equation (9.166), that $P(z) + P(-z) = 2$, then the product filter must be normalized by $3\pi/4\sqrt{2}$, yielding

$$P(z) = \frac{1}{4\sqrt{2}} \left(-z^3 + 3z + 4\sqrt{2} + 3z^{-1} - z^{-3} \right). \quad (9.283)$$

The corresponding frequency response $P(e^{j\omega})$ can be plotted in MATLAB as follows (note that, since $P(z)$ is symmetric, then $P(e^{j\omega})$ is real):

```
p = (1/4*sqrt(2))*[-1 0 3 4*sqrt(2) 3 0 -1];
[P,w] = freqz(p);
plot(w,abs(P));
```

as shown in Figure 9.56.

Now we must factorize $P(z)$ into $H_0(z)H_0(z^{-1})$. One way to do that is to find the zeros of $P(z)$ using the function `tf2zpk` from MATLAB:

```
p = (1/(4*sqrt(2)))*[-1 0 3 4*sqrt(2) 3 0 -1];
[zi pi k] = tf2zpk(p,1)
```

which indicate that we have a double zero at $e^{j3\pi/4}$, another double zero at $e^{-j3\pi/4}$, a zero at $(\sqrt{2} + 1)$ and a zero at $(\sqrt{2} - 1)$, as plotted in Figure 9.57.

Hence, $P(z)$ can be expressed as

$$P(z) = -\frac{1}{4\sqrt{2}}z^{-3}(z - e^{j3\pi/4})^2(z - e^{-j3\pi/4})^2(z - \sqrt{2} - 1)(z - \sqrt{2} + 1). \quad (9.284)$$

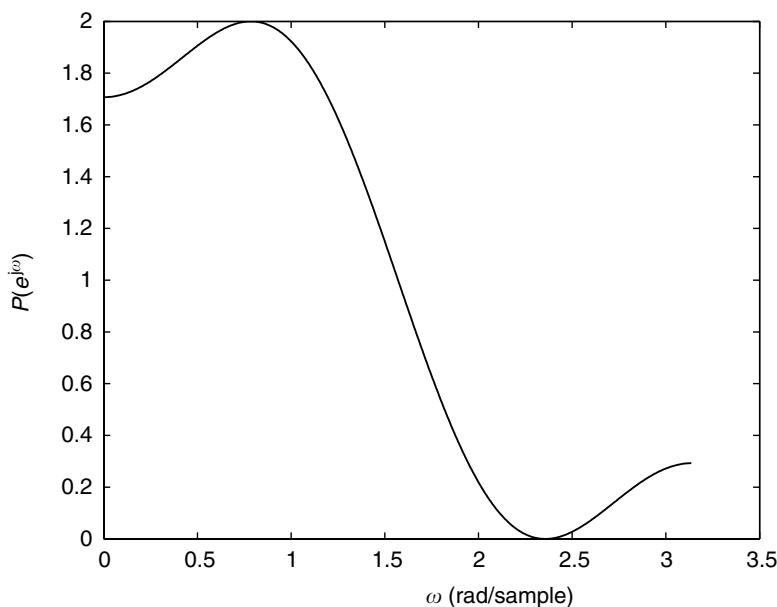


Fig. 9.56. Plot of the frequency response of product filter $P(e^{j\omega})$ from Experiment 9.1.

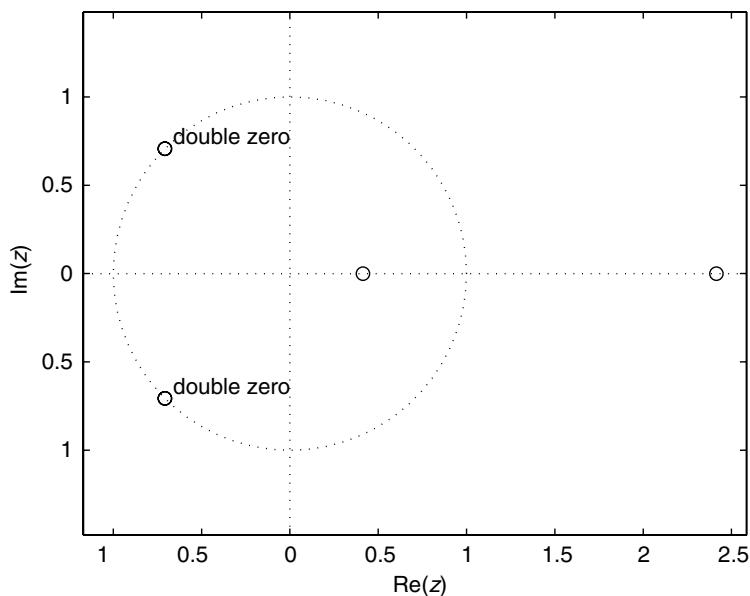


Fig. 9.57. Zeros of the product filter $P(z)$ from Experiment 9.1.

Grouping the complex conjugate poles together, we have

$$\begin{aligned}
 P(z) &= -\frac{1}{4\sqrt{2}}z^{-3}(z^2 + \sqrt{2}z + 1)^2(z - \sqrt{2} - 1)(z - \sqrt{2} + 1) \\
 &= -\frac{1}{4\sqrt{2}}(1 + \sqrt{2}z^{-1} + z^{-2})(z^2 + \sqrt{2}z + 1)[1 - (\sqrt{2} + 1)z^{-1}](z - \sqrt{2} + 1) \\
 &= -\frac{1 - \sqrt{2}}{4\sqrt{2}}(1 + \sqrt{2}z^{-1} + z^{-2})(1 + \sqrt{2}z + z^2)[1 - (\sqrt{2} + 1)z^{-1}] \left(1 + \frac{z}{1 - \sqrt{2}}\right) \\
 &= \frac{\sqrt{2} - 1}{4\sqrt{2}}(1 + \sqrt{2}z^{-1} + z^{-2})(1 + \sqrt{2}z + z^2)[1 - (\sqrt{2} + 1)z^{-1}][1 - (1 + \sqrt{2})z].
 \end{aligned} \tag{9.285}$$

Since $P(z) = H_0(z)H_0(z^{-1})$, we can choose

$$\begin{aligned}
 H_0(z) &= \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}(1 + \sqrt{2}z^{-1} + z^{-2})[1 - (\sqrt{2} + 1)z^{-1}] \\
 &= \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}\left[1 - z^{-1} - (1 + \sqrt{2})z^{-2} - (1 + \sqrt{2})z^{-3}\right].
 \end{aligned} \tag{9.286}$$

Supposing an overall delay of $(2\Delta + 1) = N = 3$ samples, the highpass analysis, lowpass synthesis, and highpass synthesis filters, from Equations (9.161), (9.167), and (9.168), are

$$\left. \begin{array}{l} H_1(z) = -z^{-3}H_0(-z^{-1}) \\ G_0(z) = z^{-3}H_0(z^{-1}) \\ G_1(z) = -H_0(-z) \end{array} \right\}, \tag{9.287}$$

which correspond to a CQF filter bank described by

$$\left. \begin{array}{l} H_0(z) = \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}\left[1 - z^{-1} - (1 + \sqrt{2})z^{-2} - (1 + \sqrt{2})z^{-3}\right] \\ H_1(z) = \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}\left[-(1 + \sqrt{2}) + (1 + \sqrt{2})z^{-1} - z^{-2} - z^{-3}\right] \\ G_0(z) = \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}\left[-(1 + \sqrt{2}) - (1 + \sqrt{2})z^{-1} - z^{-2} + z^{-3}\right] \\ G_1(z) = \sqrt{\frac{\sqrt{2} - 1}{4\sqrt{2}}}\left[-1 - z^{-1} + (1 + \sqrt{2})z^{-2} - (1 + \sqrt{2})z^{-3}\right] \end{array} \right\}. \tag{9.288}$$

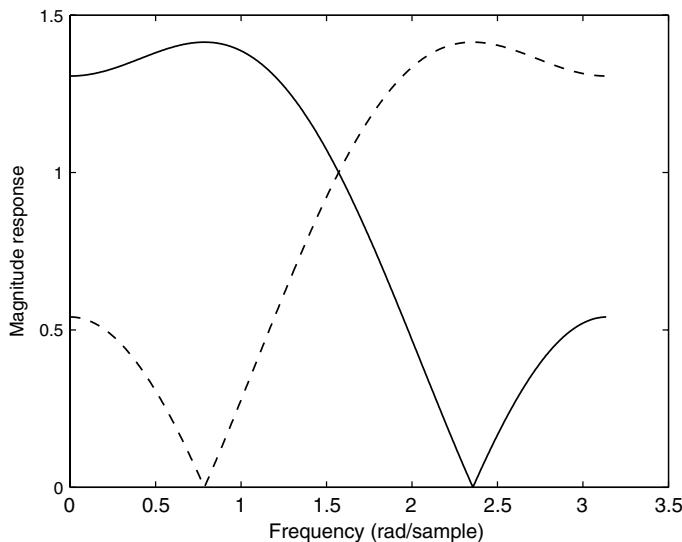


Fig. 9.58. Magnitude responses of the lowpass and highpass filters of the analysis and synthesis banks from Experiment 9.1: $H_0(z)$ (solid line); $H_1(z)$ (dashed line).

The magnitude responses of the lowpass and highpass filters for the analysis and synthesis banks can be plotted using the MATLAB commands below and are depicted in Figure 9.58.

```
c = sqrt((sqrt(2)-1)/(4*sqrt(2)));
h0 = c*[1 -1 -(1+sqrt(2)) -(1+sqrt(2))];
[H0,w] = freqz(h0);
plot(w,abs(H0)); hold;
h1 = c*[-(1+sqrt(2)) (1+sqrt(2)) -1 -1];
[H1,w] = freqz(h1);
plot(w,abs(H1));
```

Experiment 9.2

Design a filter bank with $M = 10$ sub-bands using the LOT.

Since the LOT to be designed has $M = 10$, as in Example 9.11, the matrix \mathbf{L}_0 can be built using Equation (9.242) with the matrices \mathbf{C}_e and \mathbf{C}_o defined in Equations (9.251) and (9.252) respectively. A MATLAB script to compute \mathbf{L}_0 is

```
C = dctmtx(10);
Ce = C([1:2:9],:); Co = C([2:2:10],:);
I = eye(10); J = fliplr(I);
L0 = 0.5*[Ce-Co (Ce-Co)*J; Ce-Co -(Ce-Co)*J];
```

In order to complete the design, we have to compute $\mathbf{L}_{\text{LOT}} = \mathbf{L}_1 \mathbf{L}_0$. Hence, we must find the matrix \mathbf{L}_1 , which is an orthogonal matrix as given in Equation (9.245), where \mathbf{L}_2

is determined by $((M/2) - 1) = 4$ rotation angles θ_i , for $i = 0, 1, 2, 3$ as follows:

$$\mathbf{L}_2 = \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1 \mathbf{T}_0, \quad (9.289)$$

where

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{I}_i & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}(\theta_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{3-i} \end{bmatrix}, \quad i = 0, 1, 2, 3 \quad (9.290)$$

and

$$\mathbf{Y}(\theta_i) = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix}. \quad (9.291)$$

The MATLAB code to compute \mathbf{L}_{LOT} given \mathbf{L}_0 and the rotation angles $\theta_0, \theta_1, \theta_2$, and θ_3 in a vector \mathbf{t} is

```
function y = LOT(t,L0)
Y = zeros(2,2,3); T = zeros(5,5,3); L2 = eye(5);
for i=1:4
    Y(:,:,i) = [cos(t(i)), -sin(t(i)); sin(t(i)), cos(t(i))];
    T(:,:,i) = [eye(i-1), zeros(i-1,2), zeros(i-1,4-i);
                zeros(2,i-1), Y(:,:,i), zeros(2,4-i);
                zeros(4-i,i-1), zeros(4-i,2), eye(4-i)];
    L2 = T(:,:,i)*L2;
end;
L1 = [eye(5), zeros(5,5); zeros(5,5), L2];
y = L1*L0;
```

Therefore, these four rotation angles should be such that a certain optimization criterion is satisfied. It can be, for example, the summation of the energy in the stopband of all bands. In this experiment we choose them such that the maximum energy compaction on the transform coefficients is obtained; that is, most energy concentrates on the smallest number of transform coefficients. In order to do so, it is assumed that the input signal is an autoregressive (AR) process generated by filtering a white noise with a first-order lowpass digital filter with a single pole at $z = 0.9$.

From Exercise 1.31, we have that if a WSS process $\{X\}$ with autocorrelation $R_X(n)$ is input to a linear system with impulse response $h(n)$, then the autocorrelation $R_Y(n)$ at its output is

$$R_Y(n) = \sum_{k=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} R_X(n-k)h(k+r)h(r). \quad (9.292)$$

Therefore, since the PSD of white noise is $R_X(n) = \sigma^2 \delta(n)$ and the impulse response of a stable digital filter with a pole at $z = \rho$ is $h(n) = \rho^n u(n)$, we have that the PSD of an AR

process $\{W\}$ with a pole at $z = \rho$ is

$$\begin{aligned}
 R_W(n) &= \sum_{k=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} \sigma^2 \delta(n-k) h(k+r) h(r) \\
 &= \sigma^2 \sum_{r=-\infty}^{\infty} h(n+r) h(r) \\
 &= \sigma^2 \sum_{r=-\infty}^{\infty} \rho^{n+r} u(n+r) \rho^r u(r) \\
 &= \sigma^2 \rho^n \sum_{r=-\infty}^{\infty} \rho^{2r} u(n+r) u(r) \\
 &= \sigma^2 \rho^n \sum_{r=\max\{-n,0\}}^{\infty} \rho^{2r} \\
 &= \sigma^2 \frac{\rho^{n+2 \max\{-n,0\}}}{1 - \rho^2} \\
 &= \frac{\sigma^2}{1 - \rho^2} \rho^{|n|}.
 \end{aligned} \tag{9.293}$$

Therefore, the autocorrelation of the output of the k th analysis filter $h_k(n)$, when an AR process with a pole at $z = \rho$ is input to it, from Equations (9.292) and (9.293), is

$$\begin{aligned}
 R_{Y_k}(n) &= \sum_{l=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} R_W(n-l) h_k(l+r) h_k(r) \\
 &= \sum_{l=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} \frac{\sigma^2}{1 - \rho^2} \rho^{|n-l|} h_k(l+r) h_k(r) \\
 &= \frac{\sigma^2}{1 - \rho^2} \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} \rho^{|n-s+r|} h_k(s) h_k(r)
 \end{aligned} \tag{9.294}$$

and, consequently, its variance is

$$\sigma_{Y_k}^2 = R_{Y_k}(0) = \frac{\sigma^2}{1 - \rho^2} \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} \rho^{|s-r|} h_k(s) h_k(r). \tag{9.295}$$

For a stationary process, its variance is equal to the variance of its decimated output. Since, in this case, the analysis filters have length $M = 20$ and $\rho = 0.9$, we have that the variance of the decimated output of the k th analysis filter is

$$\sigma_k^2 = \frac{\sigma^2}{1 - \rho^2} \sum_{s=0}^{19} \sum_{r=0}^{19} 0.9^{|s-r|} h_k(s) h_k(r). \tag{9.296}$$

It can be shown that a good measure of the energy concentration at the output of an M -band orthogonal filter bank is given by the coding gain (Jayant & Noll, 1984)

$$G = \frac{\frac{1}{M} \sum_{k=0}^{M-1} \sigma_k^2}{\prod_{k=0}^{M-1} (\sigma_k^2)^{1/M}}. \quad (9.297)$$

A MATLAB code for computing the coding gain of a given LOT matrix \mathbf{L}_{LOT} when the AR pole is equal to rho is

```
function y = CG(Llot,rho)
sigma = zeros(1,10);
for k=1:10,
    for s=1:20,
        for r=1:20,
            sigma(k) = sigma(k) + rho^(abs(s-r))
            *Llot(k,s)*Llot(k,r);
        end;
    end;
end;
y = (sum(sigma.^2/10)/(prod(sigma.^2)^(1/10));
```

Now, we should find the vector $\mathbf{t} = [\theta_0, \theta_1, \theta_2, \theta_3]$ that maximizes the coding gain; that is, provides the maximum value of $\text{CG}(\text{LOT}(\mathbf{t}, \mathbf{L}_0), 0.9)$. This can be done by using either the optimization routine `fminsearch` or `fminunc` from the MATLAB Optimization Toolbox. The MATLAB code can be as follows:

```
function [coding_gain,tf] = LOTtest(t)
tf = fminsearch(@LotCG,t);
coding_gain = 1/LotCG(tf);
```

Running the routines above for $\text{rho} = 0.9$ with the initial point $\mathbf{t} = [0 \ 0 \ 0 \ 0]$, we get

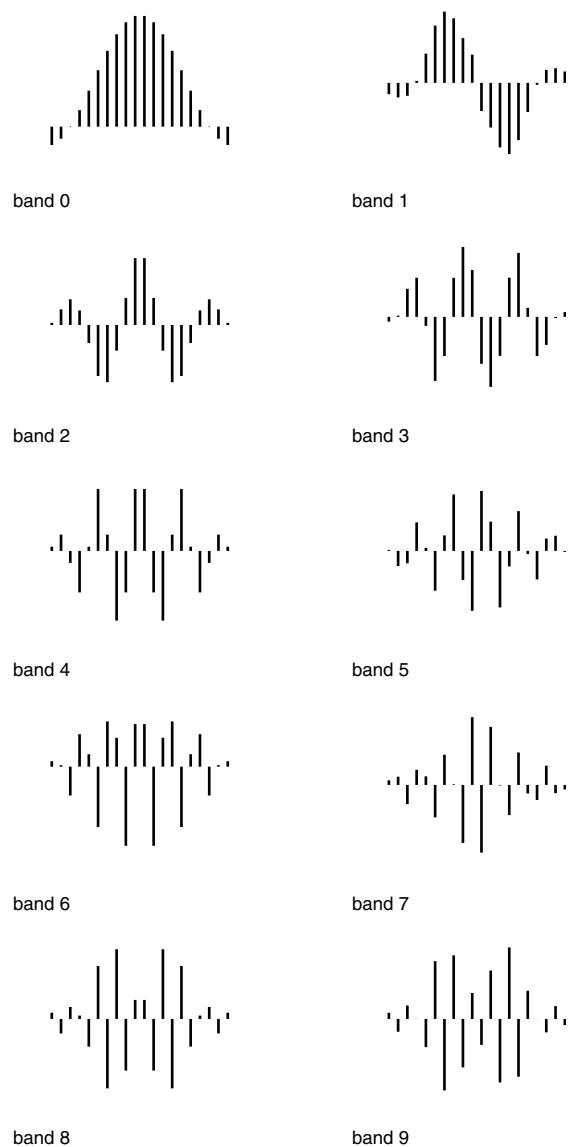
```
coding_gain = 133.7581
tf = [0.4648 0.5926 -1.1191 -0.0912]
```

The resulting analysis filters' impulse responses are shown in Figure 9.59. The coefficients of the first analysis filter are listed in Table 9.5. Observe that the LOT has linear phase, and thus only the first half of the coefficients are shown.

Figure 9.60 depicts the normalized magnitude responses of the analysis filters for a LOT filter bank with $M = 10$ sub-bands. The impulse responses and magnitude responses of the filters are shown in increasing order of peak frequency. \triangle

Table 9.5.LOT analysis filter coefficients $h(0)$ to $h(9)$: band 0.

$h(0) = -0.0627$	$h(4) = 0.1231$	$h(7) = 0.3162$
$h(1) = 0.0411$	$h(5) = 0.1931$	$h(8) = 0.3573$
$h(2) = 0.0000$	$h(6) = 0.2596$	$h(9) = 0.3790$
$h(3) = 0.0566$		

**Fig. 9.59.**

Impulse responses of the filters of a 10-band LOT.

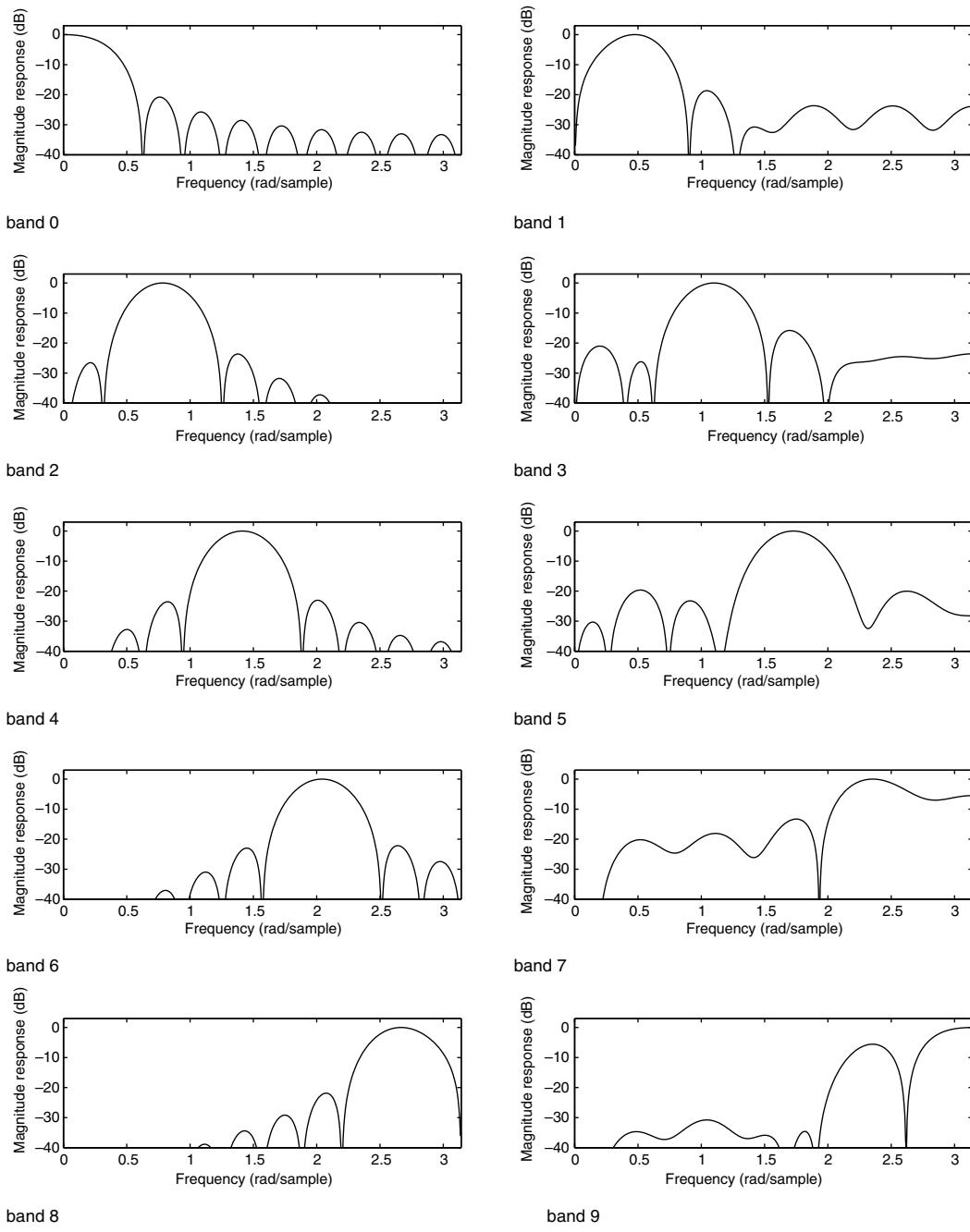


Fig. 9.60. Magnitude responses of the filters of a 10-band LOT.

9.12 Filter banks with MATLAB

The functions described below are from the MATLAB Wavelet Toolbox.

- **dyaddown:** Dyadic downsampling.

Input parameters:

- the input vector or matrix x ;
- the integer evenodd ; if evenodd is even, then the output corresponds to the even samples of the input; if evenodd is odd, then the output corresponds to the odd samples of the input;
- the parameter ' type ', when x is a matrix; if ' $\text{type} = 'c'$, only the rows are downsampled; if ' $\text{type} = 'r'$, only the columns are downsampled; if ' $\text{type} = 'm'$, both rows and columns are downsampled.

Output parameter: the vector or matrix y containing the downsampled signal.

Example:

```
x=[1.0 2.7 2.4 5.0]; evenodd=1;
y=dyaddown(x, evenodd);
```

- **dyadup:** Dyadic upsampling.

For a complete list of input and output parameters, see command `dyadown`.

Example:

```
x=[0.97 -2.1 ; 3.7 -0.03]; evenodd=0;
y=dyadup(x, 'm', evenodd);
```

- **qmff:** Generates a quadrature mirror filter of the input.

Input parameters:

- The input vector or matrix x .
- The integer p . If p is even, then the output corresponds to x in reversed order with the signs changed for the even index entries. If p is odd, then the output corresponds to x in reversed order, with the signs of the odd index entries changed.

Output parameter: the vector y containing the output filter coefficients.

Example:

```
y=qmf(x, 0);
```

9.13 Summary

In this chapter we have discussed the concept of filter banks. They are used in several digital signal processing applications; for instance, in signal analysis, coding, and transmission. Several examples of filter banks were considered, including the QMF, CQF, cosine-modulated, and biorthogonal perfect reconstruction filter banks.

Several tools for signal analysis were discussed in the framework of filter banks, with special emphasis given to block transforms and LOT. Also, functions from the MATLAB Wavelet Toolbox were described.

This chapter includes extensive coverage of a very broad topic where we tried to select techniques widely used in practice. It is expected that its study will equip the reader to follow the more advanced literature in the subject without difficulty.

9.14 Exercises

- 9.1 Show that, for an M -band filter bank whose analysis and synthesis filters are ideal filters, perfect reconstruction demands that their frequency responses are

$$\left. \begin{aligned} H_0(e^{j\omega}) &= M, & |\omega| &\in [0, \pi/M] \\ H_{2k-1}(e^{j\omega}) &= \begin{cases} M, & |\omega| \in ((2k-1)\pi/M, 2k\pi/M) \\ \frac{M}{2}, & |\omega| = 2k\pi/M \end{cases} \\ H_{2k}(e^{j\omega}) &= \begin{cases} M, & |\omega| \in (2k\pi/M, (2k+1)\pi/M) \\ \frac{M}{2}, & |\omega| = 2k\pi/M \end{cases} \end{aligned} \right\} \text{ for } k \neq 0.$$

Hint: Consider signals of the type $A_l \cos[(2l\pi/M)n]$, for $l \in \mathbb{Z}$.

- 9.2 Show that a perfect reconstruction linear-phase filter bank with causal filters must be such that $H_0(z)H_1(-z)$ has an odd number of coefficients and that all but one of its odd powers of z must be zero.
- 9.3 Prove, using an argument in the frequency domain, that the system depicted in Figure 9.10 has a transfer function equal to z^{-M+1} .
- 9.4 Find the matrices $\mathbf{E}(z)$ and $\mathbf{R}(z)$ for the filter bank described by Equations (9.131)–(9.134), and verify that their product is equal to a pure delay.
- 9.5 Modify the causal analysis and synthesis filters of the filter bank in equations (9.131)–(9.134) so that they constitute a zero-delay perfect reconstruction filter bank. Do the same for the filter banks in Equations (9.135)–(9.138). Suggest a general method to transform any perfect reconstruction causal filter bank into a zero-delay filter bank.
- 9.6 Using the relation deduced in Exercise 10.2, design a two-band linear-phase perfect reconstruction filter bank such that the lowpass analysis filter has z transform equal to $(1 + z^{-1})^5$ and the highpass analysis filter has order 5 with one zero at $z = 1$. Plot the frequency responses of the filters and comment on how “good” this filter bank is.
- 9.7 Design a Johnston two-band filter bank with 64 coefficients.
- 9.8 Design a Johnston two-band filter bank with at least 60 dB of stopband attenuation.
- 9.9 Consider a perfect reconstruction filter bank which satisfies the QMF condition, $H_1(z) = H_0(-z)$, and has the following lowpass analysis filter:

$$H_0(z) = z^{-5} + 5z^{-3} + z^{-2} + 6z^{-1} + 4.$$

Design the stable synthesis filters.

9.10 Repeat Exercise 9.9 for the lowpass analysis filter equal to

$$H_0(z) = \frac{1}{4}z^{-5} + z^{-3} + z^{-2} + z^{-1} + 2.$$

9.11 Repeat Exercise 9.9 for the lowpass analysis filter equal to

$$H_0(z) = z^{-3} - z^{-2} + z^{-1} + 1.$$

9.12 Design a CQF bank with at least 55 dB of stopband attenuation.

9.13 Given the lowpass analysis filter of a two-band FIR perfect reconstruction filter bank

$$H_0(z) = z^{-3} + az^{-2} + bz^{-1} + 2,$$

determine the analysis and synthesis filters and discuss the class of filter bank to which they belong. Deduce the relation between a and b , assuming they are nonzero, in order to achieve perfect reconstruction.

9.14 Design a two-band QMF filter bank satisfying the following specifications:

$$\delta_p = 0.1$$

$$\delta_r = 0.05$$

$$\Omega_p = 0.4 \frac{\Omega_s}{2}$$

$$\Omega_r = 0.6 \frac{\Omega_s}{2}$$

$$\Omega_s = 2\pi \text{ rad/s.}$$

9.15 Show that a halfband filter can be designed from a Hilbert transformer as follows:

$$h(n) = 0.5 \left[\delta(n) + (-1)^{(n-1)/2} h_h(n) \right],$$

where $h(n)$ is the halfband impulse response and $h_h(n)$ is the impulse response of the Hilbert transformer. Note that $h_h(n) = 0$ for even n (see Equation (5.18)). Show also that its z transform is equal to

$$H(z) = 0.5 [1 - jH_h(-jz)].$$

9.16 Design a two-band perfect reconstruction filter bank using the CQF design satisfying the following specifications:

$$\delta_p = 0.05$$

$$\delta_r = 0.05$$

$$\Omega_p = 0.45 \frac{\Omega_s}{2}$$

$$\Omega_r = 0.55 \frac{\Omega_s}{2}$$

$$\Omega_s = 2\pi \text{ rad/s.}$$

- 9.17 Design a perfect reconstruction filter bank using the cosine-modulated filter bank with six sub-bands with at least 40 dB of stopband attenuation and 0.5 dB of passband ripple. Try to use $\rho = \pi/8M$.
- 9.18 Design a perfect reconstruction filter bank using the cosine-modulated filter bank with 10 sub-bands with at least 40 dB of stopband attenuation and 0.5 dB of passband ripple. Try to use $\rho = \pi/6M$.
- 9.19 Design a perfect reconstruction filter bank using the cosine-modulated filter bank with 10 sub-bands with at least 35 dB of stopband attenuation and 1 dB of passband ripple. Try to use $\rho = \pi/8M$.
- 9.20 Depict in detail the structure of the cosine-modulated filter bank utilizing $M = 2$ bands.
- 9.21 Design a cosine-modulated filter bank with $M = 5$ sub-bands and at least 40 dB of stopband attenuation.
- 9.22 Design a cosine-modulated filter bank with $M = 15$ sub-bands with at least 20 dB of stopband attenuation.
- 9.23 Express the DFT as an M -band filter bank. Plot the magnitude responses of the analysis filters for $M = 8$.
- 9.24 Show that $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{C}}_2$ defined in Equations (9.237) and (9.238) satisfy Equations (9.233) and (9.234).
- 9.25 Design a fast LOT-based filter bank with at least eight sub-bands.
- 9.26 Prove the relationship in Equation (9.210).
- 9.27 Show that the relations in Equations (9.256)–(9.258) are valid.
- 9.28 For Example 9.12, is an orthogonal solution possible? Compute $\mathbf{E}^{-1}(z)$ for the proof.
- 9.29 In Example 9.12, we could attempt to generalize the orthogonal realization of the LOT by allowing the matrix \mathbf{L}_1 of Figure 9.45 to be a full matrix and design a simple biorthogonal filter. Will this solve the problem at hand? Justify your answer.
- 9.30 Propose an alternative and simpler structure to that of Figure 9.46 in Example 9.12. The simplified structure should be based on Equation (9.267).
- 9.31 Design a linear-phase lapped biorthogonal transform with $M = 6$ sub-bands aiming at minimizing the stopband ripple of the sub-filters.
- 9.32 Design a linear-phase lapped biorthogonal transform with $M = 10$ sub-bands.
- 9.33 Design a length-16 linear-phase lapped biorthogonal transform with $M = 8$ sub-bands aiming at minimizing the stopband ripple of the sub-filters.
- 9.34 Design a GenLOT with $M = 10$ sub-bands having at least 25 dB of stopband attenuation.
- 9.35 Design a GenLOT with $M = 8$ sub-bands having at least 25 dB of stopband attenuation.
- 9.36 Design a GenLOT with $M = 6$ sub-bands having at least 20 dB of stopband attenuation.

- 9.37 Show that if a filter bank has linear-phase analysis and synthesis filters with the same lengths $N = LM$, then the following relations for the polyphase matrices are valid:

$$\mathbf{E}(z) = z^{-L+1} \mathbf{D} \mathbf{E}(z^{-1}) \mathbf{J}$$

$$\mathbf{R}(z) = z^{-L+1} \mathbf{J} \mathbf{R}(z^{-1}) \mathbf{D},$$

where \mathbf{D} is a diagonal matrix whose entries are 1 if the corresponding filter is symmetric and -1 otherwise.

- 9.38 Consider an M -band linear-phase filter bank with perfect reconstruction with all analysis and synthesis filters having the same length $N = LM$. Show that, for a polyphase matrix $\mathbf{E}_1(z)$ corresponding to a linear-phase filter bank, then $\mathbf{E}(z) = \mathbf{E}_2(z) \mathbf{E}_1(z)$ will also lead to a linear-phase perfect reconstruction filter bank if

$$\mathbf{E}_2(z) = z^{-L} \mathbf{D} \mathbf{E}_2(z^{-1}) \mathbf{D},$$

where \mathbf{D} is a diagonal matrix with entries 1 or -1 , as described in Exercise 9.37, and L is the order of $\mathbf{E}_2(z)$. Also show that

$$\mathbf{E}_{2,i} = \mathbf{D} \mathbf{E}_{2,L-i} \mathbf{D}.$$

10.1 Introduction

In Chapter 9 we dealt with filter banks, which are important in several applications. In this chapter, wavelet transforms are considered. They come from the area of functional analysis and generate great interest in the signal processing community, because of their ability to represent and analyze signals with varying time and frequency resolutions. Their digital implementation can be regarded as a special case of critically decimated filter banks. Multiresolution decompositions are then presented as an application of wavelet transforms. The concepts of regularity and number of vanishing moments of a wavelet transform are then explored. Two-dimensional wavelet transforms are introduced, with emphasis on image processing. Wavelet transforms of finite-length signals are also dealt with. We wrap up the chapter with a Do-it-yourself section followed by a brief description of functions from the MATLAB Wavelet Toolbox which are useful for wavelets implementation.

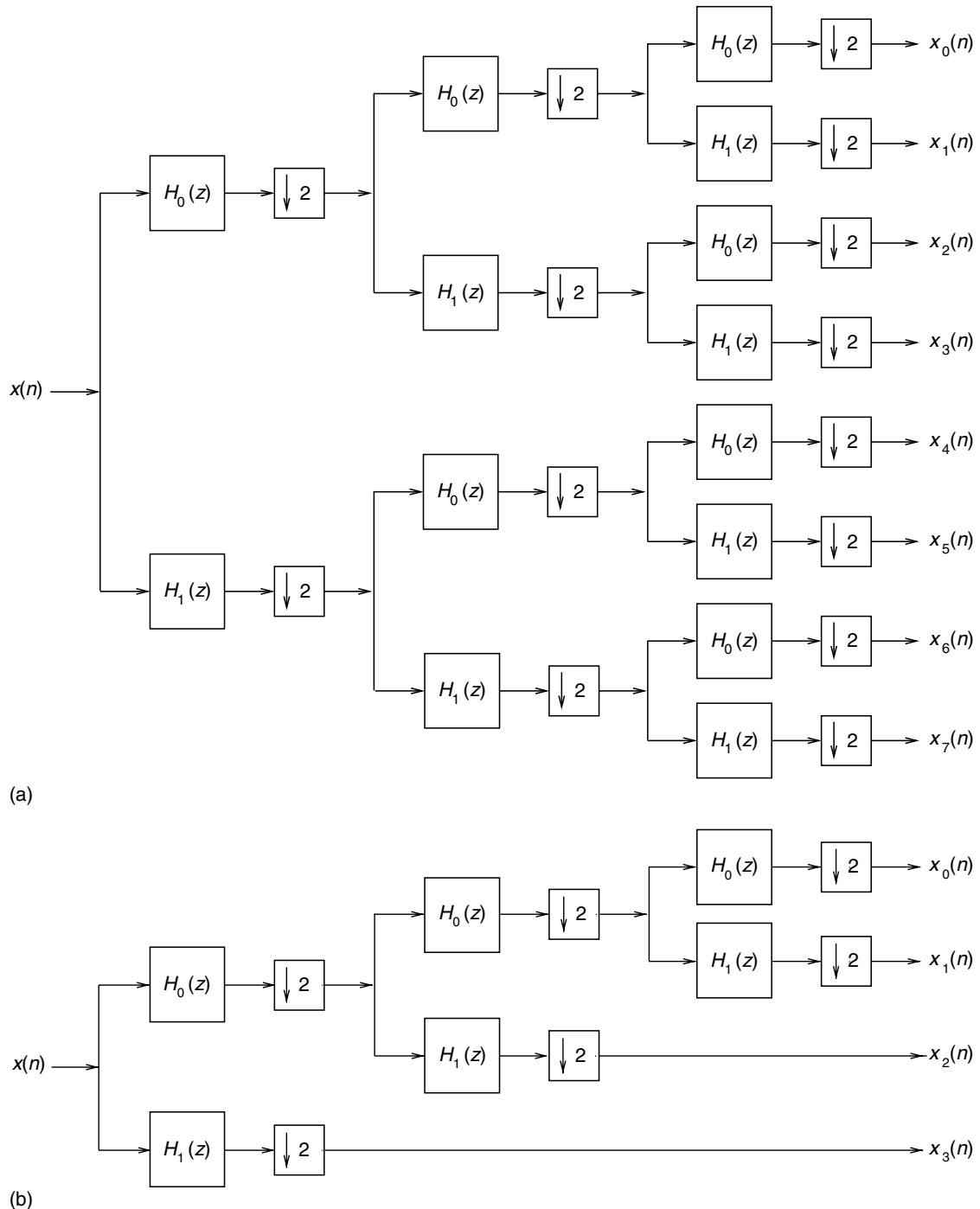
10.2 Wavelet transforms

Wavelet transforms are a relatively recent development in functional analysis that have attracted a great deal of attention from the signal processing community (Daubechies, 1991). The wavelet transform of a function belonging to $\mathcal{L}^2(\mathbb{R})$, the space of the square integrable functions, is its decomposition in a base formed by expansions, compressions, and translations of a single mother function $\psi(t)$, called a wavelet.

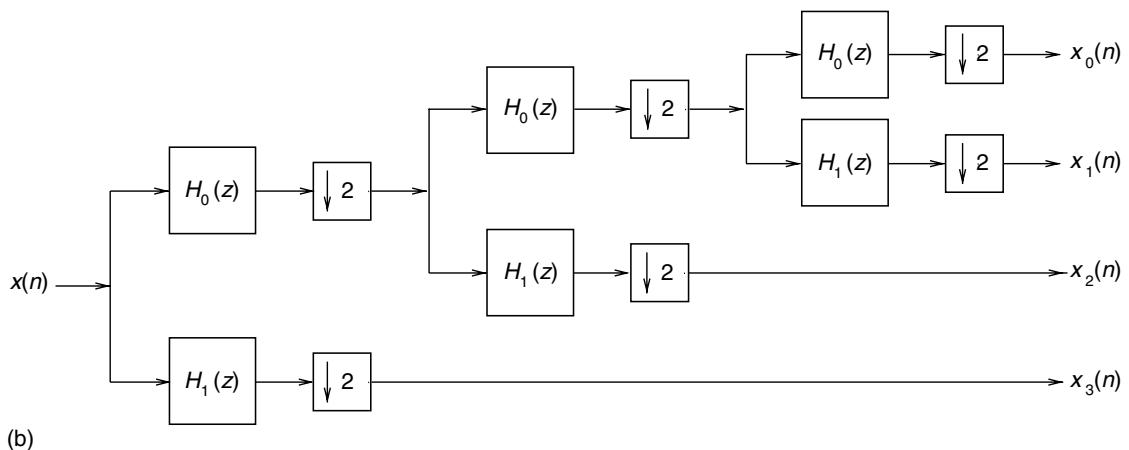
The applications of wavelet transforms range from quantum physics to signal coding. It can be shown that for digital signals the wavelet transform is a special case of critically decimated filter banks (Vetterli & Herley, 1992). In fact, its numerical implementation relies heavily on that approach. In what follows, we give a brief introduction to wavelet transforms, emphasizing their relation to filter banks. In the literature there is plenty of good material analyzing wavelet transforms from different points of view. Examples are Daubechies (1991), Vetterli and Kovačević (1995), Strang and Nguyen (1996), and Mallat (1999).

10.2.1 Hierarchical filter banks

The cascading of two-band filter banks can produce many different kinds of critically decimated decompositions. For example, one can make a 2^k -band uniform decomposition,



(a)



(b)

Fig. 10.1. Hierarchical decompositions: (a) eight-band uniform; (b) three-stage octave-band.

as depicted in Figure 10.1a, for $k = 3$. Another common type of hierarchical decomposition is the octave-band decomposition, in which only the lowpass band is further decomposed. In Figure 10.1b, one can see a three-stage octave-band decomposition. In these figures, the synthesis bank is not drawn, because it is entirely analogous to the analysis bank.

10.2.2 Wavelets

Consider the octave-band analysis and synthesis filter banks depicted in Figure 10.2, where the lowpass bands are recursively decomposed into lowpass and highpass channels. In this framework, the outputs of the lowpass channels after an $(S + 1)$ -stage decomposition are $x_{S,n}$ and the outputs of the highpass channels are $c_{S,n}$, with $S \geq 1$.

Applying the noble identities to Figure 10.2, we arrive at Figure 10.3. After $(S + 1)$ stages, and before decimation by a factor of $2^{(S+1)}$, the z transforms of the analysis lowpass and highpass channels, $H_{\text{low}}^{(S)}(z)$ and $H_{\text{high}}^{(S)}(z)$, are

$$H_{\text{low}}^{(S)}(z) = \frac{X_S(z)}{X(z)} = \prod_{k=0}^S H_0(z^{2^k}) \quad (10.1)$$

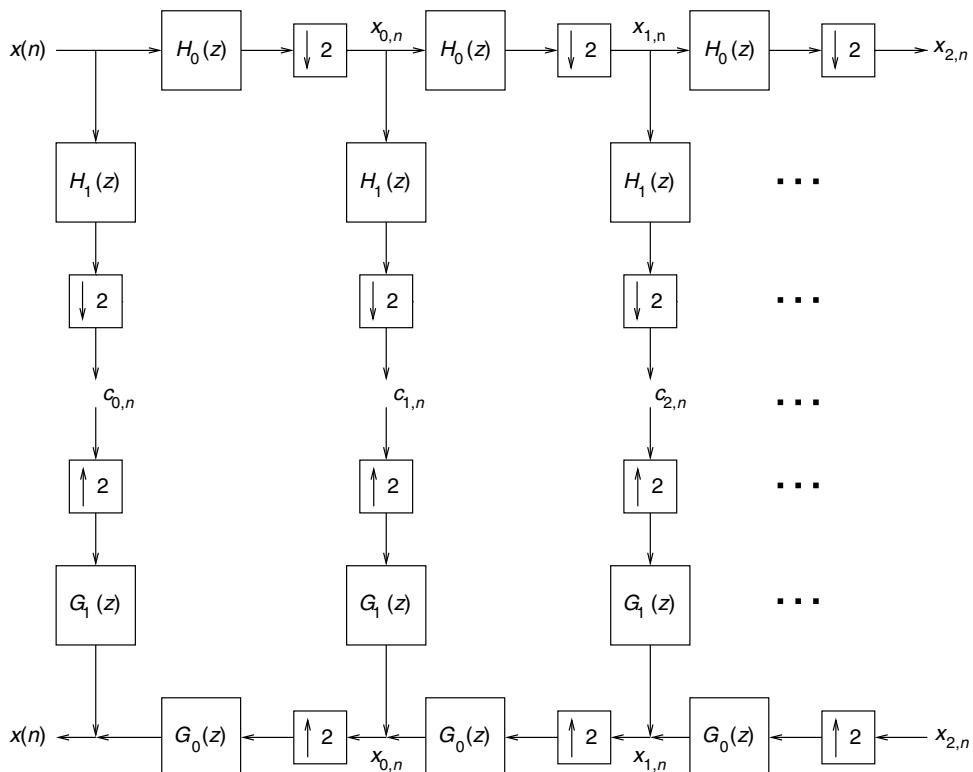


Fig. 10.2. Octave-band analysis and synthesis filter banks.

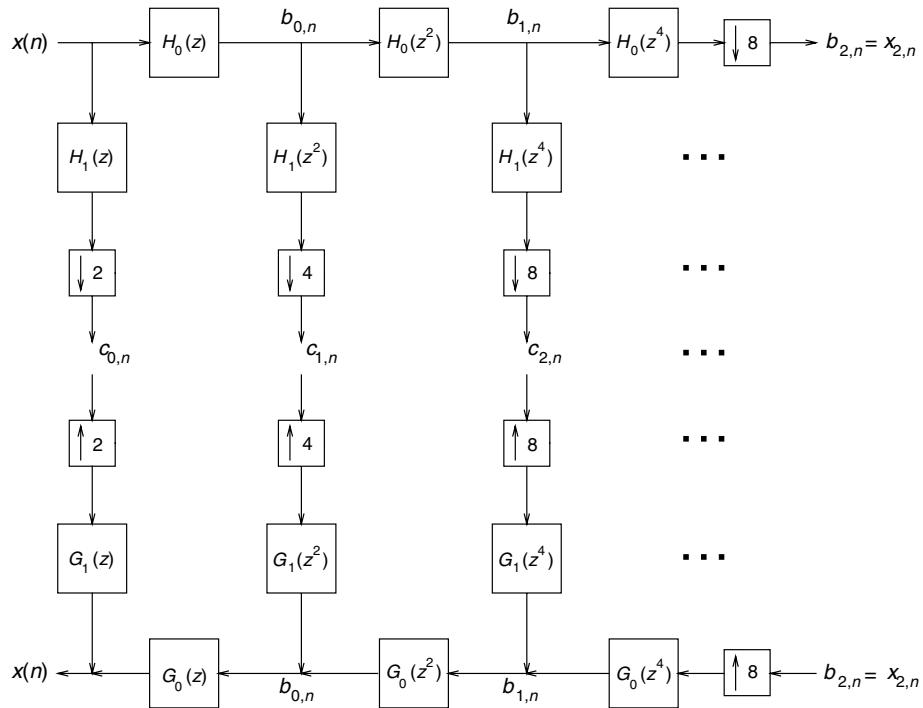


Fig. 10.3. Octave-band analysis and synthesis filter banks after the application of the noble identities.

and

$$H_{\text{high}}^{(S)}(z) = \frac{C_S(z)}{X(z)} = H_1(z^{2^S}) H_{\text{low}}^{(S-1)}(z) \quad (10.2)$$

respectively. For the synthesis channels, the results are analogous; that is:

$$G_{\text{low}}^{(S)}(z) = \prod_{k=0}^S G_0(z^{2^k}) \quad (10.3)$$

$$G_{\text{high}}^{(S)}(z) = G_1(z^{2^S}) G_{\text{low}}^{(S-1)}(z). \quad (10.4)$$

If $H_0(z)$ has enough zeros at $z = -1$, it can be shown (Vetterli and Kovačević, 1995; Strang and Nguyen, 1996; Mallat, 1999) that the envelope of the impulse response of the filters in Equation (10.2) has the same shape for all S . In other words, this envelope can be represented by expansions and contractions of a single function $\psi(t)$, as seen in Figure 10.4 for the analysis filter bank.

In fact, in this setup, the envelopes before and after the decimators are the same. However, it must be noted that, after decimation, we cannot refer to impulse responses in the usual way, because the decimation operation is not time invariant (see Section 8.3).

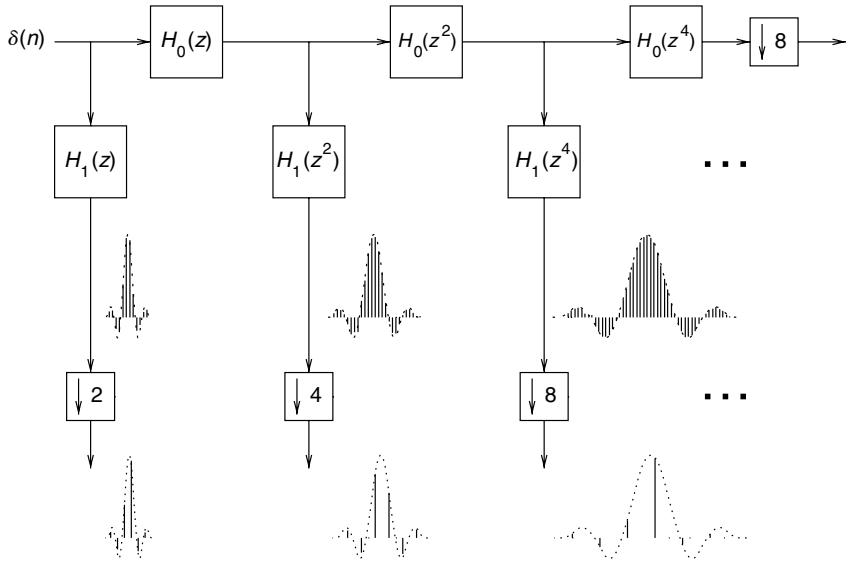


Fig. 10.4. The impulse responses of the filters from Equation (10.2) have the same shape for every stage.

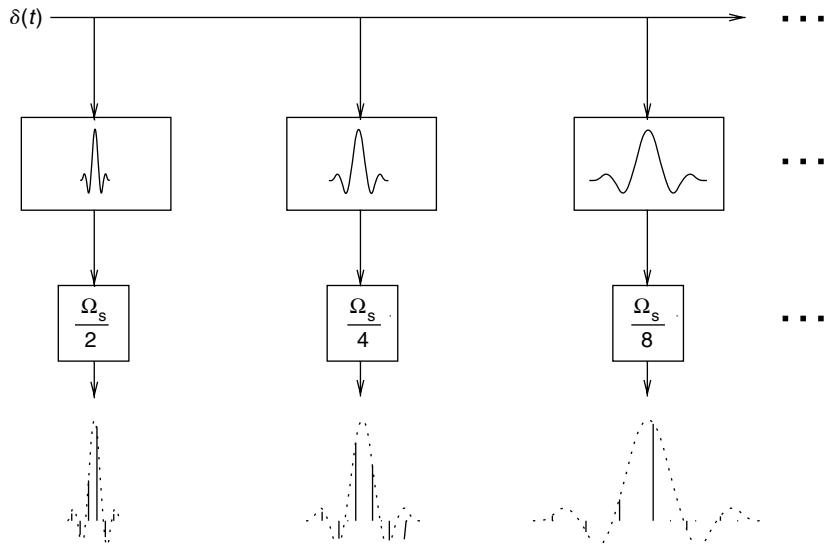


Fig. 10.5. Equivalent system to the one in Figure 10.4.

If Ω_s is the sampling rate at the input of the system in Figure 10.4, then we have that this system has the same output as the one in Figure 10.5, where the boxes represent continuous-time filters with impulse responses equal to the envelopes of these signals in Figure 10.4. Note that, in this case, sampling with frequency Ω_s/k is equivalent to decimating by k .

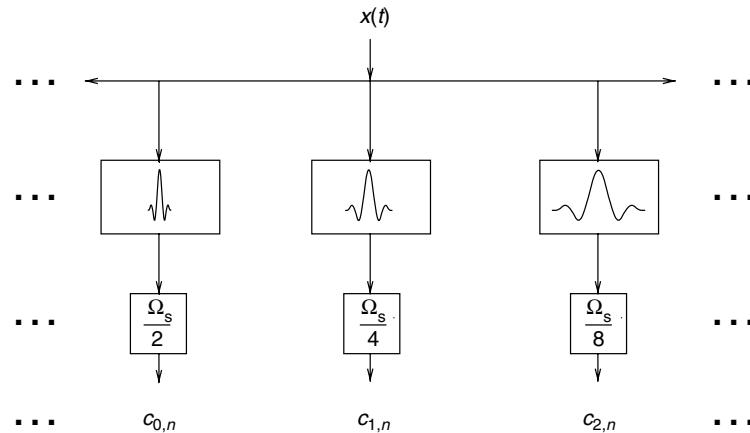


Fig. 10.6. Wavelet transform of a continuous signal $x(t)$.

We then observe that the impulse responses of the continuous-time filters of Figure 10.5 are expansions and contractions of a single mother function $\psi(t)$, with the highest sampling rate being $\Omega_s/2$, as stated above. Then, each channel added to the right has an impulse response with double the width and a sampling rate half of the previous one. There is no impediment in adding channels also to the left of the channel with sampling frequency $\Omega_s/2$. In such cases, each new channel to the left has an impulse response with half the width and a sampling rate twice the one to the right. If we keep adding channels to both the right and the left indefinitely, then we arrive at Figure 10.6, where the input is $x(t)$ and the output is referred to as the wavelet transform of $x(t)$. The mother function $\psi(t)$ is called the wavelet, or, more specifically, the analysis wavelet (Daubechies, 1988; Mallat, 1999).

Assuming, without loss of generality, that $\Omega_s = 2\pi$ (that is, $T_s = 1$), it is straightforward to derive from Figure 10.6 that the wavelet transform of a signal $x(t)$ is given by¹

$$c_{m,n} = \int_{-\infty}^{\infty} 2^{-m/2} \psi^*(2^{-m}t - n) x(t) dt. \quad (10.5)$$

From Figures 10.3–10.6 and Equation (10.2), one can see that the wavelet $\psi(t)$ is a bandpass function, because each channel is a cascade of several lowpass filters and a highpass filter with superimposing passbands. Also, when the wavelet is expanded in time by two, its bandwidth decreases by two, as seen in Figure 10.7. Therefore, the decomposition in Figure 10.6 and Equation (10.5) is, in the frequency domain, as shown in Figure 10.8.

In a similar manner, the envelopes of the impulse responses of the equivalent synthesis filters after interpolation (see Figure 10.3 and Equation (10.4)) are expansions and contractions of a single mother function $\bar{\psi}(t)$. Using similar reasoning to that leading to Figures 10.4–10.6, one can obtain the continuous-time signal $x(t)$ from the wavelet coefficients $c_{m,n}$

¹ Actually, in this expression, the impulse responses of the filters are expansions and contractions of $\psi^*(-t)$. In addition, the constant $2^{-m/2}$ is included because, if $\psi(t)$ has unit energy, which can be assumed without loss of generality, $2^{-m/2}\psi(2^{-m}t - n)$ will also have unit energy.

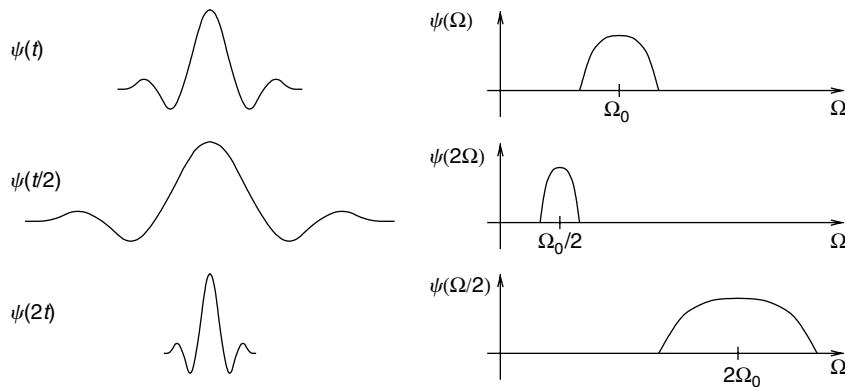


Fig. 10.7. Expansions and contractions of the wavelet in the time and frequency domains.

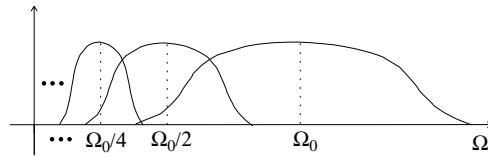


Fig. 10.8. Wavelet transform in the frequency domain.

as (Vetterli and Kovačević, 1995; Mallat, 1999)

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} 2^{-m/2} \bar{\psi}(2^{-m}t - n). \quad (10.6)$$

Equations (10.5) and (10.6) represent the direct and inverse wavelet transforms of a continuous-time signal $x(t)$. The wavelet transform of the corresponding discrete-time signal $x(n)$ is merely the octave-band decomposition in Figures 10.2 and 10.3. A natural question to ask at this point is: How are the continuous-time signal $x(t)$ and the discrete-time signal $x(n)$ related, if they generate the same wavelet coefficients? In addition, how can the analysis and synthesis wavelets be derived from the filter bank coefficients and vice versa? These questions can be answered using the concept of scaling functions, seen in the next section.

10.2.3 Scaling functions

By examining Figure 10.6 and Equations (10.5) and (10.6), we observe that the values of m which are associated with the “width” of the filters range from $-\infty$ to $+\infty$. Since all signals encountered in practice are somehow band-limited, one can assume, without loss of generality, that the outputs of the filters with impulse responses $\psi(2^{-m}t)$ are zero, for $m < 0$. Therefore, in practice, m can then vary only from 0 to $+\infty$. Examining Figures 10.2–10.4,

we observe that $m \rightarrow +\infty$ means that the lowpass channels will be indefinitely decomposed. However, in practice, the number of stages of decomposition is finite and, after S stages, we have S bandpass channels and one lowpass channel. Therefore, if we restrict the number of decomposing stages in Figures 10.2–10.6 and add a lowpass channel, then we can modify Equation (10.6) so that m assumes only a finite number of values. This can be done by noting that if $H_0(z)$ has enough zeros at $z = -1$, the envelopes of the analysis lowpass channels given in Equation (10.1) will also be expansions and contractions of a single function $\phi(t)$, which is called the analysis scaling function. Likewise, the envelopes of the synthesis lowpass channels are expansions and contractions of the synthesis scaling function $\bar{\phi}(t)$ (Vetterli and Kovačević, 1995; Mallat, 1999). Therefore, if we make an $(S + 1)$ -stage decomposition, Equation (10.6) becomes

$$x(t) = \sum_{m=0}^{S-1} \sum_{n=-\infty}^{\infty} c_{m,n} 2^{-m/2} \bar{\psi}(2^{-m}t - n) + \sum_{n=-\infty}^{\infty} x_{S,n} 2^{-S/2} \bar{\phi}(2^{-S}t - n), \quad (10.7)$$

where

$$x_{S,n} = \int_{-\infty}^{\infty} 2^{-S/2} \bar{\phi}^*(2^{-S}t - n) x(t) dt. \quad (10.8)$$

Hence, the wavelet transform is, in practice, described as in Equations (10.5), (10.7), and (10.8). The summations in n will, in general, depend on the supports (that is, the regions where the functions are nonzero) of the signal, wavelets, and scaling functions (Daubechies, 1988; Mallat, 1999).

10.3 Relation between $x(t)$ and $x(n)$

Equation (10.8) shows how to compute the coefficients of the lowpass channel after an $(S + 1)$ -stage wavelet transform. In Figure 10.2, $x_{S,n}$ are the outputs of a lowpass filter $H_0(z)$ after $(S + 1)$ stages. Since in Figure 10.3 the discrete-time signal $x(n)$ can be regarded as the output of a lowpass filter after “zero” stages, we can say that $x(n)$ would be equal to $x_{-1,n}$. In other words, equivalence of the outputs of the octave-band filter bank of Figure 10.2 and the wavelet transform given by Equations (10.5) and (10.6) occurs only if the digital signal input to the filter bank of Figure 10.2 is equal to $x_{-1,n}$. From Equation (10.8), this means

$$x(n) = \int_{-\infty}^{\infty} \sqrt{2} \bar{\phi}^*(2t - n) x(t) dt. \quad (10.9)$$

Such an equation can be interpreted as $x(n)$ being the signal $x(t)$ digitized with a band-limiting filter having $\sqrt{2}\bar{\phi}(-2t)$ as its impulse response. Therefore, a possible way to compute the wavelet transform of a continuous-time signal $x(t)$ is depicted in Figure 10.9, in which $x(t)$ is passed through a filter having as impulse response the scaling function

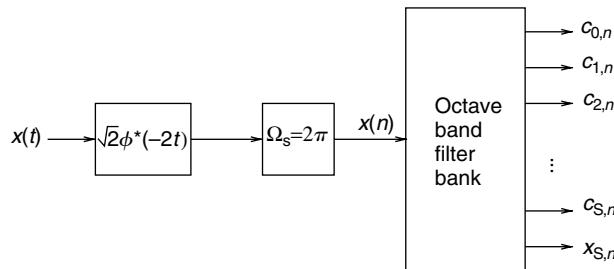


Fig. 10.9.

Practical way to compute the wavelet transform of a continuous-time signal.

contracted-in-time by 2 and sampled with $T_s = 1$, or equivalently $\Omega_s = 2\pi$. The resulting digital signal is then the input to the octave-band filter bank in Figure 10.2 whose filter coefficients will be as determined later on in Section 10.6 by Equations (10.125) and (10.127). At this point, it is important to note that, strictly speaking, the wavelet transform is only defined for continuous-time signals. However, it is common practice to refer to the wavelet transform of a discrete-time signal $x(n)$ as the output of the filter bank in Figure 10.2 (Vetterli and Kovačević, 1995). One should also note that, in order for the output signals in Figures 10.4 and 10.5 to be entirely equivalent, the input signal in Figure 10.4 must not be the discrete-time impulse, but the sampled impulse response of the filter $\sqrt{2}\phi(-2t)$. This is nothing less than a sampled version of the scaling function contracted-in-time by 2.

10.4 Wavelet transforms and time-frequency analysis

In time-frequency analysis one is interested to know how the frequency content of a signal varies in time. Wavelet transforms are a powerful tool for that purpose, and in this section we approach the wavelet transform from a time-frequency analysis point of view. We do so by first analyzing the short-time Fourier transform, highlighting its limitations using the uncertainty principle. Then we introduce the continuous-time wavelet transform as a way to overcome some of the limitations of the short-time Fourier transform. Finally, we arrive at the wavelet transform by sampling the continuous-time wavelet transform.

10.4.1 The short-time Fourier transform

A common form of signal representation and analysis is through its decomposition into its frequency components. A classical way of doing so is through the Fourier transform $X(\Omega)$ of a function $x(t)$ defined as

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt. \quad (10.10)$$

The Fourier transform computes the frequency content of a signal taking into consideration its entire duration, from $t = -\infty$ to $t = \infty$. However, it is sometimes interesting to compute the frequency content of a signal only around a certain time location. For example, if someone speaks “The baby wants a ball,” one may be interested in analyzing the frequency content only of the article “a,” and not of the phrase as a whole. Clearly, the Fourier transform is not suitable for such an analysis, for it will always take into account the full duration of the signal. Therefore, a tool for analyzing the local frequency content of a signal is desirable. A generalization of the Fourier transform, the short-time Fourier transform (STFT) is such a tool. It can be defined as (Gabor, 1946)

$$X_F(\Omega_0, b) = \int_{-\infty}^{\infty} x(t)g(t - b) e^{-j\Omega_0 t} dt. \quad (10.11)$$

The STFT is equivalent to the Fourier transform of the windowed function $x(t)g(t - b)$. The window function $g(t)$ is in general “concentrated” around $t = 0$, and its purpose is to isolate the values of the function $x(t)$ around $t = b$ prior to the computation of the Fourier transform. Figure 10.10 shows a typical window $g(t)$. The STFT has two independent variables: the frequency Ω and the position b of the data window. For each value of b , the STFT gives the spectral content $X_F(\Omega, b)$ of $x(t)$ around $t = b$.

The STFT also has a dual interpretation. If $G(\Omega)$ is the Fourier transform of $g(t)$, then one can use the Parseval’s theorem to show that Equation (10.11) is equivalent to (Pouliakis & Seely, 1988)

$$X_F(\Omega_0, b) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega)G(\Omega - \Omega_0) e^{-j\Omega b} d\Omega. \quad (10.12)$$

Therefore, for each Ω_0 , the STFT also gives how the frequency components of $x(t)$ around $\Omega = \Omega_0$ (as filtered by $G(\Omega - \Omega_0)$) evolve in time.

Besides being concentrated around $t = 0$, the window function $g(t)$ is in general chosen such that $G(\Omega)$ is also concentrated around $\Omega = 0$. A common choice for $g(t)$ is a Gaussian function, which is concentrated around zero in both the time and frequency domains. A good estimate of the “width” of both $g(t)$ and $G(\Omega)$ is provided, respectively, by their standard

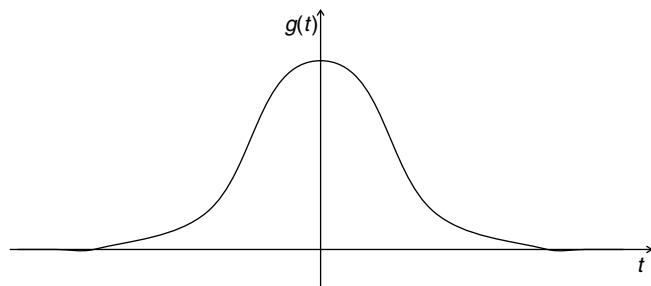


Fig. 10.10. Typical window function for the STFT.

deviations σ_b and σ_Ω . The respective variances are given by (Mallat, 1989a)

$$\sigma_b^2 = \frac{\int_{-\infty}^{\infty} t^2 |g(t)|^2 dt}{\int_{-\infty}^{\infty} |g(t)|^2 dt} \quad (10.13)$$

$$\sigma_\Omega^2 = \frac{\int_0^{\infty} \Omega^2 |G(\Omega)|^2 d\Omega}{\int_0^{\infty} |G(\Omega)|^2 d\Omega}. \quad (10.14)$$

Examining Equation (10.11), one can observe that the computation of $X_F(\Omega_0, b)$ depends mainly on the values of $x(t)$ in the interval $t \in [b - \sigma_b, b + \sigma_b]$. Alternatively, Equation (10.12) shows that $X_F(\Omega_0, b)$ depends mainly on the values of $X(\Omega)$ in the interval $[\Omega - \sigma_\Omega, \Omega + \sigma_\Omega]$. This is equivalent to saying that the STFT analyzes slices of the signal, having duration $2\sigma_b$, through filters of constant bandwidth, equal to $2\sigma_\Omega$.

From the above, we can conclude that the smaller the σ_b , the better a feature can be localized in the time domain; in other words, the better is the time resolution of the STFT. Alternatively, the smaller the σ_Ω , the better the frequency resolution of the STFT. This implies that the time and frequency resolutions of the STFT depend only on $g(t)$ and, therefore, are fixed, independent of the particular point (b, Ω) in the time-frequency space, as illustrated in Figure 10.11.

It is important to point out that the time and frequency resolutions of the STFT, as defined in Equations (10.13) and (10.14), cannot be arbitrarily small, as stated below (Vetterli & Kovačević, 1995; Mallat, 1999).

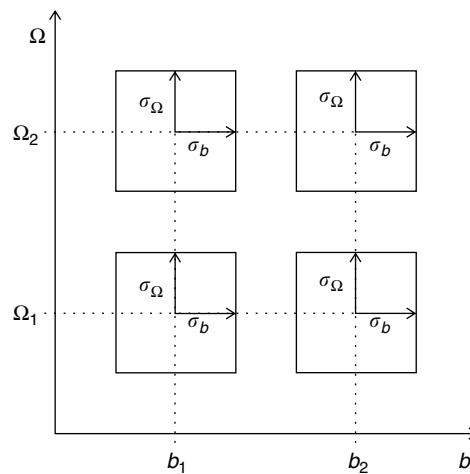


Fig. 10.11. Resolution cells in the time × frequency plane for the STFT.

Property 10.1 (Uncertainty Principle). Whenever $g(t)$ decays faster than $1/\sqrt{t}$ for $t \rightarrow \pm\infty$, then

$$\sigma_{\Omega}^2 \sigma_b^2 \geq \frac{1}{4}. \quad (10.15)$$

This result implies that there is a maximum resolution that can be jointly achieved in frequency and time by any linear transform. The equality holds for Gaussian signals; that is, for

$$g(t) = \alpha e^{-\kappa t^2}. \quad (10.16)$$

Outline

Using the Schwartz inequality (Steele, 2004):

$$\left| \int_{-\infty}^{\infty} t g(t) \frac{dg(t)}{dt} dt \right|^2 \leq \int_{-\infty}^{\infty} |tg(t)|^2 dt \int_{-\infty}^{\infty} \left| \frac{dg(t)}{dt} \right|^2 dt. \quad (10.17)$$

Since the Fourier transform of $dg(t)/dt$ is $j\Omega G(\Omega)$, one can apply Parseval's theorem in the third integral to get

$$\int_{-\infty}^{\infty} \left| \frac{dg(t)}{dt} \right|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\Omega G(\Omega)|^2 d\Omega; \quad (10.18)$$

therefore, Equation (10.17) becomes

$$\left| \int_{-\infty}^{\infty} t g(t) \frac{dg(t)}{dt} dt \right|^2 \leq \frac{1}{2\pi} \int_{-\infty}^{\infty} |tg(t)|^2 dt \int_{-\infty}^{\infty} |\Omega G(\Omega)|^2 d\Omega. \quad (10.19)$$

Since

$$2g(t) \frac{dg(t)}{dt} = \frac{d[g^2(t)]}{dt} \quad (10.20)$$

we have that

$$\begin{aligned} \int_{-\infty}^{\infty} t g(t) \frac{dg(t)}{dt} dt &= \frac{1}{2} \int_{-\infty}^{\infty} t \frac{d[g^2(t)]}{dt} dt \\ &= \frac{1}{2} tg^2(t) \Big|_{-\infty}^{\infty} - \frac{1}{2} \int_{-\infty}^{\infty} g^2(t) dt. \end{aligned} \quad (10.21)$$

Since $g(t)$ decays faster than $1/\sqrt{t}$ for $t \rightarrow \pm\infty$, we have that

$$\lim_{t \rightarrow \pm\infty} tg^2(t) = 0, \quad (10.22)$$

and then, supposing $g(t)$ real, Equation (10.21) becomes

$$\int_{-\infty}^{\infty} t g(t) \frac{dg(t)}{dt} dt = -\frac{1}{2} \int_{-\infty}^{\infty} g^2(t) dt = -\frac{1}{2} \int_{-\infty}^{\infty} |g(t)|^2 dt. \quad (10.23)$$

Substituting the above equation in Equation (10.19), we get

$$\left| \frac{1}{2} \int_{-\infty}^{\infty} |g(t)|^2 dt \right|^2 \leq \frac{1}{2\pi} \int_{-\infty}^{\infty} |tg(t)|^2 dt \int_{-\infty}^{\infty} |\Omega G(\Omega)|^2 d\Omega. \quad (10.24)$$

Using Parseval's theorem again, we have that

$$\int_{-\infty}^{\infty} |g(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |G(\Omega)|^2 d\Omega; \quad (10.25)$$

therefore:

$$\left| \int_{-\infty}^{\infty} |g(t)|^2 dt \right|^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} |G(\Omega)|^2 d\Omega \int_{-\infty}^{\infty} |g(t)|^2 dt. \quad (10.26)$$

Replacing the above expression into Equation (10.24), we conclude that

$$\frac{1}{8\pi} \int_{-\infty}^{\infty} |g(t)|^2 dt \int_{-\infty}^{\infty} |G(\Omega)|^2 d\Omega \leq \frac{1}{2\pi} \int_{-\infty}^{\infty} |tg(t)|^2 dt \int_{-\infty}^{\infty} |\Omega G(\Omega)|^2 d\Omega; \quad (10.27)$$

therefore:

$$\left(\frac{\int_{-\infty}^{\infty} |t|^2 |g(t)|^2 dt}{\int_{-\infty}^{\infty} |g(t)|^2 dt} \right) \left(\frac{\int_{-\infty}^{\infty} |\Omega|^2 |G(\Omega)|^2 d\Omega}{\int_{-\infty}^{\infty} |G(\Omega)|^2 d\Omega} \right) \geq \frac{1}{4}, \quad (10.28)$$

which, from the definitions of Equations (10.13) and (10.14), gives the desired result.

If $g(t)$ is a Gaussian, that is

$$g(t) = \alpha e^{-\kappa t^2}, \quad (10.29)$$

then we have that

$$G(\Omega) = \alpha \sqrt{\frac{\pi}{\kappa}} e^{-\Omega^2/4\kappa} \quad (10.30)$$

and then

$$g^2(t) = \alpha^2 e^{-2\kappa t^2}; \quad G^2(\Omega) = \alpha^2 \frac{\pi}{\kappa} e^{-\Omega^2/2\kappa}. \quad (10.31)$$

Since $g^2(t)$ and $G^2(\Omega)$ are still Gaussian, then $\sigma_b^2 = 1/4\kappa$ and $\sigma_\Omega^2 = \kappa$, in such a way that

$$\sigma_b^2 \sigma_\Omega^2 = \frac{1}{4}. \quad (10.32)$$

Therefore, for the Gaussian window function, relationship (10.15) holds with the equality sign, indicating that it represents the best possible time-frequency resolution compromise.

Note that the requirements of the window in this case are different from the ones in Section 5.4. There, we want $g(t)$ to be concentrated as much as possible in the frequency domain, while having ripples with the smallest amplitude possible. This is so because in that case we are interested only in the frequency-domain properties, while in the STFT we want the best compromise between time- and frequency-domain representations.

In some cases, the fixed spatial and frequency resolutions of the STFT become in general a major drawback. This usually happens with highly nonstationary signals, which have features of very different sizes and in varying degrees of resolution. This is so because, once the function $g(t)$ is fixed, its time and frequency resolutions are also fixed, and only those features of size comparable to that of $g(t)$ can be conveniently analyzed. Therefore, it is desirable to have a transform with windows of different sizes, so that it can adapt to the features to be analyzed. Wavelet transforms are a class of transforms that have exactly this property.

10.4.2 The continuous-time wavelet transform

The continuous wavelet transform (Rioul & Vetterli, 1991) of a signal $x(t)$ belonging to $L^2(\mathbb{R})$, the space of square integrable functions in \mathbb{R} , is its decomposition into a set of basis functions comprising expansions and translations of a mother function $\psi(t)$. Therefore, by defining the basis functions $\psi_{a,b}(t)$ as

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right), \quad (10.33)$$

the continuous wavelet transform can be written as (Daubechies, 1991)

$$X_W(a, b) = \int_{-\infty}^{\infty} x(t) \psi_{a,b}^*(t) dt, \quad (10.34)$$

where “*” denotes complex conjugation. Throughout this discussion, wavelets will be assumed to be real, and so $\psi(t) = \psi^*(t)$.

Alternatively, if the Fourier transform of $\psi_{a,b}(t)$ is $\Psi_{a,b}(\omega)$, then Equation (10.34) can be rewritten in the frequency domain, using Parseval’s theorem, as

$$X_W(a, b) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_{a,b}^*(\omega) X(\omega) d\omega. \quad (10.35)$$

It can be shown that $x(t)$ can be recovered from its wavelet transform $X_W(a, b)$ using the following expression (Daubechies, 1991):

$$x(t) = C_\psi^{-1} \int_0^\infty \frac{da}{a^2} \int_{-\infty}^{\infty} X_W(a, b) \psi_{a,b}(t) db, \quad (10.36)$$

where the constant C_ψ is such that

$$C_\psi = 2\pi \int_0^\infty \frac{|\Psi(\omega)|^2}{|\omega|} d\omega = 2\pi \int_{-\infty}^0 \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty. \quad (10.37)$$

If $\psi(t)$ is continuous, then Equation (10.37) can only be satisfied if (Daubechies, 1991)

$$\int_{-\infty}^\infty \psi(t) dt = 0. \quad (10.38)$$

The above equation is equivalent to $\Psi(0) = 0$, which implies that $\psi(t)$ must be a bandpass function.

The factor a is called the *scale* of the basis function. The larger it is, the wider the basis function is in the time domain and, therefore, the narrower it is in the frequency domain. Conversely, the smaller the scale a , the narrower the basis function is in the time domain and the wider it is in the frequency domain. Figure 10.7 illustrates this point. From this, one can infer that, in the wavelet transform, time and frequency resolutions vary and there is a trade-off between both. In order to be more precise, we first note that, from Equation (10.38), $\psi(t)$ must be bandpass. Supposing that $\psi(t)$ is real, then $\Psi(\Omega)$ is conjugate symmetric. Now let t_0 and Ω_0 be such that

$$\int_{-\infty}^\infty (t - t_0) |\psi(t)|^2 dt = 0 \quad (10.39)$$

$$\int_0^\infty (\Omega - \Omega_0) |\Psi(\Omega)|^2 d\Omega = 0. \quad (10.40)$$

We can define, similar to Equations (10.13) and (10.14), the variances of $\psi(t)$ and of the positive frequencies of its Fourier transform $\Psi(\omega)$ as (Mallat, 1989a)

$$\sigma_b^2 = \frac{\int_{-\infty}^\infty (t - t_0)^2 |\psi(t)|^2 dt}{\int_{-\infty}^\infty |\psi(t)|^2 dt} \quad (10.41)$$

$$\sigma_\Omega^2 = \frac{\int_0^\infty (\Omega - \Omega_0)^2 |\Psi(\Omega)|^2 d\Omega}{\int_0^\infty |\Psi(\Omega)|^2 d\Omega}. \quad (10.42)$$

Note that $\psi(t)$ is usually chosen such that $t_0 = 0$.

The above equations imply that the standard deviations of $\psi_{a,b}(t)$ and its Fourier transform are $a\sigma_b$ and σ_Ω/a respectively. Also, the center frequency of $\Psi_{a,b}(\Omega)$ is Ω_0/a . Therefore, from Equation (10.34), the wavelet transform $X_W(a, b)$ depends mainly on the values of $x(t)$ in the interval $t \in [b - a\sigma_b, b + a\sigma_b]$. From Equation (10.35), $X_W(a, b)$ depends mainly on the values of $X(\Omega)$ in the frequency interval $\Omega \in [(\Omega_0/a) - (\sigma_\Omega/a), (\Omega_0/a) + (\sigma_\Omega/a)]$. This implies that for large a (that is, for small

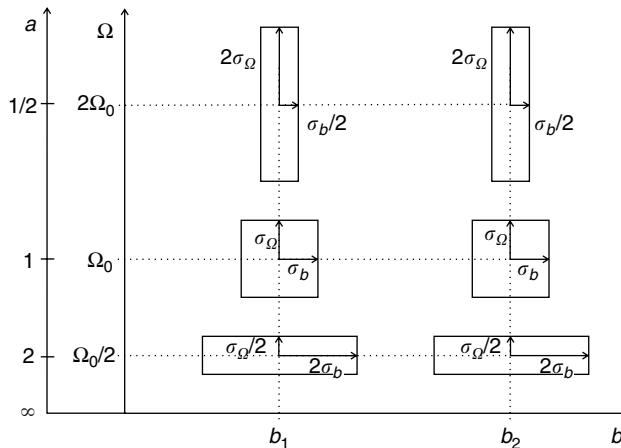


Fig. 10.12. Resolution cells in the time \times frequency plane for the wavelet transform.

frequencies) the wavelet transform has low time resolution and high frequency resolution. Conversely, for small values of a , which correspond to large frequencies, the wavelet transform has high time resolution and low frequency resolution. This can be illustrated by the resolution cells in Figure 10.12, where one may note that the uncertainty principle from relation (10.15) still holds.

From the above analysis, one can see that the wavelet transform is adequate for analyzing signals having features of different sizes. For each feature size, there is a scale a in which it will be best analyzed or represented. This is particularly useful in image processing, which is thoroughly explored in Section 10.9.

There is another way of interpreting the fact that the wavelet transform depends mainly on values of $X(\Omega)$ within the interval $[(\Omega_0/a) - (\sigma_\Omega/a), (\Omega_0/a) + (\sigma_\Omega/a)]$. Since $\psi(t)$ can be viewed as the impulse response of a bandpass filter, this is equivalent to stating that the wavelet $\psi_{a,b}(t)$ is the impulse response of a bandpass filter having center frequency Ω_0/a and bandwidth $2\sigma_\Omega/a$. Thus, the $\psi_{a,b}(t)$ functions represent a set of bandpass filters whose values of the quality factor Q (defined as the ratio between the center frequency Ω_0 and the filter bandwidth $2\sigma_\Omega$) are independent of the scale a . Therefore, a wavelet transform is equivalent to the analysis of a signal in the frequency domain using bandpass filters having variable center frequencies, which depend on the scale a , but with constant Q .

10.4.3 Sampling the continuous-time wavelet transform: the discrete wavelet transform

As can be seen from Equation (10.34), the wavelet transform maps a one-dimensional function into a two-dimensional function. This dimensionality increase makes it extremely redundant. Thus, it looks natural that the original signal can be recovered from the wavelet transform computed only on a discrete grid. This is indeed the case, and Ingrid Daubechies (1990) has made an extensive investigation of this problem.

By looking at the resolution cells in Figure 10.12, we can see that, when the scale a increases, the central frequency and the width of the resolution cells in the frequency direction decrease, and so more resolution cells are needed to cover that region of the $\Omega \times b$ plane. Therefore, a natural choice for the discretization of a would be $a = a_0^m$, with $a_0 > 1$ and $m \in \mathbb{Z}$. Since the discretization of b corresponds to a sampling in time, its sampling frequency must be proportional to the bandwidth of the signal to be sampled, which in turn is inversely proportional to the scale a . Therefore, it is intuitive to choose $b = nb_0a_0^m$. With these choices of a and b , the discrete wavelet transform $X_W(m, n)$ of $x(t)$ becomes

$$X_W(m, n) = a_0^{-m/2} \int_{-\infty}^{\infty} \psi_{m,n}^*(t)x(t) dt \quad (10.43)$$

$$\psi_{m,n}(t) = a_0^{-m/2}\psi(a_0^{-m}t - nb_0). \quad (10.44)$$

If $a_0 = 2$ and $b_0 = 1$, then there are choices of $\psi(t)$ such that the functions $\psi_{m,n}(t)$, for $m, n \in \mathbb{Z}$, form an orthonormal basis of $L^2(\mathbb{R})$, the space of square integrable functions in \mathbb{R} (Daubechies, 1990). This implies that any function $x(t) \in L^2(\mathbb{R})$ can be expressed as

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} \psi_{m,n}(t) \quad (10.45)$$

$$c_{m,n} = \int_{-\infty}^{\infty} \psi_{m,n}^*(t)x(t) dt. \quad (10.46)$$

The $c_{m,n}$ are the wavelet transform coefficients of $x(t)$. Note that the above equations are the same as Equations (10.5) and (10.6), in the case that $\bar{\psi}(t) = \psi(t)$. This is so because the diagram depicted in Figure 10.6 implements exactly the discretization of the continuous-time wavelet transform given by Equations (10.43) and (10.44) when $a_0 = 2$ and $b_0 = 1$. This implies that the samples of a continuous-time wavelet transform can be computed using the scheme in Figure 10.9, provided that the envelopes of the iterated filter banks in Figure 10.4 correspond to expansions and translations of $\psi(t)$.

It is interesting to observe that in this discretization of the wavelet transform, for every increment in m , the value of a doubles. This implies doubling the width in the time domain and halving the width in the frequency domain. The equivalent sampling grid is shown in Figure 10.13. By referring to Equation (10.35), this is equivalent to having a signal analyzed in frequency channels having widths of one octave, as illustrated in Figure 10.8.

If $x(t)$ is equal to $\psi_{k,l}(t)$, then Equation (10.45) becomes

$$\psi_{k,l}(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} \psi_{m,n}(t). \quad (10.47)$$

In order for the above equation to be valid, one must have $c_{m,n} = \delta(m - k)\delta(n - l)$. In this case, Equation (10.46) implies that

$$\int_{-\infty}^{\infty} \psi_{m,n}^*(t)\psi_{k,l}(t) dt = \delta(m - k)\delta(n - l). \quad (10.48)$$

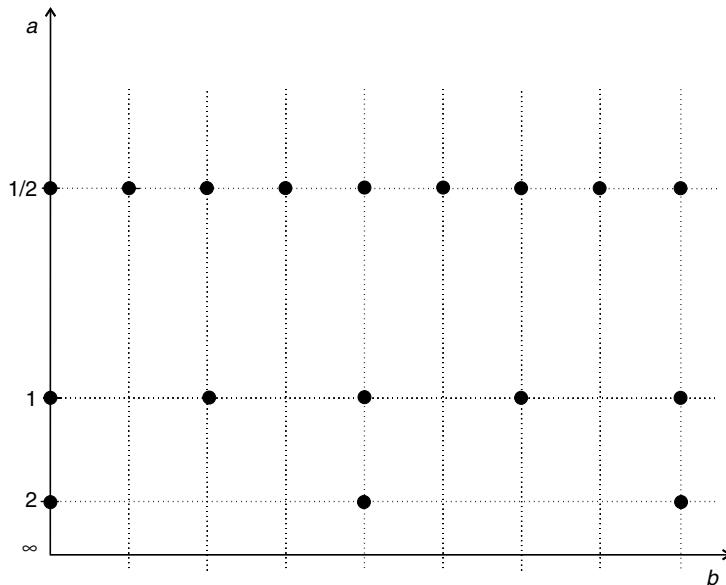


Fig. 10.13. Discretization of the wavelet transform in Equations (10.43) and (10.44) for $a_0 = 2$ and $b_0 = 1$.

This is equivalent to saying that the functions $\psi_{m,n}(t)$, for all $m, n \in \mathbb{Z}$, are orthonormal (Kolmogorov & Fomin, 1962). As detailed in Section 3.6.2, orthonormality is usually expressed using the inner product notation as

$$\langle \psi_{k,l}(t), \psi_{m,n}(t) \rangle = \delta(m - k)\delta(n - l), \quad (10.49)$$

where the inner product $\langle g(t), f(t) \rangle$ between the two functions $f(t)$ and $g(t)$ is defined as

$$\langle g(t), f(t) \rangle = \int_{-\infty}^{\infty} f^*(t)g(t) dt. \quad (10.50)$$

Since Equations (10.45) and (10.46) are valid for any function $x(t) \in L^2(\mathbb{R})$, then it can be said that the functions $\psi_{m,n}(t)$, for all $m, n \in \mathbb{Z}$, form an orthonormal basis of $L^2(\mathbb{R})$.

Other choices of a_0 can lead to other orthonormal bases. For example, Kovacević (1991) has described some discrete wavelet transforms using fractional values of a_0 . However, we will restrict our presentation to the dyadic cases; that is, to wavelet transforms in which $a_0 = 2$ and $b_0 = 1$.

The wavelet transform as defined by Equations (10.5) and (10.6), namely,

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} \bar{\psi}_{m,n}(t) \quad (10.51)$$

$$c_{m,n} = \int_{-\infty}^{\infty} x(t) \psi_{m,n}^*(t) dt, \quad (10.52)$$

where

$$\psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n) \quad (10.53)$$

$$\bar{\psi}_{m,n}(t) = 2^{-m/2}\bar{\psi}(2^{-m}t - n), \quad (10.54)$$

is not orthogonal. In this case, we refer to it as a biorthogonal wavelet transform (refer to Section 9.4.3 for a discussion on orthogonality and biorthogonality).

Biorthogonal wavelet transforms are characterized by two wavelets: the analysis wavelet, $\psi(t)$ and the synthesis wavelet $\bar{\psi}(t)$ (Cohen *et al.*, 1992; Vetterli & Herley, 1992). Equations (10.51)–(10.54) indicate that any function $x(t) \in L^2(\mathbb{R})$ can be decomposed as a linear combination of contractions, expansions, and translations of the synthesis wavelet $\bar{\psi}(t)$. The weights of the expansion can be computed via the inner product of $x(t)$ with expansions, contractions, and translations of the analysis wavelet $\psi(t)$.

Functions $\psi_{m,n}(t)$ do not comprise an orthogonal set, so neither do the functions $\bar{\psi}_{m,n}(t)$. However, functions $\psi_{m,n}(t)$ are orthogonal to $\bar{\psi}_{m,n}(t)$ (see Figure 9.24 and Equations (9.97)–(9.100)). This means that

$$\langle \psi_{m,n}(t), \bar{\psi}_{k,l}(t) \rangle = \delta(m - k)\delta(n - l). \quad (10.55)$$

10.5 Multiresolution representation

The concept of multiresolution signal representation (Mallat, 1989b) provides interesting insights on wavelet transforms, as well as a deeper understanding of their connection with filter banks.

Suppose a function $\phi(t)$ such that the set $\phi(t - n)$, for all $n \in \mathbb{Z}$, is orthonormal. Define V_0 as the space generated by this set. Analogously, define V_m as the space generated by $2^{-m/2}\phi(2^{-m}t - n)$.

Suppose also that $\phi(t)$ is the solution of the following two-scale difference equation (Daubechies, 1988):

$$\phi(t) = \sum_{n=-\infty}^{\infty} c_n \sqrt{2}\phi(2t - n). \quad (10.56)$$

Since the orthonormality of the set $\phi(t - n)$ along t implies the orthonormality of the set $\sqrt{2}\phi(2t - n)$, we have that c_n is related to $\phi(t)$ by the following equation:

$$c_n = \int_{-\infty}^{\infty} \phi(t) \sqrt{2}\phi^*(2t - n) dt. \quad (10.57)$$

From Equation (10.56), it follows immediately by induction that, if $i, j \in \mathbb{Z}$, with $i > j$, there are constants α_n^{ij} such that

$$\phi(2^{-i}t) = \sum_{n=-\infty}^{\infty} \alpha_n^{ij} \phi(2^{-j}t - n). \quad (10.58)$$

This means that the functions that generate the space V_i are also in V_j . This implies that $V_i \subset V_j$, for $i > j$. By induction, then:

$$\cdots \supset V_{-2} \supset V_{-1} \supset V_0 \supset V_1 \supset \cdots \quad (10.59)$$

Interpreting V_0 as the space of functions having resolution 2^0 , V_{-1} can be interpreted as the space of functions having the higher resolution 2^1 , which contains V_0 ; V_1 can be interpreted as the space of functions having the lower resolution 2^{-1} , which is contained in V_0 . Hence, the V_m , for increasing m , can be viewed as spaces of decreasing resolution.

The function $\phi(t)$ in Equation (10.56) must also be such that

$$\bigcup_{j=-\infty}^{\infty} V_j = L^2(\mathbb{R}) \quad (10.60)$$

$$\bigcap_{j=-\infty}^{\infty} V_j = \{0\}. \quad (10.61)$$

Defining now W_j as the orthogonal complement of V_j in V_{j-1} (Kolmogorov & Fomin, 1962):

$$W_j \perp V_j \text{ and } W_j \oplus V_j = V_{j-1}, \quad (10.62)$$

where \oplus denotes the orthogonal sum operation, which corresponds to the linear closure of two orthogonal spaces. In this context, W_j can be seen as the amount of “detail” added when going from resolution V_j to the larger resolution V_{j-1} . This hierarchy of spaces is depicted in Figure 10.14.

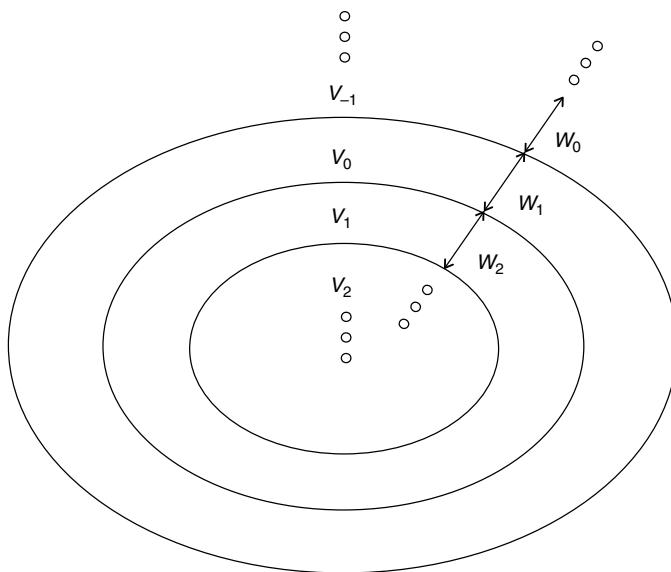


Fig. 10.14.

Geometric representation of multiresolution spaces.

From Equations (10.59) and (10.62), any function $g(t) \in W_0$ also belongs to V_{-1} , as also seen in Figure 10.14. Therefore, such $g(t)$ can be expressed as

$$g(t) = \sum_{n=-\infty}^{\infty} d_n \sqrt{2} \phi(2t - n). \quad (10.63)$$

If we define $\psi(t) \in V_{-1}$ as

$$\psi(t) = \sum_{n=-\infty}^{\infty} (-1)^n c_{1-n} \sqrt{2} \phi(2t - n), \quad (10.64)$$

where the c_n are as in Equation (10.56), then it can be shown that $\psi(t) \in W_0$, and $\psi(t - n)$, for all $n \in \mathbb{Z}$, is an orthonormal basis for W_0 (Daubechies, 1988; Mallat, 1989b); see Exercise 10.5. More generally, $2^{-m/2} \psi(2^{-m} t - n)$ is an orthonormal basis for W_m .

From Equation (10.62), this implies that the functions $\psi(t - n)$ are orthogonal to the functions $\phi(t - m)$. Summarizing the orthogonality conditions, we have

$$\langle \phi(t - m), \phi(t - n) \rangle = \delta(m - n) \quad (10.65)$$

$$\langle \psi(t - m), \psi(t - n) \rangle = \delta(m - n) \quad (10.66)$$

$$\langle \psi(t - m), \phi(t - n) \rangle = 0. \quad (10.67)$$

From Equations (10.60)–(10.62) one can derive that

$$\cdots \oplus W_{-2} \oplus W_{-1} \oplus W_0 \oplus W_1 \cdots = L^2(\mathbb{R}). \quad (10.68)$$

Thus, the functions $\psi_{m,n}(t) = 2^{-m/2} \psi(2^{-m} t - n)$, for all $m, n \in \mathbb{Z}$, constitute an orthonormal basis of $L^2(\mathbb{R})$. This is equivalent to saying that any $f(t) \in L^2(\mathbb{R})$ can be written as

$$f(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \alpha_{m,n} \psi_{m,n}(t) \quad (10.69)$$

and

$$\alpha_{m,n} = \int_{-\infty}^{\infty} \psi_{m,n}^*(t) f(t) dt. \quad (10.70)$$

These equations are the same as Equations (10.45) and (10.46) respectively, indicating that they represent a discrete wavelet transform of $f(t)$ with mother wavelet $\psi(t)$. The wavelet transform coefficients $\alpha_{m,n}$ correspond to the projection of $f(t)$ onto a “detail space” W_m of resolution m . Therefore, a wavelet transform performs the decomposition of a signal into spaces of different resolutions. In the literature, these kinds of decomposition are in general referred to as multiresolution decompositions (Mallat, 1989b). They are another way of stating the property that wavelet transforms can conveniently represent features of different sizes; that is, in different resolutions, as discussed in Section 10.4.2.

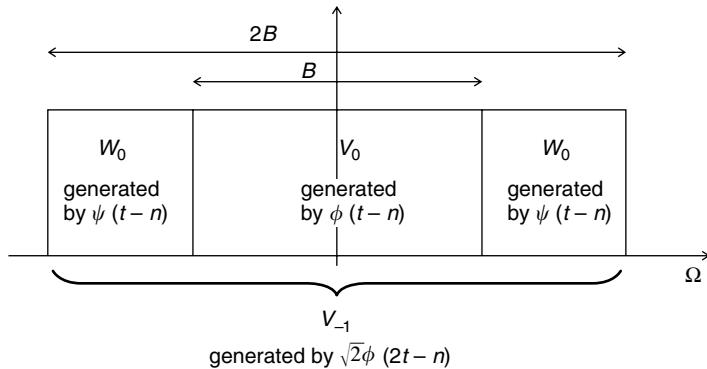


Fig. 10.15. Multiresolution decomposition in the frequency domain.

In the frequency domain the multiresolution decompositions can be understood in the following way: V_0 is the space generated by $\phi(t - n)$ and V_{-1} is the space generated by $\sqrt{2}\phi(2t - n)$, which has double the bandwidth of $\phi(t - n)$. Therefore, V_{-1} also contains functions with double the frequency content of V_0 , which means double the time resolution. W_0 is composed of the functions that are in V_{-1} but not in V_0 and, therefore, is contained in the bandpass region between the passbands of $\phi(t - n)$ and $\sqrt{2}\phi(2t - n)$, which should be the passband of $\psi(t - n)$. This spectral reasoning is clarified in Figure 10.15.

Note that the function $\phi(t)$ is the same as the scaling function defined in Section 10.2.3. It is often referred to as the scaling function of the multiresolution representation, while $\psi(t)$ is referred to as its wavelet.

Despite having nice properties, orthogonal wavelet transforms have an important limitation. As we have seen in Section 9.7, orthogonal filter banks cannot have linear phase. Since wavelets are the envelope of the impulse response of iterated filter banks (see Figures 10.2–10.6), the wavelet $\psi(t)$ cannot, at the same time, be orthogonal and be the impulse response of a linear-phase filter (Vetterli & Herley, 1992). This is particularly significant in image processing applications, because the phase of an image signal carries very important information (Field, 1993). Thus, it is beneficial to use the biorthogonal wavelet transforms described in Equations (10.51)–(10.55), which can have linear phase. In the following we analyze the biorthogonal multiresolution representation.

10.5.1 Biorthogonal multiresolution representation

Let $\psi(t)$ and $\bar{\psi}(t)$ be the analysis and synthesis wavelets respectively and let $\phi(t)$ and $\bar{\phi}(t)$ be the corresponding analysis and synthesis scaling functions respectively. Analogous to the orthogonal case, let $\phi(t)$ and $\bar{\phi}(t)$ be such that

$$\phi(t) = \sum_{n=-\infty}^{\infty} c_n \sqrt{2}\phi(2t - n) \quad (10.71)$$

$$\bar{\phi}(t) = \sum_{n=-\infty}^{\infty} \bar{c}_n \sqrt{2} \bar{\phi}(2t - n). \quad (10.72)$$

Supposing also that

$$\langle 2^{-m/2}\phi(2^{-m}t - n), 2^{-m/2}\bar{\phi}(2^{-m}t - k) \rangle = \delta(n - k), \quad (10.73)$$

we have

$$c_n = \int_{-\infty}^{\infty} \phi(t) \sqrt{2} \bar{\phi}^*(2t - n) dt \quad (10.74)$$

$$\bar{c}_n = \int_{-\infty}^{\infty} \bar{\phi}(t) \sqrt{2} \phi^*(2t - n) dt. \quad (10.75)$$

Functions $\phi(t)$ and $\bar{\phi}(t)$ defined this way generate two hierarchies of subspaces as the one in Equation (10.59) (Cohen *et al.*, 1992):

$$\phi(t) : \cdots \supset V_{-2} \supset V_{-1} \supset V_0 \supset V_1 \cdots \quad (10.76)$$

$$\bar{\phi}(t) : \cdots \supset \bar{V}_{-2} \supset \bar{V}_{-1} \supset \bar{V}_0 \supset \bar{V}_1 \cdots \quad (10.77)$$

Suppose also, for V_j and \bar{V}_j , that

$$\bigcup_{j=-\infty}^{\infty} V_j = L^2(\mathbb{R}) \quad (10.78)$$

$$\bigcap_{j=-\infty}^{\infty} V_j = \{0\} \quad (10.79)$$

$$\bigcup_{j=-\infty}^{\infty} \bar{V}_j = L^2(\mathbb{R}) \quad (10.80)$$

$$\bigcap_{j=-\infty}^{\infty} \bar{V}_j = \{0\}. \quad (10.81)$$

Defining W_j and \bar{W}_j such that

$$V_{j-1} = V_j + W_j \quad (10.82)$$

$$\bar{V}_{j-1} = \bar{V}_j + \bar{W}_j \quad (10.83)$$

$$W_j \perp \bar{V}_j \text{ and } \bar{W}_j \perp V_j. \quad (10.84)$$

Here, $A + B$ denotes the linear closure of A and B , or the subspace generated by all the linear combinations of functions in A and B (Kolmogorov & Fomin, 1962). It must be noted that the linear closure $+$ in Equations (10.82) and (10.83) differs from the orthogonal sum \oplus

operation in Equation (10.62), because, in the biorthogonal multiresolution representation considered in the present section, V_j and W_j are not orthogonal to each other.

Once again, W_j can be interpreted as the amount of “detail” added when going from “resolution” V_j to V_{j-1} , and \overline{W}_j is the amount of “detail” added when going from “resolution” \overline{V}_j to \overline{V}_{j-1} . However, unlike the orthogonal case, the W_j in one set of spaces is orthogonal only to the \overline{V}_j in the other set of spaces, not to V_j .

Let $\psi(t)$ and $\overline{\psi}(t)$ be bases for W_j and \overline{W}_j respectively. Since, from Equations (10.82) and (10.83), $W_j \in V_{j-1}$ and $\overline{W}_j \in \overline{V}_{j-1}$, $\psi(t)$ and $\overline{\psi}(t)$ can be expressed as

$$\psi(t) = \sum_{n=-\infty}^{\infty} d_n \sqrt{2} \phi(2t - n) \quad (10.85)$$

$$\overline{\psi}(t) = \sum_{n=-\infty}^{\infty} \overline{d}_n \sqrt{2} \overline{\phi}(2t - n), \quad (10.86)$$

where, from Equation (10.73):

$$d_n = \int_{-\infty}^{\infty} \psi(t) \sqrt{2} \overline{\phi}^*(2t - n) dt \quad (10.87)$$

$$\overline{d}_n = \int_{-\infty}^{\infty} \overline{\psi}(t) \sqrt{2} \phi^*(2t - n) dt. \quad (10.88)$$

The functions $\phi(t)$, $\overline{\phi}(t)$, $\psi(t)$, and $\overline{\psi}(t)$ defined as above satisfy the following biorthogonality conditions:

$$\langle \phi(t), \overline{\phi}(t - m) \rangle = \delta(m) \quad (10.89)$$

$$\langle \psi(t), \overline{\psi}(t - m) \rangle = \delta(m) \quad (10.90)$$

$$\langle \phi(t), \overline{\psi}(t - m) \rangle = 0 \quad (10.91)$$

$$\langle \psi(t), \overline{\phi}(t - m) \rangle = 0. \quad (10.92)$$

From Equations (10.76)–(10.83), we also get that

$$\cdots + \overline{W}_{-2} + \overline{W}_{-1} + \overline{W}_0 + \overline{W}_1 + \cdots = L^2(\mathbb{R}). \quad (10.93)$$

The above equation together with Equations (10.51) and (10.53) imply that, as in the orthogonal case, a biorthogonal wavelet transform involves the projection of a function onto the detail spaces \overline{W}_j (Cohen *et al.*, 1992).

In the next section we use the concept of multiresolution decomposition to present how wavelet transforms of digital signals can be computed, as well as their relation to the two-band perfect reconstruction filter banks described in Section 9.5.

10.6 Wavelet transforms and filter banks

In the real world, every function is measured with a finite resolution. Without any loss of generality, we can assume that \bar{V}_0 is this resolution.²

Since \bar{V}_j is the space generated by the functions $2^{-j/2}\bar{\phi}(2^{-j}t - n)$, the projection $x_j(t)$ of $x(t)$ onto \bar{V}_j is equal to

$$x_j(t) = \sum_{n=-\infty}^{\infty} x_{j,n} 2^{-j/2} \bar{\phi}(2^{-j}t - n) \quad (10.94)$$

$$x_{j,n} = \int_{-\infty}^{\infty} x(t) 2^{-j/2} \bar{\phi}(2^{-j}t - n) dt. \quad (10.95)$$

Equation (10.95) can be interpreted as the coefficients $x_{j,n}$ being obtained by filtering $x(t)$ with a continuous-time filter having impulse response $2^{-j/2}\bar{\phi}(-2^{-j}t)$, and sampling the resulting continuous-time function at the sampling times $t_n = 2^j n$. Referring to Figure 10.15, the filtering process would serve to reduce the bandwidth of $x(t)$, and consequently its time resolution. Therefore, the resulting function $x_j(t)$, having limited resolution, can be represented unambiguously by the coefficients $x_{j,n}$.

Since, from Equation (10.71), $\phi(t) = \sum_{k=-\infty}^{\infty} c_k \sqrt{2} \phi(2t - k)$, we have

$$2^{-j/2}\bar{\phi}(2^{-j}t - n) = \sum_{k=-\infty}^{\infty} c_k 2^{(1-j)/2} \phi(2^{1-j}t - 2n - k), \quad (10.96)$$

which, when substituted into Equation (10.95), results in

$$\begin{aligned} x_{j,n} &= \int_{-\infty}^{\infty} x(t) \sum_{k=-\infty}^{\infty} c_k^* 2^{(1-j)/2} \phi^*(2^{1-j}t - 2n - k) dt \\ &= \sum_{k=-\infty}^{\infty} c_k^* \int_{-\infty}^{\infty} x(t) 2^{(1-j)/2} \phi^*(2^{1-j}t - 2n - k) dt. \end{aligned} \quad (10.97)$$

The comparison of the above expression with Equation (10.95) implies that

$$x_{j,n} = \sum_{k=-\infty}^{\infty} c_k^* x_{j-1,2n+k}. \quad (10.98)$$

By defining

$$h_0(k) = c_{-k}^*, \quad (10.99)$$

² Since orthogonal wavelet transforms are a special case of the biorthogonal wavelet transforms, only the biorthogonal case will be analyzed here.

Equation (10.98) can be rewritten as

$$x_{j,n} = \sum_{k=-\infty}^{\infty} h_0(k) x_{j-1,2n-k}. \quad (10.100)$$

This equation means that the coefficients $x_{j,n}$ of the approximation of $x(t)$ at resolution 2^{-j} can be obtained, from the coefficients $x_{j-1,n}$ of the approximation of $x(t)$ at the higher resolution 2^{1-j} , by filtering them with a digital filter having impulse response $h_0(k)$, with $k \in \mathbb{Z}$, and subsampling the result by a factor of 2, as detailed in Equation (8.13) and depicted in Figure 10.2. This result is not surprising, because, since the resolution of the space V_j is half of the one of the space V_{j-1} , then V_j should roughly have half of the frequency content of V_{j-1} (as determined by the filtering operation with $h_0(k)$) and should, therefore, be represented by a nonredundant transform with only half of the number of coefficients (as yielded by the decimation-by-2 operation).

Since \overline{W}_j is the space generated by the functions $2^{-j/2}\overline{\psi}(2^{-j}t - n)$, the projection $\check{x}_j(t)$ of $x(t)$ onto \overline{W}_j is equal to

$$\check{x}_j(t) = \sum_{n=-\infty}^{\infty} \check{x}_{j,n} 2^{-j/2} \overline{\psi}(2^{-j}t - n) \quad (10.101)$$

$$\check{x}_{j,n} = \int_{-\infty}^{\infty} x(t) 2^{-j/2} \psi^*(2^{-j}t - n) dt. \quad (10.102)$$

Similar to Equation (10.95), Equation (10.102) can be interpreted as the coefficients $\check{x}_{j,n}$ being obtained by filtering $x(t)$ with a continuous-time filter having impulse response $2^{-j/2}\psi(-2^{-j}t)$ and sampling the resulting continuous-time function at the sampling times $t_n = 2^j n$.

As, from Equation (10.85), $\psi(t) = \sum_{k=-\infty}^{\infty} d_k \sqrt{2} \phi(2t - k)$, we have

$$2^{-j/2}\psi(2^{-j}t - n) = \sum_{k=-\infty}^{\infty} d_k 2^{(1-j)/2} \phi(2^{1-j}t - 2n - k), \quad (10.103)$$

which, when substituted in Equation (10.102), results in

$$\begin{aligned} \check{x}_{j,n} &= \int_{-\infty}^{\infty} x(t) \sum_{k=-\infty}^{\infty} d_k^* 2^{(1-j)/2} \phi^*(2^{1-j}t - 2n - k) dt \\ &= \sum_{k=-\infty}^{\infty} d_k^* \int_{-\infty}^{\infty} x(t) 2^{(1-j)/2} \phi^*(2^{1-j}t - 2n - k) dt. \end{aligned} \quad (10.104)$$

Comparing this expression with Equation (10.102), we have that

$$\check{x}_{j,n} = \sum_{k=-\infty}^{\infty} d_k^* x_{j-1,2n+k}. \quad (10.105)$$

By defining

$$h_1(k) = d_{-k}^*, \quad (10.106)$$

Equation (10.105) can be rewritten as

$$\check{x}_{j,n} = \sum_{k=-\infty}^{\infty} h_1(k) x_{j-1,2n-k}. \quad (10.107)$$

This relationship indicates that the coefficients of the detail signal $\check{x}_j(t)$ can be obtained from the coefficients of $x_{j-1}(t)$ by filtering them with $h_1(k)$ and subsampling by a factor of 2, as detailed in Equation (8.13) and also illustrated in Figure 10.2.

Equations (10.100) and (10.107) show how to go from resolution 2^{1-j} to the smaller resolution 2^{-j} . Assuming that the digital representation of $x(t)$ is given by the coefficients $x_{-1,n}$ (see Section 10.3), we have that the discrete wavelet transform of $x(t)$, the coefficients $\check{x}_{j,n}$, can be computed recursively from Equations (10.100) and (10.107). This is again illustrated in Figure 10.2, where $\check{x}_{j,n} = c_{j,n}$.

Now we will deal with the problem of going from resolution 2^{-j} to the higher resolution 2^{1-j} , with the help of the detail signal. Being able to do so is equivalent to being able to recover the digital representation of the signal from its wavelet coefficients.

From Equation (10.83), the projection $x_{j-1}(t)$ of $x(t)$ onto \overline{V}_{j-1} can be decomposed as the sum of the projection $x_j(t)$ of $x(t)$ onto \overline{V}_j and the projection $\check{x}_j(t)$ of $x(t)$ onto the detail space \overline{W}_j ; that is:

$$x_{j-1}(t) = x_j(t) + \check{x}_j(t). \quad (10.108)$$

Then, substituting Equations (10.94) and (10.101) into Equation (10.108), we have

$$x_{j-1}(t) = \sum_{k=-\infty}^{\infty} x_{j,k} 2^{-j/2} \bar{\phi}(2^{-j}t - k) + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} 2^{-j/2} \bar{\psi}(2^{-j}t - k). \quad (10.109)$$

However, since

$$x_{j-1}(t) = \sum_{l=-\infty}^{\infty} x_{j-1,l} 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - l), \quad (10.110)$$

from Equation (10.75) and then from equation (10.109), we have that

$$\begin{aligned} x_{j-1,l} &= \int_{-\infty}^{\infty} x_{j-1}(t) 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - l) dt \\ &= \sum_{k=-\infty}^{\infty} x_{j,k} \int_{-\infty}^{\infty} 2^{-j/2} \bar{\phi}(2^{-j}t - k) 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - l) dt \\ &\quad + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} \int_{-\infty}^{\infty} 2^{-j/2} \bar{\psi}(2^{-j}t - k) 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - l) dt. \end{aligned} \quad (10.111)$$

From Equations (10.72) and (10.86):

$$2^{-j/2}\bar{\phi}(2^{-j}t - k) = \sum_{n=-\infty}^{\infty} \bar{c}_n 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - 2k - n) \quad (10.112)$$

$$2^{-j/2}\bar{\psi}(2^{-j}t - k) = \sum_{n=-\infty}^{\infty} \bar{d}_n 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - 2k - n). \quad (10.113)$$

Substituting the above equations into Equation (10.111), we have

$$\begin{aligned} x_{j-1,l} &= \sum_{k=-\infty}^{\infty} x_{j,k} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \bar{c}_n 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - 2k - n) 2^{(1-j)/2} \phi^*(2^{1-j}t - l) dt \\ &\quad + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \bar{d}_n 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - 2k - n) 2^{(1-j)/2} \phi^*(2^{1-j}t - l) dt. \end{aligned} \quad (10.114)$$

From Equation (10.89):

$$\langle 2^{(1-j)/2} \phi(2^{1-j}t - n), 2^{(1-j)/2} \bar{\phi}(2^{1-j}t - m) \rangle = \delta(m - n). \quad (10.115)$$

Equation (10.114) then becomes

$$\begin{aligned} x_{j-1,l} &= \sum_{k=-\infty}^{\infty} x_{j,k} \sum_{n=-\infty}^{\infty} \bar{c}_n \delta(2k + n - l) + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} \sum_{n=-\infty}^{\infty} \bar{d}_n \delta(2k + n - l) \\ &= \sum_{k=-\infty}^{\infty} x_{j,k} \bar{c}_{l-2k} + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} \bar{d}_{l-2k}. \end{aligned} \quad (10.116)$$

By defining

$$\bar{c}_n = g_0(n) \quad (10.117)$$

$$\bar{d}_n = g_1(n), \quad (10.118)$$

Equation (10.116) can be rewritten as

$$x_{j-1,l} = \sum_{k=-\infty}^{\infty} x_{j,k} g_0(l - 2k) + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} g_1(l - 2k). \quad (10.119)$$

This equation means that the coefficients of the approximation $x_{j-1,l}$ of a signal at resolution 2^{1-j} can be obtained from the coefficients of the approximation of the signal at the smaller resolution 2^{-j} and the coefficients of the corresponding detail signal. In order to accomplish this, it suffices to upsample the coefficients of the approximation $x_{j,k}$ by a factor of two and to filter them with a digital filter having impulse response $g_0(k)$, summing the result to the

coefficients of the detail signal, $\check{x}_{j,k}$, upsampled by a factor of 2 and filtered by a digital filter with impulse response $g_1(k)$, as detailed in Equation (8.21) and presented in Figure 10.2.

Summarizing Equations (10.99), (10.100), (10.106), (10.107), (10.118), and (10.119), we have

$$\left. \begin{array}{l} c_n^* = h_0(-n) \\ d_n^* = h_1(-n) \\ \bar{c}_n = g_0(n) \\ \bar{d}_n = g_1(n) \\ x_{j,n} = \sum_{k=-\infty}^{\infty} h_0(k) x_{j-1,2n-k} \\ \check{x}_{j,n} = \sum_{k=-\infty}^{\infty} h_1(k) x_{j-1,2n-k} \\ x_{j-1,n} = \sum_{k=-\infty}^{\infty} x_{j,k} g_0(n-2k) + \sum_{k=-\infty}^{\infty} \check{x}_{j,k} g_1(n-2k) \end{array} \right\}. \quad (10.120)$$

From the above equations, we have that Equations (10.71)–(10.75) and (10.85)–(10.88) become

$$\phi(t) = \sum_{n=-\infty}^{\infty} h_0^*(n) \sqrt{2} \phi(2t+n) \quad (10.121)$$

$$\bar{\phi}(t) = \sum_{n=-\infty}^{\infty} g_0(n) \sqrt{2} \bar{\phi}(2t-n) \quad (10.122)$$

$$\psi(t) = \sum_{n=-\infty}^{\infty} h_1^*(n) \sqrt{2} \phi(2t+n) \quad (10.123)$$

$$\bar{\psi}(t) = \sum_{n=-\infty}^{\infty} g_1(n) \sqrt{2} \bar{\phi}(2t-n). \quad (10.124)$$

Therefore, the filter coefficients can be obtained from the wavelets and scaling functions by the following expressions:

$$h_0(n) = \int_{-\infty}^{\infty} \phi^*(t) \sqrt{2} \bar{\phi}(2t+n) dt \quad (10.125)$$

$$g_0(n) = \int_{-\infty}^{\infty} \bar{\phi}(t) \sqrt{2} \phi^*(2t-n) dt \quad (10.126)$$

$$h_1(n) = \int_{-\infty}^{\infty} \psi^*(t) \sqrt{2} \bar{\phi}(2t+n) dt \quad (10.127)$$

$$g_1(n) = \int_{-\infty}^{\infty} \bar{\psi}(t) \sqrt{2} \phi^*(2t-n) dt \quad (10.128)$$

From the above equations, we can confirm that, in the orthogonal case, when $\phi(t) = \bar{\phi}(t)$ and $\psi(t) = \bar{\psi}(t)$, we have that $h_0(n) = g_0^*(-n)$ and $h_1(n) = g_1^*(-n)$. Note that this is similar to the condition for orthogonality of filter banks in Equation (9.90).

The Fourier transforms of the wavelets, scaling functions, and filters can be related (Vetterli & Kovačević, 1995; Mallat, 1999) by taking the Fourier transforms of Equations (10.121)–(10.124). In the case of Equation (10.121) we have that

$$\begin{aligned}\Phi(\Omega) &= \sum_{n=-\infty}^{\infty} h_0^*(n) \frac{\sqrt{2}}{2} e^{j(\Omega/2)n} \Phi\left(\frac{\Omega}{2}\right) \\ &= \frac{1}{\sqrt{2}} \Phi\left(\frac{\Omega}{2}\right) \sum_{n=-\infty}^{\infty} h_0^*(n) e^{j(\Omega/2)n} \\ &= \frac{1}{\sqrt{2}} \Phi\left(\frac{\Omega}{2}\right) H_0^*(e^{j\Omega/2})\end{aligned}\quad (10.129)$$

Analogously, taking the Fourier transforms of Equations (10.122)–(10.124) we have

$$\bar{\Phi}(\Omega) = \frac{1}{\sqrt{2}} \bar{\Phi}\left(\frac{\Omega}{2}\right) G_0(e^{j\Omega/2}) \quad (10.130)$$

$$\Psi(\Omega) = \frac{1}{\sqrt{2}} \Phi\left(\frac{\Omega}{2}\right) H_1^*(e^{j\Omega/2}) \quad (10.131)$$

$$\bar{\Psi}(\Omega) = \frac{1}{\sqrt{2}} \bar{\Phi}\left(\frac{\Omega}{2}\right) G_1(e^{j\Omega/2}) \quad (10.132)$$

and then, solving the recursions in Equations (10.129)–(10.132), we get

$$\Phi(\Omega) = \prod_{n=1}^{\infty} \frac{1}{\sqrt{2}} H_0^*(e^{j\Omega/2^n}) \quad (10.133)$$

$$\bar{\Phi}(\Omega) = \prod_{n=1}^{\infty} \frac{1}{\sqrt{2}} G_0(e^{j\Omega/2^n}) \quad (10.134)$$

$$\Psi(\Omega) = \frac{1}{\sqrt{2}} H_1^*(e^{j\Omega/2}) \prod_{n=2}^{\infty} \frac{1}{\sqrt{2}} H_0^*(e^{j\Omega/2^n}) \quad (10.135)$$

$$\bar{\Psi}(\Omega) = \frac{1}{\sqrt{2}} G_1(e^{j\Omega/2}) \prod_{n=2}^{\infty} \frac{1}{\sqrt{2}} G_0(e^{j\Omega/2^n}). \quad (10.136)$$

It is important to notice that, for the wavelet transform to be defined, the corresponding filter bank must provide perfect reconstruction. Also, it is interesting to note the similarity of the above equations with Equations (10.1)–(10.4) for the iterated filters of an octave-band filter bank.

10.6.1 Relations between the filter coefficients

Substituting Equations (10.121) and (10.122) in Equation (10.89), we have

$$\left\langle \sum_{k=-\infty}^{\infty} h_0^*(k)\sqrt{2}\phi(2t+k), \sum_{n=-\infty}^{\infty} g_0(n)\sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = \delta(m) \quad (10.137)$$

and then

$$\sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_0(k)g_0(n) \left\langle \sqrt{2}\phi(2t+k), \sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = \delta(m), \quad (10.138)$$

leading to

$$\sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_0(k)g_0(n)\delta(k+2m+n) = \delta(m). \quad (10.139)$$

This implies that

$$\sum_{n=-\infty}^{\infty} h_0(-2m-n)g_0(n) = \delta(m), \quad (10.140)$$

which is the same as writing

$$(h_0 * g_0)(-2m) = \delta(m), \quad (10.141)$$

where $*$ denotes the convolution operation between two discrete-time sequences.

Equation (10.141) means that all the even indexed elements of $(h_0 * g_0)$ are equal to zero with the exception of the zeroth element, which is equal to 1. This is equivalent to stating that the even powers of $H_0(z)G_0(z)$ are equal to zero with the exception of z^0 , which is equal to 1. This implies that

$$H_0(z)G_0(z) + H_0(-z)G_0(-z) = 2. \quad (10.142)$$

Now, substituting Equations (10.123) and (10.124) in Equation (10.90), we have

$$\begin{aligned} & \left\langle \sum_{k=-\infty}^{\infty} h_1^*(k)\sqrt{2}\phi(2t+k), \sum_{n=-\infty}^{\infty} g_1(n)\sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = \delta(m) \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_1(k)g_1(n) \left\langle \sqrt{2}\phi(2t+k), \sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = \delta(m) \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_1(k)g_1(n)\delta(k+2m+n) = \delta(m). \end{aligned} \quad (10.143)$$

This implies that

$$\sum_{n=-\infty}^{\infty} h_1(-2m-n)g_1(n) = \delta(m), \quad (10.144)$$

which can be rewritten as

$$(h_1 * g_1)(-2m) = \delta(m). \quad (10.145)$$

Equation (10.145) means that all the even powers of $H_1(z)G_1(z)$ are equal to zero with the exception of z^0 , which is equal to 1; that is:

$$H_1(z)G_1(z) + H_1(-z)G_1(-z) = 2. \quad (10.146)$$

Substituting Equations (10.121) and (10.124) in Equation (10.91), we have

$$\begin{aligned} & \left\langle \sum_{k=-\infty}^{\infty} h_0^*(k)\sqrt{2}\phi(2t+k), \sum_{n=-\infty}^{\infty} g_1(n)\sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = 0 \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_0(k)g_1(n) \left\langle \sqrt{2}\phi(2t+k), \sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = 0 \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_0(k)g_1(n)\delta(k+2m+n) = 0. \end{aligned} \quad (10.147)$$

This implies that

$$\sum_{n=-\infty}^{\infty} h_0(-2m-n)g_1(n) = 0, \quad (10.148)$$

which can be rewritten as

$$(h_0 * g_1)(-2m) = 0. \quad (10.149)$$

This is equivalent to saying that all the even powers of $H_0(z)G_1(z)$ are equal to zero; that is:

$$H_0(z)G_1(z) + H_0(-z)G_1(-z) = 0. \quad (10.150)$$

Finally, substituting Equations (10.122) and (10.123) in Equation (10.92), we have

$$\begin{aligned} & \left\langle \sum_{k=-\infty}^{\infty} h_1^*(k)\sqrt{2}\phi(2t+k), \sum_{n=-\infty}^{\infty} g_0(n)\sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = 0 \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_1(k)g_0(n) \left\langle \sqrt{2}\phi(2t+k), \sqrt{2}\bar{\phi}(2t-2m-n) \right\rangle = 0 \\ & \Rightarrow \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_1(k)g_0(n)\delta(k+2m+n) = 0. \end{aligned} \quad (10.151)$$

This implies that

$$\sum_{n=-\infty}^{\infty} h_1(-2m-n)g_0(n) = 0, \quad (10.152)$$

which can be rewritten as

$$(h_1 * g_0)(-2m) = 0. \quad (10.153)$$

This is equivalent to saying that all the even powers of $H_1(z)G_0(z)$ are equal to zero; that is:

$$H_1(z)G_0(z) + H_1(-z)G_0(-z) = 0. \quad (10.154)$$

Summarizing, Equations (10.142), (10.146), (10.150), and (10.154) form the following system of equations:

$$H_0(z)G_0(z) + H_0(-z)G_0(-z) = 2 \quad (10.155)$$

$$H_1(z)G_0(z) + H_1(-z)G_0(-z) = 0 \quad (10.156)$$

$$H_0(z)G_1(z) + H_0(-z)G_1(-z) = 0 \quad (10.157)$$

$$H_1(z)G_1(z) + H_1(-z)G_1(-z) = 2, \quad (10.158)$$

which can be written in matrix form as

$$\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} G_0(z) \\ G_0(-z) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (10.159)$$

$$\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} G_1(z) \\ G_1(-z) \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}. \quad (10.160)$$

Then, from Equation (10.159) we have

$$\begin{bmatrix} G_0(z) \\ G_0(-z) \end{bmatrix} = \frac{1}{H_0(z)H_1(-z) - H_0(-z)H_1(z)} \begin{bmatrix} H_1(-z) & -H_0(-z) \\ -H_1(z) & H_0(z) \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad (10.161)$$

and from Equation (10.160) we have

$$\begin{bmatrix} G_1(z) \\ G_1(-z) \end{bmatrix} = \frac{1}{H_0(z)H_1(-z) - H_0(-z)H_1(z)} \begin{bmatrix} H_1(-z) & -H_0(-z) \\ -H_1(z) & H_0(z) \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix}. \quad (10.162)$$

If one wants linear-phase filters, then we should look for FIR filters, as thoroughly discussed in Section 9.5. If the solutions for $G_0(z)$ and $G_1(z)$ in Equations (10.161) and (10.162) have to be FIR, then the condition

$$H_0(z)H_1(-z) - H_0(-z)H_1(z) = cz^{-r} \quad (10.163)$$

must be satisfied (Vetterli, 1986; Vetterli & Le Gall, 1989) in such a way that, from Equation (10.161), we have

$$G_0(z) = \frac{2}{c} z^r H_1(-z) \quad (10.164)$$

$$G_0(-z) = \frac{2}{c} z^r (-H_1(z)). \quad (10.165)$$

Comparing Equation (10.164) with Equation (10.165), we conclude that

$$(-1)^r = -1 \Rightarrow r = 2l + 1, \text{ with } l \in \mathbb{Z}. \quad (10.166)$$

Equation (10.164) then becomes

$$G_0(z) = \frac{2}{c} z^{2l+1} H_1(-z). \quad (10.167)$$

Now, from Equation (10.162), we have

$$G_1(z) = \frac{2}{c} z^{2l+1} (-H_0(-z)) \quad (10.168)$$

$$G_1(-z) = \frac{2}{c} z^{2l+1} H_0(z). \quad (10.169)$$

Therefore, the conditions that must be satisfied by the filters $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$, as given by Equations (10.142), (10.167), and (10.168), can be summarized as

$$H_0(z)G_0(z) + H_0(-z)G_0(-z) = 2 \quad (10.170)$$

$$G_0(z) = \frac{2}{c} z^{2l+1} H_1(-z) \quad (10.171)$$

$$G_1(z) = -\frac{2}{c} z^{2l+1} H_0(-z). \quad (10.172)$$

Note that these equations are very similar to Equations (9.123), (9.124), and (9.125). One important difference is that, in the present derivation, the overall delay has been forced to be zero by the biorthonormality of the wavelet transform. In fact, Equations (10.171) and (10.172) can be obtained by making $\Delta = -\frac{1}{2}$ in Equations (9.124) and (9.125), corresponding to an overall delay of $(2\Delta + 1) = 0$.

These conditions are obviously valid for purely orthogonal systems, a special case of biorthogonal ones. As we have seen earlier in this section, for orthogonal wavelets one has that $\phi(t) = \bar{\phi}(t)$ and $\psi(t) = \bar{\psi}(t)$, and then

$$h_0(n) = g_0^*(-n) \quad (10.173)$$

$$h_1(n) = g_1^*(-n). \quad (10.174)$$

In the z -transform domain, the above conditions correspond to

$$G_0(z) = H_0^*((z^{-1})^*) \quad (10.175)$$

$$G_1(z) = H_1^*((z^{-1})^*) \quad (10.176)$$

in such a manner that the conditions given by Equations (10.170) to (10.172) then become

$$H_0(z)H_0^*((z^{-1})^*) + H_0(-z)H_0^*(-(z^{-1})^*) = 2 \quad (10.177)$$

$$H_0^*((z^{-1})^*) = z^{2l+1}H_1(-z). \quad (10.178)$$

Substituting z by $e^{j\omega}$, Equation (10.177) can be rewritten as

$$\left|H_0(e^{j\omega})\right|^2 + \left|H_0(e^{j(\omega+\pi)})\right|^2 = 2. \quad (10.179)$$

This is the power complementary condition that arises in the project of CQF filter banks, as detailed in Section 9.7. This is not surprising, since CQF filter banks are orthogonal and, therefore, generate orthogonal wavelet transforms.

10.7 Regularity

From Equations (10.133)–(10.136), one can see that the wavelets and scaling functions are derived from the filter bank coefficients by infinite products. Therefore, in order for a wavelet to be defined, these infinite products must converge. In other words, a wavelet transform is not necessarily defined for every two-band perfect reconstruction filter bank. In fact, there are cases in which the envelope of the impulse responses of the equivalent filters of Equations (10.1)–(10.4) is not the same for every S (Vetterli & Kovačević, 1995; Mallat, 1999).

The regularity of a wavelet or scaling function is roughly speaking the number of continuous derivatives that a wavelet has. It gives a measure of the extent of convergence of the products in Equations (10.133)–(10.136). In order to define regularity more formally, we first define the following concept (Rioul, 1992; Mallat, 1999).

Definition 10.1. A function $f(t)$ is Lipschitz continuous of order α , with $0 < \alpha \leq 1$, if for all $x, h \in \mathbb{R}$, we have

$$|f(x+h) - f(x)| \leq ch^\alpha, \quad (10.180)$$

where c is a constant.

Using this definition, we have the regularity concept.

Definition 10.2. The Hölder regularity of a scaling function $\phi(t)$, such that $d^N\phi(t)/dt^N$ is Lipschitz continuous of order α , is $r = (N + \alpha)$, where N is integer and $0 < \alpha \leq 1$ (Rioul, 1992; Mallat, 1999).

It can be shown that, in order for a scaling function $\phi(t)$ to be regular, $H_0(z)$ must have enough zeros at $z = -1$. In addition, supposing that $\phi(t)$ generated by $H_0(z)$, as given in

Equation (10.133), has regularity r ; if we take

$$H'_0(z) = \left(\frac{1+z^{-1}}{2} \right) H_0(z), \quad (10.181)$$

then $\phi'(t)$ generated by $H'_0(z)$ will have regularity $(r+1)$ (Rioul, 1992; Mallat, 1999).

The regularity of a wavelet is the same as the regularity of the corresponding scaling function (Rioul, 1992).

10.7.1 Additional constraints imposed on the filter banks due to the regularity condition

If $\phi(t)$ and $\bar{\phi}(t)$ are regular, then the products in Equations (10.133) and (10.134) must converge. Using $\Omega = 0$ in Equation (10.129), we have

$$\Phi(0) = \frac{1}{\sqrt{2}} H_0^*(1) \Phi(0) \Rightarrow H_0(1) = \sqrt{2} \quad (10.182)$$

and then, from Equation (10.133), we get

$$\Phi(0) = 1. \quad (10.183)$$

Analogously, using $\Omega = 0$ in Equation (10.130), we have that

$$G_0(1) = \sqrt{2}, \quad (10.184)$$

which, when substituted in Equation (10.134), demands that

$$\bar{\Phi}(0) = 1. \quad (10.185)$$

Another condition can be imposed by substituting Equations (10.182) and (10.184) in Equation (10.170) for $z = 1$, leading to

$$H_0(-1)G_0(-1) = 0, \quad (10.186)$$

indicating that the product $H_0(z)G_0(z)$ must have a zero at $z = -1$.

In the orthogonal case, since, from Equation (10.175), $G_0(z) = H_0^*((z^{-1})^*)$, it can be concluded that the simple convergence of the product in Equation (10.133) forces the presence of one zero at $z = -1$. On the other hand, in the biorthogonal case, Equation (10.186) imposes a weaker condition, where only one of either $H_0(z)$ or $G_0(z)$ must have a zero at $z = -1$. Below, however, we see that an additional constraint forces $H_0(z)$ and $G_0(z)$ to present a zero at $z = -1$ even in the biorthogonal case.

As seen in Equation (10.38), a wavelet $\psi(t)$ must present a bandpass response in such a way that its Fourier transform at $\Omega = 0$ must be zero; that is, $\Psi(0) = 0$. Thus, substituting

$\Omega = 0$ in Equations (10.131) and (10.132), we have

$$\Psi(0) = \frac{1}{\sqrt{2}} H_1^*(1) \Phi(0) = 0 \quad (10.187)$$

$$\bar{\Psi}(0) = \frac{1}{\sqrt{2}} G_1(1) \bar{\Phi}(0) = 0. \quad (10.188)$$

Since $\Phi(0) = \bar{\Phi}(0) = 1$, then both $H_1(1)$ and $G_1(1)$ must be zero.

Summarizing all results in this subsection, for a regular wavelet, the filters must satisfy the following extra conditions:

$$H_0(-1) = 0 \quad (10.189)$$

$$G_0(-1) = 0 \quad (10.190)$$

$$H_0(1) = \sqrt{2} \quad (10.191)$$

$$G_0(1) = \sqrt{2}. \quad (10.192)$$

Equations (10.189) and (10.190) imply that the filters $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$ have to be normalized in order to generate a wavelet transform. Remember that, when deriving the wavelet transform from the octave-band filter bank in Section 10.2.2, it was supposed that the lowpass filters had enough zeros at $z = -1$. In fact, what was meant there was that the wavelets should be regular.

It is interesting to note that the conditions $H_0(1) = \sqrt{2}$ and $H_0(-1) = 0$ imply that $H_0(z)$ is a lowpass filter to a certain extent, the same being true for $G_0(z)$. These, together with Equations (10.171) and (10.172), imply that $H_1(z)$ and $G_1(z)$ are highpass filters.

Therefore, by referring once again to Figure 10.2, a wavelet transform can be viewed as an octave band sub-band analysis/synthesis system, in which the low-frequency band is recursively divided in low- and high-frequency bands. This implies that, in the frequency domain, a wavelet transform is equivalent to the frequency decomposition depicted in Figure 10.8

10.7.2 A practical estimate of regularity

There are several approaches to estimate the regularity of a scaling function or wavelet. Examples can be found in Daubechies (1988), Rioul (1992), and Villemoes (1992). The regularity estimate presented here is the one proposed by Rioul (1992).

Next, we describe how to estimate the regularity of the synthesis wavelet and scaling function. The regularity of the analysis wavelet and scaling function follows by substituting $G_0(z)$ by $H_0(z)$.

Supposing that $G_0(z)$ has at least $(N + 1)$ zeros at $z = -1$, the auxiliary function $F_N(z)$ is defined such that

$$G_0(z) = G_0(1) \left(\frac{1+z}{2} \right)^N F_N(z). \quad (10.193)$$

Let $(f_N^j)_n$ be the sequence whose z transform $F_N^j(z)$ is given by the following expression:

$$F_N^j(z) = \prod_{k=1}^j F_N(z^{2^{k-1}}). \quad (10.194)$$

Define α_N^j such that

$$2^{-j\alpha_N^j} = \max_{0 \leq n \leq 2^{j-1}} \left\{ \sum_{k=-\infty}^{\infty} |(f_N^j)_{n+k2^j}| \right\}. \quad (10.195)$$

Then, the Hölder regularity of the synthesis wavelet and scaling function is

$$r = N + \alpha_N, \quad \text{where } \alpha_N = \lim_{j \rightarrow \infty} \alpha_N^j. \quad (10.196)$$

The main advantage of this estimate is that it can be easily implemented in a digital computer, and it converges reasonably fast.

Figure 10.16 shows examples of wavelets with different regularities. For example, Figure 10.16a corresponds to the analysis wavelet generated by the filter bank described by Equations (9.135)–(9.138), and Figure 10.16b corresponds to the analysis wavelet generated by the filter bank described by Equations (9.131)–(9.134). From these figures, we notice that higher values of regularity correspond to smoother wavelets, as described above.

10.7.3 Number of vanishing moments

The presence of zeros at $z = -1$ for $H_0(z)$ and $G_0(z)$ lends an interesting property to the respective wavelets $\bar{\psi}(t)$ and $\psi(t)$ regarding their number of vanishing moments (Antonini *et al.*, 1992; Daubechies, 1993).

Suppose that $H_0(z)$ has N zeros at $z = -1$, or, in an equivalent way, $H_0(e^{j\omega})$ has N zeros at $\omega = \pi$. From Equation (10.172), this implies that $G_1(z)$ has N zeros at $z = 1$. Hence, $G_1(e^{j\omega})$ has N zeros at $\omega = 0$, and then

$$\left. \frac{d^n G_1(e^{j\omega})}{d\omega^n} \right|_{\omega=0} = 0, \quad \text{for } n = 0, 1, \dots, (N-1). \quad (10.197)$$

From Equation (10.132), one has

$$\frac{d\Psi(\Omega)}{d\Omega} = \frac{1}{2\sqrt{2}} \left[\frac{dG_1(e^{j\Omega/2})}{d\Omega} \bar{\Phi}\left(\frac{\Omega}{2}\right) + G_1(e^{j\Omega/2}) \frac{d\bar{\Phi}(\Omega/2)}{d\Omega} \right]; \quad (10.198)$$

it then follows that

$$\left. \frac{d^n \bar{\Psi}(\Omega)}{d\Omega^n} \right|_{\Omega=0} = 0, \quad \text{for } n = 0, 1, \dots, (N-1). \quad (10.199)$$

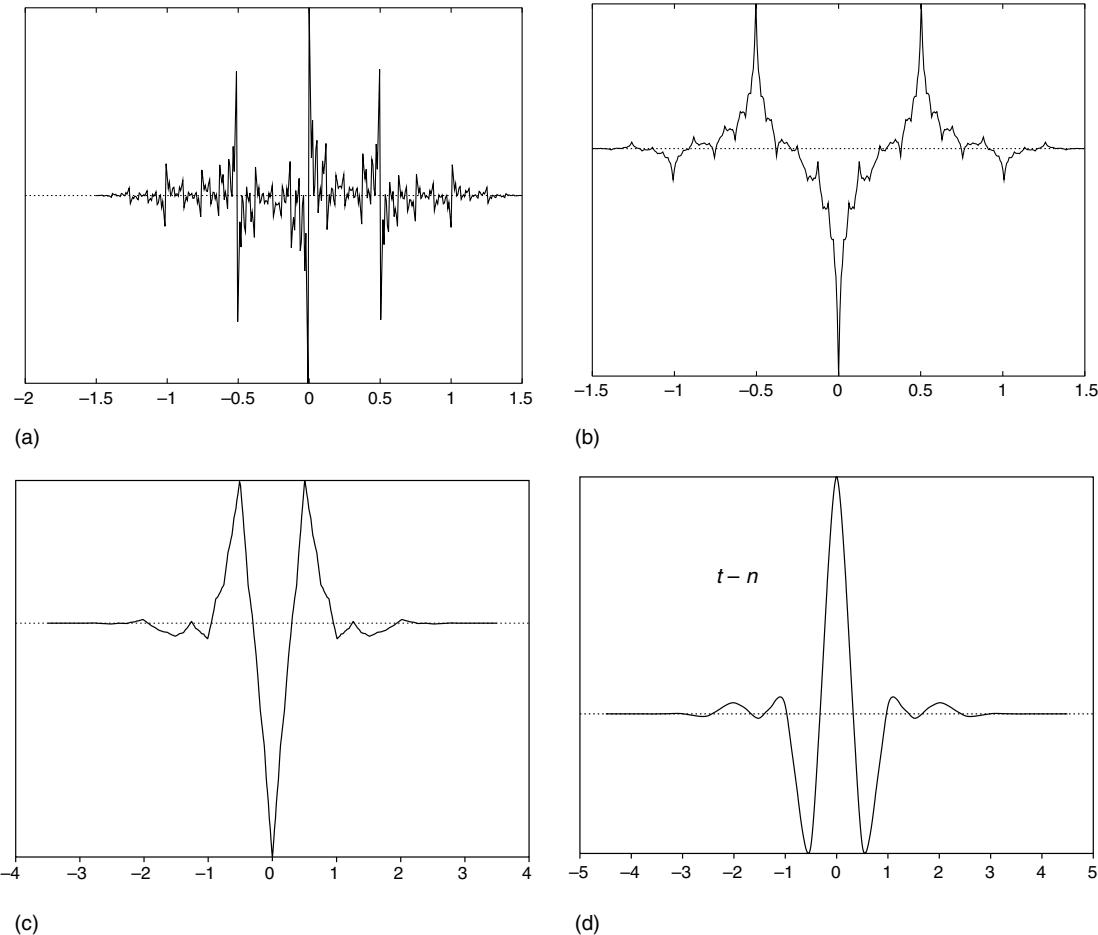


Fig. 10.16. Examples of wavelets with different regularities. (a) regularity = -1; (b) regularity = 0; (c) regularity = 1; (d) regularity = 2.

By definition:

$$\bar{\Psi}(\Omega) = \int_{-\infty}^{\infty} \bar{\psi}(t) e^{-j\Omega t} dt \quad (10.200)$$

such that

$$\frac{d^n \bar{\Psi}(\Omega)}{d\Omega^n} = \int_{-\infty}^{\infty} \bar{\psi}(t) (-jt)^n e^{-j\Omega t} dt = (-j)^n \int_{-\infty}^{\infty} t^n \bar{\psi}(t) e^{-j\Omega t} dt. \quad (10.201)$$

Therefore, conditions (10.199) correspond to

$$\int_{-\infty}^{\infty} t^n \bar{\psi}(t) dt = 0, \quad \text{for } n = 0, 1, \dots, (N-1), \quad (10.202)$$

which are equivalent to the synthesis wavelet $\bar{\psi}(t)$ having N vanishing moments.

Applying a similar reasoning, one can conclude that if $G_0(z)$ has N zeros at $z = -1$, then the analysis wavelet $\psi(t)$ has N vanishing moments. Referring to Equation (10.52), this means that the wavelet coefficients of any polynomial function of degree less than or equal to N are zero. Also, by referring to Equation (10.102), this implies that the coefficients $\check{x}_{j,n}$ of such a polynomial are equal to zero; therefore, the polynomial function $x(t)$ is represented only by the lowpass coefficients $x_{j,n}$, as given in Equation (10.95).

If a function $x(t)$ is analytic, then it can be expanded into a Taylor series as follows (Apostol, 1967):

$$x(t) = \sum_{k=0}^{\infty} \frac{1}{k!} \left. \frac{d^k x(t)}{dt^k} \right|_{t=t_0} (t - t_0)^k. \quad (10.203)$$

Therefore, if the analysis wavelet has N vanishing moments, only the terms of the expansion for $k > N$ will generate nonzero wavelet coefficients. If these terms are negligible, then the wavelet coefficients $\check{x}_{j,n}$ will be very small. This property can be useful in signal compression applications, because such functions could be represented by a few significant coefficients $x_{j,n}$ (Equation (10.95)), and negligible $\check{x}_{i,n}$, for $i \leq j$ (Equation (10.102)), (Antonini *et al.*, 1992).

10.8 Examples of wavelets

Every two-band perfect reconstruction filter bank with $H_0(z)$ having enough zeros at $z = -1$ has corresponding analysis and synthesis wavelets and scaling functions. For example, the filter bank described by Equations (9.11)–(9.14), normalized such that Equation (9.161) is satisfied, generates the so-called Haar wavelet. It is the only orthogonal wavelet that has linear phase (Vetterli & Kovačević, 1995; Mallat, 1999). The corresponding scaling function and wavelets are shown in Figure 10.17.

The wavelets and scaling functions corresponding to the symmetric short-kernel filter bank, described by Equations (9.131)–(9.134), are depicted in Figure 10.18.

A good example of an orthogonal wavelet is the Daubechies wavelet whose filters have length 4. It is also an example of CQF banks, seen in Section 9.7. The filters are

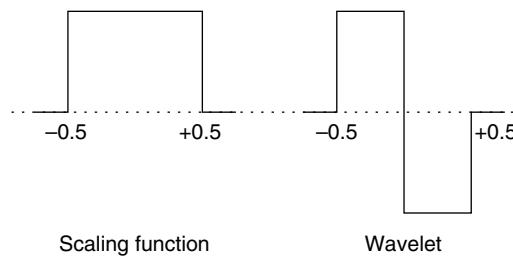


Fig. 10.17.

Haar wavelet and scaling function.

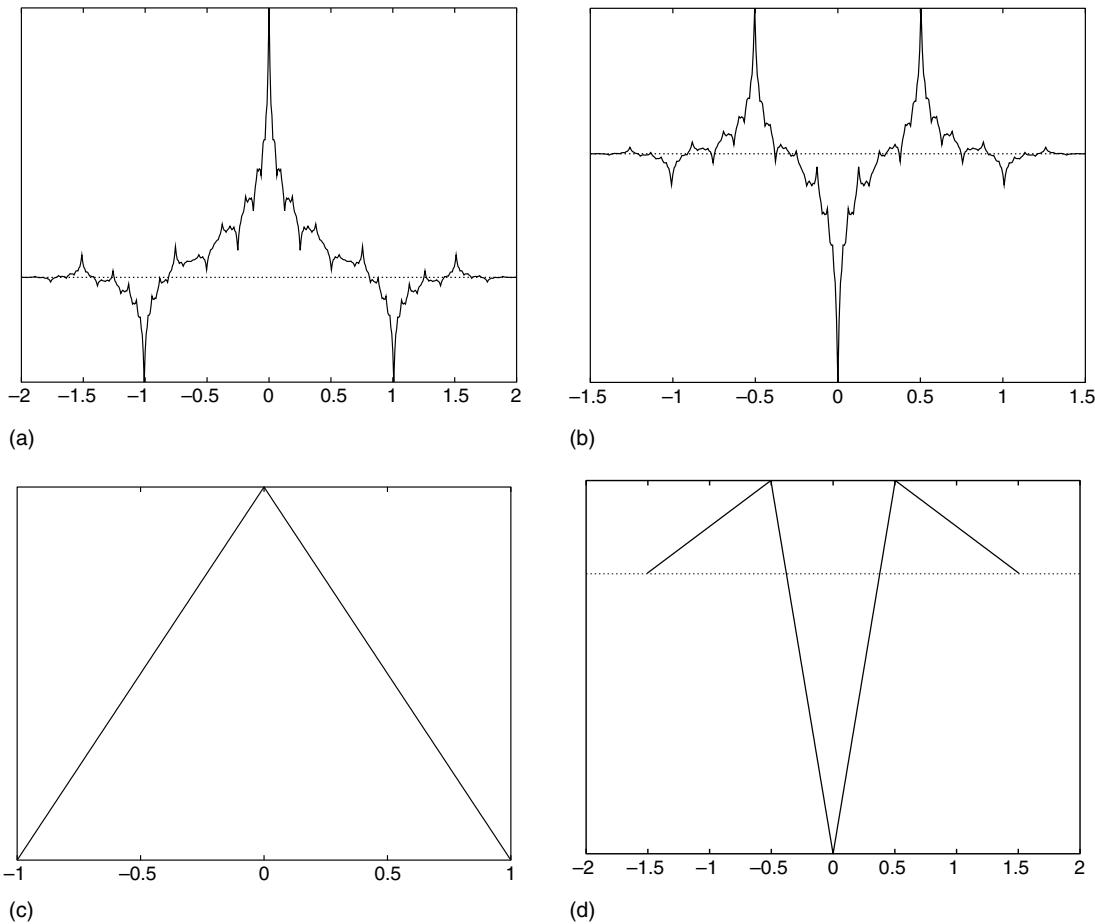


Fig. 10.18. The “symmetric short-kernel” wavelet transform (Equations (9.131)–(9.134)): (a) analysis scaling function; (b) analysis wavelet; (c) synthesis scaling function; (d) synthesis wavelet.

(Daubechies, 1988)

$$H_0(z) = +0.482\,9629 + 0.836\,5163z^{-1} + 0.224\,1439z^{-2} - 0.129\,4095z^{-3} \quad (10.204)$$

$$H_1(z) = -0.129\,4095 - 0.224\,1439z^{-1} + 0.836\,5163z^{-2} - 0.482\,9629z^{-3} \quad (10.205)$$

$$G_0(z) = -0.129\,4095 + 0.224\,1439z^{-1} + 0.836\,5163z^{-2} + 0.482\,9629z^{-3} \quad (10.206)$$

$$G_1(z) = -0.482\,9629 + 0.836\,5163z^{-1} - 0.224\,1439z^{-2} - 0.129\,4095z^{-3}. \quad (10.207)$$

Since the wavelet transform is orthogonal, the analysis and synthesis scaling functions and wavelets are the same. These are depicted in Figure 10.19. It is important to notice that, unlike the biorthogonal wavelets in Figure 10.18, these orthogonal wavelets are nonsymmetric, and, therefore, do not have linear phase.

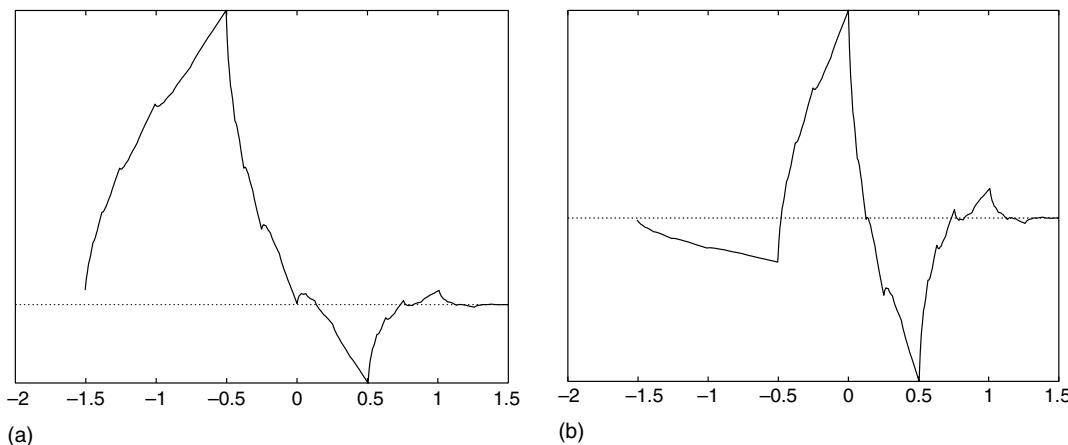


Fig. 10.19. Daubechies wavelet transform of length 4 (Equations (10.204)–(10.207)): (a) scaling function; (b) wavelet.

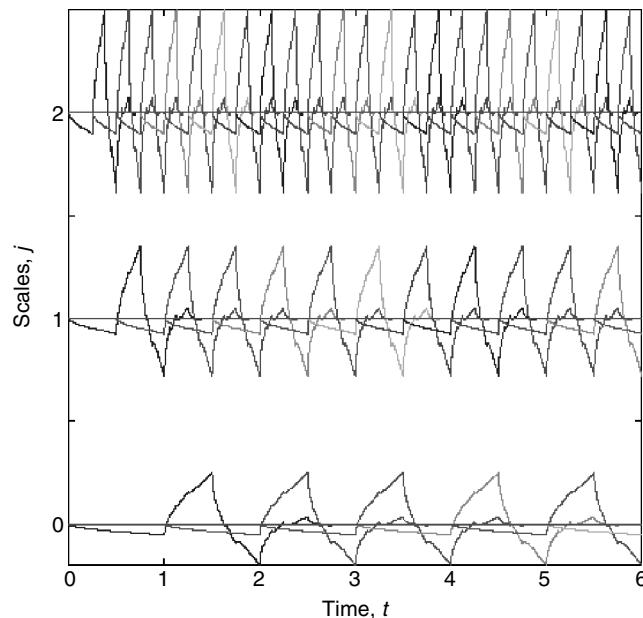


Fig. 10.20. Basis functions of a Daubechies wavelet transform of length 4, showing several scales and displacements.

Figure 10.20 shows the basis functions of a Daubechies wavelet of length 4, with several scales and displacements.

When implementing a wavelet transform using the scheme in Figure 10.2, it is essential that the delay introduced by each analysis/synthesis stage is compensated. Failure to do so may result in the loss of the perfect reconstruction property.

10.9 Wavelet transforms of images

One application where wavelet transforms are extremely useful is image processing. The varying degrees of time and frequency resolutions provided by their basis functions are well suited to images in general, since images tend to have features of varying sizes. For example, in a picture of a house with a person at the window, the basis function with a large scale will analyze conveniently the house as a whole. The person at the window will be best analyzed at a smaller scale, and the eyes of the person at an even smaller scale. This property of images is illustrated in Figure 10.21.

For applying wavelet transforms to images, a two-dimensional wavelet transform should be defined. This can be done in a variety of ways. The simplest form is the separable one (Mersereau & Dudgeon, 1984), where a two-dimensional wavelet transform is computed by applying one-dimensional wavelet transforms to every row in the image and then applying one-dimensional wavelet transforms to every column of the result. It can, therefore, be implemented by using the filter banks, as in Equations (10.125)–(10.128), in the horizontal and vertical directions of an image. More precisely, the two-dimensional z transforms of the analysis and synthesis filter banks, $H_{ij}(z_1, z_2)$ and $G_{ij}(z_1, z_2)$ respectively, are defined as (Vetterli & Kovačević, 1995; Mallat, 1999):

$$H_{00}(z_1, z_2) = H_0(z_1)H_0(z_2) \quad (10.208)$$

$$H_{01}(z_1, z_2) = H_0(z_1)H_1(z_2) \quad (10.209)$$



Fig. 10.21. Image showing features of different sizes.

$$H_{10}(z_1, z_2) = H_1(z_1)H_0(z_2) \quad (10.210)$$

$$H_{11}(z_1, z_2) = H_1(z_1)H_1(z_2) \quad (10.211)$$

$$G_{00}(z_1, z_2) = G_0(z_1)G_0(z_2) \quad (10.212)$$

$$G_{01}(z_1, z_2) = G_0(z_1)G_1(z_2) \quad (10.213)$$

$$G_{10}(z_1, z_2) = G_1(z_1)G_0(z_2) \quad (10.214)$$

$$G_{11}(z_1, z_2) = G_1(z_1)G_1(z_2). \quad (10.215)$$

In this framework, note that the variable z_1 corresponds to filtering the rows of the images and z_2 to filtering its columns.

If the filter banks above are applied recursively to the sub-band resulting from the low-pass filtering and subsampling in the horizontal and vertical directions, then we get an octave-band two-dimensional sub-band decomposition. Figure 10.22 depicts the process of generating a two-stage wavelet transform; that is, an octave-band sub-band decomposition.

Therefore, the resulting two-dimensional separable wavelet transform is defined by one scaling function and three wavelets, for the analysis and synthesis cases. The analysis

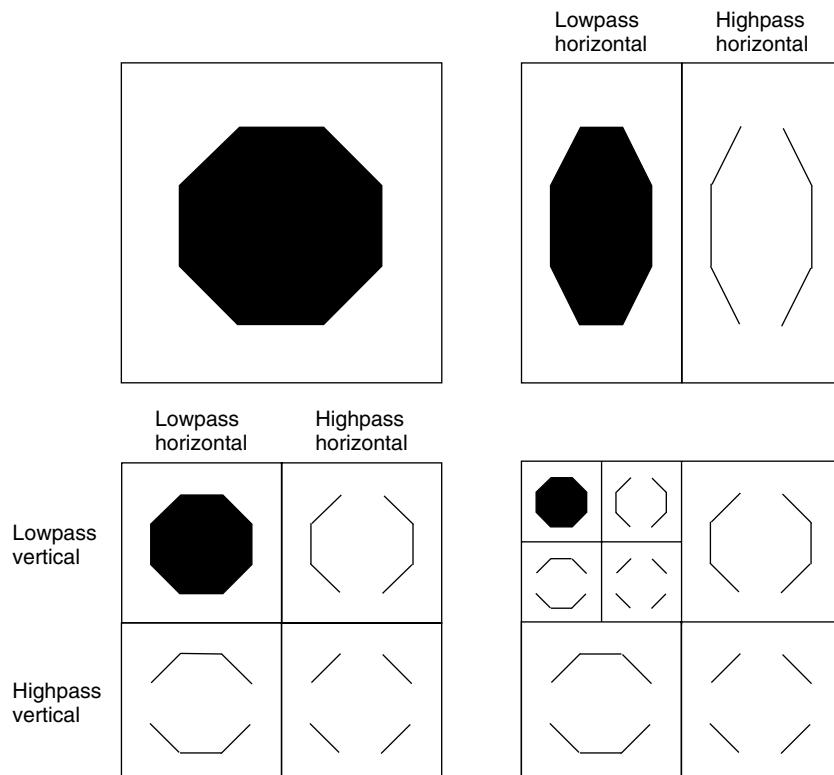


Fig. 10.22. Process of generating a two-stage wavelet transform of an image.

wavelets are then (Mallat, 1989b)

$$\phi_{00}(x_1, x_2) = \phi(x_1)\phi(x_2) \quad (10.216)$$

$$\psi_{01}(x_1, x_2) = \phi(x_1)\psi(x_2) \quad (10.217)$$

$$\psi_{10}(x_1, x_2) = \psi(x_1)\phi(x_2) \quad (10.218)$$

$$\psi_{11}(x_1, x_2) = \psi(x_1)\psi(x_2), \quad (10.219)$$

where x_1 corresponds to the horizontal direction and x_2 to vertical direction. Similarly, the synthesis wavelets are

$$\bar{\phi}_{00}(x_1, x_2) = \bar{\phi}(x_1)\bar{\phi}(x_2) \quad (10.220)$$

$$\bar{\psi}_{01}(x_1, x_2) = \bar{\phi}(x_1)\bar{\psi}(x_2) \quad (10.221)$$

$$\bar{\psi}_{10}(x_1, x_2) = \bar{\psi}(x_1)\bar{\phi}(x_2) \quad (10.222)$$

$$\bar{\psi}_{11}(x_1, x_2) = \bar{\psi}(x_1)\bar{\psi}(x_2). \quad (10.223)$$

The scaling functions $\phi_{00}(x_1, x_2)$ and $\bar{\phi}_{00}(x_1, x_2)$ are impulse responses of two-dimensional filters that are lowpass both in the vertical and horizontal directions. The wavelets $\psi_{01}(x_1, x_2)$ and $\bar{\psi}_{01}(x_1, x_2)$ are impulse responses of two-dimensional filters that are lowpass in the horizontal direction and highpass in the vertical direction. This leads to the corresponding wavelet coefficients being mainly related to image information in the horizontal direction. Similarly, the coefficients corresponding to the wavelets $\psi_{10}(x_1, x_2)$ and $\bar{\psi}_{10}(x_1, x_2)$ are related to image information in the vertical direction, and the coefficients corresponding to the wavelets $\psi_{11}(x_1, x_2)$ and $\bar{\psi}_{11}(x_1, x_2)$ are related to image information in the diagonal direction. The frequency decomposition obtained through such a wavelet transform is schematically represented in Figure 10.23, where H_i corresponds to coefficients in the horizontal direction and resolution i (wavelet $\psi_{01}(x_1/2^i, x_2/2^i)$). Analogously, V_i (wavelet $\psi_{10}(x_1/2^i, x_2/2^i)$) and D_i (wavelet $\psi_{11}(x_1/2^i, x_2/2^i)$) correspond to the vertical and diagonal directions respectively.

The same reasoning can be extended to multiple dimensions; that is, a one-dimensional wavelet transform can be applied in each dimension, generating multidimensional separable wavelet transforms.

The directionality of the sub-bands of a wavelet transform is represented schematically in Figure 10.22, where the horizontal, vertical, and diagonal bands can be clearly identified. The original octagon image and its wavelet transform are shown in Figure 10.24. Note that, besides the predominantly horizontal, vertical, and diagonal orientations, the sub-bands of similar orientations tend to be similar to each other.

Figure 10.25 shows the wavelet transform of the image shown in Figure 10.21. On the upper plot, each scale has been normalized such that it occupies the full dynamic range. On the lower plot, its absolute value is displayed in logarithmic scale. We can note that the small scales (high-frequency bands) tend to represent details more well localized in space, while the large scales (low-frequency bands) tend to represent only larger objects

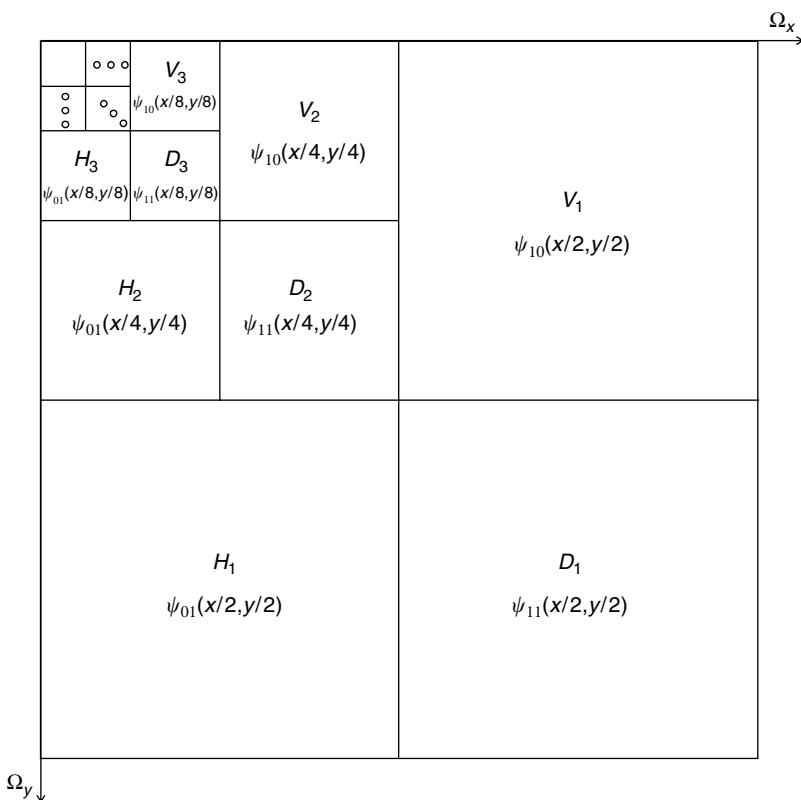
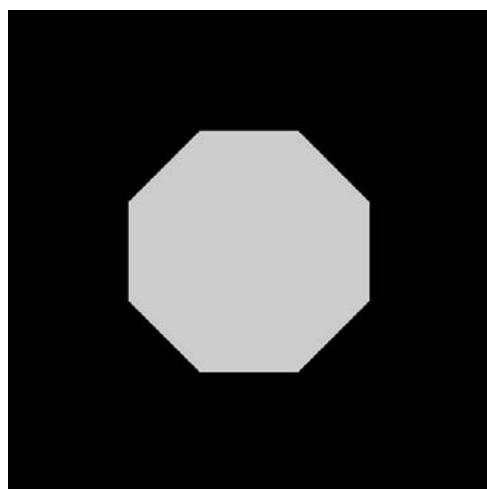
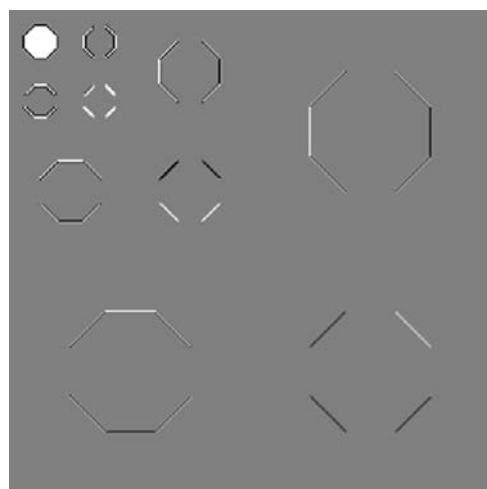


Fig. 10.23. Frequency decomposition obtained with a two-dimensional separable wavelet transform.



(a)



(b)

Fig. 10.24. (a) Original octagon image; (b) corresponding wavelet transform.



(a)



(b)

Fig. 10.25. Wavelet transform of the image in Figure 10.21: (a) each scale has been adjusted to fit the full dynamic range; (b) for the lowest frequency band, the logarithm of the coefficients' absolute value plus one is displayed, while for the other bands, twice this logarithm is displayed, with gray corresponding to zero. Note that many wavelet coefficients are very close to zero.

and, therefore, with worse localization in space. We can also note the directionality of the bands, as well as the similarity among the bands of similar orientations. In addition, by looking at the bands on a logarithmic scale in Figure 10.25b, we can see that the wavelet transform is quite effective in concentrating the energy of an image in a small number of coefficients. This is one of the main reasons why it has been successfully used in image compression schemes (Taubman & Marcellin, 2001; Sayood, 2005).

10.10 Wavelet transforms of finite-length signals

Often, the signals that one wants to filter have finite length. One example has been given in Section 10.9, where we have computed wavelet transforms of images that are, by nature, of finite length. One problem that arises when one considers finite-length signals is that a length- N signal, when filtered by a length- K impulse response FIR filter, yields an output signal of length $(N + K - 1)$. As seen above, in wavelet transforms, a two-band filter bank is recursively applied to the lowpass band of the previous stage. Therefore, the lengths of the signals increase for each decomposition stage. As a consequence, the number of samples of the wavelet transform tends to be larger than the number of samples of the signal. This is particularly inconvenient when one uses wavelet transforms in order to generate compact representations, as is the case, for example, of the JPEG2000 standard for image compression (Taubman & Marcellin, 2001). Therefore, a way to overcome this problem of increased number of samples in the wavelet transform is highly desirable. In this section we analyze signal extensions as a way to compute wavelet transforms that have as many coefficients as signal samples.

10.10.1 Periodic signal extension

The most straightforward way to avoid the increase in a signal length when it is filtered is to consider that it is periodic. This is so because, when one filters a periodic signal of period N , the filtered signal also has period N . Hence, for a periodic signal one needs to know the results for just one period, and, in this way, the effective signal length is not increased after filtering. The periodic extension of a length- N signal $x(n)$ is

$$x'(n) = x(n \bmod N). \quad (10.224)$$

Such a periodic extension is illustrated at the top of Figure 10.26.

When computing the wavelet transform one performs subsampling of each band by a factor of two. Therefore, for such a scheme to work, it is important that, besides the filtered signal being periodic with period N , its subsampled versions should also be periodic. In other words, its even and odd polyphase components

$$e'_0(l) = x'(2l) \quad (10.225)$$

and

$$e'_1(l) = x'(2l + 1) \quad (10.226)$$

respectively must also be periodic with period N . From Equation (10.224), if N is even, then we have that

$$e'_0\left(l + \frac{N}{2}\right) = x'\left(2l + N\right) = x'(2l) = e'_0(l) \quad (10.227)$$

$$e'_1\left(l + \frac{N}{2}\right) = x'\left(2l + N + 1\right) = x'(2l + 1) = e'_1(l). \quad (10.228)$$

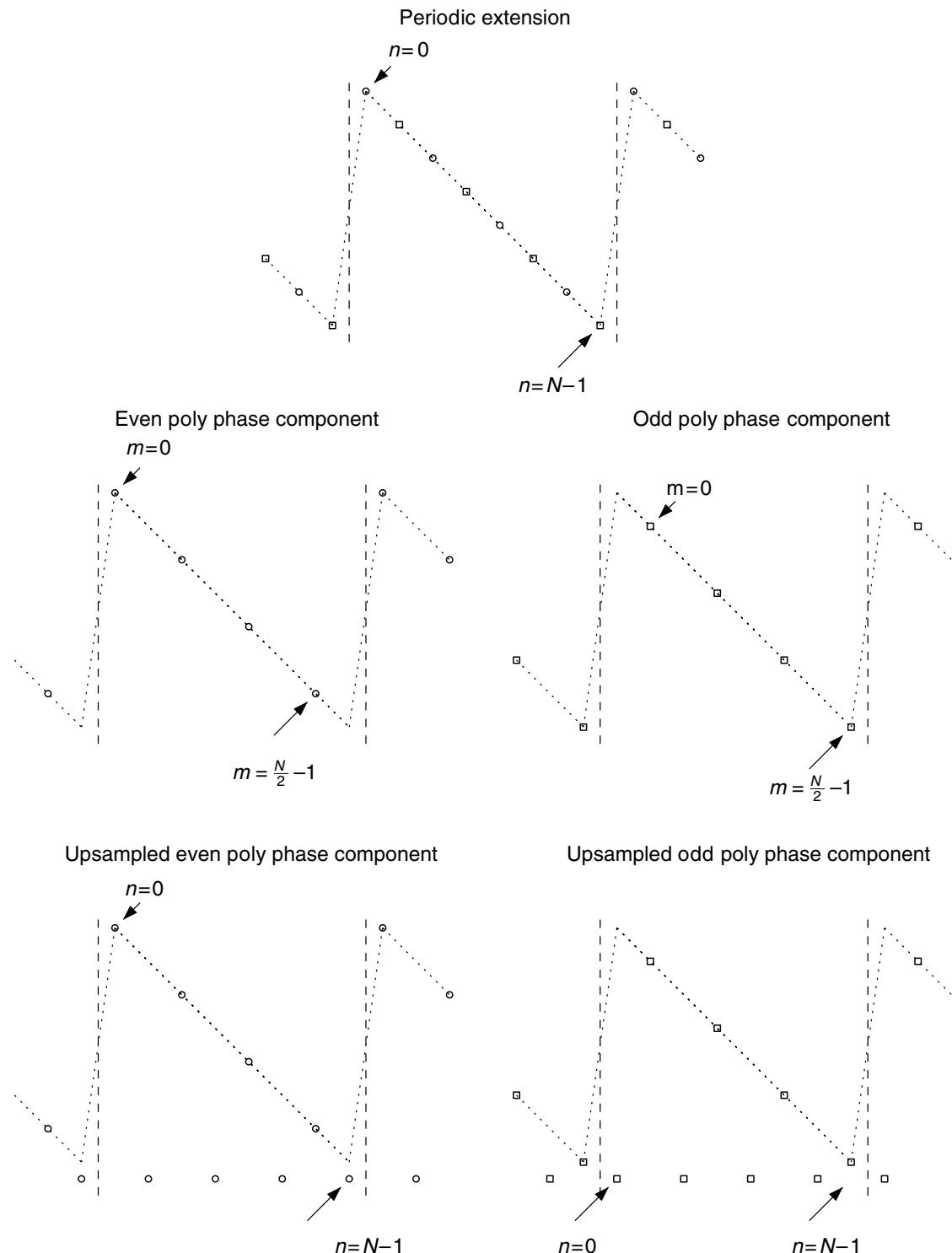


Fig. 10.26. From top to bottom: periodic extension of a signal, its even and odd polyphase components, and their upsampled versions.

therefore, we conclude that the polyphase components are periodic with period $N/2$. Since a sub-band is a filtered polyphase component, then the sub-bands are also periodic with period $N/2$, and thus the total number of samples in the two sub-bands is also equal to N . This is illustrated in the second row of Figure 10.26. Note that if the number of samples N is odd, then the polyphase components are only periodic with period N , and thus the total number of samples in the sub-bands is $2N$ (see Exercise 10.4). This is inefficient, and is one of the main reasons why periodic extensions of odd-length signals are seldom used. Because of this, in this section, we restrict ourselves to extensions of even-length signals.

In the synthesis part of the process, the sub-bands are upsampled and filtered. As can be seen at the bottom of Figure 10.26, the upsampled polyphase components are also periodic. Therefore, no matter which component is chosen during subsampling, the signal resulting from synthesis is also periodic with period N , having only N independent samples.

One drawback of the periodic extension can be understood by looking again at the top of Figure 10.26, which shows the original signal extended periodically. We see that the periodic extension will in general have discontinuities around $n = 0$ and $n = N - 1$ that are not part of the original signal. These discontinuities tend to appear with large energy in the detail bands of its wavelet transform (see Experiment 10.1), which is quite undesirable in many applications. For example, as can be seen in Figure 10.25, the wavelet transform has the energy concentrated in a relatively small number of coefficients, yielding compact representations. However, if one uses periodic extensions, then the discontinuities introduced will appear as high-energy coefficients in the details bands, thus diminishing the energy compaction properties of the wavelet transform. Therefore, whenever possible, it is preferable to use symmetric extensions. Such extensions will be dealt with in the next section.

10.10.2 Symmetric signal extensions

A commonly used form of signal extension is the symmetric extension. It avoids the discontinuities that arise when performing a periodic extension. Discrete-time signals have two types of symmetry: whole-sample symmetry and half-sample symmetry. In whole-sample symmetry, the axis of symmetry intersects a sample, while in half-sample symmetry it falls between samples. Mathematically, a signal $x(n)$ is whole-sample symmetric around $n = K$ if

$$x(K - n) = x(K + n), \quad \text{for all } n \in \mathbb{Z}. \quad (10.229)$$

On the other hand, a signal is half-sample symmetric around “sample” $K - \frac{1}{2}$, with $K \in \mathbb{Z}$, if

$$x(K - 1 - n) = x(K + n), \quad \text{for all } n \in \mathbb{Z}. \quad (10.230)$$

Examples of the whole- and half-sample symmetries are depicted in Figures 10.27a and 10.27b respectively.

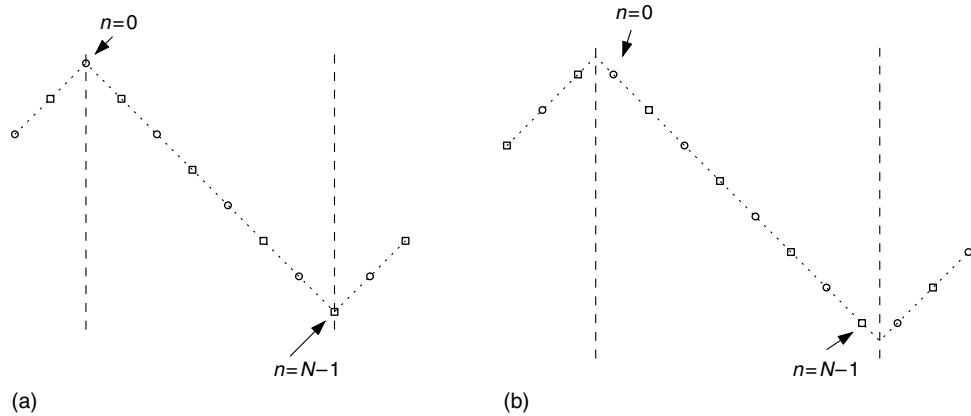


Fig. 10.27. (a) Whole-sample symmetry; (b) half-sample symmetry.

From Equation (10.229), if a signal $x(n)$ of length N is extended symmetrically with whole-sample symmetry around both $n = 0$ and $n = N - 1$, the resulting signal $x'(n)$ is a periodic signal with period $2N - 2$ that is given by

$$x'(n) = \begin{cases} x(n), & 0 \leq n \leq N - 1 \\ x(-n), & -N + 1 \leq n \leq 0 \\ x(2N - 2 - n), & N - 1 \leq n \leq 2N - 2 \end{cases}. \quad (10.231)$$

Likewise, from Equation (10.230), if a signal $x(n)$ of length N is extended symmetrically with half-sample symmetry around both $n = 0$ and $n = N - 1$, the resulting signal $x'(n)$ is a periodic signal with period $2N$ such that

$$x'(n) = \begin{cases} x(n), & 0 \leq n \leq N - 1 \\ x(-n - 1), & -N \leq n \leq -1 \\ x(2N - 1 - n), & N \leq n \leq 2N - 1 \end{cases}. \quad (10.232)$$

If the j th polyphase components of $x(n)$ and $x'(n)$ are $e_j(l)$ and $e'_j(l)$ respectively, then we have, from Equation (10.231), that for whole-sample symmetry (as in the periodic case, we restrict ourselves to the case that N is even)

$$e'_0(l) = \begin{cases} x(2l) = e_0(l), & 0 \leq l \leq (N/2) - 1 \\ x(-2l) = e_0(-l), & -(N/2) + 1 \leq l \leq 0 \\ x(2N - 2 - n) = e_0(N - 1 - l), & (N/2) - 1 \leq l \leq N - 1 \end{cases} \quad (10.233)$$

$$e'_1(l) = \begin{cases} x(2l + 1) = e_1(l), & 0 \leq l \leq (N/2) - 1 \\ x(-2l - 1) = e_1(-l), & -N/2 \leq l \leq -1 \\ x(2N - 2l - 3) = e_1(N - 2 - l), & (N/2) - 1 \leq l \leq N - 2 \end{cases}. \quad (10.234)$$

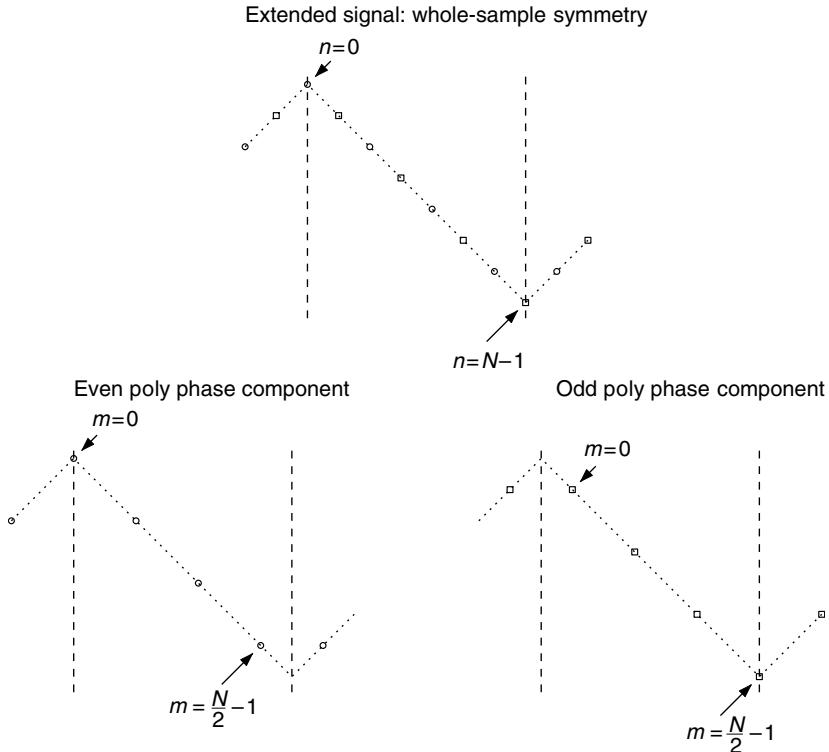


Fig. 10.28. Top: length- N signal extended with whole-sample symmetry around zero and around $N - 1$; bottom: corresponding polyphase components.

Therefore, the above equations, with the help of Equations (10.229)–(10.232), imply that, for signals which are extended with whole-sample symmetry, the even polyphase component $e'_0(l)$ is whole-sample symmetric around zero and half-sample symmetric around $(N/2) - 1$. Likewise, the odd polyphase component $e'_1(l)$ is half-sample symmetric around zero and whole-sample symmetric around $(N/2) - 1$. This situation is illustrated in Figure 10.28.

On the other hand, we have, from Equation (10.232), that for half-sample symmetry (again, N is restricted to be even)

$$e'_0(l) = \begin{cases} x(2l) = e_0(l), & 0 \leq l \leq (N/2) - 1 \\ x(-2l - 1) = e_1(-l - 1), & -N/2 \leq l \leq -1 \\ x(2N - 1 - n) = e_1(N - 1 - l), & N/2 \leq l \leq N - 1 \end{cases} \quad (10.235)$$

$$e'_1(l) = \begin{cases} x(2l + 1) = e_1(l), & 0 \leq l \leq (N/2) - 1 \\ x(-2l - 2) = e_0(-l - 1), & -N/2 \leq l \leq -1 \\ x(2N - 2l - 2) = e_0(N - 1 - l), & N/2 \leq l \leq N - 1 \end{cases} . \quad (10.236)$$

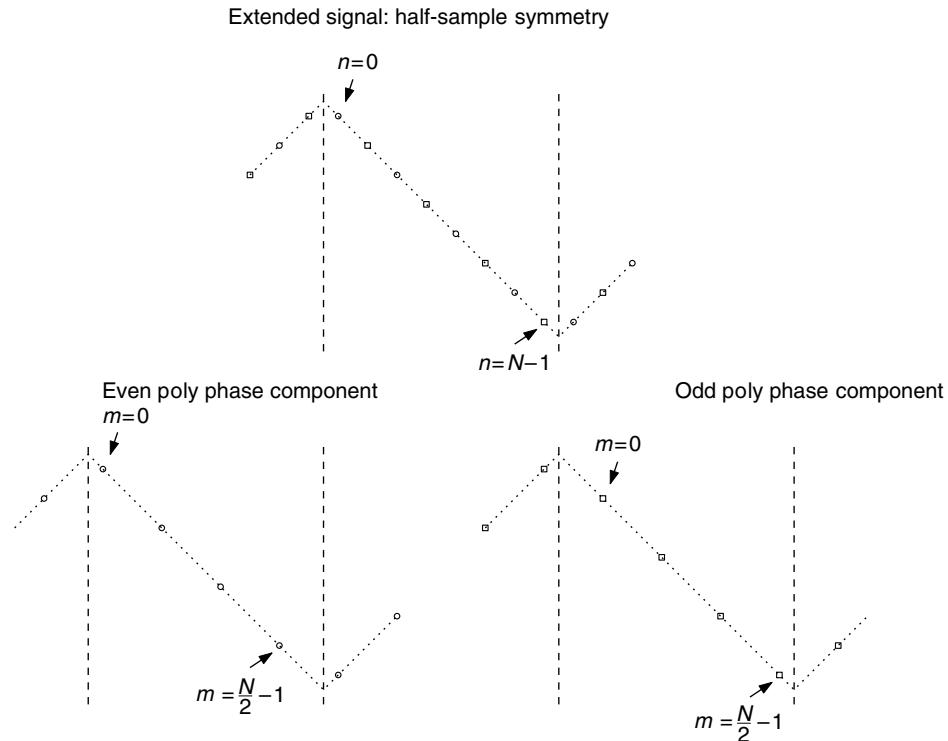


Fig. 10.29. Top: length- N signal extended with half-sample symmetry around zero and around $N - 1$; bottom: corresponding polyphase components.

Therefore, the above equations imply that, for signals which are extended with half-sample symmetry, neither of their polyphase components are symmetric, as illustrated in Figure 10.29.

In order for a symmetric extension to be usable for the computation of a wavelet transform, three conditions must be satisfied:

- (a) The signal must remain symmetric after the application of the analysis filter.
- (b) The signal filtered by the analysis filters must remain symmetric after subsampling by a factor of 2.
- (c) The upsampled signal must be symmetric before application of the synthesis filters.

Condition (a) above demands that the analysis filters have linear phase, since they are the only ones that do not destroy the symmetry of signals input to it. As seen in Section 4.2.3, linear-phase filters can have either integer delays (even order) or an integer plus $\frac{1}{2}$ delay (odd order). It is important to notice that a signal with whole-sample symmetry, when delayed by an integer number of samples, remains whole-sample symmetric. On the other hand, a signal with whole-sample symmetry becomes half-sample symmetric when delayed by half

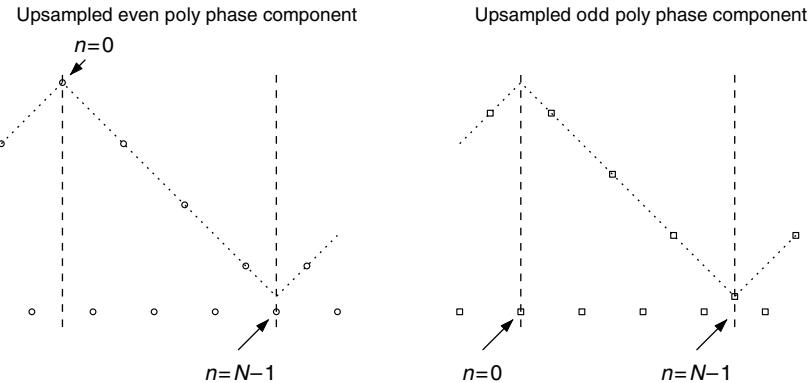


Fig. 10.30. The even and odd polyphase components of a whole-sample symmetric length- N signal at both $n = 0$ and $n = N - 1$ after upsampling. It can be observed that both are whole-sample symmetric.

sample. Likewise, a signal with half-sample symmetry becomes whole-sample symmetric when delayed by half sample.

Condition (b) demands that the output signals for the analysis filters have to be whole-sample symmetric around both zero and $N - 1$. This is so because, as seen above and illustrated in Figures 10.28 and 10.29, the polyphase components of half-sample symmetric signals are not symmetric. Therefore, since the output of an analysis filter must be whole-sample symmetric, we have two cases, depending on the order of the analysis filters: if its delay is integer (even order), then the signal must be extended with whole-sample symmetry; if its delay is an integer plus $\frac{1}{2}$ (odd order), then the signal must be extended with half-sample symmetry.

Condition (c) is automatically satisfied provided that the subsampled signals input to the interpolators are symmetric. This is illustrated in Figure 10.30. Note that for both polyphase components the symmetry of their interpolated versions is whole-sample.

From the design restrictions for the two-band linear-phase filter banks, as presented in Section 9.5 (after Equation (9.126)), we have that useful two-band linear-phase filter banks should have either all filters with even orders or all filters with odd orders. Table 10.1 summarizes how the symmetric extensions in each stage of the two-band filter bank process should be in the two cases. Note that, once again we restrict ourselves to the case when the signal length N is even (see Exercise 10.8).

It is important to note that in wavelet transforms we have to apply a two-band filter bank recursively to the lowpass bands. In the even-order case, for instance, if we take the even polyphase component as the signal after subsampling, then it is whole-sample symmetric around zero and half-sample symmetric around $(N/2) - 1$. Although this is the signal that we must upsample to perform the synthesis stage, in order for us to further decompose it we have to generate a slightly different signal. For example, if we are going to use an even-order filter bank for the next stage, we must first generate from it a signal that is whole-sample symmetric at both ends. We do so by first taking its samples from $m = 0$ to $m = (N/2) - 1$ and extending them using whole-sample symmetry at both ends.

Table 10.1.

Types of symmetric extension in the two-band analysis and synthesis process for both even order and odd order filter banks.

Stage of the filtering process	Symmetry	
	Even order	Odd order
Before analysis filter	whole (0) / whole ($N - 1$)	half (0) / half ($N - 1$)
After analysis filter	whole (0) / whole ($N - 1$)	whole (0) / whole ($N - 1$)
Even polyphase component	whole (0) / half ($(N/2) - 1$)	whole (0) / half ($(N/2) - 1$)
Odd polyphase component	half (0) / whole ($(N/2) - 1$)	half (0) / whole ($(N/2) - 1$)
After upsampling	whole (0) / whole ($N - 1$)	whole (0) / whole ($N - 1$)
After synthesis	whole (0) / whole ($N - 1$)	half (0) / half ($N - 1$)

10.11 Do-it-yourself: wavelet transforms

Experiment 10.1

Here, we see how wavelets can be used to analyze nonstationary signals. We start by generating a signal composed of a sequence of five sinusoids of different frequencies, corrupted by spikes. The beginning of each sinusoid is specified in the `pos_sin` variable and the corresponding period is defined in `T_sin`. For the spikes, their amplitude and time positions are as given by the `amp_imp` and `pos_imp` variables respectively. The MATLAB code to generate it is as follows:

```
N = 2000; t = [0:N];
x = zeros(size(t));
pos_sin = [0 600 1080 1380 1680 2000];
T_sin = [100 40 20 10 5];
for i = 1:5,
    m = 1 + pos_sin(i); n = pos_sin(i+1);
    x(m:n) = sin(2*pi*t(1:n-m+1)/T_sin(i));
end;
amp_imp = [3 -2 2 2.5 -2.5];
pos_imp = [200 372 1324 1343 1802];
T_imp = [5 25 5 5 5 ];
for i = 1:5,
    m = 1 + pos_imp(i); n = 1 + pos_imp(i)+fix(T_imp(i)/2);
    x(m:n) = x(m:n) + amp_imp(i)*sin(2*pi*t(1:n-m+1)/T_imp(i)).^2;
end;
```

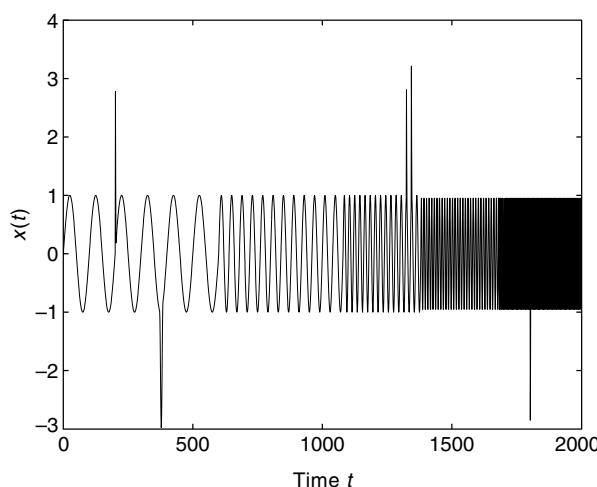
The resulting signal `x` has $(N+1) = 2001$ samples, and the sinusoids have, in sequence, periods of 100, 40, 20, 10 and 5 samples. The spikes consist of one period of a sine-squared waveform. Four of them have duration of three samples and another one (the second from left to right) has a period of 23 samples. The third and fourth spikes are very close, only 19 samples apart. The signal is depicted in Figure 10.31.

In this experiment, we decompose the signals with the wavelet `bior4.4`, which is the biorthogonal linear-phase wavelet used in the JPEG2000 standard for image compression

Table 10.2.

Coefficients of the analysis and synthesis 9–7 (*bior4.4*) wavelet.

$h_0(0) = 0.0378$	$h_1(0) = -0.0645$	$g_0(0) = -0.0645$	$g_1(0) = -0.0378$
$h_0(1) = -0.0238$	$h_1(1) = 0.0407$	$g_0(1) = -0.0407$	$g_1(1) = -0.0238$
$h_0(2) = -0.1106$	$h_1(2) = 0.4181$	$g_0(2) = 0.4181$	$g_1(2) = 0.1106$
$h_0(3) = 0.3774$	$h_1(3) = -0.7885$	$g_0(3) = 0.7885$	$g_1(3) = 0.3774$
$h_0(4) = 0.8527$	$h_1(4) = 0.4181$	$g_0(4) = 0.4181$	$g_1(4) = -0.8527$
$h_0(5) = 0.3774$	$h_1(5) = 0.0407$	$g_0(5) = -0.0407$	$g_1(5) = 0.3774$
$h_0(6) = -0.1106$	$h_1(6) = -0.0645$	$g_0(6) = -0.0645$	$g_1(6) = 0.1106$
$h_0(7) = -0.0238$			$g_1(7) = -0.0238$
$h_0(8) = 0.0378$			$g_1(8) = -0.0378$

**Fig. 10.31.** Signal for Experiment 10.1.

(Taubman & Marcellin, 2001), also referred to as the 9–7 wavelet. The coefficients of the analysis and synthesis filters are shown in Table 10.2.

In order to load the analysis and synthesis filters we use the MATLAB command

```
[Lo_D, Hi_D, Lo_R, Hi_R] = wfilters('bior4.4');
```

and compute the five-stage wavelet transform using

```
[C, S] = wavedec(x, 5, Lo_D, Hi_D);
```

Following this approach, vector C stores the wavelet coefficients and vector S stores the lengths of the sub-bands. We compute the detail sub-bands using the command `detcoef`, and the approximation coefficients (lowpass bands for each scale) using the command `appcoef`, as follows:

```
D1 = detcoef(C, S, 1);
D2 = detcoef(C, S, 2);
D3 = detcoef(C, S, 3);
```

```

D4 = detcoef(C,S,4);
D5 = detcoef(C,S,5);
A1 = appcoef(C,S,Lo_R,Hi_R,1);
A2 = appcoef(C,S,Lo_R,Hi_R,2);
A3 = appcoef(C,S,Lo_R,Hi_R,3);
A4 = appcoef(C,S,Lo_R,Hi_R,4);
A5 = appcoef(C,S,Lo_R,Hi_R,5);

```

The plots of the detail bands are shown in Figure 10.32, and the plots of the approximation bands are shown in Figure 10.33. Note that the sample scale has been normalized in order to make the comparison easier along the time axis.

By observing Figure 10.32, we can see that the highest frequency detail band (second plot from top) contains essentially the spikes of three samples duration. The wider spike does not appear in this band, since its resolution is not high enough for that. Note that, although some of the five-sample period sinusoid is still present, by performing a simple thresholding in this band, one can easily have a signal composed only of these short-duration spikes, and their locations can be easily determined. In addition, although there are traces of these spikes up to the fourth detail band, the two spikes that are closer together can only be distinguished up to the second band. Also, the wider spike can only be detected from the third band onwards. From these observations, we can see that the wavelet transform is

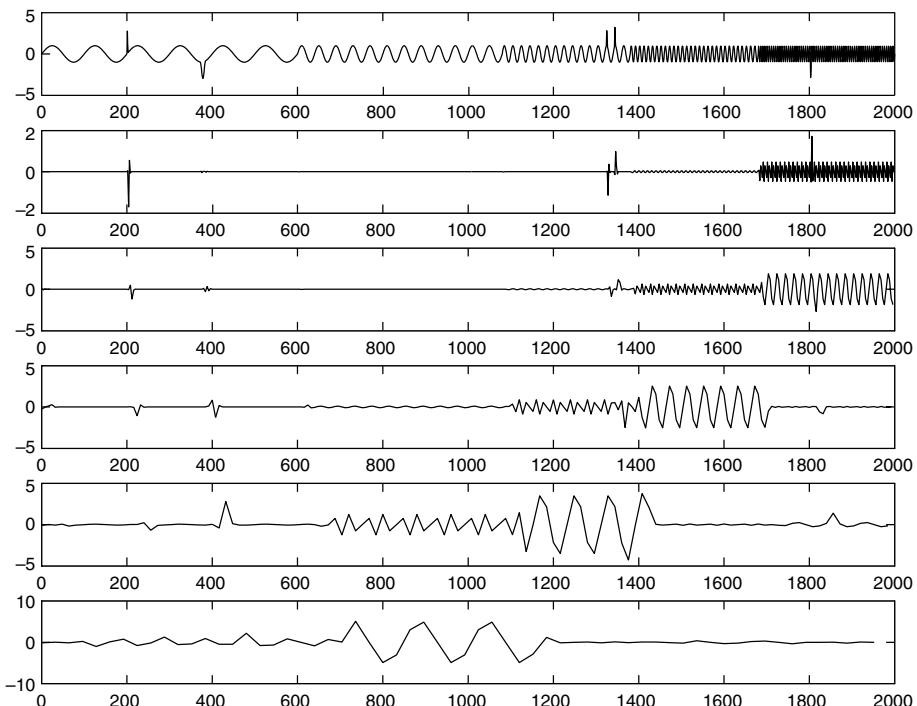


Fig. 10.32.

Detail bands for the signal from Experiment 10.1. The top plot corresponds to the original signal, and the details bands are shown, from top to bottom, in increasing scale (decreasing frequency) order.

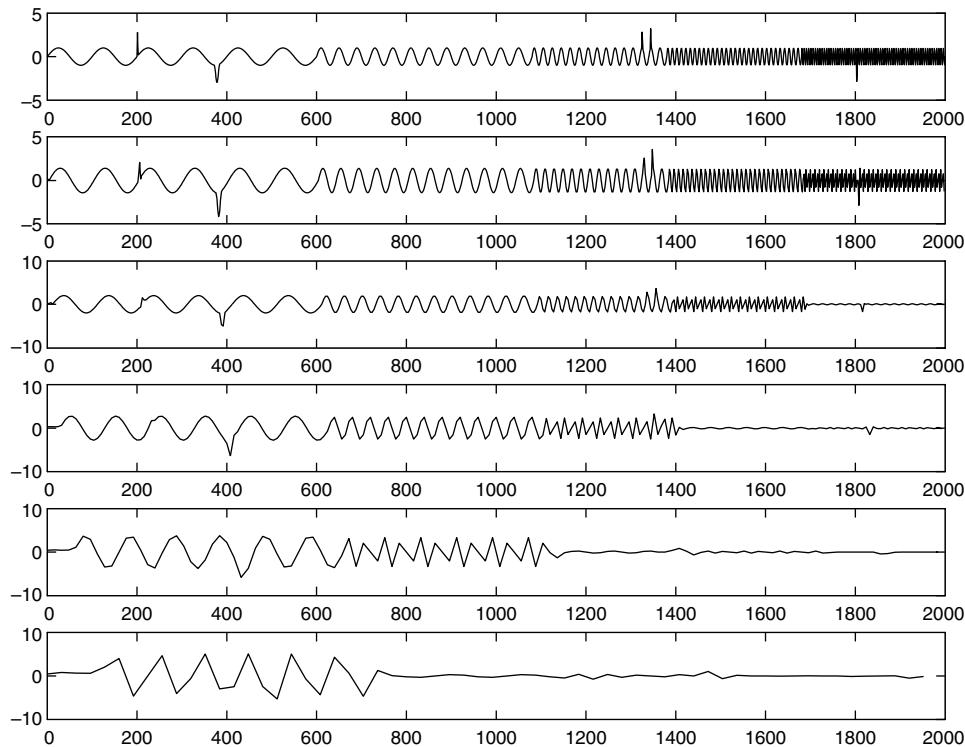


Fig. 10.33. Approximation bands for the signal from Experiment 10.1. The top plot corresponds to the original signal, and the approximation bands are shown, from top to bottom, in increasing scale (decreasing frequency) order.

good at detecting transient phenomena. A good way to perform this is to look for correlation across bands. All the spikes in the signal tend to appear at least in three consecutive detail bands. Note also that each band shows preferentially one sinusoid, highlighting the bandpass nature of the detail bands.

Figure 10.33 shows the approximation bands. There, we can see the decreasing levels of details present in these bands as the scale increases (frequency decreases). This highlights the lowpass nature of these bands.

The reader is encouraged to try different wavelets with this signal, and also a different number of decomposition stages. The MATLAB Wavelet Toolbox has several other signals that can be used for processing. They are usually under the directory `wavedemo` in the Wavelet Toolbox. Using these signals, the reader can play around with the wavelet transforms. This will help develop a good feeling of this important signal processing tool.

Experiment 10.2

In this experiment, we investigate the use of wavelet analysis to perform denoising of a given signal. We use as an example the signal `1eleccum` from the MATLAB Wavelet Toolbox.

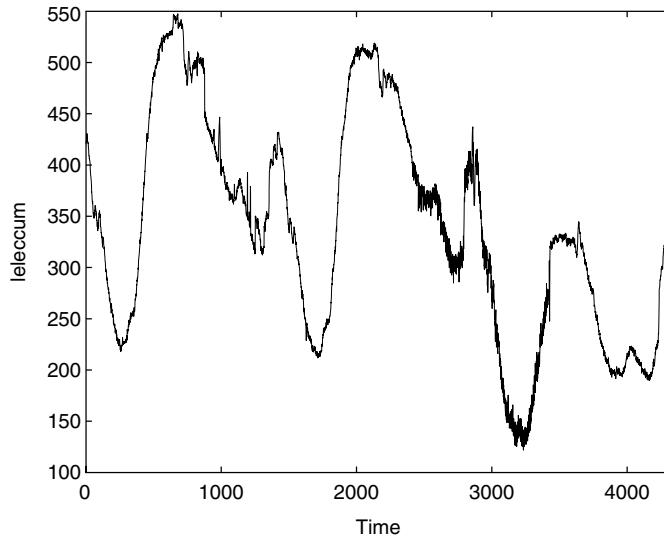


Fig. 10.34. Signal for Experiment 10.2.

It can be loaded with the command

```
load leleccum;
```

which creates a variable `leleccum` of length 4320, as depicted in Figure 10.34.

This signal is corrupted by noise. Wavelet transforms can be successfully used to perform signal denoising. Since noise is usually wideband, the mere lowpass filtering of the corrupted signal is not the most effective way of reducing the noise. Let us start by computing the five-stage wavelet transform of the signal, using the Daubechies-4 orthogonal filter bank. Its analysis and synthesis filters are shown in Table 10.3.

In order to load the analysis and synthesis filters, we use the MATLAB command

```
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters('db4');
```

and compute the five-stage wavelet transform using

```
[C,S] = wavedec(leleccum,5,Lo_D,Hi_D);
```

The detail bands and the fifth-level approximation band can be computed using the following commands:

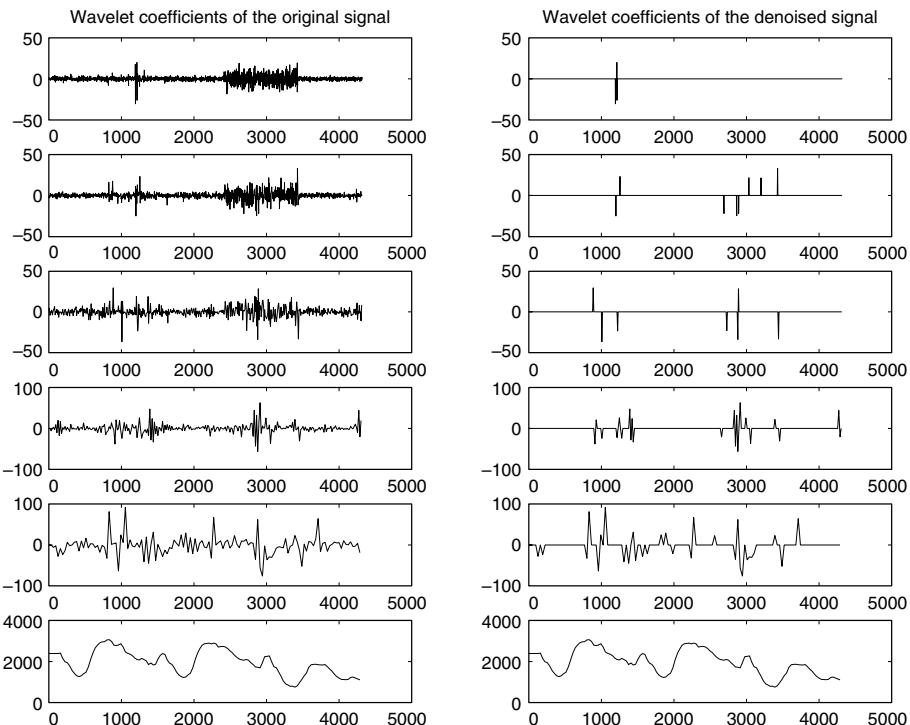
```
A5 = appcoef(C,S,Lo_R,Hi_R,5);
D1 = detcoef(C,S,1);
D2 = detcoef(C,S,2);
D3 = detcoef(C,S,3);
D4 = detcoef(C,S,4);
D5 = detcoef(C,S,5);
```

The results from these commands are plotted on the left-hand side of Figure 10.35, with the scale increasing from top to bottom. The bottom-most plot corresponds to the approximation

Table 10.3.

Coefficients of the analysis and synthesis Daubechies 4 wavelet (db4).

$h_0(0) = -0.0106$	$h_1(0) = -0.2304$	$g_0(0) = 0.2304$	$g_1(0) = -0.0106$
$h_0(1) = 0.0329$	$h_1(1) = 0.7148$	$g_0(1) = 0.7148$	$g_1(1) = -0.0329$
$h_0(2) = 0.0308$	$h_1(2) = -0.6309$	$g_0(2) = 0.6309$	$g_1(2) = 0.0308$
$h_0(3) = -0.1870$	$h_1(3) = -0.0280$	$g_0(3) = -0.0280$	$g_1(3) = 0.1870$
$h_0(4) = -0.0280$	$h_1(4) = 0.1870$	$g_0(4) = -0.1870$	$g_1(4) = -0.0280$
$h_0(5) = 0.6309$	$h_1(5) = 0.0308$	$g_0(5) = 0.0308$	$g_1(5) = -0.6309$
$h_0(6) = 0.7148$	$h_1(6) = -0.0329$	$g_0(6) = 0.0329$	$g_1(6) = 0.7148$
$h_0(7) = 0.2304$	$h_1(7) = -0.0106$	$g_0(7) = -0.0106$	$g_1(7) = -0.2304$

**Fig. 10.35.**

Fifth-stage approximation band and detail bands for the signals from Experiment 10.2. Details bands are shown, from top to bottom, in increasing scale order. The bottom-most plot corresponds to the approximation band. Left: original signal; right: denoised signal.

band of the fifth stage. By looking at these plots, one can see that, if we apply a magnitude threshold of around 20, then most of the noisy coefficients can be made null. We can perform this using the commands

```
Ct = zeros(size(C));
Ct(find(abs(C)>20)) = C(find(abs(C)>20));
xt = waverec(Ct,S,Lo_R,Hi_R);
```

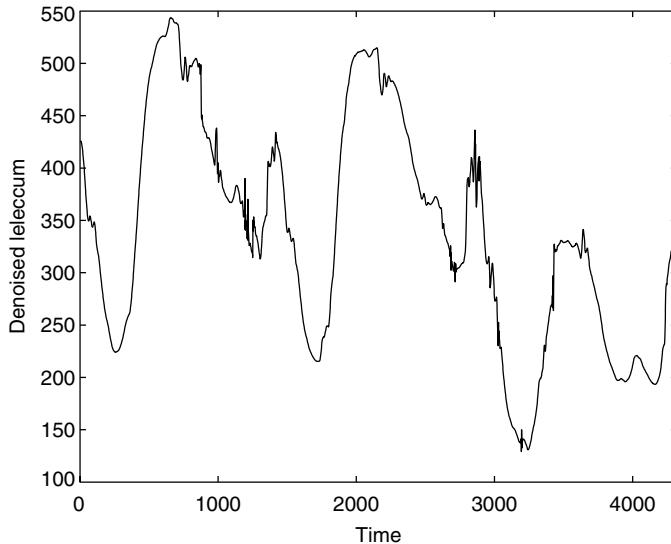


Fig. 10.36. Denoised signal from Experiment 10.2.

The reconstructed bands can be computed by

```
At5 = appcoef(Ct,S,Lo_R,Hi_R,5);
Dt1 = detcoef(Ct,S,1);
Dt2 = detcoef(Ct,S,2);
Dt3 = detcoef(Ct,S,3);
Dt4 = detcoef(Ct,S,4);
Dt5 = detcoef(Ct,S,5);
```

The right-hand side of Figure 10.35 shows the sub-bands after applying the threshold. Figure 10.36 shows the denoised signal. One can see that the wavelet is able to perform effective denoising. Note that while the thresholding process sets to zero most noisy coefficients, it can also set to zero some signal coefficients, which may lead to a quality loss in the reconstructed signal. Therefore, in wavelet denoising it is an important matter to find a threshold that gives a good trade-off between noise elimination and quality of the reconstructed signal. The reader is encouraged to explore this experiment further by trying out different threshold values, as well as different wavelets. In the MATLAB Wavelet Toolbox there are several signals corrupted with noise. Examples are the signals cnoislop, ex1nfix, ex2nfix, ex3nfix, heavysin, mishmash, nbump1, nelec, ndoppr1, noischir, wnoislop, and wntrs1, among others. The reader is also encouraged to experiment with them.

10.12 Wavelets with MATLAB

The functions described below are from the MATLAB Wavelet Toolbox. This toolbox includes many predefined wavelets, divided into families. For example, among others we

have the Daubechies, the biorthogonal, the Coiflet, and the Symmlet families (Mallat, 1999). Most functions require the desired wavelet in a family to be specified. The functions which involve wavelet transform computation also permit the specification of the filter bank coefficients.

- **waveinfo:** Gives information on wavelet families.

Input parameter: the name `wfname` of the wavelet family. Use the command `waveinfo` with no parameters to see a list of names and a brief description of the wavelet families.
Example:

```
wfname='coif'; waveinfo('wfname');
```

- **wfilters:** Computes the coefficients of the analysis and synthesis filters given a wavelet transform.

Input parameter: the name `wname` of the wavelet transform. Available wavelet names are:

- Daubechies: 'db1' or 'haar', 'db2', ..., 'db50'
- Coiflet: 'coif1', ..., 'coif5'
- Symmlet: 'sym2', ..., 'sym8'
- Biorthogonal: 'bior1.1', 'bior1.3', 'bior1.5',
 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8',
 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7',
 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'.

Output parameters:

- a vector `Lo_D` containing the decomposition lowpass filter coefficients;
- a vector `Hi_D` containing the decomposition highpass filter coefficients;
- a vector `Lo_R` containing the reconstruction lowpass filter coefficients;
- a vector `Hi_R` containing the reconstruction highpass filter coefficients.

Example:

```
wname='db5';
[Lo_D,Hi_D,Lo_R,Hi_R]=wfilters(wname);
```

- **dbwavf:** Compute the coefficients of the two-scale difference equation (see Equation (10.56)) given a wavelet transform from the Daubechies family.

Input parameter: the name `wname` of the wavelet transform from the Daubechies family.
See description of the function `wfilters` for a list of available names.

Output parameter: a vector `F` containing the coefficients of the two-scale difference equation.

Example:

```
wname='db5';
F=dbwavf(wname);
```

- The functions `coifwavf` and `symwavf` are the equivalent to the function `dbwavf` for the Coiflet and Symmlet families. Please refer to the MATLAB Wavelet Toolbox documentation for details.
- **orthfilt:** Computes the coefficients of the analysis and synthesis filters given the coefficients of the two-scale difference equation of an orthogonal wavelet (see Equation (10.56)).

Input parameter: a vector W containing the coefficients of the two-scale difference equation.

Output parameters: see command wfilters.

Example:

```
wname='coif4'; W=coifwavf(wname);
[Lo_D,Hi_D,Lo_R,Hi_R]=orthfilt(W);
```

- **biorwavf:** Computes the coefficients of the analysis and synthesis two-scale difference equations (see Equation (10.56)) given a wavelet transform from the biorthogonal family. Input parameter: the name wname of the wavelet transform from the biorthogonal family. See description of the function wfilters for a list of available names.

Output parameters:

- a vector RF containing the coefficients of the two-scale synthesis difference equation;
- a vector DF containing the coefficients of the two-scale analysis difference equation.

Example:

```
wname='bior2.2';
[RF,DF]=biorwavf(wname);
```

- **biorfilt:** Computes the coefficients of the analysis and synthesis filters given the coefficients of the analysis and synthesis two-scale difference equations of a biorthogonal wavelet (see Equation (10.56)).

Input parameters:

- a vector DF containing the coefficients of the two-scale analysis difference equation.
- a vector RF containing the coefficients of the two-scale synthesis difference equation.

Output parameters: see command wfilters.

Example:

```
wname='bior3.5'; [RF,DF]=biorwavf(wname);
[Lo_D,Hi_D,Lo_R,Hi_R]=biorfilt(DF,RF);
```

- **dwt:** Single-stage discrete one-dimensional wavelet decomposition.

Input parameters:

- A vector x containing the input signal.
- A vector Lo_D containing the analysis lowpass filter.
- A vector Hi_D containing the analysis highpass filter.
- There is the possibility of providing, instead of Lo_D and Hi_D, wname, the name of the wavelet transform. See description of the function wfilters for a list of available names.

Output parameters:

- a vector cA, containing the approximation coefficients (lowpass band);
- a vector cD, containing the detail coefficients (highpass band).

Example:

```
Lo_D=[-0.0625 0.0625 0.5 0.5 0.0625 -0.0625];
Hi_D=[-0.5 0.5];
load leleccum; x=leleccum;
[cA,cD]=dwt(x,Lo_D,Hi_D);
```

- **idwt**: Single-stage discrete one-dimensional wavelet reconstruction.

Input parameters:

- A vector **cA** containing the approximation coefficients (lowpass band).
- A vector **cD** containing the detail coefficients (highpass band).
- A vector **Lo_D** containing the analysis lowpass filter.
- A vector **Hi_D** containing the analysis highpass filter.
- There is the possibility of providing, instead of **Lo_D** and **Hi_D**, **wname**, the name of the wavelet transform. See description of the function **wfilters** for a list of available names.

Output parameter: a vector **x** containing the output signal.

Example:

```
x=idwt(cA,cD,'bior2.2');
```

- **dwtmode**: Sets the type of signal extension at the signal boundaries for wavelet transform calculations.

Input parameters:

- '**zpd**' sets the extension mode to zero-padding (default mode);
- '**sym**' sets the extension mode to symmetric (boundary-value replication – half-sample symmetry);
- '**symw**' sets the extension mode to symmetric (whole-sample symmetry);
- '**asym**' sets the extension mode to antisymmetric (boundary-value replication – half-sample symmetry);
- '**asymw**' sets the extension mode to antisymmetric (whole-sample symmetry);
- '**spd**' sets the extension mode to smooth padding (first derivative interpolation at the edges).
- '**spd0**' sets the extension mode to smooth padding of order zero (constant extension at the edges).
- '**ppd**' sets the extension mode to periodic padding (periodic extension at the edges).
- '**per**' is similar to '**ppd**' and produces the smallest length decomposition.

Example:

```
dwtmode('sym'); load leleccum; x=leleccum;
[cA,cD]=dwt(x,'bior1.3');
```

- **wavedec**: Performs multiple-stage one-dimensional wavelet decomposition.

Input parameters:

- A vector **x** containing the input signal.
- The number of stages **n**.
- A vector **Lo_D** containing the analysis lowpass filter.
- A vector **Hi_D** containing the analysis highpass filter.
- There is the possibility of providing, instead of **Lo_D** and **Hi_D**, **wname**, the name of the wavelet transform. See the description of the function **wfilters** for a list of available names.

Output parameters:

- a vector **c** containing the full wavelet decomposition;
- a vector **l** containing the number of elements of each band in the vector **c**.

Example:

```
load sumsin; x=sumsin; n=3;
[c,l]=wavedec(x,n,'dB1');
indx1=1+l(1)+l(2); indx2=indx1+l(3)-1;
plot(c(indx1:indx2)); %Plots 2nd stage coefficients
plot(c(1:l(1))); %Plots lowpass coefficients
```

- **waverec:** Performs multiple-stage one-dimensional wavelet reconstruction.

Input parameters:

- A vector **c** containing the full wavelet decomposition.
- A vector **l** containing the number of elements of each band in **c**.
- A vector **Lo_D** containing the analysis lowpass filter.
- A vector **Hi_D** containing the analysis highpass filter.
- There is the possibility of providing, instead of **Lo_D** and **Hi_D**, **wname**, the name of the wavelet transform. See the description of the function **wfilters** for a list of available names.

Output parameter: a vector **x** containing the output signal.

Example:

```
x=waverec(c,l,'dB1');
```

- **upwlev:** Performs single-stage reconstruction of a multiple-stage one-dimensional wavelet decomposition.

Input parameters: see function **waverec**.

Output parameters:

- the vector **nc** containing the full wavelet decomposition of the output coefficients obtained after one stage of wavelet reconstruction;
- the vector **nl** containing the number of elements of each band in the vector **nc**.

Example:

```
[nc, nl]=upwlev(c, l, 'dB1');
```

- **appcoef:** Extracts one-dimensional approximation coefficients.

Input parameters:

- A vector **c** containing the full wavelet decomposition.
- A vector **l** containing the number of elements of each band in the vector **c**.
- The number of stages **n**.
- A vector **Lo_D** containing the analysis lowpass filter.
- A vector **Hi_D** containing the analysis highpass filter.
- There is the possibility of providing, instead of **Lo_D** and **Hi_D**, **wname**, the name of the wavelet transform. See the description of the function **wfilters** for a list of available names.

Output parameters:

- the vector **a** containing the approximation coefficients for the given number of stages.

Example:

```
A=appcoef(c, l, 'bior4.4', 3);
```

- **detcoef:** Extracts one-dimensional approximation coefficients.

Input parameters:

- a vector c containing the full wavelet decomposition;
- a vector l containing the number of elements of each band in the vector c ;
- The number of stages n .

Output parameters:

- the vector a containing the approximation coefficients for the given number of stages.

Example:

```
D=detcoef(c,l,3);
```

- The interested reader may also refer to the MATLAB documentation commands **upcoef** and **wrccoef**, as well as to the commands **appcoef2**, **detcoef2**, **dwt2**, **idwt2**, **upcoef2**, **upwlev2**, **wavedec2**, **waverec2**, and **wrccoef2**, which apply to two-dimensional signals.

- **wavefun:** Generates approximations of wavelets and scaling functions given a wavelet transform.

Input parameters:

- The name $wname$ of the wavelet transform. See description of the function **wfilters** for a list of available names.
- The number of iterations $iter$ used in the approximation.

Output parameters:

- a vector ϕ_1 containing the analysis scaling function;
- a vector ψ_1 containing the analysis wavelet;
- a vector ϕ_2 containing the synthesis scaling function;
- a vector ψ_2 containing the synthesis wavelet
(in the case of orthogonal wavelets, only the analysis scaling functions and wavelets are output);
- a grid $xval$ containing 2^{iter} points.

Example:

```
iter=7; wname='dB5';
[phi,psi,xval]=wavefun(wname,iter);
plot(xval,psi);
```

- The MATLAB Wavelet Toolbox provides the command **wavemenu** that integrates most of the commands above into a graphics user interface.
- The command **wavedemo** gives a tour through most of the commands. For further information, type ‘**help command**’, or refer to the Wavelet Toolbox reference manual.

10.13 Summary

In this chapter we have discussed the subject of wavelet transforms. We started with hierarchical filter banks and from them introduced wavelet transforms. We then analyzed the STFT and the continuous-time wavelet transform. Multiresolution decompositions were

then defined and conditions for designing wavelet transforms have been developed. We also analyzed the concepts of regularity and number of vanishing moments, and gave a brief introduction to wavelet transforms of images. We finished the chapter with a discussion on the computation of wavelet transforms of finite-length signals, followed by a hands-on Do-it-yourself section. Also, functions from the MATLAB Wavelet Toolbox were described, allowing one to take full advantage of such an important tool for digital signal processing.

10.14 Exercises

10.1 Deduce Equations (10.1)–(10.4).

10.2 Consider a two-band linear-phase filter bank whose product filter $P(z) = H_0(z)H_1(-z)$ of order $(4M - 2)$ can be expressed as

$$P(z) = z^{-2M+1} + \sum_{k=0}^{M-1} a_{2k} \left(z^{-2k} + z^{-4M+2+2k} \right).$$

Show that such a $P(z)$:

- (a) Can have at most $2M$ zeros at $z = -1$.
- (b) Has exactly $2M$ zeros at $z = -1$ provided that its coefficients satisfy the following set of M equations:

$$\sum_{k=0}^{M-1} a_{2k} (2M - 1 - 2k)^{2n} = \frac{1}{2} \delta(n), \quad n = 0, 1, \dots, (M - 1).$$

Use the fact that a polynomial root has multiplicity p if it is also the root of the first $p - 1$ derivatives of the polynomial.

10.3 Compute the analysis and synthesis wavelets and scaling functions corresponding to the analysis filters in Table 10.4.

Hint: Refer to Figure 10.4 and use the MATLAB functions `dyadup` and `conv` to compute the iterated impulse responses.

Table 10.4.

Coefficients of the lowpass and highpass analysis filters corresponding to the wavelet in Exercises 10.3, 10.11, and 10.14.

$h_0(0) = -0.0234$	$h_1(0) = -0.0502$
$h_0(1) = -0.0528$	$h_1(1) = -0.1128$
$h_0(2) = 0.7833$	$h_1(2) = 0.6444$
$h_0(3) = 0.7833$	$h_1(3) = -0.6444$
$h_0(4) = -0.0528$	$h_1(4) = 0.1128$
$h_0(5) = -0.0234$	$h_1(5) = 0.0502$

- 10.4 Show that the sub-bands from a two-band decomposition of a periodic signal with odd period N have $2N$ independent samples.

Hint: Begin by showing that the polyphase components of a periodic signal with odd period N also have period N .

- 10.5 Show that $\psi(t)$ as defined in Equation (10.64) is orthogonal to $\phi(t - n)$ and that $\psi(t - n)$, for $n \in \mathbb{Z}$, is an orthonormal basis for W_0 (Vetterli & Kovačević, 1995; Mallat, 1999).

- 10.6 For a two-band perfect reconstruction filter bank, assume the analysis filter $H_0(z)$ and the synthesis filter $G_0(z)$ satisfy the condition of Equation (10.142). Assume also that these filters were designed to generate a wavelet so that they have enough zeros placed at $z = 1$.

- (a) Show that a filter bank consisting of an analysis lowpass filter whose impulse response is

$$\hat{h}_0(n) = \frac{1}{2} [h_0(n) + h_0(n - 1)]$$

and the synthesis lowpass filter impulse response is

$$\frac{1}{2} \hat{g}_0(n) = \left[g_0(n) - \frac{1}{2} \hat{g}_0(n - 1) \right]$$

still represents a perfect reconstruction filter bank.

- (b) Show this procedure affects the number of zeros of $\hat{H}_0(z)$ and $\hat{H}_1(z)$ at $z = -1$.
 10.7 Apply the technique, known as *balancing*, described in Exercise 10.6, to the filters of Equations (10.204) and (10.205) and comment on the observed results.
 10.8 Develop a table similar to Table 10.1 for computing symmetric extensions of odd-length signals. Note in this case that for odd order the lowpass and highpass bands may not have the same lengths.
 10.9 Quantization of wavelet transforms: use the signal generated in Experiment 10.1 and compute its N -stage wavelet transform with both the `bior4.4` and the `db4` wavelets. Use $N = 3$, $N = 6$, and $N = 8$. Quantize its coefficients using several quantization step sizes q . Use for the quantized value of x the function `q*round(x/q)`. Then reconstruct the signal from the quantized lowpass band and detail bands and observe the result. Repeat this exercise for the signal `leleccum` used in Experiment 10.2.
 10.10 Repeat Exercise 10.9, this time using as input the image `cameraman.tif` from the MATLAB Image Processing Toolbox. Observe how the increase in the quantization step sizes affects the reconstructed image quality.

Hint: Read the image using the command `imread`, with the care of converting the read matrix from type `uint8` to `double`. Then use the command `wavedec2` to compute the wavelet decomposition given the analysis and synthesis filters. To reconstruct the image after quantization, use the command `waverec2`. In order to display an image, use the command `imshow`.

10.11 Repeat Exercise 10.10 using the filter bank whose analysis lowpass and highpass filters are listed in Table 10.4. Compare the results with those obtained in Exercise 10.10 for the same quantization step sizes.

10.12 Repeat Exercises 10.10 and 10.11, this time comparing the results obtained from the use of the periodic and symmetric extensions. Refer to the MATLAB function `dwtmode`, which sets the type of signal extension used when computing wavelet transforms.

10.13 Develop a MATLAB program to compute the regularity of a wavelet and scaling function given its corresponding lowpass and highpass filters.

Hint: Refer to Figure 10.4 and use the MATLAB functions `dyadup` and `conv` to compute the iterated impulse response in Equation (10.194). Remember that one must first factorize the z transforms of the analysis and synthesis filters according to Equation (10.193). In order to do so, the functions `zp2tf` and `tf2zp` can be used. You can also use the fact that the wavelet `db4` and the analysis and synthesis lowpass and highpass filters of the `bior4.4` wavelets have four zeros at $z = -1$. Try out the wavelets used in Experiments 10.1 and 10.2, as well as wavelets from the MATLAB Wavelet Toolbox (see MATLAB command `wfilters` in Section 10.12).

10.14 Compute the number of vanishing moments of the analysis and synthesis wavelets of Exercise 10.3, using, for instance, the MATLAB function `tf2zp`.

10.15 Compute the STFT of the signal generated in Experiment 10.1 and compare it with the wavelet transform obtained in Experiment 10.1, using the MATLAB command `spectrogram`. Describe the differences in the representations of sinusoids and impulses in both cases.

10.16 If one wants to compute the wavelet transform of a continuous signal, then one has to assume that its digital representation is derived as in Equation (10.9) and Figure 10.9. However, the digital representation of a signal is usually obtained by filtering it with a band-limiting filter and sampling it at or above the Nyquist frequency. Supposing that the continuous signal, in order to be digitized, is filtered with a filter arbitrarily close to the ideal one, find an expression for the mean-square error made in the computation of the wavelet transform. Discuss the implications of this result in the accuracy of signal processing operations carried out in the wavelet transform domain.

11.1 Introduction

This chapter starts by addressing some implementation methods for digital filtering algorithms and structures. The implementation of any building block of digital signal processing can be performed using a software routine on a simple personal computer. In this case, the designer's main concern becomes the description of the desired filter as an efficient algorithm that can be easily converted into a piece of software. In such cases, the hardware concerns tend to be noncritical, except for some details such as memory size, processing speed, and data input/output.

Another implementation strategy is based on specific hardware, especially suitable for the application at hand. In such cases, the system architecture must be designed within the speed constraints at a minimal cost. This form of implementation is mainly justified in applications that require high processing speed or in large-scale production. The four main forms of appropriate hardware for implementing a given system are:

- The development of a specific architecture using basic commercial electronic components and integrated circuits (Jackson *et al.*, 1968; Peled & Liu, 1974, 1985; Freeny, 1975; Rabiner & Gold, 1975; Wanhammar, 1981).
- The use of programmable logic devices (PLDs), such as field-programmable gate arrays (FPGAs), which represent an intermediate integrated stage between discrete hardware and full-custom integrated circuits or digital signal processors (DSPs) (Skahill, 1996).
- The design of a dedicated integrated circuit for the application at hand using computer-automated tools for a very large scale integration (VLSI) design. The VLSI technology allows several basic processing systems to be integrated in a single silicon chip. The basic goal in designing application-specific integrated circuits (ASICs) is to obtain a final design satisfying very strict specifications with respect to, for example, chip area, power consumption, processing speed, testability, and overall production cost (Skahill, 1996). These requirements are often achieved with the aid of computer-automated design systems.
- The use of a commercially available general-purpose DSP to implement the desired system. The complete hardware system must also include external memory, input- and output-data interfaces, and sometimes analog-to-digital and digital-to-analog converters. There are several commercial DSPs available today, which include features such as fixed- or floating-point operation, several ranges of clock speed and price, internal memory,

and very fast multipliers (Analog Devices, Inc., 2004a,b, 2005, 2009; Texas Instruments, 2006, 2008; Freescale Semiconductor, 2007). In summary, the DSPs are meant to deal with signal processing tasks requiring intensive math calculations. As an example, most mobile phones include a DSP.

The state-of-the-art implementation of digital signal processing systems falls beyond the scope of this book. Instead, we address some fundamental concepts related to hardware implementation.

This chapter starts by discussing the most widely used binary number representations in digital signal processing. Then, we introduce the basic elements necessary for the implementation of systems for digital signal processing, in particular the digital filters extensively covered by this book. Distributed arithmetic is presented as a design alternative for digital filters that eliminates the necessity of multiplier elements. These very basic implementation concepts illustrate issues related to hardware implementation of digital signal processing systems.

It is certain that in practice a digital signal processing system is implemented by software on a digital computer, either using a general-purpose DSP, or using dedicated hardware for the given application. In either case, quantization errors are inherent due to the finite-precision arithmetic. These errors are of the following types:

- Errors due to the quantization of the input signals into a set of discrete levels, such as the ones introduced by the analog-to-digital converter.
- Errors in the frequency response of filters, or in transform coefficients, due to the finite-wordlength representation of multiplier constants.
- Errors made when internal data, like outputs of multipliers, are quantized before or after subsequent additions.

All these error forms depend on the type of arithmetic utilized in the implementation. If a digital signal processing routine is implemented on a general-purpose computer, since floating-point arithmetic is in general available, this type of arithmetic becomes the most natural choice. On the other hand, if the building block is implemented on some special-purpose hardware or a fixed-point DSP, then fixed-point arithmetic may be the best choice, because it is less costly in terms of hardware and simpler to design. A fixed-point implementation usually implies a lot of savings in terms of chip area as well.

For a given application, the quantization effects are key factors to be considered when assessing the performance of a digital signal processing algorithm. In this chapter, the various quantization effects are introduced along with the most widely used formulations for their analyses. In particular, the product and coefficient quantization effects are discussed in some detail along with the techniques to scale the internal signals in order to avoid frequent overflows. The chapter also discusses strategies to eliminate zero-input and constant-input granular limit cycles and to avoid sustained overflow oscillations. The Do-it-yourself section illustrates some finite precision issues in a practical example.

11.2 Binary number representation

11.2.1 Fixed-point representations

In most of the cases when digital signal processing systems are implemented using fixed-point arithmetic, the numbers are represented in one of the following forms: sign-magnitude, one's-complement, and two's-complement formats. These representations are described in this subsection, where we implicitly assume that all numbers have been previously scaled to the interval $x \in (-1, 1)$.

In order to clarify the definitions of the three types of number representation given below, we associate, for every positive number x such that $x < 2$, a function that returns its representation in base 2, $\mathcal{B}(x)$, which is defined by

$$\mathcal{B}(x) = x_0.x_1x_2 \cdots x_n \quad (11.1)$$

and

$$x = \mathcal{B}^{-1}(x_0.x_1x_2 \cdots x_n) = x_0 + x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}. \quad (11.2)$$

11.2.1.1 Sign-magnitude representation

The sign-magnitude representation of a given number consists of a sign bit followed by a binary number representing its magnitude. That is:

$$[x]_M = s_x.x_1x_2x_3 \cdots x_n, \quad (11.3)$$

where s_x is the sign bit and $x_1x_2x_3 \cdots x_n$ represents the magnitude of the number in base 2; that is, in binary format. Here, we use $s_x = 0$ for positive numbers and $s_x = 1$ for negative numbers. This means that the number x is given by

$$x = \begin{cases} \mathcal{B}^{-1}(0.x_1x_2 \cdots x_n) = x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}, & \text{for } s_x = 0 \\ -\mathcal{B}^{-1}(0.x_1x_2 \cdots x_n) = -(x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}), & \text{for } s_x = 1. \end{cases} \quad (11.4)$$

11.2.1.2 One's-complement representation

The one's-complement representation of a number is given by

$$[x]_{1c} = \begin{cases} \mathcal{B}(x), & \text{if } x \geq 0 \\ \mathcal{B}(2 - 2^{-n} - |x|), & \text{if } x < 0, \end{cases} \quad (11.5)$$

where \mathcal{B} is defined by Equations (11.1) and (11.2). Notice that, in the case of positive numbers, the one's-complement and the sign-magnitude representations are identical. However, for negative numbers, the one's complement is generated by changing all the 0s to 1s and all the 1s to 0s in the sign-magnitude representation of its absolute value. As before $s_x = 0$ for positive numbers and $s_x = 1$ for negative numbers.

11.2.1.3 Two's-complement representation

The two's-complement representation of a number is given by

$$[x]_{2c} = \begin{cases} \mathcal{B}(x), & \text{if } x \geq 0 \\ \mathcal{B}(2 - |x|), & \text{if } x < 0, \end{cases} \quad (11.6)$$

where \mathcal{B} is defined by Equations (11.1) and (11.2). Again, for positive numbers, the two's-complement representation is identical to the sign-magnitude representation. The representation of a negative number in two's complement can be obtained by adding 1 to the least significant bit of the number represented in one's complement. Then, as in the one's-complement case, $s_x = 0$ for positive numbers and $s_x = 1$ for negative numbers.

If the two's-complement representation of a number x is given by

$$[x]_{2c} = s_x.x_1x_2 \cdots x_n, \quad (11.7)$$

then, from Equation (11.6), we have that

- for $s_x = 0$, then

$$x = \mathcal{B}^{-1}([x]_{2c}) = \mathcal{B}^{-1}(0.x_1x_2 \cdots x_n) = x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}; \quad (11.8)$$

- for $s_x = 1$, then

$$2 - |x| = 2 + x = \mathcal{B}^{-1}([x]_{2c}) = \mathcal{B}^{-1}(s_x.x_1x_2 \cdots x_n) \quad (11.9)$$

and then

$$\begin{aligned} x &= -2 + \mathcal{B}^{-1}(1.x_1x_2 \cdots x_n) \\ &= -2 + 1 + x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n} \\ &= -1 + x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}. \end{aligned} \quad (11.10)$$

From Equations (11.8) and (11.10), we have that x can be generally expressed as

$$x = -s_x + x_12^{-1} + x_22^{-2} + \cdots + x_n2^{-n}. \quad (11.11)$$

This short notation is very useful for introducing the multiplication operation of binary numbers represented in two's-complement format, as given in Section 11.3.1.

The one's-complement and two's-complement representations are more efficient for addition implementations, whereas the sign-magnitude representation is more efficient for the implementation of multiplications (Hwang, 1979). Overall, the most widely used binary code is the two's-complement representation.

Example 11.1. Represent the number -0.1875 using 8 bits (including the sign bit) in one's-complement and two's-complement representations.

Solution

We first obtain the standard binary representation for the absolute value of the number, using the following procedure. Multiply 0.1875 by 2 and associate a bit 0, if the product is less than 1.0, or a bit 1, if the product is greater than or equal to 1.0. In this last case, subtract 1.0 from the product, to bring the partial result back to below 1.0. Repeat this process for each required bit, yielding, in this case, the following computations:

$$\begin{array}{rcl} \text{Multiplication by 2} & & \text{Bit} \\ \hline 0.1875 \times 2 = 0.375 & 0 \\ 0.375 \times 2 = 0.75 & 0 \\ 0.75 \times 2 = 1.5 & 1 \\ 0.5 \times 2 = 1.0 & 1 \\ 0.0 \times 2 = 0 & 0 \\ 0.0 \times 2 = 0 & 0 \\ 0.0 \times 2 = 0 & 0 \end{array} \quad (11.12)$$

Hence, the binary representation of 0.1875 is 0.0011000. By changing all bits 0 by bits 1 and vice versa, we obtain the one's-complement representation 1.1100111 of -0.1875 .

The two's-complement representation can be computed by adding 1 to the least significant bit of the one's-complement representation, yielding 1.1101000. \triangle

11.2.2 Signed power-of-two representation

An alternative representation of a number consisting of weighted sums and subtractions is referred to as the signed-digit (SD) representation. The main particular case of SD is the signed power-of-two (SPT) representation, also known as radix-two SD code. The SPT representation allows a multiplication to be performed as a very simple sequence of shifts, additions, and subtractions, which is very attractive from the hardware implementation point of view (Hwang, 1979). A given number x is represented in SPT as

$$x = x_0 + x_1 2^{-1} + x_2 2^{-2} + \cdots + x_n 2^{-n}, \quad (11.13)$$

where $x_i \in \{-1, 0, 1\} = \{\bar{1}, 0, 1\}$. Note that no sign digit is required in the SPT format. For instance, the number -0.1875 can be represented in SPT format as

$$x_0 x_1 x_2 x_3 x_4 = \begin{cases} 000\bar{1}\bar{1} \\ 00\bar{1}01 \\ 00\bar{1}1\bar{1} \\ 0\bar{1}101 \\ 0\bar{1}11\bar{1} \end{cases}. \quad (11.14)$$

Since the SPT representation of a number is not unique, representations with the maximum number of 0 digits, which may also not be unique, are desirable from the computational point of view. If, in addition, we avoid two consecutive nonzero digits, the so-called canonic

SD (CSD) representation arises, which can be shown to be unique (Hwang, 1979). For instance, the second SPT representation in Equation (11.14) is the corresponding CSD code for -0.1875 .

Given an $(n+1)$ -bit two's-complement representation $[x]_{2c} = s_x.x_1x_2 \cdots x_n$ of a number, it can be transformed into a CSD representation $[x']_{\text{CSD}} = x'_0x'_1 \cdots x'_n$ through the following algorithm.

- (i) Initialization. Set the auxiliary variables $x_{-1} = s_x$ and $x_{n+1} = \delta_{n+1} = 0$.
- (ii) For $i = n, (n-1), \dots, 0$ determine, in sequence:

$$\theta_i = x_i \oplus x_{i+1} \quad (11.15)$$

$$\delta_i = \overline{\delta_{i+1}} \times \theta_i \quad (11.16)$$

$$x'_i = (1 - 2x_{i-1})\delta_i \quad (11.17)$$

where $\overline{[\cdot]}$, \times , and \oplus represent the binary complement, AND, and exclusive-OR operations, respectively.

The main advantage of the SPT representation over the two's-complement is that the inclusion of SD leads to a reduction in the number of nonzero digits, simplifying subsequent arithmetic operations. Using the CSD representation, for instance, the number of nonzero digits tends to $(3n + 4)/9$ as n increases (Reitwiesner, 1960; Yu, 2003). The hardware reduction property of the CSD representation has been exploited in several specialized filter design approaches, such as those in Horng *et al.* (1991) and Yu and Lim (2002).

11.2.3 Floating-point representation

Using floating-point representation, a number is represented as

$$x = x_m 2^c, \quad (11.18)$$

where x_m is the mantissa and c is the number exponent, with $\frac{1}{2} \leq |x_m| < 1$. For example, the number $-0.001\ 875$ can be represented in floating point by 1.1101000×2^{-2} , where the mantissa has a two's-complement representation.

When compared with fixed-point representations, the main advantage of the floating-point representation is its large dynamic range. Its main disadvantage is that its implementation is more complex. For example, in floating-point arithmetic, the mantissa must be quantized after both multiplications and additions, whereas in fixed-point arithmetic quantization this is required only after multiplications. In this text, we deal with both representation systems. However, we put the focus more on the fixed-point arithmetic, since it is more prone to errors, thus requiring more attention.

11.3 Basic elements

11.3.1 Properties of the two's-complement representation

Since the two's-complement representation is vital to the understanding of the arithmetic operations described in the following sections, here we supplement the introduction given in Section 11.2, giving some other properties of the two's-complement representation.

We start by repeating Equation (11.11), which states that if the two's-complement representation of x is given by

$$[x]_{2c} = s_x \cdot x_1 x_2 \cdots x_n, \quad (11.19)$$

then the value of x is determined by

$$x = -s_x + x_1 2^{-1} + x_2 2^{-2} + \cdots + x_n 2^{-n}. \quad (11.20)$$

One of the advantages of the two's-complement representation is that if A and B are represented in two's complement, then $C = A - B$, represented in two's complement, can be computed by adding A to the two's complement of B . Also, given x as in Equation (11.20), we have that

$$\frac{x}{2} = -s_x 2^{-1} + x_1 2^{-2} + x_2 2^{-3} + \cdots + x_n 2^{-n-1}, \quad (11.21)$$

and since $-s_x 2^{-1} = -s_x + s_x 2^{-1}$, then Equation (11.21) can be rewritten as

$$\frac{x}{2} = -s_x + s_x 2^{-1} + x_1 2^{-2} + x_2 2^{-3} + \cdots + x_n 2^{-n-1}; \quad (11.22)$$

that is, the two's-complement representation of $x/2$ is given by

$$\left[\frac{x}{2} \right]_{2c} = s_x \cdot s_x x_1 x_2 \cdots x_n. \quad (11.23)$$

This property implies that dividing a number represented in two's complement by 2 is equivalent to shifting all of its bits one position to the right, with the extra care of repeating the sign bit. This property is very important in the development of the multiplication algorithm, which involves multiplications by 2^{-j} .

11.3.2 Serial adder

A very economic implementation of digital filters is achieved through the use of so-called serial arithmetic. Such an approach performs a given operation by processing the bits representing a given binary number one by one serially. The overall result tends to be very efficient in terms of required hardware, power consumption, modularity, and ease of interconnection between serial-bit cells. The main drawback is clearly the associated processing speed, which tends to be very slow, when compared with other approaches.

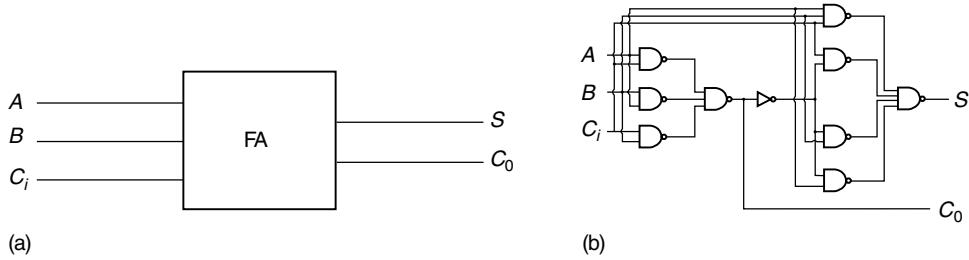


Fig. 11.1. Full adder: (a) symbol; (b) logic circuit.

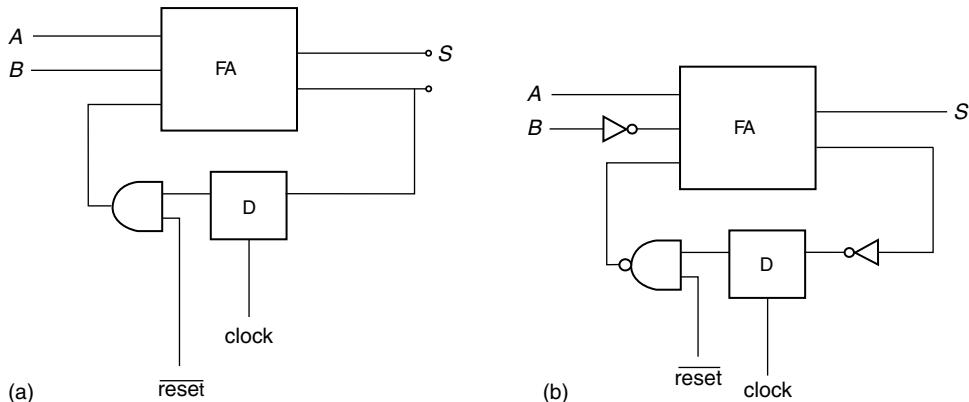


Fig. 11.2. Serial implementations using the full adder as a basic block: (a) adder; (b) subtractor.

An important basic element for the implementation of all signal processing systems is the full adder whose symbol and respective logic circuit are shown in Figure 11.1. Such a system presents two output ports: one equal to the sum of the two input bits A and B and the other, usually referred to as the carry bit, corresponding to the possible extra bit generated by the addition operation. A third input port C_i is used to allow the carry bit from a previous addition to be taken into account in the current addition.

A simple implementation of a serial adder for the two's-complement arithmetic, based on the full adder, is illustrated in Figure 11.2a. In such a system, the two input words A and B must be serially fed to the adder starting with their least-significant bit. A D-type flip-flop (labeled D) is used to store the carry bit from a given 1-bit addition, saving it to be used in the addition corresponding to the next significant bit. A reset signal is used to clear this D-type flip-flop before starting the addition of two other numbers, forcing the carry input to be zero at the beginning of the summation.

Figure 11.2b depicts the serial subtractor for the two's-complement arithmetic. This structure is based on the fact that $A - B$ can be computed by adding A to the two's complement of B , which can be determined by inverting B and summing 1 in the least-significant bit. The representation of B in two's complement is commonly done with an inverter at the input of B and with a NAND gate, substituting the AND gate, at the carry input. Then, at

the beginning of the summation, the signal `reset` is asserted, and the carry input becomes 1. An extra inverter must be placed at the carry output because, with the use of the NAND gate, the carry output fed back to the D-type flip-flop must be inverted.

11.3.3 Serial multiplier

The most complex basic element for digital signal processing is the multiplier. In general, the product of two numbers is determined as a sum of partial products, as performed in the usual multiplication algorithm. Naturally, partial products involving a bit equal to zero do not need to be performed or taken into account. For that matter, there are several filter designs in the literature that attempt to represent all filter coefficients as the sum of a minimum number of nonzero bits (Pope, 1985; Ulbrich, 1985).

Let A and B be two numbers of m and n bits respectively that can be represented using two's-complement arithmetic as

$$[A]_{2c} = s_A \cdot a_1 a_2 \cdots a_m \quad (11.24)$$

$$[B]_{2c} = s_B \cdot b_1 b_2 \cdots b_n \quad (11.25)$$

such that, from Equation (11.20), A and B are given by

$$A = -s_A + a_1 2^{-1} + a_2 2^{-2} + \cdots + a_m 2^{-m} \quad (11.26)$$

$$B = -s_B + b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}. \quad (11.27)$$

Using two's-complement arithmetic, the product $P = AB$ is given by

$$\begin{aligned} P &= (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})(-s_B + b_1 2^{-1} + \cdots + b_n 2^{-n}) \\ &= (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})b_n 2^{-n} \\ &\quad + (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})b_{n-1} 2^{-n+1} \\ &\quad \vdots \\ &\quad + (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})b_1 2^{-1} \\ &\quad - (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})s_B. \end{aligned} \quad (11.28)$$

We can perform this multiplication in a step-by-step manner by first summing the terms multiplied by b_n and b_{n-1} , taking the result and adding to the term multiplied by b_{n-2} , and so on. Let us develop this reasoning a bit further. The sum of the first two terms can be written as

$$\begin{aligned} C &= b_n 2^{-n} A + b_{n-1} 2^{-n+1} A \\ &= (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})b_n 2^{-n} \\ &\quad + (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})b_{n-1} 2^{-n+1} \end{aligned}$$

$$= 2^{-n+1} [b_n 2^{-1} (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m}) \\ + b_{n-1} (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m})]. \quad (11.29)$$

From Equation (11.22), the above equation becomes

$$C = 2^{-n+1} \left[b_n (-s_A + s_A 2^{-1} + a_1 2^{-2} + \cdots + a_m 2^{-m-1}) \\ + b_{n-1} (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m}) \right], \quad (11.30)$$

which can be represented in the form of the multiplication algorithm as

$$\begin{array}{r} (\dots s_A s_A a_1 \dots a_{m-1} a_m) \times b_n \\ + (\dots s_A a_1 a_2 \dots a_m) \times b_{n-1}; \\ \hline s_C c_1 c_2 c_3 \dots c_{m+1} c_{m+2} \end{array}$$

that is:

$$C = 2^{-n+2} (-s_C + c_1 2^{-1} + c_2 2^{-2} + c_3 2^{-3} + \cdots + c_{m+1} 2^{-m-1} + c_{m+2} 2^{-m-2}). \quad (11.31)$$

Note that the sum of two positive numbers is always positive, and the sum of two negative numbers is always negative. Therefore, $s_C = s_A$. In fact, since in two's-complement arithmetic the sign has to be extended, the more compact representation for the summation above would be

$$\begin{array}{r} (\dots s_A s_A s_A s_A a_1 \dots a_{m-1} a_m) \times b_n \\ + (\dots s_A s_A s_A a_1 a_2 \dots a_m) \times b_{n-1}; \\ \dots s_A s_A c_1 c_2 c_3 \dots c_{m+1} c_{m+2} \end{array}$$

In the next step, C should be added to $b_{n-2} 2^{-n+2} A$. This yields

$$D = (-s_A + c_1 2^{-1} + \cdots + c_{m+2} 2^{-m-2}) 2^{-n+2} \\ + (-s_A + a_1 2^{-1} + \cdots + a_m 2^{-m}) b_{n-2} 2^{-n+2}, \quad (11.32)$$

which can be represented in the compact form of the multiplication algorithm as

$$\begin{array}{r} (\dots s_A s_A c_1 c_2 \dots c_m c_{m+1} c_{m+2}) \\ + (\dots s_A s_A a_1 a_2 \dots a_m) \times b_{n-2}; \\ \dots s_A d_1 d_2 d_3 \dots d_{m+1} d_{m+2} d_{m+3} \end{array}$$

which is equivalent to

$$D = 2^{-n+3} (-s_A + d_1 2^{-1} + d_2 2^{-2} + \cdots + d_{m+2} 2^{-m-2} + d_{m+3} 2^{-m-3}). \quad (11.33)$$

This process goes on until we obtain the next-to-last partial sum Y . If B is positive, then s_B is zero and the final product Z is equal to Y ; that is:

$$Z = Y = -s_A + y_1 2^{-1} + \cdots + y_m 2^{-m} + y_{m+1} 2^{-m-1} + \cdots + y_{m+n} 2^{-m-n}, \quad (11.34)$$

such that the two's complement of Z is given by

$$[Z]_{2c} = s_A \cdot z_1 z_2 \cdots z_m z_{m+1} \cdots z_{m+n} = s_A \cdot y_1 y_2 \cdots y_m y_{m+1} \cdots y_{m+n}. \quad (11.35)$$

On the other hand, if B is negative, then $s_B = 1$ and Y still needs to be subtracted from $s_B A$. This can be represented as

$$\begin{array}{r} (s_A \quad y_1 \quad y_2 \quad \dots \quad y_m \quad y_{m+1} \quad \dots \quad y_{m+n}) \\ - (s_A \quad a_1 \quad a_2 \quad \dots \quad a_m) \\ \hline s_Z \quad z_1 \quad z_2 \quad \dots \quad z_m \quad z_{m+1} \quad \dots \quad z_{m+n} \end{array} \times s_B.$$

The full precision two's-complement multiplication of A , with $(m + 1)$ bits, by B , with $(n + 1)$ bits, should be represented with $(m + n + 1)$ bits. If we want to represent the final result using just the same number of bits as A – that is, $(m + 1)$ – then we can use either rounding or truncation. For truncation, it suffices to disregard the bits $z_{m+1}, z_{m+2}, \dots, z_{m+n}$. For rounding, we must add to the result, prior to truncation, a value equal to $\Delta = 2^{-m-1}$, such that

$$[\Delta]_{2c} = 0. \underbrace{0 \cdots 0}_{m \text{ zeros}} 1. \quad (11.36)$$

By looking at the algorithmic representation of the last partial sum above, one sees that to add Δ to the result is equivalent to summing bit 1 at position $(m + 1)$, which is the n th position from the rightmost bit, z_{m+n} . Since it does not matter whether this bit is summed in the last or the first partial sum, it suffices to sum this bit during the first partial sum. Then the rounding can be performed by summing the number

$$[Q]_{2c} = 1. \underbrace{0 \cdots 0}_{n-1 \text{ zeros}} \quad (11.37)$$

to the first partial sum. In addition, since only $(m + 1)$ bits of the product will be kept, we need to keep, from each partial sum, its $(m + 1)$ most significant bits.

The main idea behind a serial multiplier is to perform each partial sum using a serial adder such as the one depicted in Figure 11.1, taking care to introduce proper delays between the serial adder (SA) blocks, to align each partial sum properly with $b_j A$. This scheme is depicted in Figure 11.3, where the rounding signal Q is introduced in the first partial sum, and the least significant bits of A and Q are input first.

In the scheme of Figure 11.3, depending on the values of the delay elements, one serial adder can begin to perform one partial sum as soon as the first bit of the previous partial sum becomes available. When this happens, we have what is called a pipelined architecture. The main idea behind the concept of pipelining is to use delay elements at strategic points of the system that allow one operation to start before the previous operation has finished.

Figure 11.4 shows a pipelined serial multiplier. In this multiplier, the input A is in two's-complement format, while the input B is assumed to be positive. In this figure, the cells labeled D are D-type flip-flops, all sharing the same clock line, which is omitted for convenience, and the cells labeled SA represent the serial adder depicted in Figure 11.2a. The

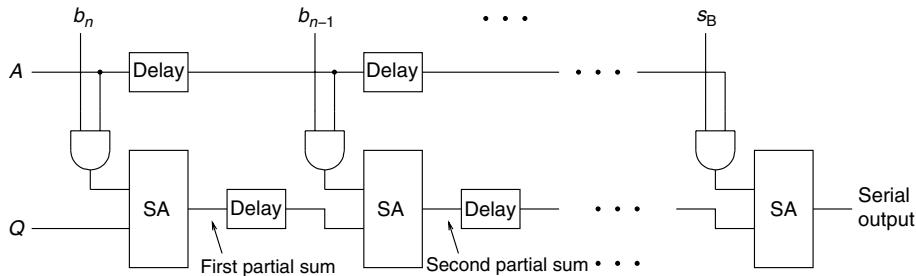


Fig. 11.3. Schematic representation of a serial multiplier.

latch element used is shown in Figure 11.5. In this case, if the enable signal is high, after the clock, the output y becomes equal to the input x ; otherwise, the output y keeps its previous value.

In Figure 11.4, since B is assumed to be positive, then $s_B = 0$. Also, the rounding signal Q is such that

$$[Q]_{2c} = \begin{cases} 00 \dots 00, & \text{for truncation} \\ \dots 01 \underbrace{0 \dots 0}_{n+1 \text{ zeros}}, & \text{for rounding} \end{cases} . \quad (11.38)$$

In the case of rounding, it is important to note that $[Q]_{2c}$ has two more zeros to the right than the one in Equation (11.37). This is so because this signal must be synchronized with input $b_n A$, which is delayed by 2 bits before entering the serial adder of the first cell. Finally, the control signal CT is equal to

$$CT = 00 \underbrace{1 \dots 1}_m 0 \quad (11.39)$$

It should be noticed that all signals A , B , Q , and CT should be input from right to left; that is, starting from their least-significant bits. Naturally, the output signal is generated serially, also starting from its least-significant bit, the first bit coming out after $(2n + 3)$ clock cycles and the entire word taking a total of $(2n + m + 3)$ cycles to be calculated.

In the literature, it has been shown that the multiplication of two words of lengths $(m + 1)$ and $(n + 1)$ using a serial or serial-parallel multiplier takes at least $(m + n + 2)$ clock cycles (Hwang, 1979), even in the cases where the final result is quantized with $(m + 1)$ bits. Using a pipelined architecture, it is possible to determine the $(m + 1)$ -bit quantized product at every $(m + 1)$ clock cycles.

In the serial multiplier of Figure 11.4, each basic cell performs one partial sum of the multiplication algorithm. As the result of cell j is being generated, it is directly fed to cell $(j + 1)$, indicating the pipelined nature of the multiplier. A detailed explanation of the behavior of the circuit is given below:

- (i) After the $(2j + 1)$ th clock cycle, b_{m-j} will be at the input of the upper latch of cell $j + 1$. The first 0 of the control signal CT will be at input ct_{j+1} at this time, which will reset the serial adder and enable the upper latch of cell $j + 1$. Therefore, after the $(2j + 2)$ th

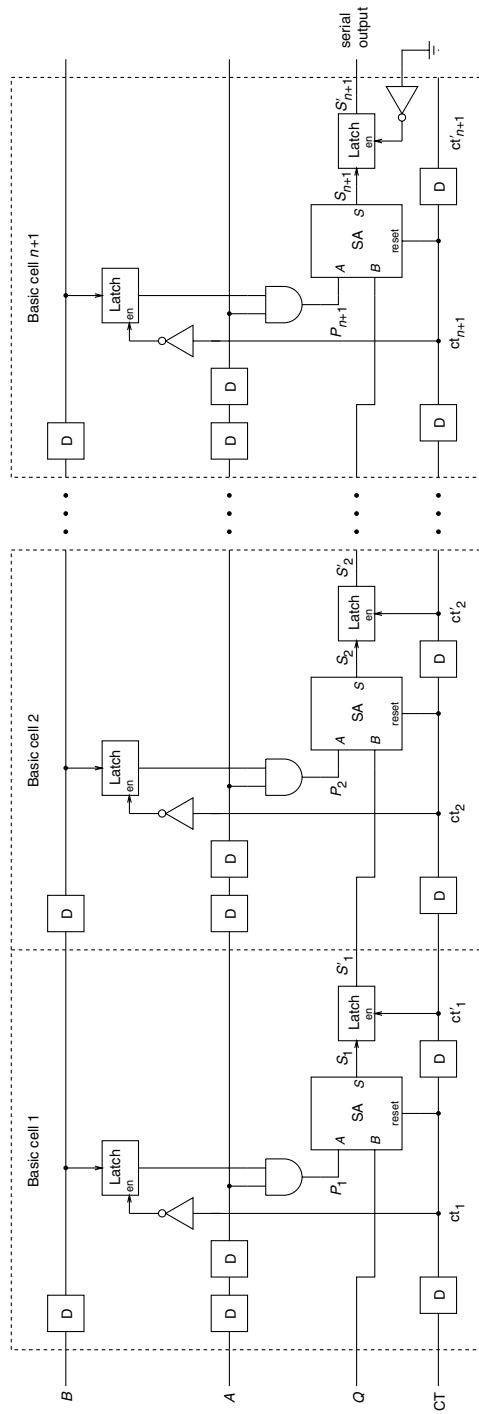


Fig. 11.4. Basic architecture of the pipelined serial multiplier. Clock lines have been suppressed for convenience.

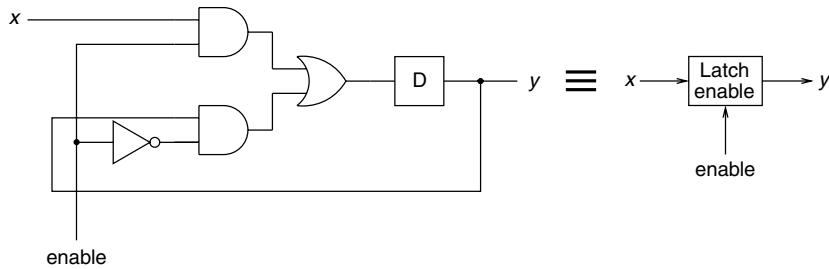


Fig. 11.5. Latch element.

clock cycle, b_{m-j} will be at the output of the upper latch, and will remain there for m clock cycles (the number of 1s of the signal CT).

- (ii) After the $(2j + 2)$ th clock cycle, a_m will be at the input of the AND gate of cell $j + 1$; therefore, $a_m b_{m-j}$ will be the P_{j+1} input of the serial adder. Since at this time the first 0 of the control signal CT will be at input ct'_{j+1} , the lower latch becomes on hold. Therefore, although the first bit of the $(j + 1)$ th partial sum is at S_{j+1} after the $(2j + 2)$ th clock cycle, it will not be at S'_{j+1} after the $(2j + 3)$ th clock cycle, and it will be discarded. Since in the next $m + 1$ clock cycles the bit 1 will be at ct'_{j+1} , the remaining $m + 1$ bits of the $(j + 1)$ th partial sum will pass from S_{j+1} to S'_{j+1} , which is input to the next basic cell.
- (iii) After the $(2j + 2 + k)$ th clock cycle, $a_{m-k} b_{m-j}$ will be the P_{j+1} input of the serial adder of cell $j + 1$. Therefore, after the $(2j + 2 + m)$ th clock cycle, $s_A b_{m-j}$ will be at the P_{j+1} input of the serial adder of cell $j + 1$. This is equivalent to saying that the last bit of the $(j + 1)$ th partial sum will be at the input of the lower latch of cell $j + 1$ at this time. Since then ct'_{j+1} is still 1, then, after the $(2j + 3 + m)$ th clock cycle the last bit of the partial sum of cell $j + 1$ will be at the output of the lower latch of cell $j + 1$. But from this time on, ct'_{j+1} will be zero; therefore, the output of the latch of cell $j + 1$ will hold the last bit of the $(j + 1)$ th partial sum. Since this last bit represents the sign of the $(j + 1)$ th partial sum, it performs the sign extension necessary in two's-complement arithmetic.
- (iv) Since there is no need to perform sign extension of the last partial sum, the lower latch of the last cell is always enabled, as indicated in Figure 11.4.
- (v) Since each basic cell but the last one only outputs m bits apart from the sign extensions, then the serial output at the last cell will contain the product either truncated or rounded to $m + 1$ bits, depending on the signal Q .

Example 11.2. Verify how the pipelined serial multiplier depicted in Figure 11.4 processes the product of the binary numbers $A = 1.1001$ and $B = 0.011$.

Solution

We have that $m = 4$ and $n = 3$. Therefore, the serial multiplier has four basic cells. We expect the least-significant bit of the truncated product to be output after $(2n + 3) = 9$ clock cycles and its last bit after $(2n + m + 3) = 13$ clock cycles. Supposing that the quantization signal Q is zero, which corresponds to a truncated result, we have that (the variable t indicates the clock cycle after which the values of the signals are given; $t = 0$

means the time just before the first clock pulse):

t	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_1	0	0	0	0	0	0	0	1	1	0	0	1	0	0
S_1	0	0	0	0	0	0	0	1	1	0	0	1	0	0
ct'_1	0	0	0	0	0	0	0	1	1	1	1	0	0	0
S'_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
P_2	0	0	0	0	0	1	1	0	0	1	0	0	0	0
S_2	0	0	0	0	0	1	0	1	0	1	0	0	0	0
ct'_2	0	0	0	0	0	1	1	1	0	0	0	0	0	0
S'_2	1	1	1	1	1	0	1	0	0	0	0	0	0	0
P_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_3	1	1	1	1	1	0	1	0	0	0	0	0	0	0
ct'_3	0	0	0	1	1	1	1	0	0	0	0	0	0	0
S'_3	1	1	1	1	0	1	0	0	0	0	0	0	0	0
P_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_4	1	1	1	1	0	1	0	0	0	0	0	0	0	0
ct'_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S'_4	1	1	1	0	1	0	0	0	0	0	0	0	0	0

And the computed product is 1.1101.

For rounding, since $n = 3$, we have, from Equation (11.38), that $[Q]_{2c} = \dots 0010000$. Thus, the operation of the serial multiplier is as follows:

t	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P_1	0	0	0	0	0	0	0	1	1	0	0	1	0	0
S_1	0	0	0	0	0	0	0	1	1	1	0	1	0	0
ct'_1	0	0	0	0	0	0	0	1	1	1	1	0	0	0
S'_1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
P_2	0	0	0	0	0	1	1	0	0	1	0	0	0	0
S_2	0	0	0	0	0	1	0	1	1	1	0	0	0	0
ct'_2	0	0	0	0	0	1	1	1	1	0	0	0	0	0
S'_2	1	1	1	1	1	0	1	1	0	0	0	0	0	0
P_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_3	1	1	1	1	1	0	1	1	0	0	0	0	0	0
ct'_3	0	0	0	1	1	1	1	0	0	0	0	0	0	0
S'_3	1	1	1	1	0	1	0	0	0	0	0	0	0	0
P_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_4	1	1	1	1	0	1	0	0	0	0	0	0	0	0
ct'_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S'_4	1	1	1	0	1	0	0	0	0	0	0	0	0	0

That is, the rounded product is also 1.1101.

△

The general multiplier for two's-complement arithmetic (without the positive restriction in one of the factors) can be obtained from the multiplier seen in Figure 11.4 by a slight modification of the connections between the last two basic cells. In fact, from the representation given in Equation (11.20), we note that the general multiplication of any two numbers represented in two's complement is equal to the multiplication of a positive number and a two's-complement number, except that in the last step we must perform the subtraction of the product between the data and the coefficient sign bit from the partial result obtained at that point. Then, we must perform a subtraction to obtain $S_{n+1} = S'_n - s_A A$. It can be shown that, using two's-complement arithmetic, $X - Y = Y + \overline{X}$, where \overline{X} represents the inversion of all bits of X (see Exercise 11.2). Therefore, in the last cell, we should invert S'_n before it is input to the $(n+1)$ th serial adder and then invert its output. Thus, the connections to the last basic cell should be modified as shown in Figure 11.6. An alternative implementation for the rounding quantization, instead of using the Q signal, consists of forcing the carry signal in serial adder of the n th basic cell to 1 while it is performing the addition of $a_m b_1$ to the $(n-1)$ th partial sum. Other versions of the pipelined serial multiplier can be found in Lyon (1976).

So far, we have focused on single-precision multipliers; that is, the final product is quantized, either by rounding or truncation. As seen in previous chapters, in many cases it is desirable that the final product presents double precision to avoid, for instance, nonlinear oscillations. Such operations can be easily performed with the basic multipliers seen so far, artificially doubling the precision of the two multiplying factors by juxtaposing the adequate number of bits 0 to the right of each input number. For example, the exact multiplication of

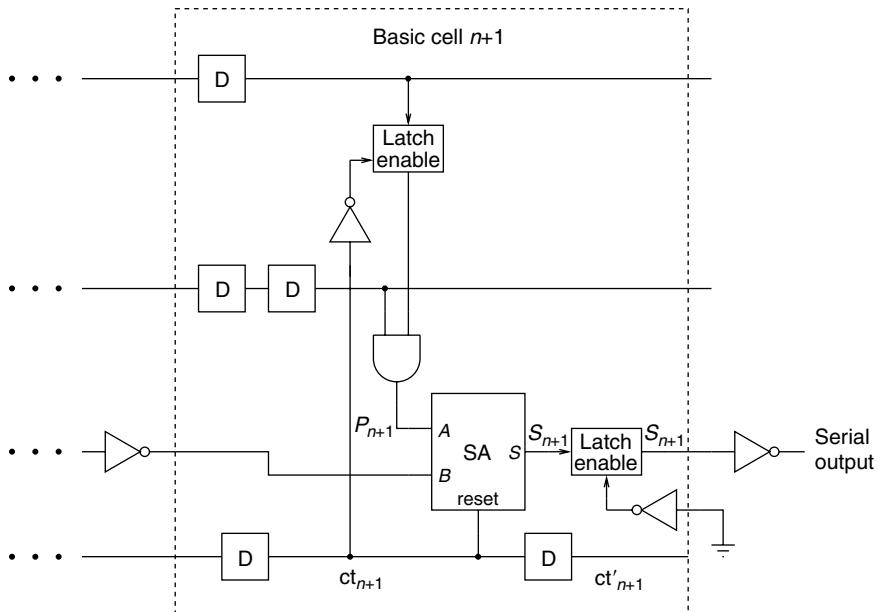


Fig. 11.6. Tail-end of the modified general multiplier for two's-complement arithmetic.

two inputs with $(n + 1)$ bits each is obtained by doubling the number of bits in each input by adding $(n + 1)$ bits 0 to the right of each number. Naturally, in this case, we need $(n + 1)$ more basic cells, and the complete operation takes twice as long to be implemented as basic single-precision multiplication.

For binary representations other than the two's complement, the implementation of the corresponding serial multipliers can be found, for instance, in Jackson *et al.* (1968), Freeny (1975), and Wanhammar (1981, 1999).

11.3.4 Parallel adder

Parallel adders can be easily constructed by interconnecting several full adders as shown in Figure 11.7, where we can observe that the carry signal in each cell propagates to the next cell through the overall adder structure. The main time-consuming operation in this realization is the propagation of the carry signal that must go through all the full adders to form the desired sum. This problem can be reduced, as proposed in Lyon (1976), at the cost of an increase in the hardware complexity.

11.3.5 Parallel multiplier

A parallel multiplier is usually implemented as a matrix of basic cells (Freeny, 1975; Rabiner & Gold, 1975), in which the internal data propagates horizontally, vertically, and diagonally in an efficient and ordered way. In general, such multipliers occupy a very large area of silicon and consume significant power. For these reasons they are only used in cases where time is a major factor in the overall performance of the given digital processing system, as in major DSP chips of today.

Since a digital filter is basically composed of delays, adders, and multipliers (using serial or parallel arithmetic), we now have all the tools necessary to implement a digital filter. The implementation is carried out by properly interconnecting such elements according to a certain realization, which should be chosen from among the ones seen in the previous chapters. If, in the resulting design, the sampling rate multiplied by the number of bits

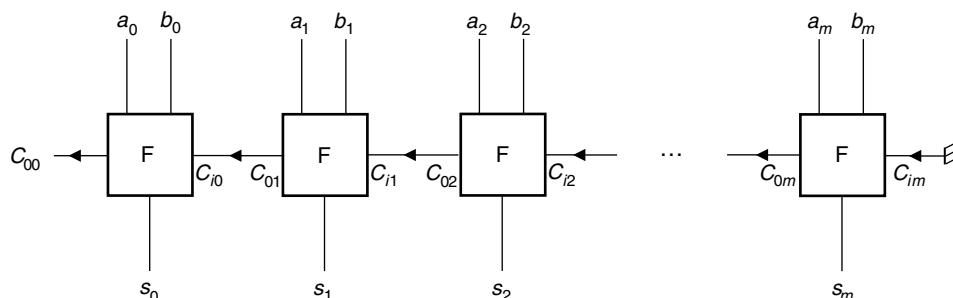


Fig. 11.7. Block diagram of the parallel adder.

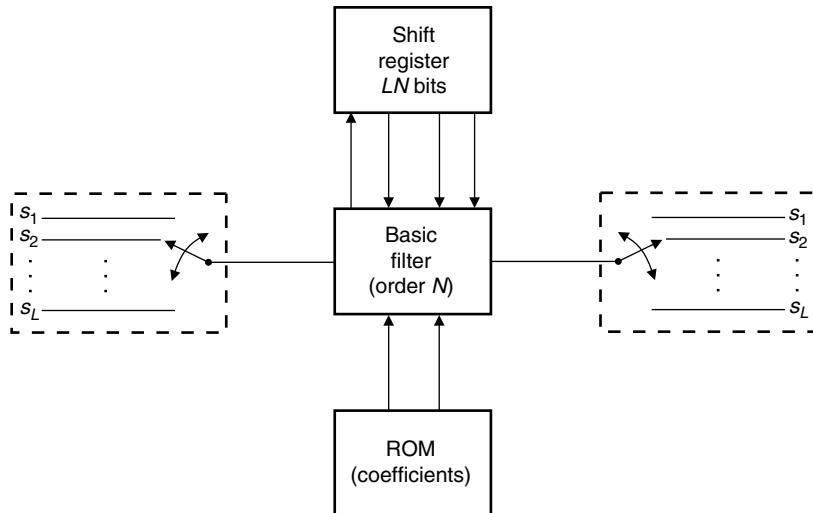


Fig. 11.8. Fully multiplexed realization of digital filters.

used to represent the signals is below the reachable processing speed, then multiplexing techniques can be incorporated to optimize the use of hardware resources, as described in Jackson *et al.* (1968). There are two main multiplexing approaches. In the first one, a single filter processes several input signals that are appropriately multiplexed. In the second one, the filter coefficients are multiplexed, resulting in several distinct transfer functions. It is always possible to combine both approaches, if desired, as suggested in Figure 11.8, where a memory element stores several sets of coefficients corresponding to different digital filters to be multiplexed.

11.4 Distributed arithmetic implementation

An alternative approach for implementing digital filters, the so-called distributed arithmetic (Peled & Liu, 1974, 1985; Wanhammar, 1981), avoids the explicit implementation of the multiplier element. Such a concept first appeared in the open literature in 1974 (Peled & Liu, 1974), although a patent describing it had been issued in December 1973 (Croisier *et al.*, 1973). The basic idea behind the concept of distributed arithmetic is to perform the summation of the products between filter coefficients and internal signals without using multipliers.

For example, suppose one wishes to calculate the following expression:

$$y = \sum_{i=1}^N c_i X_i, \quad (11.40)$$

where c_i are the filter coefficients and X_i are a set of signals represented in two's complement with $(b+1)$ bits. Assuming that the X_i are properly scaled such that $|X_i| < 1$, we can rewrite Equation (11.40) as

$$y = \sum_{i=1}^N c_i \left(\sum_{j=1}^b x_{ij} 2^{-j} - x_{i0} \right), \quad (11.41)$$

where x_{ij} corresponds to the j th bit of x_i and x_{i0} to its sign. By reversing the summation order in this equation, the following relationship applies:

$$y = \sum_{j=1}^b \left(\sum_{i=1}^N c_i x_{ij} \right) 2^{-j} - \sum_{i=1}^N c_i x_{i0}. \quad (11.42)$$

If we define the function $s(\cdot)$ of N binary variables z_1, z_2, \dots, z_N as

$$s(z_1, z_2, \dots, z_N) = \sum_{i=1}^N c_i z_i, \quad (11.43)$$

then Equation (11.42) can be written as

$$y = \sum_{j=1}^b s(x_{1j}, x_{2j}, \dots, x_{Nj}) 2^{-j} - s(x_{10}, x_{20}, \dots, x_{N0}). \quad (11.44)$$

Using the notation $s_j = s(x_{1j}, x_{2j}, \dots, x_{Nj})$, the above equation can then be written as

$$y = \{ \dots [(s_b 2^{-1} + s_{b-1}) 2^{-1} + s_{b-2}] 2^{-1} + \dots + s_1 \} 2^{-1} - s_0. \quad (11.45)$$

The value of s_j depends on the j th bit of all signals that determine y . Equation (11.45) gives a methodology to compute y : we first compute s_b , then divide the result by two with a right-shift operation, add the result to s_{b-1} , then divide the result by two, add it to s_{b-2} , and so on. In the last step, s_0 must be subtracted from the last partial result.

Overall, the function $s(\cdot)$ in Equation (11.43) can assume at most 2^N possible distinct values, since all of its N variables are binary. Thus, an efficient way to compute $s(\cdot)$ is to predetermine all its possible values, which depend on the values of the coefficients c_i , and store them in a memory unit whose addressing logic must be based on the synchronized data content.

The distributed arithmetic implementation of Equation (11.40) is as shown in Figure 11.9. In this implementation, the words X_1, X_2, \dots, X_N are stored in shift-registers SR_1, SR_2, \dots, SR_N respectively, each one with $(b+1)$ bits. The N outputs of the shift-registers are used to address a ROM unit. Then, once the words are loaded in the shift-registers, after the j th right-shift, the ROM will be addressed with $x_{1,b-j} x_{2,b-j} \dots x_{N,b-j}$ and the ROM will output s_{b-j} . This value is then loaded into register A to be added to the partial result from another register, B , which stores the previous accumulated value. The result is divided by two (see Equation (11.45)). For each calculation, register A is initialized with s_b and

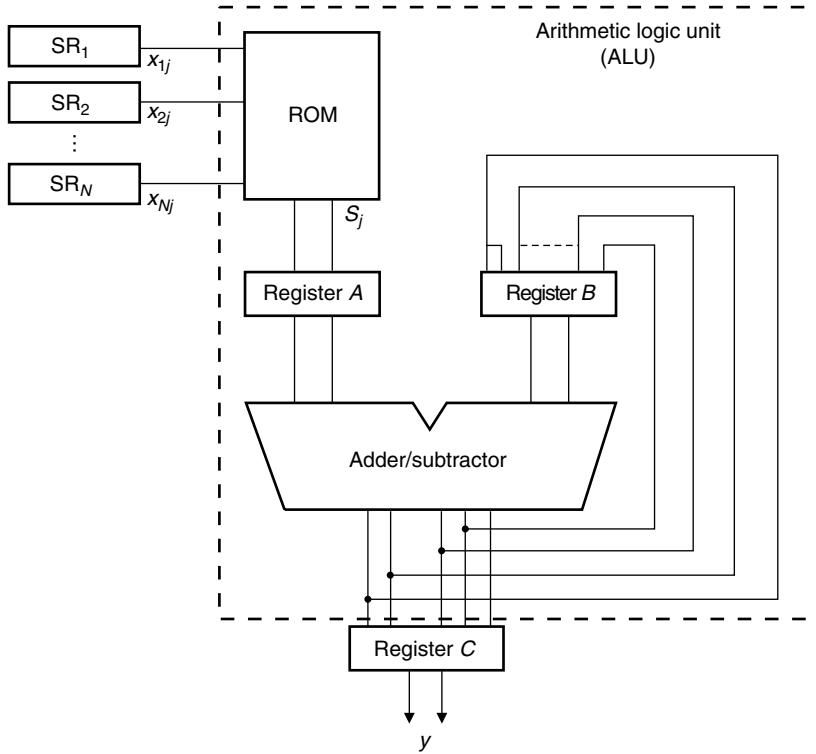


Fig. 11.9. Basic architecture for implementing Equation (11.40) using distributed arithmetic.

register B is reset to contain only zeros. Register C is used to store the final sum, obtained from the subtraction of s_0 from the next-to-last sum. Naturally, all the above calculations could have been implemented without register A , whose importance lies in the possibility of accessing the memory unit simultaneously to the adder/subtractor operation, thus forming a pipelined architecture.

Using the implementation illustrated in Figure 11.9, the complete time interval for computing one sample of the output y entails $(b + 1)$ clock cycles and, therefore, depends solely on the number of bits of each word and is not a function of the number N of partial products involved to form y . The duration of the clock cycle, in this case, is determined by the maximum value between the times of a memory access and a computation of an addition or subtraction operation. The number of bits b used to represent each word greatly affects the memory size, along with the number N of partial sums, which should not be made too large in order to limit the memory access time. The value of b depends basically on the dynamic range necessary to represent $|s_j|$ and the precision required to represent the coefficients c_i to avoid large coefficient quantization effects.

We can greatly increase the overall processing speed for computing y , as in Equation (11.40), by reading the desired values of s_j in parallel and using several adders in parallel to determine all necessary partial sums, as described in Peled & Liu (1974).

This distributed arithmetic can be used to implement several of the digital filter structures presented in this book. For instance, the second-order direct-form realization implementing the transfer function

$$H(z) = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}, \quad (11.46)$$

which corresponds to the difference equation

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2), \quad (11.47)$$

can be mapped onto the implementation seen in Figure 11.10, where the present and past values of the input signal are used in conjunction with the past values of the output signal to address the memory unit. In other words, we make $X_i = x(n-i)$, for $i = 0, 1, 2$, $X_3 = y(n-1)$, and $X_4 = y(n-2)$. The function $s(\cdot)$, which determines the content of the memory unit, is then given by

$$s(z_1, z_2, z_3, z_4, z_5) = b_0 z_1 + b_1 z_2 + b_2 z_3 - a_1 z_4 - a_2 z_5. \quad (11.48)$$

Therefore, at a given instant, the quantity s_j is

$$s_j = b_0 x_j(n) + b_1 x_j(n-1) + b_2 x_j(n-2) - a_1 y_j(n-1) - a_2 y_j(n-2). \quad (11.49)$$

As the number of partial sums required to calculate y is $N = 5$, the memory unit in this example should have 32 positions of L bits, where L is the number of bits established to represent s_j .

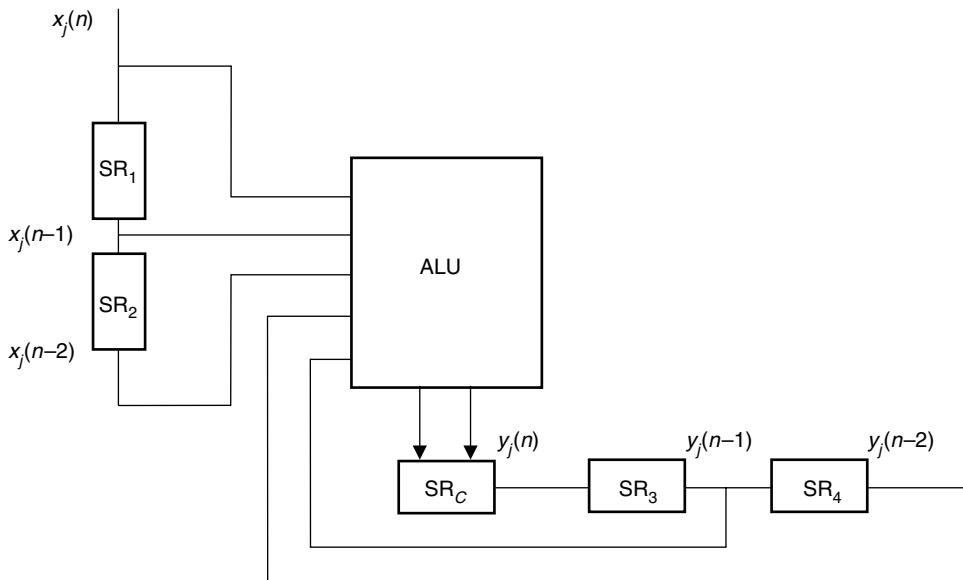


Fig. 11.10. Second-order direct-form filter implemented with distributed arithmetic.

In general, all coefficients b_i and a_i in Equation (11.48) should be already quantized. This is because it is easier to predict the quantization effects at the stage of filter design (following the theory presented later in this chapter) than to analyze such effects after quantizing s_j as given in Equation (11.48). In fact, performing the quantization on s_j can even introduce nonlinear oscillations at the output of a given structure which was initially shown to be immune to limit cycles. Limit cycles are dealt with in Section 11.8, and such a quantization effect is illustrated in Example 11.13.

For the second-order state-variable realization as given in Figure 13.5, the distributed arithmetic implementation is shown in Figure 11.11. In this case, the contents of the memory units are generated by

$$s_{1j} = a_{11}x_{1j}(n) + a_{12}x_{2j}(n) + b_1x_j(n); \text{ for the ROM of the ALU}_1 \quad (11.50)$$

$$s_{2j} = a_{22}x_{2j}(n) + a_{21}x_{1j}(n) + b_2x_j(n); \text{ for the ROM of the ALU}_2 \quad (11.51)$$

$$s_{3j} = c_1x_{1j}(n) + c_2x_{2j}(n) + d_j(n); \text{ for the ROM of the ALU}_3. \quad (11.52)$$

Each memory unit has $8 \times L$ words, where L is the established wordlength for the given s_{ij} .

The implementation of high-order filters using distributed arithmetic is simple when the parallel and cascade forms seen in Chapter 4 are used. Both forms can be implemented

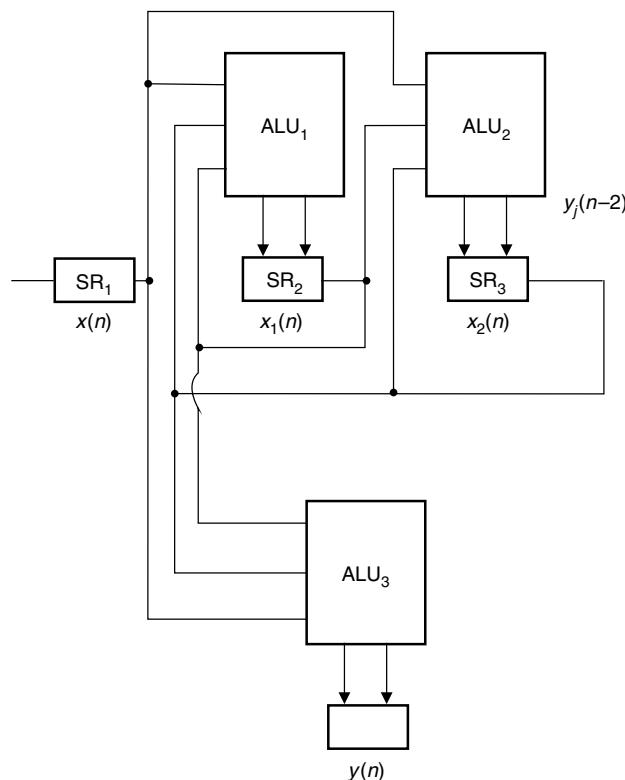


Fig. 11.11. Second-order state-space filter implemented with distributed arithmetic.

using a single block realization, whose coefficients are multiplexed in time to perform the computation of a specific second-order block. The cascade version of this type of implementation is presented in Figure 11.12. Such an approach is very efficient in terms of chip area and power consumption. Faster versions, however, are possible using both the parallel and cascade forms. For instance, the fast parallel approach is depicted in Figure 11.13, where all

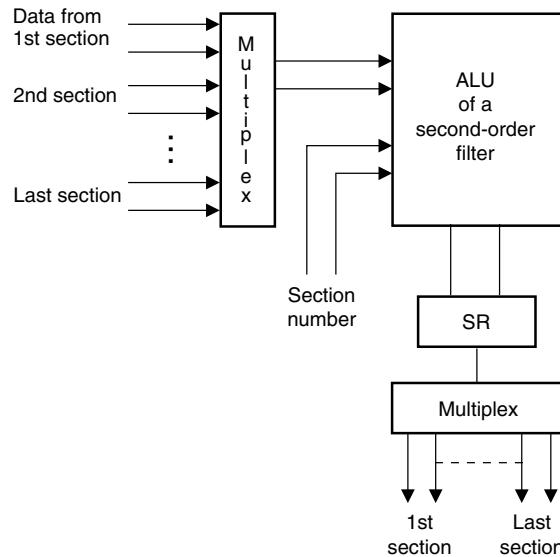


Fig. 11.12. Implementation of the cascade form using distributed arithmetic.

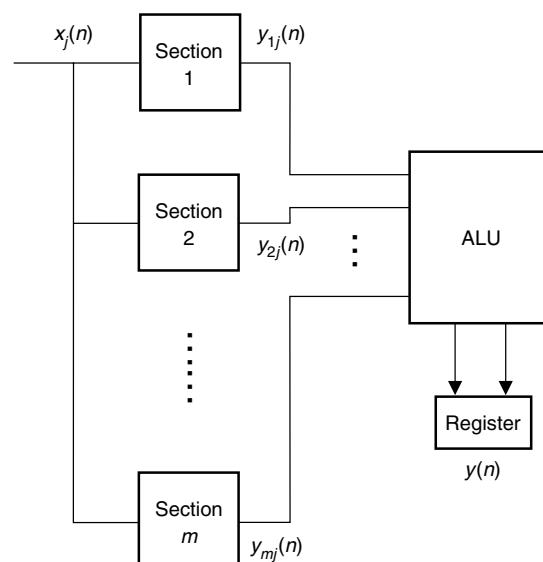


Fig. 11.13. Fast implementation of the parallel form using distributed arithmetic.

numerical calculations can be performed in parallel. The fast cascade form can be made more efficient if delay elements are added in between the blocks to allow pipelined processing.

The distributed arithmetic technique presented in this section can also be used to implement other digital filter structures (Wanhammar, 1981), as well as some specific processors for the computation of the FFT (Liu & Peled, 1975).

11.5 Product quantization

A finite-wordlength multiplier can be modeled in terms of an ideal multiplier followed by a single additive noise source $e(n)$, as shown in Figure 11.14. Three distinct approximation schemes can be employed after a multiplication, namely rounding, truncation, and magnitude truncation. We analyze their effects in numbers represented in two's complement.

Product quantization by rounding leads to a result in finite precision whose value is the closest possible to the actual value. If we assume that the dynamic range throughout the digital filter is much larger than the value of the least significant bit $q = 2^{-b}$ (b corresponds to n in Equations (11.1)–(11.11)), the probability density function of the quantization error is depicted in Figure 11.15a. The mean or expected value of the quantization error due to rounding is zero:

$$E\{e(n)\} = 0. \quad (11.53)$$

Also, it is easy to show that the variance of the noise $e(n)$ is given by

$$\sigma_e^2 = \frac{q^2}{12} = \frac{2^{-2b}}{12}. \quad (11.54)$$

In the case of truncation of a number, where the result is always less than the original value, the probability density function is as shown in Figure 11.15b, the expected value for the quantization error is

$$E\{e(n)\} = -\frac{q}{2}, \quad (11.55)$$

and the error variance amounts to

$$\sigma_e^2 = \frac{2^{-2b}}{12}. \quad (11.56)$$

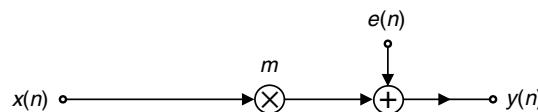


Fig. 11.14. Noise model for multiplier.

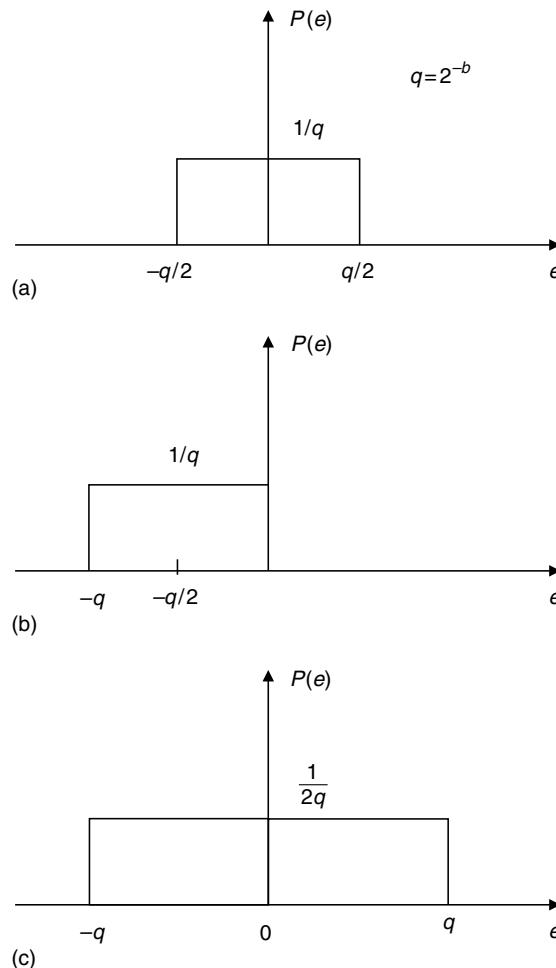


Fig. 11.15. Probability density functions for the product-quantization error: (a) rounding; (b) truncation; (c) magnitude truncation.

This type of quantization is not adequate, in general, because the errors associated with a nonzero mean value, although small, tend to propagate through the filter, and their effect can be sensed at the filter output.

If we apply magnitude truncation, which necessarily implies reducing the magnitude of the number, then the probability density function has the form depicted in Figure 11.15c. In this case, the mean value of the quantization error is

$$E\{e(n)\} = 0 \quad (11.57)$$

and the variance is

$$\sigma_e^2 = \frac{2^{-2b}}{3}. \quad (11.58)$$

Owing to these results, it is easy to understand why rounding is the most attractive form of quantization, since it generates the smallest noise variance while maintaining the mean value of the quantization error equal to zero. Magnitude truncation, besides having a noise variance four times greater than rounding, leads to a higher correlation between the signal and quantization noise (for example, when the signal is positive/negative, the quantization noise is also positive/negative), which is a strong disadvantage, as will soon be clear. However, the importance of magnitude truncation cannot be overlooked, since it can eliminate granular limit cycles in recursive digital filters, as will be shown later in this chapter.

At this point, it is interesting to study how the signal quantization affects the output signal. In order to simplify the analysis of roundoff-noise effects, the following assumptions are made regarding the filter internal signals (Jackson, 1969, 1970a,b):

- their amplitude is much larger than the quantization error;
- their amplitude is small enough that overflow never occurs;
- they have a wide spectral content.

These assumptions imply that:

- (i) The quantization errors at different instants are uncorrelated; that is, $e_i(n)$ is uncorrelated with $e_j(n+l)$, for $l \neq 0$.
- (ii) Errors at different nodes are uncorrelated; that is, $e_i(n)$ is uncorrelated with $e_j(n)$, for $i \neq j$.

From the above considerations, the contributions of different noise sources can be accounted for separately and added to determine the overall roundoff error at the filter output. However, one should note that (i) and (ii) do not hold when magnitude truncation is used, due to the inherent correlation between the signal and the quantization error. Therefore, one should bear in mind that, for the magnitude truncation scheme, the analysis that follows does not lead to accurate results.

Denoting the error variance due to each internal signal quantization by σ_e^2 and assuming that the quantization error is the outcome of a white noise process, the PSD of a given noise source is (Papoulis, 1965)

$$\Gamma_E(e^{j\omega}) = \sigma_e^2. \quad (11.59)$$

In Equation (2.252), we showed that for a linear system having transfer function $H(z)$, if a stationary signal with PSD $\Gamma_X(e^{j\omega})$ is input, then the PSD of the output is $\Gamma_Y(e^{j\omega}) = H(e^{j\omega})H(e^{-j\omega})\Gamma_X(e^{j\omega})$. Therefore, in a fixed-point digital-filter implementation, the PSD of the output noise $y(n)$ is given by

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \sum_{i=1}^K G_i(e^{j\omega})G_i(e^{-j\omega}), \quad (11.60)$$

where K is the number of multipliers of the filter and $G_i(e^{j\omega})$ is the frequency response associated with the transfer function $G_i(z)$ from each multiplier output $g_i(n)$ to the output of the filter, as indicated in Figure 11.16.

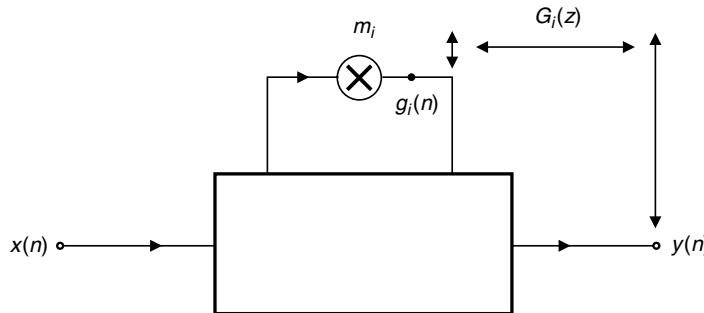


Fig. 11.16. Noise transfer function for a digital filter.

A common figure of merit for evaluating the performance of digital filters is the relative power spectral density (RPSD) of the output noise, defined in decibels as

$$\text{RPSD} = 10 \log \frac{\Gamma_Y(e^{j\omega})}{\Gamma_E(e^{j\omega})} = 10 \log \sum_{i=1}^K \left| G_i(e^{j\omega}) \right|^2. \quad (11.61)$$

This figure eliminates the dependence of the output noise on the filter wordlength, thus representing a true measure of the extent to which the output noise depends on the internal structure of the filter.

A simpler but useful performance criterion to evaluate the roundoff noise generated in digital filters is the noise gain or the relative noise variance, defined by

$$\begin{aligned}
 \frac{\sigma_y^2}{\sigma_e^2} &= \frac{1}{\pi} \int_0^\pi \sum_{i=1}^K \left(G_i(z) G_i(z^{-1}) \right)_{z=e^{j\omega}}^2 d\omega \\
 &= \frac{1}{\pi} \int_0^\pi \sum_{i=1}^K \left| G_i(e^{j\omega}) \right|^2 d\omega \\
 &= \frac{1}{\pi} \sum_{i=1}^K \int_0^\pi \left| G_i(e^{j\omega}) \right|^2 d\omega \\
 &= \sum_{i=1}^K \| G_i(e^{j\omega}) \|_2^2.
 \end{aligned} \quad (11.62)$$

Another noise source that must be accounted for in digital filters is the roundoff noise generated when the input-signal amplitude is quantized in the process of analog-to-digital conversion. Since the input-signal quantization is similar to product quantization, it can be represented by including a noise source at the input of the digital filter structure.

Example 11.3. Compute the quantization noise PSD at the output of the networks shown in Figure 11.17, assuming that fixed-point arithmetic is employed.

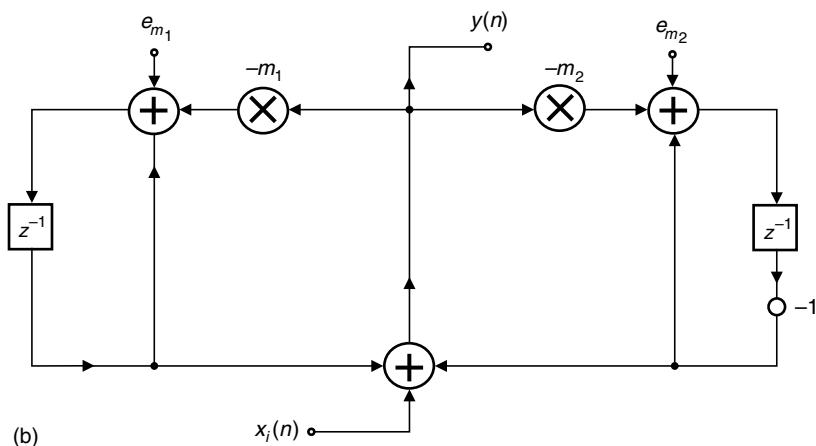
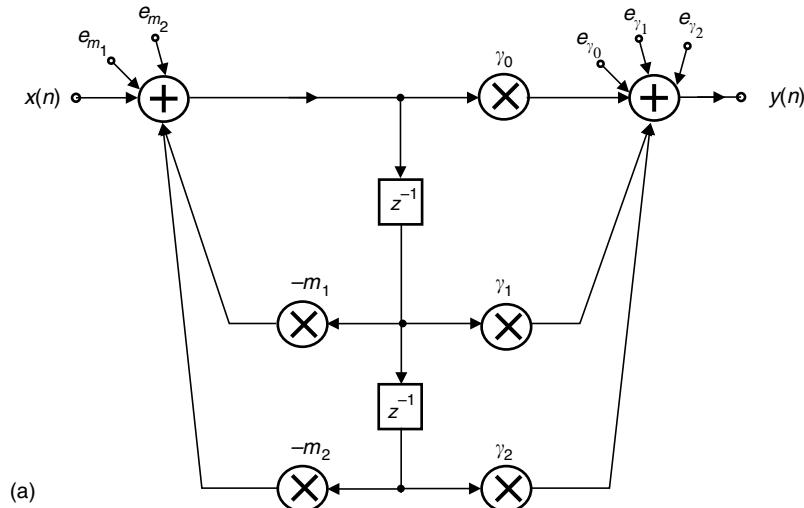


Fig. 11.17. Second-order networks.

Solution

- For the structure of Figure 11.17a, we have that

$$G_{m_1}(z) = G_{m_2}(z) = H(z) \quad (11.63)$$

$$G_{\gamma_0}(z) = G_{\gamma_1}(z) = G_{\gamma_2}(z) = 1, \quad (11.64)$$

where

$$H(z) = \frac{\gamma_0 z^2 + \gamma_1 z + \gamma_2}{z^2 + m_1 z + m_2}. \quad (11.65)$$

The output PSD is then

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(2H(e^{j\omega})H(e^{-j\omega}) + 3 \right). \quad (11.66)$$

- In the case of the structure shown in Figure 11.17b, the transfer functions from the multiplier outputs to the filter output can be readily computed from the results of Exercise 4.6, considering that for $-m_1$

$$\mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad d = 0 \quad (11.67)$$

and for $-m_2$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad d = 0. \quad (11.68)$$

The required transfer functions are given by

$$G_{m_1}(z) = \frac{z+1}{D(z)} \quad (11.69)$$

$$G_{m_2}(z) = \frac{-(z-1)}{D(z)}, \quad (11.70)$$

where

$$D(z) = z^2 + (m_1 - m_2)z + (m_1 + m_2 - 1). \quad (11.71)$$

The PSD at the output is then given by

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(G_{m_1}(e^{j\omega})G_{m_1}(e^{-j\omega}) + G_{m_2}(e^{j\omega})G_{m_2}(e^{-j\omega}) \right). \quad (11.72)$$

△

In floating-point arithmetic, quantization errors are introduced not only in products, but also in additions. In fact, the sum and product of two numbers x_1 and x_2 , in finite precision, have the following characteristics:

$$Fl\{x_1 + x_2\} = x_1 + x_2 - (x_1 + x_2)n_a \quad (11.73)$$

$$Fl\{x_1 x_2\} = x_1 x_2 - (x_1 x_2)n_p \quad (11.74)$$

where n_a and n_p are random variables with zero mean, which are independent of any other internal signals and errors in the filter. Their variances are approximately given by (Smith *et al.*, 1992)

$$\sigma_{n_a}^2 \approx 0.165 \times 2^{-2b} \quad (11.75)$$

and

$$\sigma_{n_p}^2 \approx 0.180 \times 2^{-2b} \quad (11.76)$$

respectively, where b is the number of bits in the mantissa representation, not including the sign bit. Using the above expressions, the roundoff-noise analysis for floating-point arithmetic can be done in the same way as for the fixed-point case (Bomar *et al.*, 1997).

A variant of floating-point arithmetic is the so-called block-floating-point arithmetic, which consists of representing several numbers with a shared exponent term. Block floating point is a compromise between the high-complexity hardware of floating-point arithmetic and the short dynamic range inherent to the fixed-point representation. A roundoff-error analysis of block-floating-point systems is available in Kalliojärvi and Astola (1996) and Ralev and Bauer (1999).

11.6 Signal scaling

It is possible for overflow to occur in any internal node of a digital filter structure. In general, input scaling is often required to reduce the probability of overflow occurring to an acceptable level. Particularly in fixed-point implementations, signal scaling should be applied to make the probability of overflow the same at all internal nodes of a digital filter. In such cases, the signal-to-noise ratio is maximized.

In a practical digital filter, however, a few internal signals are critical, and they should not exceed the allowed dynamic range for more than some very short periods of time. For these signals, if overflow frequently occurs, serious signal distortions will be observed at the filter output.

If either the one's- or the two's-complement representation is used, then an important fact greatly simplifies the scaling in order to avoid overflow distortions: if the sum of two or more numbers is within the available range of representable numbers, then the complete sum will always be correct irrespective of the order in which they are added, even if overflow occurs in a partial operation (Antoniou, 1993). This implies that the overflow distortions which are due to signal additions can be recovered by subsequent additions. As a consequence, when using either one's- or two's-complement arithmetic, one only needs to avoid overflow at the multiplier inputs. Therefore, they are the only points that require scaling.

In this case, then, in order to avoid frequent overflows, we should calculate the upper limit for the magnitude of each signal $x_i(n)$, for all possible types of filter inputs $u(n)$. This is shown by the analysis of the decimation-in-time FFT algorithm in the example below.

Example 11.4. Perform a roundoff-noise analysis of FFT algorithms.

Solution

Each distinct FFT algorithm requires a specific analysis of the corresponding quantization effects. In this example, we perform a roundoff-noise analysis on the radix-2 FFT algorithm with decimation in time.

The basic cell of the radix-2 FFT algorithm based on the decimation-in-time method is shown in Figure 3.10 and repeated for convenience as Figure 11.18. From Figure 11.18, we have that

$$|X_i(k)| \leq 2 \max\{|X_{ie}(k)|, |X_{io}(k)|\} \quad (11.77)$$

$$\left| X_i \left(k + \frac{L}{2} \right) \right| \leq 2 \max\{|X_{ie}(k)|, |X_{io}(k)|\} \quad (11.78)$$

Therefore, to avoid overflow in such a structure, using fixed-point arithmetic, a factor of $\frac{1}{2}$ on each cell input should be employed, as seen in Figure 11.19. There, one can clearly discern the two noise sources that result from both signal scaling and multiplication by W_L^k .

In the common case of rounding, the noise variance, as given in Equation (11.54), is equal to

$$\sigma_e^2 = \frac{2^{-2b}}{12} \quad (11.79)$$

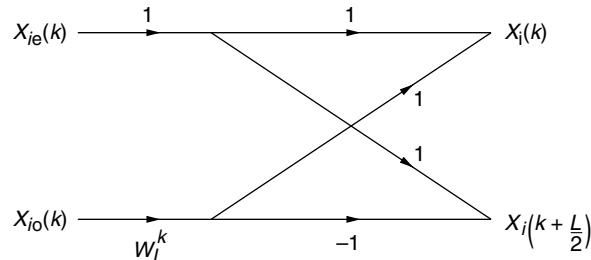


Fig. 11.18. Basic cell of the radix-2 FFT algorithm using decimation in time.

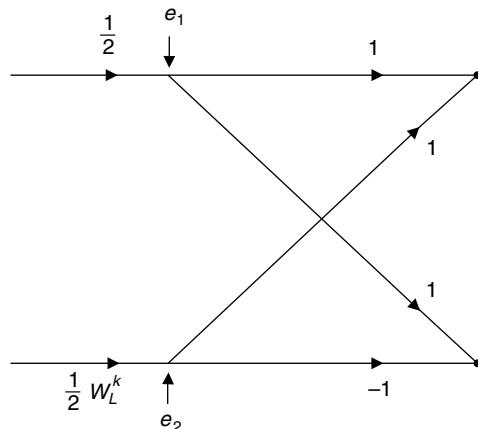


Fig. 11.19. Basic cell with input scaling.

where b is the number of fixed-point bits. In Figure 11.19, the noise source e_1 models the scaling noise on a cell input which is a complex number. By considering the real and imaginary noise contributions to be uncorrelated, we have that

$$\sigma_{e_1}^2 = 2\sigma_e^2 = \frac{2^{-2b}}{6} \quad (11.80)$$

Meanwhile, the noise source e_2 models the scaling noise due to the multiplication of two complex numbers, which involves four different terms. By considering these four terms to be uncorrelated, we have that

$$\sigma_{e_2}^2 = 4\sigma_e^2 = \frac{2^{-2b}}{3} \quad (11.81)$$

This completes the noise analysis in a single basic cell.

To extend these results to the overall FFT algorithm, let us assume that all noise sources in the algorithm are uncorrelated. Therefore, the output-noise variance can be determined by the addition of all individual noise variances, from all the basic cells involved.

Naturally, the noise generated in the first stage appears at the output scaled by $(\frac{1}{2})^{l-1}$, where l is the overall number of stages, and the noise generated at stage k is multiplied by $(\frac{1}{2})^{l-k}$. To determine the total number of cells, we note that each FFT output is directly connected to one basic cell in the last stage, two in the next-to-last stage, and so on, up to 2^{l-1} cells in the first stage, with each cell presenting two noise sources similar to e_1 and e_2 , as discussed above.

Each stage then has 2^{l-k} cells, and their output-noise contribution is given by

$$2^{l-k} (1/2)^{2l-2k} (\sigma_{e_1}^2 + \sigma_{e_2}^2) \quad (11.82)$$

Consequently, the overall noise variance in each FFT output sample is given by

$$\sigma_o^2 = (\sigma_{e_1}^2 + \sigma_{e_2}^2) \sum_{k=1}^l 2^{l-k} \left(\frac{1}{2}\right)^{2l-2k} = 6\sigma_e^2 \sum_{k=1}^l \left(\frac{1}{2}\right)^{l-k} = 6\sigma_e^2 \left(2 - \frac{1}{2^{l-1}}\right). \quad (11.83)$$

A similar analysis for other FFT algorithms can be determined in an analogous way. For a roundoff analysis on the DCT, readers should refer to Yun and Lee (1995), for instance.

△

The general case of scaling analysis is illustrated in Figure 11.20, where $F_i(z)$ and $F'_i(z)$ respectively represent the transfer functions before and after scaling from the input node to the input of the multiplier m_i , such that

$$F'_i(z) = \lambda F_i(z), \quad (11.84)$$

which also holds for the corresponding impulse responses; that is:

$$f'_i(n) = \lambda f_i(n) \quad (11.85)$$

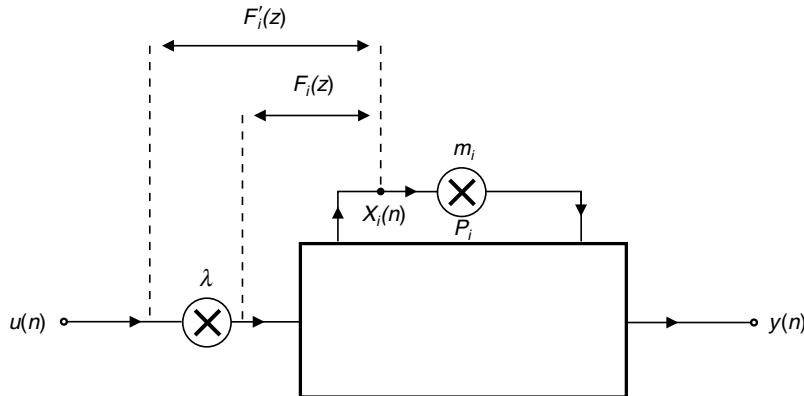


Fig. 11.20. Signal scaling.

for all n . Assuming zero initial conditions, we have that

$$x_i(n) = \sum_{k=0}^{\infty} f'_i(k)u(n-k) = \lambda \sum_{k=0}^{\infty} f_i(k)u(n-k). \quad (11.86)$$

If $u(n)$ is bounded in magnitude by u_m , for all n , then the above equation implies that

$$|x_i(n)| \leq u_m \sum_{k=0}^{\infty} |f'_i(k)| = u_m \lambda \sum_{k=0}^{\infty} |f_i(k)|. \quad (11.87)$$

If we want the magnitude of the signal $x_i(n)$ to also be upper bounded by u_m , for all types of input sequences, then the associated scaling must ensure that

$$\sum_{k=0}^{\infty} |f'_i(k)| \leq 1 \quad (11.88)$$

and therefore

$$\lambda \leq \frac{1}{\sum_{k=0}^{\infty} |f_i(k)|}. \quad (11.89)$$

This is a necessary and sufficient condition to avoid overflow for any input signal. However, the condition in Equations (11.88) and (11.89) is not useful in practice, as it cannot be easily implemented. In addition, for a large class of input signals, it leads to a very stringent scaling. A more practical scaling strategy, aimed at more specific classes of input signals, is presented in the following. Since

$$U(z) = \sum_{n=-\infty}^{\infty} u(n)z^{-n} \quad (11.90)$$

and

$$X_i(z) = F'_i(z)U(z) = \lambda F_i(z)U(z), \quad (11.91)$$

then, in the time domain, $x_i(n)$ is given by

$$x_i(n) = \frac{1}{2\pi j} \oint_C X_i(z)z^{n-1} dz, \quad (11.92)$$

where C is in the convergence region common to $F_i(z)$ and $U(z)$. Accordingly, in the frequency domain, $x_i(n)$ is given by

$$x_i(n) = \frac{\lambda}{2\pi} \int_0^{2\pi} F_i(e^{j\omega})U(e^{j\omega})e^{j\omega n} d\omega. \quad (11.93)$$

Let, now, the L_p norm be defined for any periodic function $F(e^{j\omega})$ as follows:

$$\|F(e^{j\omega})\|_p = \left(\frac{1}{2\pi} \int_0^{2\pi} |F(e^{j\omega})|^p d\omega \right)^{1/p} \quad (11.94)$$

for all $p \geq 1$, such that $\int_0^{2\pi} |F(e^{j\omega})|^p d\omega \leq \infty$.

If $F(e^{j\omega})$ is continuous, then the limit when $p \rightarrow \infty$ of Equation (11.94) exists and is given by

$$\|F(e^{j\omega})\|_\infty = \max_{0 \leq \omega \leq 2\pi} \{|F(e^{j\omega})|\}. \quad (11.95)$$

Assuming that $|U(e^{j\omega})|$ is upper bounded by U_m (that is, $\|U(e^{j\omega})\|_\infty \leq U_m$), then it clearly follows from Equation (11.93) that

$$|x_i(n)| \leq \frac{U_m \lambda}{2\pi} \int_0^{2\pi} |F_i(e^{j\omega})| d\omega; \quad (11.96)$$

that is:

$$|x_i(n)| \leq \lambda \|F_i(e^{j\omega})\|_1 \|U(e^{j\omega})\|_\infty. \quad (11.97)$$

Following a similar reasoning, we have that

$$|x_i(n)| \leq \lambda \|F_i(e^{j\omega})\|_\infty \|U(e^{j\omega})\|_1. \quad (11.98)$$

Also, from the Schwartz inequality (Jackson, 1969)

$$|x_i(n)| \leq \lambda \|F_i(e^{j\omega})\|_2 \|U(e^{j\omega})\|_2. \quad (11.99)$$

Equations (11.97)–(11.99) are special cases of a more general relation, known as the Hölder inequality (Jackson, 1969; Hwang, 1979). This states that, if $(1/p) + (1/q) = 1$, then

$$|x_i(n)| \leq \lambda \|F_i(e^{j\omega})\|_p \|U(e^{j\omega})\|_q. \quad (11.100)$$

If $|u(n)| \leq u_m$, for all n , and if its Fourier transform exists, then there is U_m such that $\|U(e^{j\omega})\|_q \leq U_m$, for any $q \geq 1$. If we then want $|x_i(n)|$ to be upper limited by U_m , for all n , Equation (11.100) indicates that a proper scaling factor should be such that

$$\lambda \leq \frac{1}{\|F_i(e^{j\omega})\|_p}. \quad (11.101)$$

In practice, when the input signal is deterministic, the most common procedures for determining λ are:

- When $U(e^{j\omega})$ is bounded and, therefore, $\|U(e^{j\omega})\|_\infty$ can be precisely determined, one may use λ as in Equation (11.101), with $p = 1$.
- When the input signal has finite energy, namely

$$E = \sum_{n=-\infty}^{\infty} u^2(n) = \|U(e^{j\omega})\|_2^2 < \infty, \quad (11.102)$$

then λ can be obtained from Equation (11.101) with $p = 2$.

- If the input signal has a dominant frequency component, such as a sinusoidal signal, this means that it has an impulse in the frequency domain. In this case, neither $\|U(e^{j\omega})\|_\infty$ nor $\|U(e^{j\omega})\|_2$ are defined, and thus only the L_1 norm can be used. Then the scaling factor comes from Equation (11.101) with $p = \infty$, which is the most strict case for λ .

For the random-input case, the above analysis does not apply directly, since the z transform of $u(n)$ is not defined. In this case, if $u(n)$ is stationary, then the PSD of an internal signal $x_i(n)$ is given by

$$\Gamma_{X_i}(e^{j\omega}) = F'_i(e^{j\omega})F'_i(e^{-j\omega})\Gamma_U(e^{j\omega}) = \lambda^2 F_i(e^{j\omega})F_i(e^{-j\omega})\Gamma_U(e^{j\omega}), \quad (11.103)$$

where $\Gamma_U(e^{j\omega})$ is the input signal PSD. Hence, the variance of the internal signal $x_i(n)$ is given by

$$\begin{aligned} \sigma_{x_i}^2 &= R_{X_i}(0) \\ &= \frac{1}{2\pi} \int_0^{2\pi} \Gamma_{X_i}(e^{j\omega}) e^{j\omega v} d\omega \Big|_{v=0} \\ &= \frac{1}{2\pi} \int_0^{2\pi} \Gamma_{X_i}(e^{j\omega}) d\omega \\ &= \frac{1}{2\pi} \int_0^{2\pi} |F'_i(e^{j\omega})|^2 \Gamma_U(e^{j\omega}) d\omega \\ &= \frac{\lambda^2}{2\pi} \int_0^{2\pi} |F_i(e^{j\omega})|^2 \Gamma_U(e^{j\omega}) d\omega. \end{aligned} \quad (11.104)$$

Applying the Hölder inequality (Equation (11.100)) to the above equation, we have that, if $(1/p) + (1/q) = 1$, then

$$\sigma_{x_i}^2 \leq \lambda^2 \|F_i^2(e^{j\omega})\|_p \|\Gamma_U(e^{j\omega})\|_q \quad (11.105)$$

or, alternatively, that

$$\sigma_{x_i}^2 \leq \lambda^2 \|F_i(e^{j\omega})\|_{2p}^2 \|\Gamma_U(e^{j\omega})\|_q. \quad (11.106)$$

For random processes, the most interesting cases in practice are:

- If we consider $q = 1$, then $p = \infty$, and noting that $\sigma_u^2 = \|\Gamma_U(e^{j\omega})\|_1$, we have, from Equation (11.106), that

$$\sigma_{x_i}^2 \leq \lambda^2 \|F_i(e^{j\omega})\|_\infty^2 \sigma_u^2 \quad (11.107)$$

and an appropriate λ , such that $\sigma_{x_i}^2 \leq \sigma_u^2$, is

$$\lambda = \frac{1}{\|F_i(e^{j\omega})\|_\infty}. \quad (11.108)$$

- If the input signal is a white noise, $\Gamma_U(e^{j\omega}) = \sigma_u^2$, for all ω , and then, from Equation (11.103), we have

$$\sigma_{x_i}^2 = \lambda^2 \|F_i(e^{j\omega})\|_2^2 \sigma_u^2 \quad (11.109)$$

and an appropriate λ is

$$\lambda = \frac{1}{\|F_i(e^{j\omega})\|_2}. \quad (11.110)$$

In practical implementations, the use of powers of two to represent the scaling multiplier coefficients is a common procedure, as long as these coefficients satisfy the constraints to control overflow. In this manner, the scaling multipliers can be implemented using simple shift operations.

In the general case when we have m multipliers, the following single scaling can be used at the input:

$$\lambda = \frac{1}{\max \{\|F_1(e^{j\omega})\|_p, \|F_2(e^{j\omega})\|_p, \dots, \|F_m(e^{j\omega})\|_p\}}. \quad (11.111)$$

For cascade and parallel realizations, a scaling multiplier is employed at the input of each section. For some types of second-order sections, used in cascade realizations, the scaling factor of a given section can be incorporated into the output multipliers of the previous section. In general, this procedure leads to a reduction in the output quantization noise. In the case of second-order sections, it is possible to calculate the L_2 and L_∞ norms of the internal transfer functions in closed form (Diniz & Antoniou, 1986; Bomar & Joseph, 1987; Laakso, 1992), as given in Chapter 13.

Example 11.5. Scale the filter shown in Figure 11.21 using the L_2 norm, aiming at a possible implementation in a fixed-point machine, and determine the relative noise variance at the scaled filter output.

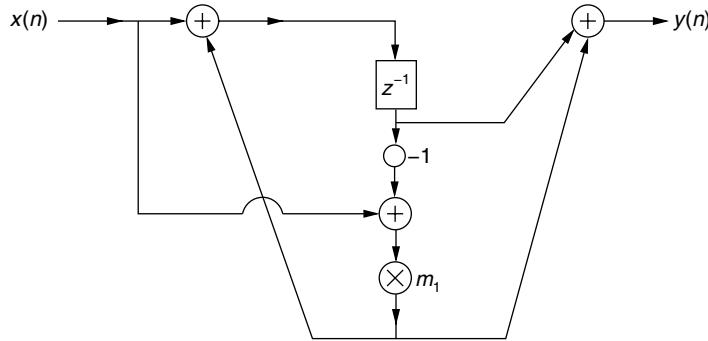


Fig. 11.21. Filter structure in Example 11.5.

Solution

Denoting the delay input as $s(n + 1)$, one can easily infer that

$$\left. \begin{array}{l} s(n + 1) = x(n) + m_1(x(n) - s(n)) \\ y(n) = s(n) + m_1(x(n) - s(n)) \end{array} \right\} \quad (11.112)$$

or equivalently

$$\left. \begin{array}{l} s(n + 1) = -m_1s(n) + (1 + m_1)x(n) \\ y(n) = (1 - m_1)s(n) + m_1x(n) \end{array} \right\}, \quad (11.113)$$

which corresponds to the state-space description characterized by

$$\mathbf{A} = -m_1; \quad \mathbf{B} = (1 + m_1); \quad \mathbf{C} = (1 - m_1); \quad d = m_1. \quad (11.114)$$

According to Equation (4.53), the transfer function for this example is then

$$H(z) = \frac{(1 - m_1)(1 + m_1)}{(z + m_1)} + m_1 = \frac{m_1 z + 1}{z + m_1}, \quad (11.115)$$

which happens to be an all-pass first-order transfer function such that

$$\|H(z)\|_2^2 = 1. \quad (11.116)$$

For scaling, the computation of the transfer function is required from the filter input to the input of the m_1 multiplier, obtained, in this case, by setting $\mathbf{C} = -1$ and $d = 1$, such that

$$F_1(z) = -\frac{1 + m_1}{z + m_1} + 1 = \frac{z - 1}{z + m_1}. \quad (11.117)$$

In order to determine the scaling factor, we need to compute

$$\|F_1(z)\|_2 = \sqrt{\frac{1}{2\pi} \int_0^{2\pi} |F_1(e^{j\omega})|^2 d\omega} = \sqrt{\frac{1}{2\pi j} \oint_C F_1(z) F_1(z^{-1}) z^{-1} dz}, \quad (11.118)$$

where the integral contour C is the z -domain unit circle. Therefore, using the residue theorem, one can write that

$$\|F_1(z)\|_2^2 = \sum_{\text{residues}} \left[F_1(z)F_1(z^{-1})z^{-1} \right], \quad (11.119)$$

where the residues are determined for all poles of $F_1(z)F_1(z^{-1})z^{-1}$ within C . In this example, assuming $|m_1| < 1$, we get

$$\begin{aligned} \|F_1(z)\|_2^2 &= \sum_{\text{residues}} \frac{(z-1)(1-z)}{(z+m_1)(1+zm_1)z} \\ &= \frac{(-m_1-1)(1+m_1)}{(1-m_1^2)(-m_1)} - \frac{1}{m_1} \\ &= \frac{2}{1-m_1}, \end{aligned} \quad (11.120)$$

leading to a scaling factor of

$$\lambda = \sqrt{\frac{1-m_1}{2}}, \quad (11.121)$$

which shall be compensated in the filter output by a gain $g = \sqrt{2/(1-m_1)}$.

The calculation of the output noise variance requires the transfer function from the multiplier output to the filter output, which, in this case, can be obtained by substituting $\mathbf{B} = d = 1$ in the state-space representation, such that

$$G_1(z) = \frac{1-m_1}{z+m_1} + 1 = \frac{z+1}{z+m_1}. \quad (11.122)$$

Again, by employing the residue theorem, we get

$$\begin{aligned} \|G_1(z)\|_2^2 &= \sum_{\text{residues}} \left[G_1(z)G_1(z^{-1})z^{-1} \right] \\ &= \sum_{\text{residues}} \frac{(1+z)^2}{(z+m_1)(1+zm_1)z} \\ &= \frac{(1-m_1)^2}{(1-m_1^2)(-m_1)} + \frac{1}{m_1} \\ &= \frac{2}{1+m_1}, \end{aligned} \quad (11.123)$$

such that

$$\begin{aligned}\frac{\sigma_y^2}{\sigma_e^2} &= \|H(z)\|_2^2 g^2 + \|G_1(z)\|_2^2 g^2 + 1 \\ &= \frac{2}{1-m_1} + \frac{2}{1+m_1} \frac{2}{1-m_1} + 1 \\ &= \frac{-m_1^2 + 2m_1 + 7}{m_1^2 - 1},\end{aligned}\tag{11.124}$$

already taking into account the scaling multiplier at the filter input and the compensating gain at the filter output. \triangle

11.7 Coefficient quantization

During the approximation step, the coefficients of a digital filter are calculated with the high accuracy inherent to the computer employed in the design. When these coefficients are quantized for practical implementations, commonly using rounding, the time and frequency responses of the realized digital filter deviate from the ideal response. In fact, the quantized filter may even fail to meet the prescribed specifications. The sensitivity of the filter response to errors in the coefficients is highly dependent on the type of structure. This fact is one of the motivations for considering alternative realizations having low sensitivity, such as those presented in Chapter 13.

Among the several sensitivity criteria that evaluate the effect of the fixed-point coefficient quantization on the digital filter transfer function, the most widely used are

$$I S_{m_i}^{H(z)}(z) = \frac{\partial H(z)}{\partial m_i}\tag{11.125}$$

$$II S_{m_i}^{H(z)}(z) = \frac{1}{H(z)} \frac{\partial H(z)}{\partial m_i}.\tag{11.126}$$

For the floating-point representation, the sensitivity criterion must take into account the relative variation of $H(z)$ due to a relative variation in a multiplier coefficient. We then must use

$$III S_{m_i}^{H(z)}(z) = \frac{m_i}{H(z)} \frac{\partial H(z)}{\partial m_i}.\tag{11.127}$$

With such a formulation, it is possible to use the value of the multiplier coefficient to determine $III S_m^{H(z)}(z)$. A simple example illustrating the importance of this fact is given by the quantization of the system

$$y(n) = (1+m)x(n), \quad \text{for } |m| \ll 1.\tag{11.128}$$

Using Equation (11.125), $IIS_{m_i}^{H(z)}(z) = 1$, regardless of the value of m , while using Equation (11.127), $III S_{m_i}^{H(z)}(z) = m/(m+1)$, indicating that a smaller value for the magnitude of m leads to a smaller sensitivity of $H(z)$ with respect to m . This is true for the floating-point representation, as long as the number of bits in the exponent is enough to represent the exponent of m .

Example 11.6. Determine the possible pole positions for a direct-form second-order section with denominator

$$D(z) = z^2 + a_1 z + a_2 \quad (11.129)$$

when the filter coefficients a_1 and a_2 are represented with 6 bits, including the sign bit, using standard binary representation.

Repeat your analysis using a state-space structure characterized by $a_{11} = a_{22} = a$ and $a_{21} = -a_{12} = \zeta$, when a and ζ are represented with 6 bits. Such a structure, when the values of at least two different coefficients are dependent on a single parameter, is referred to as a coupled-form state-space structure.

Solution

Figure 11.22a depicts the possible pole placements in the first quadrant within the z -domain unit circle for the direct-form second-order section. In the remaining quadrants, pole placements are symmetric mirrored copies of the ones seen in this figure. As can be observed, the pole grid becomes very sparse around the real axis, particularly close to $z = 0$ or $z = 1$. This explains the implementation inaccuracy achieved by this section type in applications with high sampling rate, since these cases often require a filter with poles close to the real axis. The same phenomenon occurs if the poles are required to be close to $z = 1$ or $z = -1$.

For the coupled form, the denominator polynomial becomes

$$D(z) = z^2 - 2az + a^2 + \zeta^2, \quad (11.130)$$

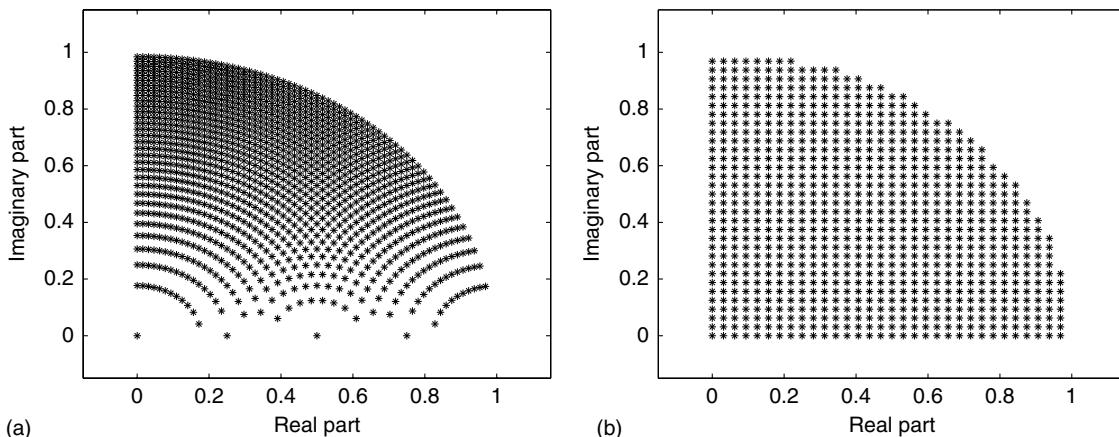


Fig. 11.22. Pole grid for second-order sections with 6-bit coefficients: (a) direct-form; (b) state-space coupled form.

where a represents the real part of the complex conjugate poles and ζ their imaginary part. The pole-placement analysis for the coefficient quantization in this structure is shown in Figure 11.22b, where we observe a uniform grid distribution around the entire quadrant. This result implies that there is no preferred region for this structure to place the poles, which is an attractive feature obtained at the cost of four multiplier coefficients to position a single pair of complex conjugate poles. \triangle

11.7.1 Deterministic sensitivity criterion

In practice, one is often interested in the variation of the magnitude of the transfer function $|H(e^{j\omega})|$ with coefficient quantization. Taking into account the contributions of all multipliers, a useful figure of merit related to this variation would be

$$S(e^{j\omega}) = \sum_{i=1}^K \left| S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right|, \quad (11.131)$$

where K is the total number of multipliers in the structure and $S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})$ is computed according to one of Equations (11.125)–(11.127), depending on the case.

However, in general, the sensitivities of $H(e^{j\omega})$ to coefficient quantization are much easier to derive than those of $|H(e^{j\omega})|$; thus, it would be convenient if the first one could be used as an estimate of the second. In order to investigate this possibility, we write the frequency response in terms of its magnitude and phase as

$$H(e^{j\omega}) = \left| H(e^{j\omega}) \right| e^{j\Theta(\omega)}. \quad (11.132)$$

Then the sensitivity measures, defined in Equations (11.125)–(11.127), can be written as

$$\left| {}_I S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right| = \sqrt{\left({}_I S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right)^2 + |H(e^{j\omega})|^2 \left(\frac{\partial \Theta(\omega)}{\partial m_i} \right)^2} \quad (11.133)$$

$$\left| {}_{II} S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right| = \sqrt{\left({}_{II} S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right)^2 + \left(\frac{\partial \Theta(\omega)}{\partial m_i} \right)^2} \quad (11.134)$$

$$\left| {}_{III} S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right| = \sqrt{\left({}_{III} S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega}) \right)^2 + |m_i|^2 \cdot \left(\frac{\partial \Theta(\omega)}{\partial m_i} \right)^2}. \quad (11.135)$$

From Equations (11.133)–(11.135), one can see that $|S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})| \geq |S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})|$. Thus, $|S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})|$ can be used as a conservative estimate of $|S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})|$, in the sense that it guarantees that the transfer function variation will be below a specified tolerance.

Moreover, it is known that low sensitivity is more critical when implementing filters with poles close to the unit circle, and, in such cases, for ω close to a pole frequency ω_0 , we can show that $|S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})| \approx |S_{m_i}^{|H(e^{j\omega})|}(e^{j\omega})|$. Therefore, we can rewrite Equation (11.131),

Table 11.1. Filter coefficients for the specifications (11.137).

Numerator coefficients	Denominator coefficients
$b_0 = 0.028\ 207\ 76$	$a_0 = 1.000\ 000\ 00$
$b_1 = -0.001\ 494\ 75$	$a_1 = -3.028\ 484\ 73$
$b_2 = 0.031\ 747\ 58$	$a_2 = 4.567\ 772\ 20$
$b_3 = 0.031\ 747\ 58$	$a_3 = -3.900\ 153\ 49$
$b_4 = -0.001\ 494\ 75$	$a_4 = 1.896\ 641\ 38$
$b_5 = 0.028\ 207\ 76$	$a_5 = -0.418\ 854\ 19$

yielding the following practical sensitivity figure of merit:

$$S(e^{j\omega}) = \sum_{i=1}^K \left| S_{m_i}^{H(e^{j\omega})}(e^{j\omega}) \right|, \quad (11.136)$$

where, depending on the case, $S_{m_i}^{H(e^{j\omega})}(e^{j\omega})$ is given by one of Equations (11.125)–(11.127).

Example 11.7. Design a lowpass elliptic filter with the following specifications:

$$\left. \begin{array}{l} A_p = 1.0 \text{ dB} \\ A_r = 40 \text{ dB} \\ \omega_p = 0.3\pi \text{ rad/sample} \\ \omega_r = 0.4\pi \text{ rad/sample} \end{array} \right\}. \quad (11.137)$$

Perform the fixed-point sensitivity analysis for the direct-order structure, determining the variation on the ideal magnitude response for an 11-bit quantization of the fractional part, including the sign bit.

Solution

The coefficients of the lowpass elliptic filter are given in Table 11.1.

For the general direct-form structure described by

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \quad (11.138)$$

it is easy to find that the sensitivities as defined in Equation (11.125) with respect to the numerator and denominator coefficients are given by

$$IS_{b_i}^{H(z)}(z) = \frac{z^{-i}}{A(z)} \quad (11.139)$$

and

$$IS_{a_i}^{H(z)}(z) = -\frac{z^{-i} H(z)}{A(z)} \quad (11.140)$$

respectively. The magnitude of these functions for the designed fifth-order elliptic filter are seen in Figure 11.23.

The figure of merit $S(e^{j\omega})$, as given in Equation (11.136), for the general direct-form realization can be written as

$$S(e^{j\omega}) = \frac{(N+1) + N|H(z)|}{|A(z)|}. \quad (11.141)$$

For this example, the function is depicted in Figure 11.24a.

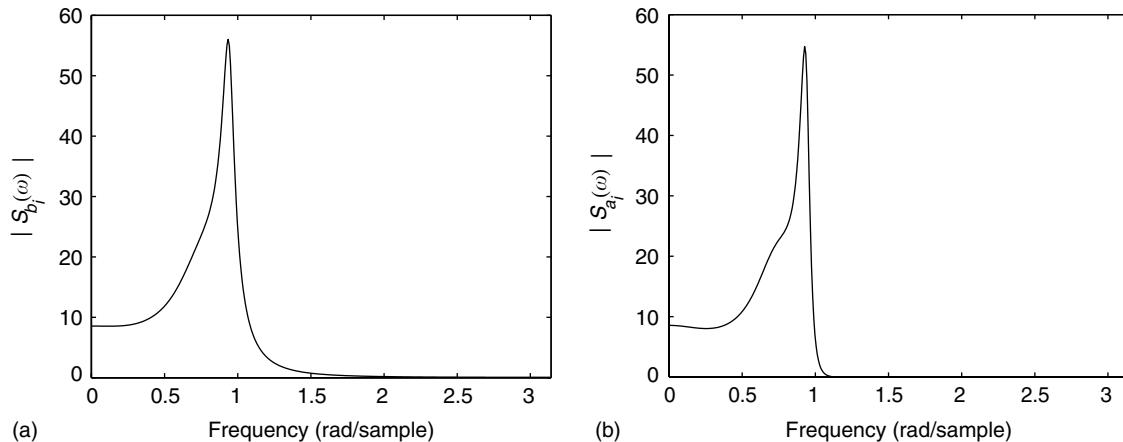


Fig. 11.23. Magnitudes of the sensitivity functions of $H(z)$ with respect to: (a) numerator coefficients b_i ; (b) denominator coefficients a_i .

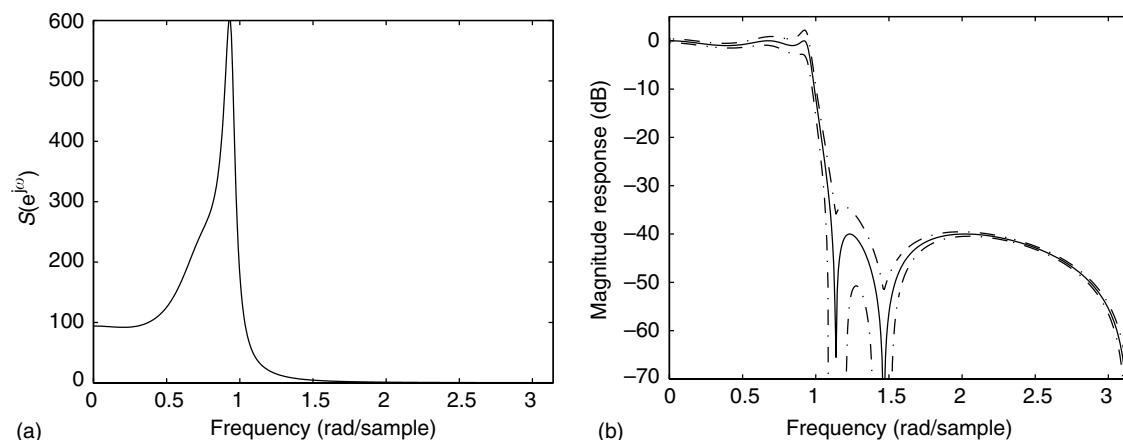


Fig. 11.24. Finite-precision analysis: (a) sensitivity measurement $S(e^{j\omega})$; (b) worst-case variation of $|H(e^{j\omega})|$ with an 11-bit fixed-point quantization.

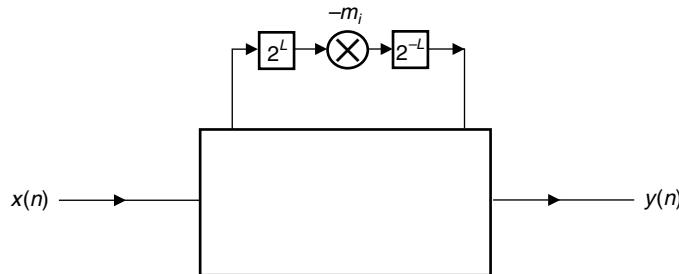


Fig. 11.25. Implementation of a multiplication in a pseudo-floating-point representation.

We can then estimate the variation of the ideal magnitude response using the approximation

$$\Delta|H(e^{j\omega})| \approx \Delta m_i S(e^{j\omega}). \quad (11.142)$$

For a fixed-point 11-bit rounding quantization, including the sign bit, $\max\{\Delta m_i\} = 2^{-11}$. In this case, Figure 11.24b depicts the ideal magnitude response of a fifth-order elliptic filter, satisfying the specifications in Equation (11.137), along with the corresponding worst-case margins due to the coefficient quantization. \triangle

It is worth noting that the sensitivity measurement given by Equation (11.136) is also useful as a figure of merit if one uses the so-called pseudo-floating-point representation, which consists of implementing multiplication between a signal and a coefficient with small magnitude in the following form, as depicted in Figure 11.25:

$$[x \times m_i]_Q = [(x \times m_i \times 2^L) \times 2^{-L}]_Q, \quad (11.143)$$

where L is the exponent of m_i when represented in floating point. Note that in the pseudo-floating-point scheme, all operations are actually performed using fixed-point arithmetic.

11.7.2 Statistical forecast of the wordlength

In the previous subsection we computed the worst-case variation in the frequency response of a digital filter with the quantization of coefficients. By worst case we meant the supposition that quantization made all the coefficients vary the maximum possible amount, and in the worst direction. However, it is unlikely that all coefficients will undergo a worst-case quantization error, and their quantization effects will accumulate in the worst possible way with respect to the resulting frequency response. Thus, it is useful to perform a more realistic, statistical analysis of the deviation in the frequency response. In this subsection we perform a statistical forecast of the wordlength necessary for a filter to satisfy a given specification.

Suppose we have designed a digital filter with frequency response $H(e^{j\omega})$ and that the ideal magnitude response is $H_d(e^{j\omega})$, with a tolerance given by $\rho(\omega)$. When the filter

coefficients are quantized, we can express the resulting magnitude response as

$$\left|H_Q(e^{j\omega})\right| = \left|H(e^{j\omega})\right| + \Delta \left|H(e^{j\omega})\right|. \quad (11.144)$$

Obviously, for a meaningful design, $\left|H_Q(e^{j\omega})\right|$ must not deviate from $H_d(e^{j\omega})$ by more than a frequency-dependent tolerance $\rho(\omega)$; that is:

$$\left|\left(\left|H_Q(e^{j\omega})\right| - H_d(e^{j\omega})\right)\right| = \left|\left|H(e^{j\omega})\right| + \Delta \left|H(e^{j\omega})\right| - H_d(e^{j\omega})\right| \leq \rho(\omega). \quad (11.145)$$

Or, more strictly:

$$\left|\Delta \left|H(e^{j\omega})\right|\right| \leq \rho(\omega) - \left|\left|H(e^{j\omega})\right| - H_d(e^{j\omega})\right|. \quad (11.146)$$

The variation in the magnitude response of the digital filter due to the variations in the multiplier coefficients m_i can be approximated by

$$\Delta \left|H(e^{j\omega})\right| \approx \sum_{i=1}^K \frac{\partial \left|H(e^{j\omega})\right|}{\partial m_i} \Delta m_i. \quad (11.147)$$

If we consider that

- The multiplier coefficients are rounded,
- The quantization errors are statistically independent, and
- All Δm_i are uniformly distributed,

then the variance of the error in each coefficient, based on Equation (11.54), is given by

$$\sigma_{\Delta m_i}^2 = \sigma_{\Delta m}^2 = \frac{2^{-2b}}{12}, \quad \text{for } i = 1, 2, \dots, K, \quad (11.148)$$

where b is the number of bits not including the sign bit.

With the assumptions above, the mean of $\Delta |H(e^{j\omega})|$ is zero and its variance is given by

$$\sigma_{\Delta |H(e^{j\omega})|}^2 \approx \sigma_{\Delta m}^2 \sum_{i=1}^K \left(\frac{\partial |H(e^{j\omega})|}{\partial m_i} \right)^2 = \sigma_{\Delta m}^2 S^2(e^{j\omega}), \quad (11.149)$$

where $S^2(e^{j\omega})$ is given by Equations (11.125) and (11.136).

If we further assume that $\Delta |H(e^{j\omega})|$ is Gaussian (Avenhaus, 1972), we can estimate the probability of $\Delta |H(e^{j\omega})|$ being less than or equal to $x\sigma_{\Delta |H(e^{j\omega})|}$ by

$$\Pr \left\{ \left| \Delta H(e^{j\omega}) \right| \leq x\sigma_{\Delta |H(e^{j\omega})|} \right\} = \frac{2}{\sqrt{\pi}} \int_0^{x/\sqrt{2}} e^{-x'^2} dx'. \quad (11.150)$$

To guarantee that Equation (11.146) holds with a probability less than or equal to the one given in Equation (11.150), it suffices that

$$x\sigma_{\Delta m}S(e^{j\omega}) \leq \rho(\omega) - \left| |H(e^{j\omega})| - H_d(e^{j\omega}) \right|. \quad (11.151)$$

Now, assume that the wordlength, including the sign bit, is given by

$$B = I + F + 1, \quad (11.152)$$

where I and F are the numbers of bits in the integer and fractional parts respectively. The value of I depends on the required order of magnitude of the coefficient, and F can be estimated from Equation (11.151) to guarantee that Equation (11.146) holds with probability as given in Equation (11.150). To satisfy the inequality in (11.151), the value of 2^{-b} , from Equation (11.148), should be given by

$$2^{-b} = \sqrt{12} \min_{\omega \in C} \left\{ \left| \frac{\rho(\omega) - |H(e^{j\omega})| - H_d(e^{j\omega})}{xS(e^{j\omega})} \right| \right\}, \quad (11.153)$$

where C is the set of frequencies not belonging to the filter transition bands. Then, an estimate for F is

$$F \approx b = -\log_2 \left(\sqrt{12} \min_{\omega \in C} \left\{ \left| \frac{\rho(\omega) - |H(e^{j\omega})| - H_d(e^{j\omega})}{xS(e^{j\omega})} \right| \right\} \right). \quad (11.154)$$

This method for estimating the wordlength is also useful in iterative procedures to design filters with minimum wordlength (Avenhaus, 1972; Crochiere & Oppenheim, 1975).

An alternative procedure that is widely used in practice to evaluate the design of digital filters with finite-coefficient wordlength is to design the filters with tighter specifications than required, quantize the coefficients, and check if the prescribed specifications are still met. Obviously, in this case, the success of the design is highly dependent on the designer's experience.

Example 11.8. Determine the total number of bits required for the filter designed in Example 11.3 to satisfy the following specifications, after coefficient quantization:

$$\left. \begin{array}{l} A_p = 1.2 \text{ dB} \\ A_r = 39 \text{ dB} \\ \omega_p = 0.3\pi \text{ rad/sample} \\ \omega_r = 0.4\pi \text{ rad/sample} \end{array} \right\}. \quad (11.155)$$

Solution

Using the specifications in Equation (11.155), we determine

$$\delta_p = 1 - 10^{-A_p/20} = 0.1482 \quad (11.156)$$

$$\delta_r = 10^{-A_r/20} = 0.0112 \quad (11.157)$$

Table 11.2.

Quantized filter coefficients for specifications (11.155).

Numerator coefficients	Denominator coefficients
$b_0 = 0.028\ 320\ 31$	$a_0 = 1.000\ 000\ 00$
$b_1 = -0.001\ 464\ 84$	$a_1 = -3.028\ 564\ 45$
$b_2 = 0.031\ 738\ 28$	$a_2 = 4.567\ 871\ 09$
$b_3 = 0.031\ 738\ 28$	$a_3 = -3.900\ 146\ 48$
$b_4 = -0.001\ 464\ 84$	$a_4 = 1.896\ 728\ 52$
$b_5 = 0.028\ 320\ 31$	$a_5 = -0.418\ 945\ 31$

and define

$$\rho(\omega) = \begin{cases} \delta_p, & \text{for } 0 \leq \omega \leq 0.3\pi \\ \delta_r, & \text{for } 0.4\pi \leq \omega \leq \pi \end{cases} \quad (11.158)$$

$$H_d(e^{j\omega}) = \begin{cases} 1, & \text{for } 0 \leq \omega \leq 0.3\pi \\ 0, & \text{for } 0.4\pi \leq \omega \leq \pi. \end{cases} \quad (11.159)$$

A reasonable certainty margin is about 90%, yielding from Equation (11.150)

$$\frac{x}{\sqrt{2}} = \text{erfinv}(0.9) = 1.1631 \Rightarrow x = 1.6449. \quad (11.160)$$

We use the filter designed in Example 11.3 as $H(e^{j\omega})$, with the corresponding sensitivity function $S(e^{j\omega})$, as given in Equation (11.141) and depicted in Figure 11.24a.

Based on these values, we can compute the number of bits F for the fractional part, using Equation (11.154), resulting in $F \approx 12.0993$, which we round to $F = 12$ bits. From Table 11.1, we observe that $I = 3$ bits are necessary to represent the integer part of the filter coefficients which fall in the range -4 to $+4$. Therefore, the total number of bits required, including the sign bit, is

$$B = I + F + 1 = 16. \quad (11.161)$$

Table 11.2 shows the filter coefficients after quantization.

Table 11.3 includes the resulting passband ripple and stopband attenuation for several values of F , from which we can clearly see that, using the predicted $F = 12$, the specifications in Equation (11.155) are satisfied, even after quantization of the filter coefficients.



Table 11.3.Filter characteristics as a function of the number of fractional bits F .

F	A_p (dB)	A_r (dB)
15	1.0100	40.0012
14	1.0188	40.0106
13	1.0174	40.0107
12	1.1625	39.7525
11	1.1689	39.7581
10	1.2996	39.7650
9	1.2015	40.0280
8	2.3785	40.2212

11.8 Limit cycles

A serious practical problem that affects the implementation of recursive digital filters is the possible occurrence of parasitic oscillations. These oscillations can be classified, according to their origin, as either granular or overflow limit cycles, as presented below.

11.8.1 Granular limit cycles

Any stable digital filter, if implemented with idealized infinite-precision arithmetic, should have an asymptotically decreasing response when the input signal becomes zero after a given instant of time $n_0 T$. However, if the filter is implemented with finite-precision arithmetic, then the noise signals generated at the quantizers become highly correlated from sample to sample and from source to source. This correlation can cause autonomous oscillations, referred to as granular limit cycles, originating from quantization performed in the least significant signal bits, as indicated in the example that follows.

Example 11.9. Suppose the filter of Figure 11.26 has the following input signal:

$$x(n) = \begin{cases} 0.111, & \text{for } n = 1 \\ 0.000, & \text{for } n \neq 1, \end{cases} \quad (11.162)$$

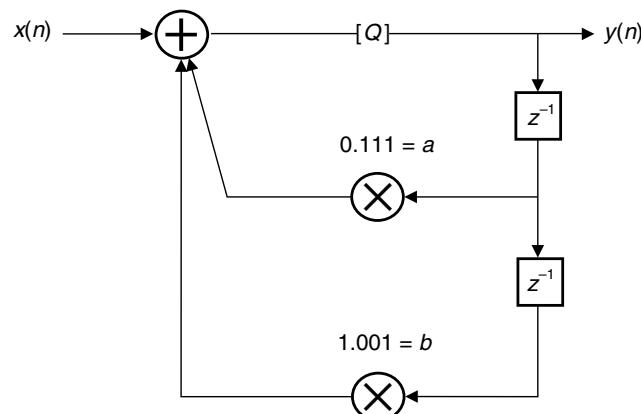
where the numbers are represented in two's complement. Determine the output signal in the case when the quantizer performs rounding, for $n = 1, 2, \dots, 40$.

Solution

The output in the time domain, assuming that the quantizer rounds the signal, is given in Table 11.4, where one can easily see that an oscillation is sustained at the output, even after the input becomes zero. \triangle

Table 11.4. Output signal of network shown in Figure 11.26: $y(n) = Q[ay(n-1) + by(n-2) + x(n)]$.

<i>n</i>	$y(n)$
1	0.111
2	$Q(0.110\ 001 + 0.000\ 000) = 0.110$
3	$Q(0.101\ 010 + 1.001\ 111) = 1.111$
4	$Q(1.111\ 001 + 1.010\ 110) = 1.010$
5	$Q(1.010\ 110 + 0.000\ 111) = 1.100$
6	$Q(1.100\ 100 + 0.101\ 010) = 0.010$
7	$Q(0.001\ 110 + 0.011\ 100) = 0.101$
8	$Q(0.100\ 011 + 1.110\ 010) = 0.011$
9	$Q(0.010\ 101 + 1.011\ 101) = 1.110$
10	$Q(1.110\ 010 + 1.101\ 011) = 1.100$
11	$Q(1.100\ 100 + 0.001\ 110) = 1.110$
12	$Q(1.110\ 010 + 0.011\ 100) = 0.010$
13	$Q(0.001\ 110 + 0.001\ 110) = 0.100$
14	$Q(0.011\ 100 + 1.110\ 010) = 0.010$
15	$Q(0.001\ 110 + 1.100\ 100) = 1.110$
16	$Q(1.110\ 010 + 1.110\ 010) = 1.100$
17	$Q(1.100\ 100 + 0.001\ 110) = 1.110$
18	$Q(1.110\ 010 + 0.011\ 100) = 0.010$
19	$Q(0.001\ 110 + 0.001\ 110) = 0.100$
20	$Q(0.011\ 100 + 1.110\ 010) = 0.010$
21	$Q(0.001\ 110 + 1.100\ 100) = 1.110$
22	$Q(1.110\ 010 + 1.110\ 010) = 1.100$
:	:

**Fig. 11.26.** Second-order section with a quantizer.

In many practical applications, where the signal levels in a digital filter can be constant or very low, even for short periods of time, limit cycles are highly undesirable and should be eliminated or at least have their amplitude bounds strictly limited.

11.8.2 Overflow limit cycles

Overflow limit cycles can occur when the magnitudes of the internal signals exceed the available register range. In order to avoid the increase of the signal wordlength in recursive digital filters, overflow nonlinearities must be applied to the signal. Such nonlinearities influence the most significant bits of the signal, possibly causing severe distortion. An overflow can give rise to self-sustained, high-amplitude oscillations, widely known as overflow limit cycles. Overflow can occur in any structure in the presence of an input signal, and input-signal scaling is crucial to reduce the probability of overflow to an acceptable level.

Example 11.10. Consider the filter of Figure 11.27 with $a = 0.9606$ and $b = -0.9849$, where the overflow nonlinearity employed is the two's complement with 3-bit quantization (see Figure 11.27). Its analytic expression is given by

$$Q(x) = \frac{1}{4}[(\lceil 4x - 0.5 \rceil + 4) \bmod 8] - 1 \quad (11.163)$$

where $\lceil x \rceil$ means the smallest integer larger than or equal to x .

Determine the output signal of such a filter for zero input, given the initial conditions $y(-2) = 0.50$ and $y(-1) = -1.00$.

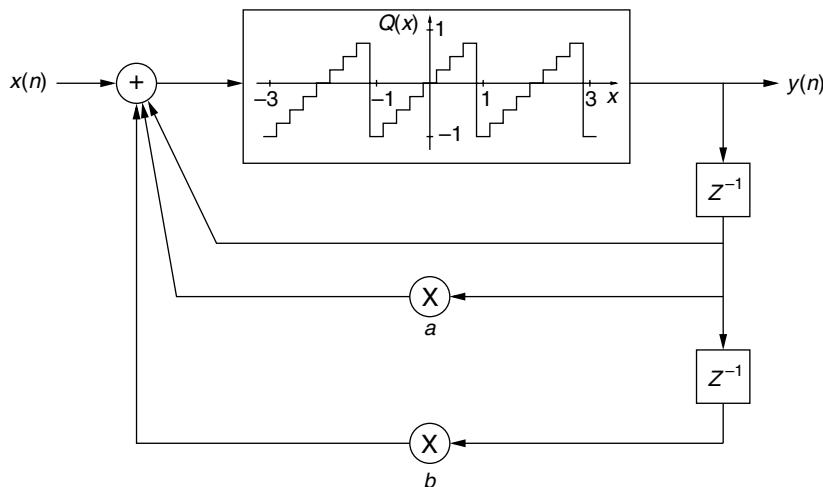


Fig. 11.27. Second-order section with an overflow quantizer.

Solution

With $a = 0.9606$, $b = -0.9849$, $y(-2) = 0.50$, and $y(-1) = -1.00$, we have that

$$\begin{aligned} y(0) &= Q[1.9606(-1.00) - 0.9849(0.50)] = Q[-2.4530] = -0.50 \\ y(1) &= Q[1.9606(-0.50) - 0.9849(-1.00)] = Q[0.0046] = 0.00 \\ y(2) &= Q[1.9606(0.00) - 0.9849(-0.50)] = Q[0.4924] = 0.50 \\ y(3) &= Q[1.9606(0.50) - 0.9849(0.00)] = Q[0.9803] = -1.00 \\ &\vdots \end{aligned} \quad (11.164)$$

Since $y(2) = y(-2)$ and $y(3) = y(-1)$, we have that, although there is no excitation, the output signal is nonzero and periodic with a period of 4, thus indicating the existence of overflow limit cycles. \triangle

A digital filter structure is considered free from overflow limit cycles if the error introduced in the filter after an overflow decreases with time in such a way that the output of the nonlinear filter (including the quantizers) converges to the output of the ideal linear filter (Claasen *et al.*, 1975).

In practice, a quantizer incorporates nonlinearities corresponding to both granular quantization and overflow. Figure 11.28 illustrates a digital filter using a quantizer that implements rounding as the granular quantization and saturation arithmetic as the overflow nonlinearity. Note that although this overflow nonlinearity is different from the one depicted in Figure 11.27, both are classified as overflow.

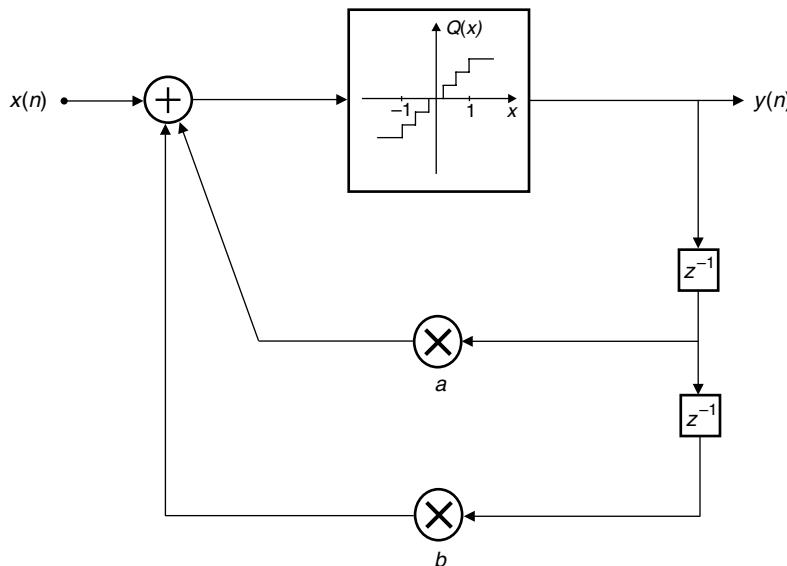


Fig. 11.28. Second-order section with a rounding and saturating quantizer.

11.8.3 Elimination of zero-input limit cycles

A general IIR filter can be depicted as in Figure 11.29a, where the linear N -port network consists of interconnections of multipliers and adders. In a recursive filter implemented with fixed-point arithmetic, each internal loop contains a quantizer. Assuming that the quantizers are placed at the delay inputs (the state variables), as shown in Figure 11.29b, we can describe the digital filter, including the quantizers, using the following state-space formulation:

$$\left. \begin{aligned} \mathbf{x}(n+1) &= [\mathbf{Ax}(n) + \mathbf{bu}(n)]_Q \\ y(n) &= \mathbf{c}^T \mathbf{x}(n) + du(n) \end{aligned} \right\}, \quad (11.165)$$

where $[x]_Q$ indicates the quantized value of x , \mathbf{A} is the state matrix, \mathbf{b} is the input vector, \mathbf{c} is the output vector, and d represents the direct connection between the input and output of the filter.

In order to analyze zero-input limit cycles, it is sufficient to consider the recursive part of the state equation given by

$$\mathbf{x}(k+1) = [\mathbf{Ax}(k)]_Q = [\mathbf{x}'(k+1)]_Q, \quad (11.166)$$

where the quantization operations $[\cdot]_Q$ are nonlinear operations such as truncation, rounding, or overflow.

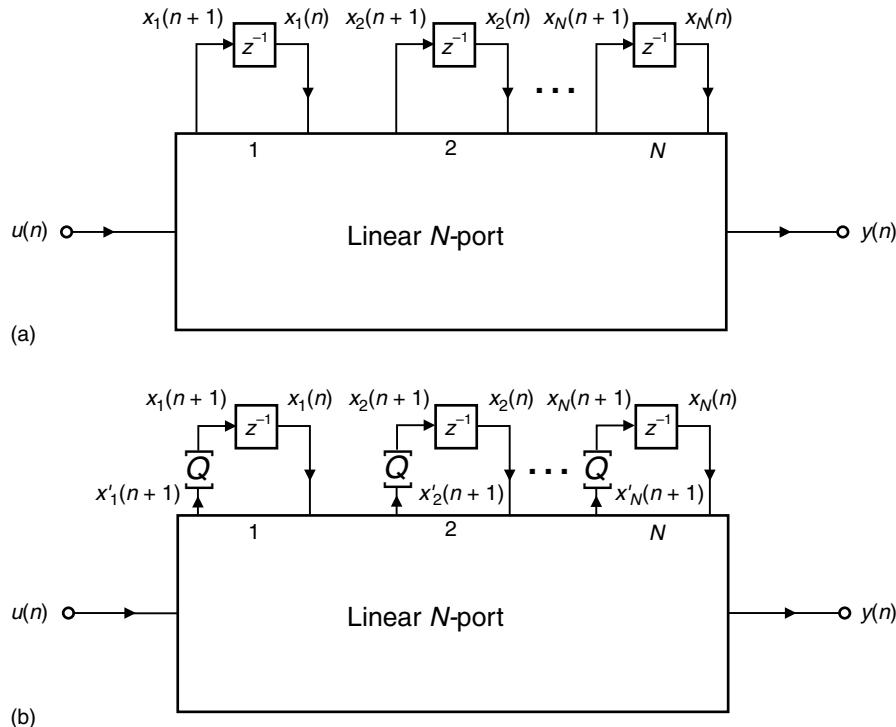


Fig. 11.29. Digital filter networks: (a) ideal; (b) with quantizers at the state variables.

The basis for the elimination of nonlinear oscillations is given by Theorem 11.1.

Theorem 11.1. *If a stable digital filter has a state matrix \mathbf{A} and, for any $N \times 1$ vector $\hat{\mathbf{x}}$ there exists a diagonal positive-definite matrix \mathbf{G} , such that*

$$\hat{\mathbf{x}}^T(\mathbf{G} - \mathbf{A}^T \mathbf{G} \mathbf{A}) \hat{\mathbf{x}} \geq 0, \quad (11.167)$$

then the granular zero-input limit cycles can be eliminated if the quantization is performed through magnitude truncation. \diamond

Proof Consider a nonnegative pseudo-energy Lyapunov function given by (Willems, 1970)

$$p(\mathbf{x}(n)) = \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n). \quad (11.168)$$

The energy variation in a single iteration can be defined as

$$\begin{aligned} \Delta p(n+1) &= p(\mathbf{x}(n+1)) - p(\mathbf{x}(n)) \\ &= \mathbf{x}^T(n+1) \mathbf{G} \mathbf{x}(n+1) - \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n) \\ &= [\mathbf{x}'^T(n+1)]_Q \mathbf{G} [\mathbf{x}'(n+1)]_Q - \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n) \\ &= [\mathbf{A} \mathbf{x}(n)]_Q^T \mathbf{G} [\mathbf{A} \mathbf{x}(n)]_Q - \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n) \\ &= [\mathbf{A} \mathbf{x}(n)]^T \mathbf{G} [\mathbf{A} \mathbf{x}(n)] - \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n) \\ &\quad - \sum_{i=1}^N (x_i'^2(n+1) - x_i^2(n+1)) g_i \\ &= \mathbf{x}^T(n) [\mathbf{A}^T \mathbf{G} \mathbf{A} - \mathbf{G}] \mathbf{x}(n) - \sum_{i=1}^N (x_i'^2(n+1) - x_i^2(n+1)) g_i, \end{aligned} \quad (11.169)$$

where g_i are the diagonal elements of \mathbf{G} .

If quantization is performed through magnitude truncation, then the errors due to granular quantization and overflow are such that

$$|x_i(n+1)| \leq |x'_i(n+1)| \quad (11.170)$$

for all i and n . Therefore, if Equation (11.167) holds, from Equation (11.169), we have that

$$\Delta p(n+1) \leq 0. \quad (11.171)$$

If a digital filter is implemented with finite-precision arithmetic, within a finite number of samples after the input signal becomes zero, the output signal will become either a periodic oscillation or zero. Periodic oscillations, with nonzero amplitude, cannot be sustained if $\Delta p(n+1) \leq 0$, as shown above. Therefore, Equations (11.167) and (11.170) are sufficient conditions to guarantee the elimination of granular zero-input limit cycles on a recursive digital filter. \square

Note that the condition given in Equation (11.167) is equivalent to requiring that \mathbf{F} be positive semidefinite, where

$$\mathbf{F} = (\mathbf{G} - \mathbf{A}^T \mathbf{G} \mathbf{A}). \quad (11.172)$$

It is worth observing that for any stable state matrix \mathbf{A} , its eigenvalues are inside the unit circle, and there will always be a positive-definite and symmetric matrix \mathbf{G} such that \mathbf{F} is symmetric and positive semidefinite. However, if \mathbf{G} is not diagonal, then the quantization process required to eliminate zero-input limit cycles is extremely complicated (Meerkötter, 1976), as the quantization operation in each quantizer is coupled to the others. On the other hand, if there is a matrix \mathbf{G} which is diagonal and positive definite such that \mathbf{F} is positive semidefinite, then zero-input limit cycles can be eliminated by simple magnitude truncation.

In the following theorem, we will state more specific conditions regarding the elimination of zero-input limit cycles in second-order systems.

Theorem 11.2. *Given a 2×2 stable state matrix \mathbf{A} , there is a diagonal positive-definite matrix \mathbf{G} , such that \mathbf{F} is positive semidefinite, if and only if (Mills et al., 1978)*

$$a_{12}a_{21} \geq 0 \quad (11.173)$$

or

$$\left. \begin{array}{l} a_{12}a_{21} < 0 \\ |a_{11} - a_{22}| + \det(\mathbf{A}) \leq 1 \end{array} \right\}. \quad (11.174)$$

◇

Proof Let $\mathbf{G} = (\mathbf{T}^{-1})^2$ be a diagonal positive-definite matrix, such that \mathbf{T} is a diagonal nonsingular matrix. Therefore, we can write \mathbf{F} as

$$\mathbf{F} = \mathbf{T}^{-1}\mathbf{T}^{-1} - \mathbf{A}^T\mathbf{T}^{-1}\mathbf{T}^{-1}\mathbf{A} \quad (11.175)$$

and then

$$\begin{aligned} \mathbf{T}^T \mathbf{F} \mathbf{T} &= \mathbf{T}^T \mathbf{T}^{-1} \mathbf{T}^{-1} \mathbf{T} - \mathbf{T}^T \mathbf{A}^T \mathbf{T}^{-1} \mathbf{T}^{-1} \mathbf{A} \mathbf{T} \\ &= \mathbf{I} - (\mathbf{T}^{-1} \mathbf{A} \mathbf{T})^T (\mathbf{T}^{-1} \mathbf{A} \mathbf{T}) \\ &= \mathbf{I} - \mathbf{M} \end{aligned} \quad (11.176)$$

with $\mathbf{M} = (\mathbf{T}^{-1} \mathbf{A} \mathbf{T})^T (\mathbf{T}^{-1} \mathbf{A} \mathbf{T})$, as $\mathbf{T}^T = \mathbf{T}$.

Since the matrix $(\mathbf{I} - \mathbf{M})$ is symmetric and real, its eigenvalues are real. This matrix is then positive semidefinite if and only if its eigenvalues are nonnegative (Strang, 1980), or, equivalently, if and only if its trace and determinant are nonnegative. We then have that

$$\det(\mathbf{I} - \mathbf{M}) = 1 + \det(\mathbf{M}) - \text{tr}(\mathbf{M}) = 1 + (\det(\mathbf{A}))^2 - \text{tr}(\mathbf{M}) \quad (11.177)$$

$$\text{tr}(\mathbf{I} - \mathbf{M}) = 2 - \text{tr}(\mathbf{M}). \quad (11.178)$$

For a stable digital filter, it is easy to verify that $\det\{\mathbf{A}\} < 1$, and then

$$\text{tr}\{\mathbf{I} - \mathbf{M}\} > \det\{\mathbf{I} - \mathbf{M}\}. \quad (11.179)$$

Hence, the condition $\det\{\mathbf{I} - \mathbf{M}\} \geq 0$ is necessary and sufficient to guarantee that $(\mathbf{I} - \mathbf{M})$ is positive semidefinite.

From the definition of \mathbf{M} , and using $\alpha = t_{22}/t_{11}$, then

$$\det\{\mathbf{I} - \mathbf{M}\} = 1 + (\det\{\mathbf{A}\})^2 - \left(a_{11}^2 + \alpha^2 a_{12}^2 + \frac{a_{21}^2}{\alpha^2} + a_{22}^2 \right). \quad (11.180)$$

By calculating the maximum of the equation above with respect to α , we get an optimal α^* such that

$$(\alpha^*)^2 = \left| \frac{a_{21}}{a_{12}} \right| \quad (11.181)$$

and then

$$\begin{aligned} \det^*\{\mathbf{I} - \mathbf{M}\} &= 1 + (\det\{\mathbf{A}\})^2 - (a_{11}^2 + 2|a_{12}a_{21}| + a_{22}^2) \\ &= (1 + \det\{\mathbf{A}\})^2 - (\text{tr}\{\mathbf{A}\})^2 + 2(a_{12}a_{21} - |a_{12}a_{21}|), \end{aligned} \quad (11.182)$$

where \det^* denotes the maximum value of the respective determinant. We now analyze two separate cases to guarantee that $\det^*\{\mathbf{I} - \mathbf{M}\} \geq 0$.

- If

$$a_{12}a_{21} \geq 0 \quad (11.183)$$

then

$$\begin{aligned} \det^*\{\mathbf{I} - \mathbf{M}\} &= (1 + \det\{\mathbf{A}\})^2 - (\text{tr}\{\mathbf{A}\})^2 \\ &= (1 + \alpha_2)^2 - (-\alpha_1)^2 \\ &= (1 + \alpha_1 + \alpha_2)(1 - \alpha_1 + \alpha_2), \end{aligned} \quad (11.184)$$

where $\alpha_1 = -\text{tr}\{\mathbf{A}\}$ and $\alpha_2 = \det\{\mathbf{A}\}$ are the filter denominator coefficients. It can be verified that, for a stable filter, $(1 + \alpha_1 + \alpha_2)(1 - \alpha_1 + \alpha_2) > 0$, and then Equation (11.183) implies that $(\mathbf{I} - \mathbf{M})$ is positive definite.

- If

$$a_{12}a_{21} < 0 \quad (11.185)$$

then

$$\begin{aligned} \det^*\{\mathbf{I} - \mathbf{M}\} &= 1 + (\det\{\mathbf{A}\})^2 - (a_{11}^2 - 2a_{12}a_{21} + a_{22}^2) \\ &= (1 - \det\{\mathbf{A}\})^2 - (a_{11} - a_{22})^2. \end{aligned} \quad (11.186)$$

This equation is greater than or equal to zero, if and only if

$$|a_{11} - a_{22}| + \det\{\mathbf{A}\} \leq 1. \quad (11.187)$$

Therefore, either Equation (11.183) or Equations (11.185) and (11.187) are the necessary and sufficient conditions for the existence of a diagonal matrix

$$\mathbf{T} = \text{diag}\{t_{11}, t_{22}\} \quad (11.188)$$

with

$$\frac{t_{22}}{t_{11}} = \sqrt{\left| \frac{a_{21}}{a_{12}} \right|} \quad (11.189)$$

such that \mathbf{F} is positive semidefinite. \square

It is worth observing that the above theorem gives the conditions for the matrix \mathbf{F} to be positive semidefinite for second-order sections. In the example below, we illustrate the limit-cycle elimination process by showing, without resorting to Theorem 11.2, that a given second-order structure is free from zero-input limit cycles. The reader is encouraged to apply the theorem to show the same result.

Example 11.11. Examine the possibility of eliminating limit cycles in the network of Figure 11.30 (Diniz & Antoniou, 1988).

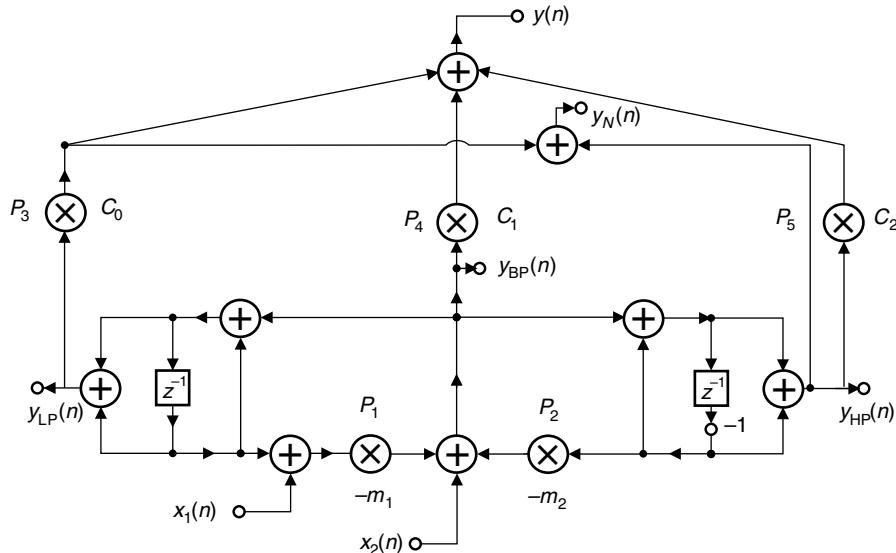


Fig. 11.30. General-purpose network.

Solution

The structure in Figure 11.30 realizes lowpass, bandpass, and highpass transfer functions simultaneously (with subscripts LP, BP, and HP respectively). The structure also realizes a transfer function with zeros on the unit circle, using the minimum number of multipliers. The characteristic polynomial of the structure is given by

$$D(z) = z^2 + (m_1 - m_2)z + m_1 + m_2 - 1. \quad (11.190)$$

In order to guarantee stability, the multiplier coefficients m_1 and m_2 should fall in the range

$$\left. \begin{array}{l} m_1 > 0 \\ m_2 > 0 \\ m_1 + m_2 < 2 \end{array} \right\}. \quad (11.191)$$

Figure 11.31 depicts the recursive part of the structure in Figure 11.30, including the quantizers.

The zero-input state-space equation for the structure in Figure 11.31 is

$$\mathbf{x}'(n+1) = \begin{bmatrix} x'_1(n+1) \\ x'_2(n+1) \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix}, \quad (11.192)$$

with

$$\mathbf{A} = \begin{bmatrix} (1 - m_1) & m_2 \\ -m_1 & (m_2 - 1) \end{bmatrix}. \quad (11.193)$$

By applying quantization to $\mathbf{x}'(n+1)$, we find

$$\mathbf{x}(n+1) = [\mathbf{x}'(n+1)]_Q = [\mathbf{Ax}(n)]_Q. \quad (11.194)$$

A quadratic positive-definite function can be defined as

$$p(\mathbf{x}(n)) = \mathbf{x}^T(n) \mathbf{G} \mathbf{x}(n) = \frac{x_1^2}{m_2} + \frac{x_2^2}{m_1}, \quad (11.195)$$

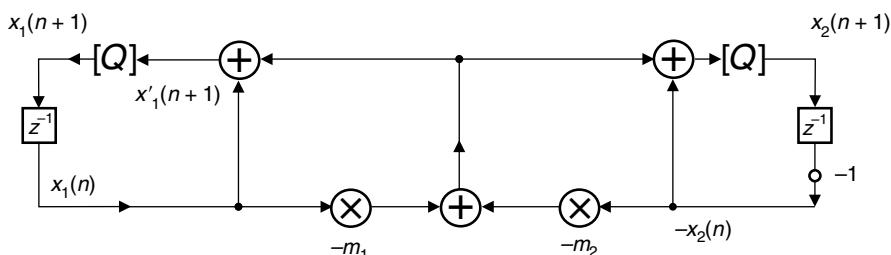


Fig. 11.31.

Recursive part of the network in Figure 11.30.

with

$$\mathbf{G} = \begin{bmatrix} 1/m_2 & 0 \\ 0 & 1/m_1 \end{bmatrix} \quad (11.196)$$

which is positive definite, since, from Equation (11.191), $m_1 > 0$ and $m_2 > 0$.

An auxiliary energy increment is then

$$\begin{aligned} \Delta p_0(n+1) &= p(\mathbf{x}'(n+1)) - p(\mathbf{x}(n)) \\ &= \mathbf{x}'^T(n+1)\mathbf{G}\mathbf{x}'(n+1) - \mathbf{x}^T(n)\mathbf{G}\mathbf{x}(n) \\ &= \mathbf{x}^T(n)[\mathbf{A}^T\mathbf{G}\mathbf{A} - \mathbf{G}]\mathbf{x}(n) \\ &= (m_1 + m_2 - 2) \left(x_1(n) \sqrt{\frac{m_1}{m_2}} - x_2(n) \sqrt{\frac{m_2}{m_1}} \right)^2. \end{aligned} \quad (11.197)$$

Since, from Equation (11.191), $m_1 + m_2 < 2$, then

$$\left. \begin{array}{l} \Delta p_0(n+1) = 0, \quad \text{for } x_1(n) = x_2(n) \frac{m_2}{m_1} \\ \Delta p_0(n+1) < 0, \quad \text{for } x_1(n) \neq x_2(n) \frac{m_2}{m_1} \end{array} \right\}. \quad (11.198)$$

Now, if magnitude truncation is applied to quantize the state variables, then $p(\mathbf{x}(n)) \leq p(\mathbf{x}'(n))$, which implies that

$$\Delta p(\mathbf{x}(n)) = p(\mathbf{x}(n+1)) - p(\mathbf{x}(n)) \leq 0 \quad (11.199)$$

and then $p(\mathbf{x}(n))$ is a Lyapunov function.

Overall, when no quantization is applied in the structure of Figure 11.30, no self-sustained oscillations occur if the stability conditions of Equation (11.191) are satisfied. If, however, quantization is applied to the structure, as shown in Figure 11.31, then oscillations may occur. Using magnitude truncation, then $|x_i(n)| \leq |x'_i(n)|$, and under these circumstances $p(\mathbf{x}(n))$ decreases during the subsequent iterations, and eventually the oscillations disappear, with

$$\mathbf{x}(n) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (11.200)$$

being the only possible equilibrium point. \triangle

11.8.4 Elimination of constant-input limit cycles

As seen above, the sufficient conditions for the elimination of zero-input limit cycles are well established. However, if the system input is a nonzero constant, then limit cycles may still occur. It is worth observing that the response of a stable linear system to a constant

input signal should also be a constant signal. In Diniz and Antoniou (1986), a theorem is presented that establishes how constant-input limit cycles can also be eliminated in digital filters in which zero-input limit cycles are eliminated. This theorem is as follows.

Theorem 11.3. Assume that the general digital filter in Figure 11.29b does not sustain zero-input limit cycles and that

$$\left. \begin{aligned} \mathbf{x}(n+1) &= [\mathbf{Ax}(n) + \mathbf{Bu}(n)]_Q \\ y(n) &= \mathbf{C}^T \mathbf{x}(n) + du(n) \end{aligned} \right\}. \quad (11.201)$$

Constant-input limit cycles can also be eliminated, by modifying the structure in Figure 11.29b, as shown in Figure 11.32, where

$$\mathbf{p} = [p_1 \ p_2 \ \cdots \ p_n]^T = (\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} \quad (11.202)$$

and $\mathbf{p}u_0$ must be representable in the machine wordlength, where u_0 is a constant input signal. \diamond

Proof Since the structure of Figure 11.29b is free from zero-input limit cycles, the autonomous system

$$\mathbf{x}(n+1) = [\mathbf{Ax}(n)]_Q \quad (11.203)$$

is such that

$$\lim_{n \rightarrow \infty} \mathbf{x}(n) = [0 \ 0 \ \cdots \ 0]^T. \quad (11.204)$$

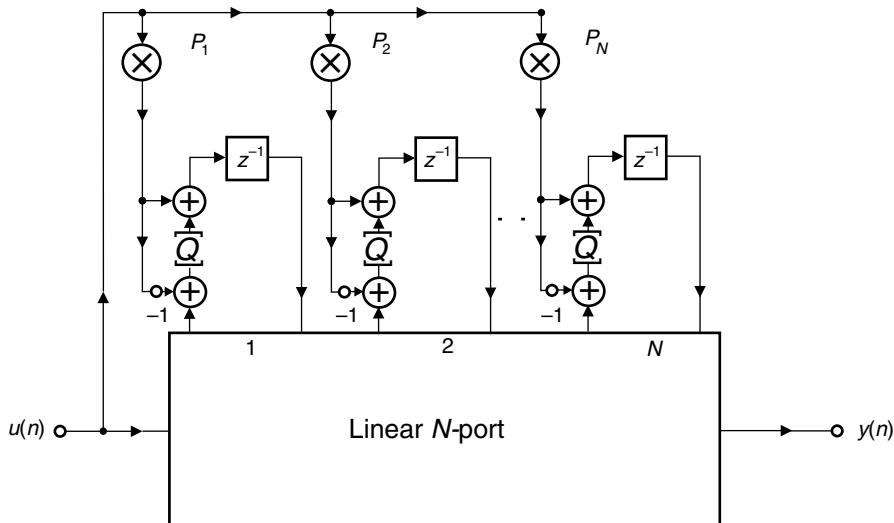


Fig. 11.32. Modified N th-order network for the elimination of constant-input limit cycles.

If \mathbf{p} is as given in Equation (11.202), then the modified structure of Figure 11.32 is described by

$$\begin{aligned}\mathbf{x}(n+1) &= [\mathbf{Ax}(n) - \mathbf{pu}_0 + \mathbf{Bu}_0]_Q + \mathbf{pu}_0 \\ &= [\mathbf{Ax}(n) - \mathbf{I}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{Bu}_0 + (\mathbf{I} - \mathbf{A})(\mathbf{I} - \mathbf{A})^{-1}\mathbf{Bu}_0]_Q + \mathbf{pu}_0 \\ &= [\mathbf{A}(\mathbf{x}(n) - \mathbf{pu}_0)]_Q + \mathbf{pu}_0.\end{aligned}\quad (11.205)$$

Defining

$$\hat{\mathbf{x}}(n) = \mathbf{x}(n) - \mathbf{pu}_0,\quad (11.206)$$

then, from Equation (11.205), we can write that

$$\hat{\mathbf{x}}(n+1) = [\mathbf{A}\hat{\mathbf{x}}(n)]_Q.\quad (11.207)$$

This is the same as Equation (11.203), except for the transformation in the state variable. Hence, as \mathbf{pu}_0 is machine representable (that is, \mathbf{pu}_0 can be exactly calculated with the available wordlength), Equation (11.207) also represents a stable system free from constant-input limit cycles. \square

If the quantization of the structure depicted in Figure 11.29b is performed using magnitude truncation, then the application of the strategy of Theorem 11.3 leads to the so-called controlled rounding proposed in Butterweck (1975).

The constraints imposed by requiring that \mathbf{pu}_0 be machine representable reduce the number of structures in which the technique described by Theorem 11.3 applies. However, there are a large number of second-order sections and wave digital filter structures in which these requirements are automatically met. In fact, a large number of research papers have been published proposing new structures which are free from zero-input limit cycles (Meerkötter & Wegener, 1975; Fettweis & Meerkötter, 1975a; Diniz & Antoniou, 1988) and free from constant-input limit cycles (Verkroost & Butterweck, 1976; Verkroost, 1977; Liu & Turner, 1983; Diniz, 1988; Sarcinelli Filho & Diniz, 1990). However, the analysis procedures for the generation of these structures are not unified, and here we have aimed to provide a unified framework leading to a general procedure to generate structures which are free from granular limit cycles.

Example 11.12. Show that, by placing the input signal at the point denoted by $x_1(n)$, the structure in Figure 11.30 is free from constant-input limit cycles.

Solution

The second-order section of Figure 11.30, with a constant input $x_1(n) = u_0$, can be described by

$$\mathbf{x}(n+1) = \mathbf{Ax}(n) + \begin{bmatrix} -m_1 \\ -m_1 \end{bmatrix} u_0\quad (11.208)$$

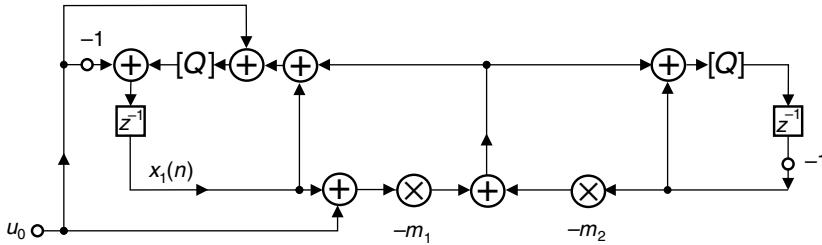


Fig. 11.33.

Elimination of constant-input limit cycles in the structure of Figure 11.30.

with \mathbf{p} such that

$$\mathbf{p} = \begin{bmatrix} m_1 & -m_2 \\ m_1 & 2 - m_2 \end{bmatrix}^{-1} \begin{bmatrix} -m_1 \\ -m_1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}. \quad (11.209)$$

Therefore, $\mathbf{p}u_0$ is clearly machine representable, for any u_0 , and the constant-input limit cycles can be eliminated, as depicted in Figure 11.33. \triangle

Example 11.13. For the second-order state-variable realization as given in Figure 4.23, discuss the elimination of constant-input limit-cycles in a distributed arithmetic implementation such as the one in Section 11.4.

Solution

In a regular implementation, such as the one in Figure 11.11, to eliminate zero-input limit cycles in the state-space realization, the state variables $x_1(n)$ and $x_2(n)$ must be calculated by ALU₁ and ALU₂ in double precision, and then properly quantized, before being loaded into the shift-registers SR₂ and SR₃. However, it can be shown that no double-precision computation is necessary to avoid zero-input limit cycles when implementing the state-space realization with the distributed arithmetic approach (De la Vega *et al.*, 1995).

To eliminate constant-input limit cycles, the state-space realization shown in Figure 11.34 requires that the state variable $x_1(n)$ must be computed by ALU₁ in double precision, and then properly quantized to be subtracted from the input signal. To perform this subtraction, register A in Figure 11.9 must be multiplexed with another register that contains the input signal $x(n)$, in order to guarantee that the signal arriving at the adder, at the appropriate instant of time, is the complemented version of $x(n)$, instead of a signal coming from the memory. In such a case, the content of the ROM of ALU₁ must be generated as

$$s'_{1j} = a_{11}x_{1j}(n) + a_{12}x_{2j}(n) + a_{11}x_j(n), \text{ for the ROM of the ALU}_1, \quad (11.210)$$

while ALU₃ is filled in the same fashion as given in Equation (11.52), and for ALU₂ the content is the same as in Equation (11.51) with b_2 replaced by a_{21} . \triangle

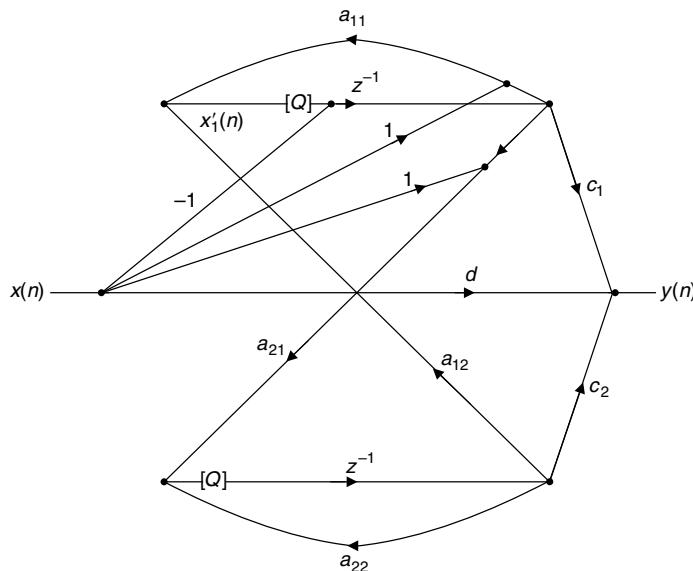


Fig. 11.34. State-space realization immune to constant-input limit cycles.

11.8.5 Forced-response stability of digital filters with nonlinearities due to overflow

The stability analysis of the forced response of digital filters that include nonlinearities to control overflow must be performed considering input signals for which, in the ideal linear system, the overflow level is never reached after a given instant n_0 . In this way, we can verify whether the real system output will recover after an overflow has occurred before instant n_0 . Although the input signals considered are in a particular class of signals, it can be shown that if the real system recovers for these signals, then it will also recover after each overflow, for any input signal, if the recovery period is shorter than the time between two consecutive overflows (Claasen *et al.*, 1975).

Consider the ideal linear system depicted in Figure 11.35a and the real nonlinear system depicted in Figure 11.35b.

The linear system illustrated in Figure 11.35a is described by the equations

$$\mathbf{f}(n) = \mathbf{Ax}(n) + \mathbf{Bu}_1(n) \quad (11.211)$$

$$\mathbf{x}(n) = \mathbf{x}(n-1) \quad (11.212)$$

and the nonlinear system illustrated in Figure 11.35b is described by the equations

$$\mathbf{f}'(n) = \mathbf{Ax}'(n) + \mathbf{Bu}_2(n) \quad (11.213)$$

$$\mathbf{x}'(n) = [\mathbf{f}'(n-1)]_{Q_0}, \quad (11.214)$$

where $[u]_{Q_0}$ denotes quantization of u , in the case where an overflow occurs.

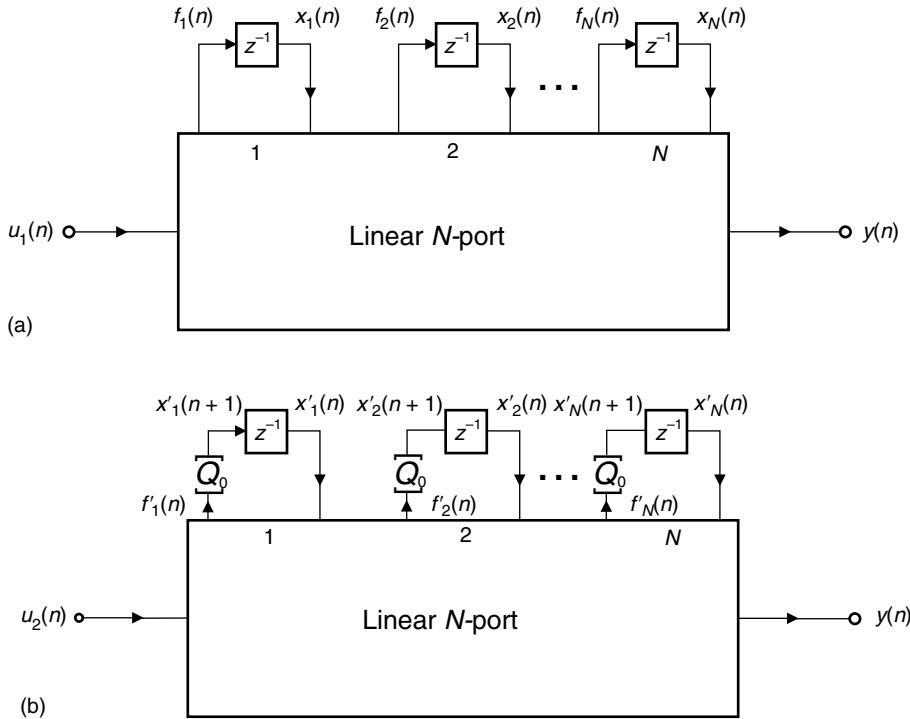


Fig. 11.35. General digital filter networks: (a) ideal; (b) with quantizers at the state variables.

We assume that the output signal of the nonlinear system is properly scaled so that no oscillation due to overflow occurs if it does not occur at the state variables.

The response of the nonlinear system of Figure 11.35b is stable if, when $u_1(n) = u_2(n)$, the difference between the outputs of the linear N -port system of Figure 11.35a, $\mathbf{f}(n)$, and the outputs of the linear N -port system of Figure 11.35b, $\mathbf{f}'(n)$, tends to zero as $n \rightarrow \infty$. In other words, if we define an error signal $\mathbf{e}(n) = \mathbf{f}'(n) - \mathbf{f}(n)$, then

$$\lim_{n \rightarrow \infty} \mathbf{e}(n) = [0 \ 0 \cdots 0]^T. \quad (11.215)$$

If the difference between the output signals of the two linear N -port systems converges to zero, then this implies that the difference between the state variables of both systems will also tend to zero. This can be deduced from Equations (11.211) and (11.213), which yield

$$\mathbf{e}(n) = \mathbf{f}'(n) - \mathbf{f}(n) = \mathbf{A}[\mathbf{x}'(n) - \mathbf{x}(n)] = \mathbf{A}\mathbf{e}'(n), \quad (11.216)$$

where $\mathbf{e}'(n) = \mathbf{x}'(n) - \mathbf{x}(n)$ is the difference between the state variables of both systems.

Equation (11.216) is equivalent to saying that $\mathbf{e}(n)$ and $\mathbf{e}'(n)$ are the output and input signals of a linear N -port system described by matrix \mathbf{A} , which is the transition matrix of the original system. Then, from Equation (11.215), the forced-response stability of the system in Figure 11.35b is equivalent to the zero-input response of the same system, regardless of the quantization characteristics $[\cdot]_{Q_0}$.

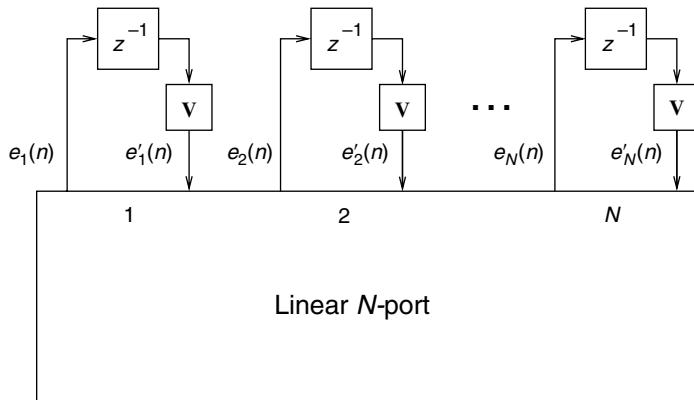


Fig. 11.36. Nonlinear system relating the signals $\mathbf{e}'(n)$ and $\mathbf{e}(n)$.

Substituting Equations (11.212) and (11.214) in Equation (11.216), we have that

$$\mathbf{e}'(n) = [\mathbf{f}'(n-1)]_{Q_0} - \mathbf{f}(n-1) = [\mathbf{e}(n-1) + \mathbf{f}(n-1)]_{Q_0} - \mathbf{f}(n-1). \quad (11.217)$$

By defining the time-varying vector $\mathbf{v}(\mathbf{e}(n), n)$ as

$$\mathbf{v}(\mathbf{e}(n), n) = [\mathbf{e}(n) + \mathbf{f}(n)]_{Q_0} - \mathbf{f}(n), \quad (11.218)$$

Equation (11.217) can be rewritten as

$$\mathbf{e}'(n) = \mathbf{v}(\mathbf{e}(n-1), (n-1)). \quad (11.219)$$

The nonlinear system described by Equations (11.216)–(11.219) is depicted in Figure 11.36.

As we saw in Section 11.8.3, a system such as the one in Figure 11.36 is free from zero-input nonlinear oscillations if the nonlinearity $\mathbf{v}(\cdot, n)$ is equivalent to the magnitude truncation; that is:

$$|\mathbf{v}(\mathbf{e}_i(n), n)| < |\mathbf{e}_i(n)|, \quad \text{for } i = 1, 2, \dots, N. \quad (11.220)$$

If we assume that the internal signals are such that $|f_i(n)| \leq 1$, for $n > n_0$, then it can be shown that Equation (11.220) remains valid whenever the quantizer Q_0 has overflow characteristics within the hatched region of Figure 11.37 (see Exercise 11.31).

Figure 11.37 can be interpreted as follows:

- If $-1 \leq x_i(n) \leq 1$, then there should be no overflow.
- If $1 \leq x_i(n) \leq 3$, then the overflow nonlinearity should be such that $2 - x_i(n) \leq Q_0(x_i(n)) \leq 1$.
- If $-3 \leq x_i(n) \leq -1$, then the overflow nonlinearity should be such that $-1 \leq Q_0(x_i(n)) \leq -2 - x_i(n)$.
- If $x_i(n) \geq 3$ or $x_i(n) \leq -3$, then $-1 \leq Q_0(x_i(n)) \leq 1$.

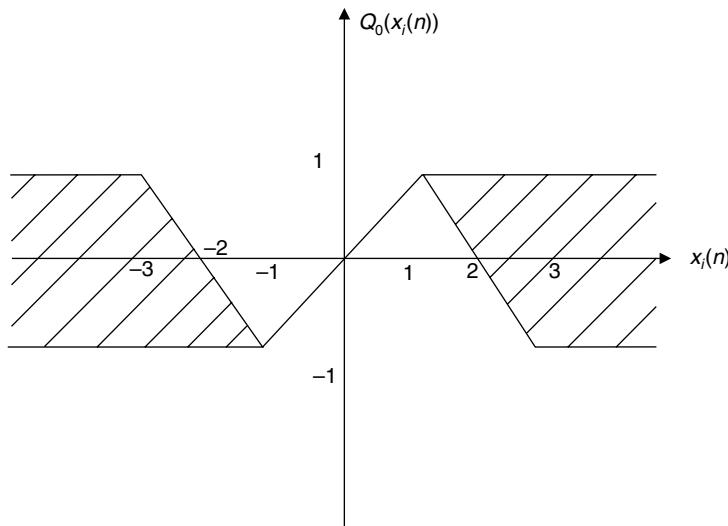


Fig. 11.37. Region for the overflow nonlinearity which guarantees forced-response stability in networks which satisfy Theorem 11.1.

It is important to note that the overflow nonlinearity of the saturation type (Equation (11.163)) satisfies the requirements in Figure 11.37.

Summarizing the above reasoning, one can state that a digital filter which is free of zero-input limit cycles, according to the condition of Equation (11.167), is also forced-input stable, provided that the overflow nonlinearities are in the hatched regions of Figure 11.37.

11.9 Do-it-yourself: finite-precision digital signal processing

Experiment 11.1

Let us play around with digital representation in MATLAB. We concentrate our efforts here on the case $-1 < x < 0$, which yields different $(n + 1)$ -bit standard, one's-complement, two's-complement, and CSD representations.

The standard sign-magnitude binary representation x_{bin} can be obtained making $s_x = 1$ and following the procedure performed in Equation (11.12), such that

```

x = abs(x); xbin = [1 zeros(1,n)];
for i=2:n+1,
    x = 2*x;
    if x >= 1,
        xbin(i) = 1; x = x-1;
    end;
end;
```

Table 11.5.

Numerical eight-digit representations of $x = -0.6875$ in Experiment 11.1.

Numerical format	[x]
Standard binary	1.1011000
One's complement	1.0100111
Two's complement	1.0101000
CSD	10101000

The one's-complement xbin1 representation of x can be determined as

$$\text{xbin1} = [1 \ \sim \text{xbin}(2:n+1)];$$

where the $\sim x$ operator determines the binary complement of x in MATLAB.

For the two's-complement representation xbin2 , we must add 1 to the least significant bit of xbin1 . This can be performed, for instance, by detecting the last 0 bit in xbin1 , which indicates the final position of the carry-over bit, such that

$$\begin{aligned}\text{xbin2} &= \text{xbin1}; \\ b &= \max(\text{find}(\text{xbin1} == 0)); \\ \text{xbin2}(b:n+1) &= \sim \text{xbin2}(b:n+1);\end{aligned}$$

We can then obtain the CSD representation xCSD from xbin2 , following the algorithm described in Section 11.2.2:

$$\begin{aligned}\text{delta} &= \text{zeros}(1,n+2); \theta = \text{zeros}(1,n+1); \text{xCSD} = \theta; \\ \text{x2aux} &= [\text{xbin2}(1) \ \text{xbin2}(0)]; \\ \text{for } i &= n:-1:1, \\ \theta(i) &= \text{xor}(\text{x2aux}(i+1), \text{x2aux}(i+2)); \\ \text{delta}(i) &= \text{and}(\sim \text{delta}(i+1), \theta(i)); \\ \text{xCSD}(i) &= (1-2*\text{x2aux}(i))*\text{delta}(i); \\ \text{end};\end{aligned}$$

Using $n = 7$ and $x = -0.6875$ with the scripts above results in the numerical representations seen in Table 11.5. \triangle

Experiment 11.2

Consider the digital filter structure depicted in Figure 11.38, whose state-space description is given by

$$\mathbf{x}(n+1) = \begin{bmatrix} (1-m_1) & -m_2 \\ m_1 & (m_2-1) \end{bmatrix} \mathbf{x}(n) + \begin{bmatrix} (2-m_1-m_2) \\ -(2-m_1-m_2) \end{bmatrix} u(n) \quad (11.221)$$

$$y(n) = [-m_1 \ -m_2] \mathbf{x}(n) + (1-m_1-m_2)u(n). \quad (11.222)$$

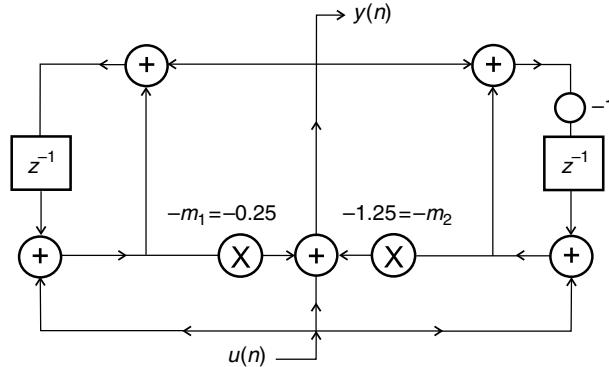


Fig. 11.38. Digital filter structure.

The corresponding transfer function is

$$H(z) = \begin{bmatrix} -m_1 & -m_2 \\ -m_1 & -m_1 \end{bmatrix} \begin{bmatrix} (z-1+m_1) & m_2 \\ -m_1 & (z-m_2+1) \end{bmatrix}^{-1} \begin{bmatrix} (2-m_1-m_2) \\ -(2-m_1-m_2) \end{bmatrix} + (1-m_1-m_2), \quad (11.223)$$

which, after some cumbersome algebraic development, becomes

$$H(z) = \frac{N(z)}{D(z)} = -\frac{(m_1 + m_2 - 1)z^2 + (m_1 - m_2)z + 1}{z^2 + (m_1 - m_2)z + (m_1 + m_2 - 1)}, \quad (11.224)$$

corresponding to an all-pass second-order block.

The transfer function from the filter input to the $-m_1$ multiplier is given by

$$F_1(z) = \frac{z^2 + 2(1 - m_2)z + 1}{z^2 + (m_1 - m_2)z + (m_1 + m_2 - 1)} \quad (11.225)$$

and the transfer function from the filter input to the $-m_2$ multiplier is

$$F_2(z) = \frac{z^2 + 2(m_1 - 1)z + 1}{z^2 + (m_1 - m_2)z + (m_1 + m_2 - 1)}. \quad (11.226)$$

Determining the L_2 norm for each scaling transfer function in a closed form is quite computationally intensive. Using MATLAB, however, this can be done numerically using a few commands, such as

```
m1 = 0.25; m2 = 1.25;
N1 = [1 2*(1-m2) 1]; N2 = [1 2*(m1-1) 1];
D = [1 (m1-m2) (m1+m2-1)];
np = 1000;
[F1,f] = freqz(N1,D,np); F1_2 = sqrt((sum(abs(F1).^2))/np);
[F2,f] = freqz(N2,D,np); F2_2 = sqrt((sum(abs(F2).^2))/np);
```

As a result, $F1_2$ and $F2_2$ are equal to 1.7332 and 1.1830 respectively. Then, we should scale the filter input by a factor of

$$\lambda = \frac{1}{\max_{i=1,2} [\|F_i(z)\|_2]} = \frac{1}{\max [1.7332, 1.1830]} = 0.5770 \quad (11.227)$$

and compensate for this by multiplying the filter output by $g = 1/\lambda = 1.7332$.

The transfer functions from the outputs of both multipliers $-m_1$ and $-m_2$ to the filter output are expressed as

$$G_1(z) = G_2(z) = H(z) \quad (11.228)$$

such that

$$\|G_1(z)\|_2 = \|G_2(z)\|_2 = 1, \quad (11.229)$$

as $H(z)$ represents an all-pass filter with unit gain. Therefore, considering the scaling operation performed as above, the output noise variance is given by

$$\sigma_y^2 = 3g^2\sigma_e^2 + \sigma_e^2 = 10.0120\sigma_e^2, \quad (11.230)$$

where the factor 3 accounts for the noise sources in the input scaling, $-m_1$, and $-m_2$ multipliers. \triangle

11.10 Finite-precision digital signal processing with MATLAB

MATLAB includes several functions that deal with decimal-integer and binary representations, such as `dec2bin`, `de2bi`, `num2bin`, and `dec2binvec` and their corresponding reverses `dec2binvec`, `bi2de`, `bin2num`, and `binvec2dec`. Also, a general quantization can be performed with the help of the `quant` command, which may operate in tandem with the `ceil`, `floor`, `round`, and `fix` operations. In addition, the MATLAB Fixedpoint Toolbox includes a complete library for fixed-point representation, which the reader is encouraged to become familiarized with, using the MATLAB `help` command.

11.11 Summary

In this chapter, some very basic concepts of implementation of digital signal processing systems were discussed.

Initially, the basic elements for such systems were described and their implementations were presented with emphasis given to the two's-complement arithmetic. In addition, the so-called distributed arithmetic was seen as a possible alternative to eliminate the use of multiplier elements in the implementation of practical digital filters.

This chapter has also presented an introduction to the finite-wordlength effects on the performance of digital signal processing systems, taking into consideration that all internal signals and parameters within these systems are quantized to a finite set of discrete values.

Starting from the basic concepts of binary number representation, the effects of quantizing the internal signals of digital filters were analyzed in some detail. A procedure to control the internal dynamic range, so that overflows are avoided, was given.

Section 11.7 introduced some tools to analyze the quantization effects on the system parameters, such as digital filter coefficients. Several forms of sensitivity measures were introduced and their merits briefly discussed. In addition, we presented a statistical method to predict the required coefficient wordlength for a filter to meet prescribed specifications.

The chapter concluded by studying granular and overflow limit cycles that arise in recursive digital filters. The emphasis was placed on procedures to eliminate these nonlinear oscillations in filters implemented with fixed-point arithmetic, which in many cases is crucial from the designer's point of view. Owing to space limitations, we could not address many useful techniques to eliminate or control the amplitude of nonlinear oscillations. An interesting example is the so-called error spectrum shaping technique (Laakso *et al.*, 1992), which is seen in detail in Section 13.2.3. This technique has also been successful in reducing roundoff noise (Diniz & Antoniou, 1985) and in eliminating limit cycles in digital filters implemented using floating-point arithmetic (Laakso *et al.*, 1994). There are also a number of research papers dealing with the analysis of the various types of limit cycle, including techniques to determine their frequencies and amplitude bounds (Munson *et al.*, 1984; Bauer & Wang, 1993).

11.12 Exercises

- 11.1 Design a circuit that determines the two's complement of a binary number in a serial fashion.
- 11.2 Show, using Equation (11.20), that, if X and Y are represented in two's-complement arithmetic, then:
 - (a) $X - Y = X + c[Y]$, where $c[Y]$ is the two's complement of Y .
 - (b) $X - Y = Y + \bar{X}$, where \bar{X} represents the number obtained by inverting all bits of X .
- 11.3 Describe an architecture for the parallel multiplier where the coefficients are represented in the two's-complement format.
- 11.4 Describe the internal circuitry of the multiplier in Figure 11.4 for 2-bit input data. Then perform the multiplication of several combinations of data, determining the internal data obtained at each step of the multiplication operation.
- 11.5 Describe an implementation of the FIR digital filter seen in Figure 4.3 using a single multiplier and a single adder multiplexed in time.
- 11.6 Describe the implementation of the FIR digital filter in Figure 4.3 using the distributed arithmetic technique. Design the architecture in detail, specifying the dimensions of the memory unit as a function of the filter order.

Table 11.6.

Coefficients of cascade realization of three second-order state-space blocks.

Coefficient	Section 1	Section 2	Section 3
a_{11}	0.8027	0.7988	0.8125
a_{12}	-0.5820	-0.5918	-0.5859
a_{21}	0.5834	0.5996	0.5859
a_{22}	0.8027	0.7988	0.8125
c_1	-0.0098	0.0469	-0.0859
c_2	-0.0273	-0.0137	0.0332
d	0.0098	0.1050	0.3594

- 11.7 Determine the content of the memory unit in a distributed arithmetic implementation of the direct-form realization of a digital filter whose coefficients are given by

$$\begin{aligned} b_0 &= b_2 = 0.078\,64 \\ b_1 &= -0.148\,58 \\ a_1 &= -1.936\,83 \\ a_2 &= 0.951\,89. \end{aligned}$$

Use 8 bits to represent all internal signals.

- 11.8 Determine the content of the memory unit in a distributed arithmetic implementation of the cascade realization of three second-order state-space blocks whose coefficients are given in Table 11.6 (scaling factor: $\lambda = 0.2832$).
- 11.9 Describe the number $-0.832\,645$ using one's-complement, two's-complement, and CSD formats.
- 11.10 Repeat Exercise 11.11.9 by constraining the wordlength to 7 bits, including the sign bit, and using rounding for quantization.
- 11.11 Describe the number $-0.000\,612\,45$ in floating-point representation, with the mantissa in two's complement.
- 11.12 Deduce the probability density function of the quantization error for the sign-magnitude and one's-complement representations. Consider the cases of rounding, truncation, and magnitude truncation.
- 11.13 Calculate the scaling factors for the radix-4 butterfly of Figure 3.16 using the L_2 and L_∞ norms.
- 11.14 Show that the L_2 norm of the transfer function

$$H(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

is

$$\|H(z)\|_2^2 = \frac{(b_1^2 + b_2^2)(1 + a_2) - 2b_1 b_2 a_1}{(1 - a_1^2 + a_2^2 + 2a_2)(1 - a_2)}.$$

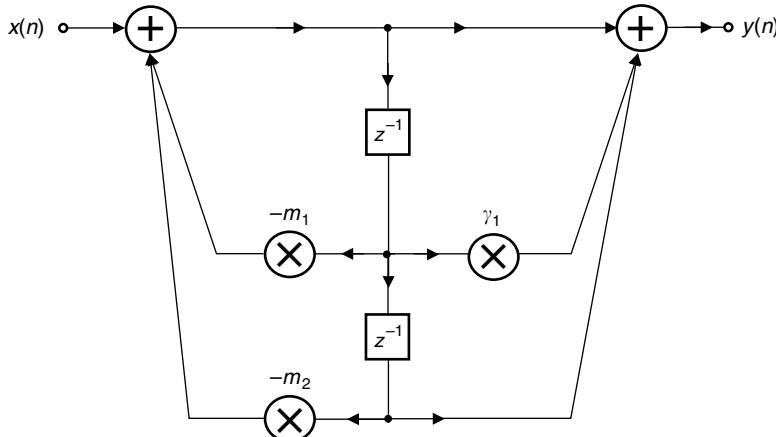


Fig. 11.39. Second-order digital filter: $m_1 = -1.933\,683$, $m_2 = 0.951\,89$, $\gamma_1 = -1.889\,37$.

11.15 Given the transfer function

$$H(z) = \frac{1 - (az^{-1})^{M+1}}{1 - az^{-1}}$$

compute the scaling factors using the L_2 and L_∞ norms assuming $|a| < 1$.

- 11.16 Calculate the scaling factor for the digital filter in Figure 11.39, using the L_2 and L_∞ norms.
- 11.17 Show that, for the Type 1 canonic direct-form realization for IIR filters (as seen in Figure 4.12), the scaling coefficient is given by

$$\lambda = \frac{1}{\left\| \frac{1}{D(z)} \right\|_p}.$$

- 11.18 Show that, for the Type 2 canonic direct-form realization for IIR filters (as depicted in Figure 4.13), the scaling coefficient is given by

$$\lambda = \frac{1}{\max\{1, \|H(z)\|_p\}}.$$

- 11.19 Calculate the relative output-noise variance in decibels for the filter of Figure 11.39 using fixed-point and floating-point arithmetics.
- 11.20 Calculate the output-noise RPSD for the filter in Figure 11.40 using fixed-point and floating-point arithmetics.
- 11.21 Derive Equations (11.133)–(11.135).
- 11.22 Plot the magnitude response of the filters designed in Exercise 4.11.12 when the coefficients are quantized to 7 bits. Compare the results with those for the original filters.
- 11.23 Calculate the maximum expected value of the transfer function deviation, given by the maximum value of the tolerance function, with a confidence factor of 95%, for

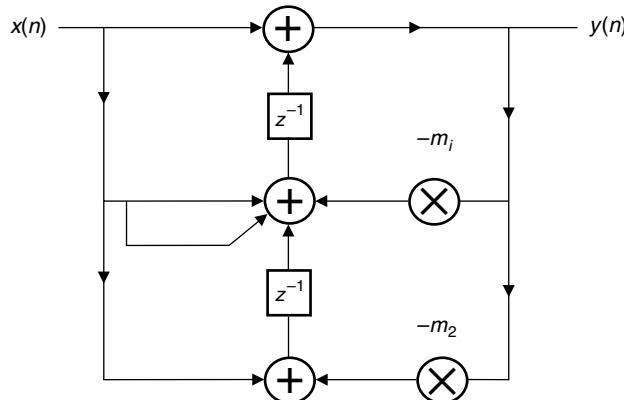


Fig. 11.40. Second-order digital filter.

the digital filter in Figure 11.39 implemented with 6 bits. Use $H(e^{j\omega}) = H_d(e^{j\omega})$ in Equation (11.151).

- 11.24 Determine the minimum number of bits a multiplier should have to keep a signal-to-noise ratio above 80 dB at its output. Consider that the type of quantization is rounding.
- 11.25 Plot the pole grid for the structure of Figure 11.30 when the coefficients are implemented in two's complement with 6 bits, including the sign bit.
- 11.26 Discuss whether granular limit cycles can be eliminated in the filter of Figure 11.40 by using magnitude truncation quantizers at the state variables.
- 11.27 Verify whether it is possible to eliminate granular and overflow limit cycles in the structure of Figure 11.17b.
- 11.28 Verify whether it is possible to eliminate granular and overflow limit cycles in the structure of Exercise 4.16.
- 11.29 Plot the overflow characteristics of simple one's-complement arithmetic.
- 11.30 Suppose there is a structure that is free from zero-input limit cycles, when employing magnitude truncation, and that it is also forced-input stable when using saturation arithmetic. Discuss whether overflow oscillations occur when the overflow quantization is removed, with the numbers represented in two's-complement arithmetic.
- 11.31 Show that Equation (11.220) is satisfied whenever the quantizer has overflow characteristics as in Figure 11.37, following the steps:
 - (i) Express inequality (11.220) as a function of $f'_i(n)$, $[f'_i(n)]_{Q_0}$, and $f_i(n)$.
 - (ii) Determine the regions of the plane $f'_i(n) \times [f'_i(n)]_{Q_0}$, as a function of $f_i(n)$, where inequality (11.220) applies.
 - (iii) Use the fact that inequality (11.220) must be valid for all $|f_i(n)| < 1$.
 - (iv) Suppose that the output of an overflow quantizer is limited to $[-1, 1]$.

12.1 Introduction

In this chapter, alternative realizations to those introduced in Chapter 5 for FIR filters are discussed.

We first present the lattice realization, highlighting its application to the design of linear-phase perfect reconstruction filter banks. Then, the polyphase structure is revisited, discussing its application in parallel processing. We also present an FFT-based realization for implementing the FIR filtering operation in the frequency domain. Such a form can be very efficient in terms of computational complexity, and is particularly suitable for off-line processing, although widely used in real-time implementations. Next, the so-called recursive running sum is described as a special recursive structure for a very particular FIR filter, which has applications in the design of FIR filters with low arithmetic complexity.

In the case of FIR filters, the main concern is to examine methods which aim at reducing the number of arithmetic operations. These methods lead to more economical realizations with reduced quantization effects. In this chapter, we also present the prefilter, the interpolation, and the frequency-response masking approaches for designing lowpass and highpass FIR filters with reduced arithmetic complexity. The frequency-response masking method can be seen as a generalization of the other two schemes, allowing the design of passbands with general widths. For bandpass and bandstop filters, the quadrature approach is also introduced.

12.2 Lattice form

Figure 12.1 shows the block diagram of a nonrecursive lattice filter of order M , which is formed by concatenating basic blocks of the form shown in Figure 12.2.

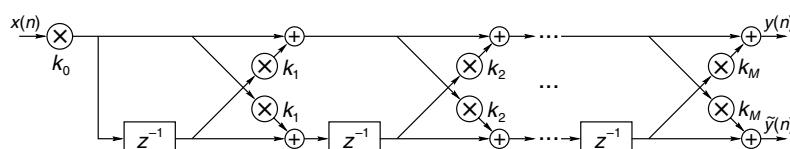


Fig. 12.1. Lattice form realization of nonrecursive digital filters.

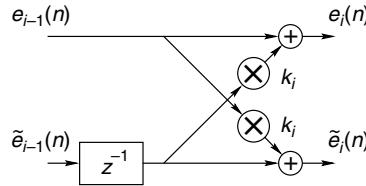


Fig. 12.2. Basic block of the nonrecursive lattice form.

To obtain a useful relation between the lattice parameters and the filter impulse response, we must analyze the recurrent relationships that appear in Figure 12.2. These equations are

$$e_i(n) = e_{i-1}(n) + k_i \tilde{e}_{i-1}(n-1) \quad (12.1)$$

$$\tilde{e}_i(n) = \tilde{e}_{i-1}(n-1) + k_i e_{i-1}(n), \quad (12.2)$$

for $i = 1, 2, \dots, M$, with $e_0(n) = \tilde{e}_0(n) = k_0 x(n)$, and $e_M(n) = y(n)$. In the frequency domain, Equations (12.1) and (12.2) become

$$\begin{bmatrix} E_i(z) \\ \tilde{E}_i(z) \end{bmatrix} = \begin{bmatrix} 1 & k_i z^{-1} \\ k_i & z^{-1} \end{bmatrix} \begin{bmatrix} E_{i-1}(z) \\ \tilde{E}_{i-1}(z) \end{bmatrix}, \quad (12.3)$$

with $E_0(z) = \tilde{E}_0(z) = k_0 X(z)$ and $E_M(z) = Y(z)$.

Defining the auxiliary polynomials

$$N_i(z) = k_0 \frac{E_i(z)}{E_0(z)} \quad (12.4)$$

$$\tilde{N}_i(z) = k_0 \frac{\tilde{E}_i(z)}{\tilde{E}_0(z)} \quad (12.5)$$

one can demonstrate by induction, using Equation (12.3), that these polynomials obey the following recurrence formulas:

$$N_i(z) = N_{i-1}(z) + k_i z^{-i} N_{i-1}(z^{-1}) \quad (12.6)$$

$$\tilde{N}_i(z) = z^{-i} N_i(z^{-1}) \quad (12.7)$$

for $i = 1, 2, \dots, M$, with $N_0(z) = \tilde{N}_0(z) = k_0$ and $N_M(z) = H(z)$. Therefore, from Equations (12.3) and (12.7), we have that

$$N_{i-1}(z) = \frac{1}{1 - k_i^2} \left(N_i(z) - k_i z^{-i} N_i(z^{-1}) \right) \quad (12.8)$$

for $i = 1, 2, \dots, M$.

Note that the recursion in Equation (12.3) implies that $N_i(z)$ has degree i . Therefore, $N_i(z)$ can be expressed as

$$N_i(z) = \sum_{m=0}^i h'_{m,i} z^{-m}. \quad (12.9)$$

Since $N_{i-1}(z)$ has degree $(i - 1)$ and $N_i(z)$ has degree i , in Equation (12.8) the highest degree coefficient of $N_i(z)$ must be canceled by the highest degree coefficient of $k_i z^{-i} N_i(z^{-1})$. This implies that

$$h'_{i,i} - k_i h'_{0,i} = 0. \quad (12.10)$$

Since, from Equation (12.9), $h'_{0,i} = N_i(\infty)$, and, from Equation (12.6), $N_i(\infty) = N_{i-1}(\infty) = \dots = N_0(\infty) = 1$, we have that Equation (12.10) is equivalent to $h'_{i,i} = k_i$. Therefore, given the filter impulse response, the k_i coefficients are determined by successively computing the polynomials $N_{i-1}(z)$ from $N_i(z)$ using Equation (12.8) and making

$$k_i = h'_{i,i} \quad (12.11)$$

for $i = M, \dots, 1, 0$.

To determine the filter impulse response $N_M(z) = H(z) = \sum_{m=0}^M h_m z^{-m}$ from the set of lattice coefficients k_i , we must first use Equation (12.6) to compute the auxiliary polynomials $N_i(z)$, starting with $N_0(z) = k_0$. The desired coefficients are then given by

$$h_m = h'_{m,M} \quad (12.12)$$

for $m = 0, 1, \dots, M$.

12.2.1 Filter banks using the lattice form

Structures similar to the lattice form are useful for the realization of critically decimated filter banks. They are referred to as lattice realizations of filter banks. As discussed in Chapter 9, using orthogonal filter banks one can only design trivial two-band filter banks with linear-phase analysis/synthesis filters. Hence, for a more general case, the solution is to employ biorthogonal filter banks. We now show one example of such structures that implement linear-phase two-band perfect reconstruction filter banks.

In order for a two-band filter bank to have linear phase, all its analysis and synthesis filters, $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$, must have linear phase. From Equation (4.15), Section 4.2.3, if we suppose that all the filters in the filter bank have the same odd order $2M + 1$, then they have linear phase if

$$\left. \begin{aligned} H_i(z) &= \pm z^{-2M-1} H_i(z^{-1}) \\ G_i(z) &= \pm z^{-2M-1} G_i(z^{-1}) \end{aligned} \right\} \quad (12.13)$$

for $i = 0, 1$.

The perfect reconstruction property holds if the analysis and synthesis polyphase matrices are related by Equation (9.118) in Section 9.5,

$$\mathbf{R}(z) = z^{-\Delta} \mathbf{E}^{-1}(z), \quad (12.14)$$

where the analysis and synthesis polyphase matrices are defined by Equations (9.3) and (9.4) as

$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \mathbf{E}(z^2) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \quad (12.15)$$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \mathbf{R}^T(z^2) \begin{bmatrix} z^{-1} \\ 1 \end{bmatrix}. \quad (12.16)$$

According to Equations (9.124) and (9.125), for perfect reconstruction the synthesis filters with $c = -1$ should satisfy

$$G_0(z) = z^{2(l-\Delta)} H_1(-z) \quad (12.17)$$

$$G_1(z) = -z^{2(l-\Delta)} H_0(-z). \quad (12.18)$$

Recall from the discussions in Section 9.5 that

$$P(z) - P(-z) = H_0(z)H_1(-z) - H_0(-z)H_1(z) = 2z^{-2l-1}. \quad (12.19)$$

Hence, $P(z)$ has linear phase, and has to be symmetric with all odd-index coefficients equal to zero, except the central coefficient of index $2l+1$, which must be equal to one. In addition, the end terms of $P(z)$ have to be of an even index, in order to cancel in $P(z) - P(-z)$. All these restrictions on $P(z)$ lead to one of the following constraints on the order of the analysis filters $H_0(z)$ and $H_1(z)$ (see details in Section 9.5):

- If both filter orders are even, they must differ by an odd multiple of 2 and the filter impulse responses must be symmetric.
- If both orders are odd, they must differ by a multiple of 4 (which includes the case where both filters are of the same order) and one filter must be symmetric and the other antisymmetric.
- If one filter order is even and the other is odd, one filter must be symmetric and the other antisymmetric, and the filter $P(z)$ degenerates into only two nonzero coefficients with all zeros on the unit circle.

Suppose, now, that we define the polyphase matrix of the analysis filters $\mathbf{E}(z)$ as having the following general form:

$$\mathbf{E}(z) = \mathcal{K}_M \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \left(\prod_{i=M}^1 \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & k_i \\ k_i & 1 \end{bmatrix} \right) \quad (12.20)$$

with

$$\mathcal{K}_M = \frac{1}{2} \prod_{i=1}^M \left(\frac{1}{1 - k_i^2} \right). \quad (12.21)$$

If we define the polyphase matrix of the synthesis filters $\mathbf{R}(z)$ as

$$\mathbf{R}(z) = \left(\prod_{i=1}^M \begin{bmatrix} 1 & -k_i \\ -k_i & 1 \end{bmatrix} \begin{bmatrix} z^{-1} & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad (12.22)$$

we then have that

$$\mathbf{R}(z) = z^{-M} \mathbf{E}^{-1}(z) \quad (12.23)$$

and perfect reconstruction is guaranteed irrespective of the values of k_i .

Also, from Equations (12.15) and (12.16), as well as Equations (12.20) and (12.22), we have that

$$\left. \begin{array}{l} H_0(z) = z^{-2M-1} H_0(z^{-1}) \\ H_1(z) = -z^{-2M-1} H_1(z^{-1}) \\ G_0(z) = z^{-2M-1} G_0(z^{-1}) \\ G_1(z) = -z^{-2M-1} G_1(z^{-1}) \end{array} \right\} \quad (12.24)$$

and linear phase is also guaranteed irrespective of the values of k_i .

Example 12.1. Prove by induction that the formulation of Equation (12.20) leads to linear-phase analysis filters.

Solution

For $M = 1$, the expression for the analysis filter bank, from Equations (12.15) and (12.20), becomes

$$\begin{aligned} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_{M=1} &= \mathcal{K}_1 \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-2} \end{bmatrix} \begin{bmatrix} 1 & \kappa_1 \\ \kappa_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \\ &= \mathcal{K}_1 \begin{bmatrix} 1 + \kappa_1 z^{-1} + \kappa_1 z^{-2} + z^{-3} \\ -1 - \kappa_1 z^{-1} + \kappa_1 z^{-2} + z^{-3} \end{bmatrix}. \end{aligned} \quad (12.25)$$

Since $H_0(z)$ and $H_1(z)$ are respectively, symmetric and antisymmetric, the filter bank is linear phase for $M = 1$. Now, in order to finish the proof, we need to show that if the filter bank is linear phase for a given M , then it is also linear phase for $M + 1$.

For a lattice structure with $(M + 1)$ stages, we have that

$$\begin{aligned} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_{M+1} &= \mathcal{K}_{M+1} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \left(\prod_{i=M+1}^1 \begin{bmatrix} 1 & k_i \\ k_i z^{-2} & z^{-2} \end{bmatrix} \right) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \\ &= \mathcal{K}_{M+1} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & k_{M+1} \\ k_{M+1} z^{-2} & z^{-2} \end{bmatrix} \left(\prod_{i=M}^1 \begin{bmatrix} 1 & k_i \\ k_i z^{-2} & z^{-2} \end{bmatrix} \right) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{1 - k_{M+1}^2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & k_{M+1} \\ k_{M+1}z^{-2} & z^{-2} \end{bmatrix} \mathcal{K}_M \left(\prod_{i=M}^1 \begin{bmatrix} 1 & k_i \\ k_iz^{-2} & z^{-2} \end{bmatrix} \right) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \\
&= \frac{1}{1 - k_{M+1}^2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & k_{M+1} \\ k_{M+1}z^{-2} & z^{-2} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_M \\
&= \frac{1}{2(1 - k_{M+1}^2)} \begin{bmatrix} (1 + k_{M+1})(1 + z^{-2}) & (1 - k_{M+1})(-1 + z^{-2}) \\ (1 + k_{M+1})(-1 + z^{-2}) & (1 - k_{M+1})(1 + z^{-2}) \end{bmatrix} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_M \\
&= \frac{1}{2} \begin{bmatrix} \frac{1 + z^{-2}}{1 - k_{M+1}} & \frac{-1 + z^{-2}}{1 + k_{M+1}} \\ \frac{-1 + z^{-2}}{1 - k_{M+1}} & \frac{1 + z^{-2}}{1 + k_{M+1}} \end{bmatrix} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_M. \tag{12.26}
\end{aligned}$$

Assuming that $[H_0(z)]_M$ is symmetric and $[H_1(z)]_M$ is antisymmetric of equal orders $(2M + 1)$ and similar coefficients, as given in Equation (12.25) for the case $M = 1$, we may write that

$$[H_0(z)]_M = z^{-2M-1} [H_0(z^{-1})]_M \tag{12.27}$$

$$[H_1(z)]_M = -z^{-2M-1} [H_1(z^{-1})]_M. \tag{12.28}$$

Therefore, for finishing the proof, we must show that $[H_0(z)]_{M+1}$ is symmetric and $[H_1(z)]_{M+1}$ is antisymmetric. Since their orders are equal to $(2(M + 1) + 1) = (2M + 3)$, we must show, according to Equations (12.27) and (12.28), that

$$[H_0(z)]_{M+1} = z^{-2M-3} [H_0(z^{-1})]_{M+1} \tag{12.29}$$

$$[H_1(z)]_{M+1} = -z^{-2M-3} [H_1(z^{-1})]_{M+1}. \tag{12.30}$$

From Equation (12.26) we have that

$$\begin{aligned}
z^{-2M-3} \begin{bmatrix} H_0(z^{-1}) \\ H_1(z^{-1}) \end{bmatrix}_{M+1} &= \frac{z^{-2M-3}}{2} \begin{bmatrix} \frac{1 + z^2}{1 - k_{M+1}} & \frac{-1 + z^2}{1 + k_{M+1}} \\ \frac{-1 + z^2}{1 - k_{M+1}} & \frac{1 + z^2}{1 + k_{M+1}} \end{bmatrix} \begin{bmatrix} H_0(z^{-1}) \\ H_1(z^{-1}) \end{bmatrix}_M \\
&\tag{12.31}
\end{aligned}$$

Substituting $[H_0(z^{-1})]_M$ and $[H_1(z^{-1})]_M$ from Equations (12.27) and (12.28) in the above equation we get

$$\begin{aligned}
z^{-2M-3} \begin{bmatrix} H_0(z^{-1}) \\ H_1(z^{-1}) \end{bmatrix}_{M+1} &= \frac{z^{-2M-3}}{2} \begin{bmatrix} \frac{1 + z^2}{1 - k_{M+1}} & \frac{-1 + z^2}{1 + k_{M+1}} \\ \frac{-1 + z^2}{1 - k_{M+1}} & \frac{1 + z^2}{1 + k_{M+1}} \end{bmatrix} z^{2M+1} \begin{bmatrix} H_0(z) \\ -H_1(z) \end{bmatrix}_M
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2} \begin{bmatrix} \frac{1+z^{-2}}{1-k_{M+1}} & \frac{1-z^{-2}}{1+k_{M+1}} \\ \frac{1-z^{-2}}{1-k_{M+1}} & \frac{1+z^{-2}}{1+k_{M+1}} \end{bmatrix} \begin{bmatrix} H_0(z) \\ -H_1(z) \end{bmatrix}_M \\
&= \frac{1}{2} \begin{bmatrix} \frac{1+z^{-2}}{1-k_{M+1}} & \frac{-1+z^{-2}}{1+k_{M+1}} \\ -\frac{-1+z^{-2}}{1-k_{M+1}} & \frac{1+z^{-2}}{1+k_{M+1}} \end{bmatrix} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}_M \\
&= \begin{bmatrix} H_0(z) \\ -H_1(z) \end{bmatrix}_{M+1}. \tag{12.32}
\end{aligned}$$

where the last step derives from Equation (12.26).

Comparing the above equation with Equations (12.29) and (12.30) we see that the lowpass and highpass filters for $(M + 1)$ stages are also symmetric and antisymmetric, which completes the proof by induction. \triangle

The realizations of the analysis and synthesis filters are shown in Figures 12.3a and 12.3b, respectively, for the same-order case, as developed in Example 12.1.

As seen above, one important property of such realizations is that both perfect reconstruction and linear phase are guaranteed irrespective of the values of k_i .¹ They are often referred to as having structurally induced perfect reconstruction and linear phase. Therefore, one possible design strategy for such filter banks is to carry out a multivariable optimization on k_i , for $i = 1, 2, \dots, M$, using a chosen objective function.

For example, one could minimize the L_2 norm of the deviation of both the lowpass filter $H_0(z)$ and the highpass filter $H_1(z)$ in both their passbands and stopbands using the

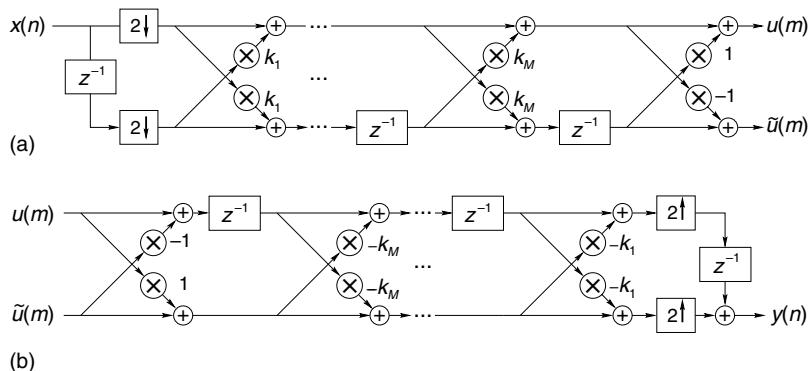


Fig. 12.3.

Lattice form realization of a linear-phase filter bank: (a) analysis filters; (b) synthesis filters.

¹ There are rare cases of perfect reconstruction FIR filter banks where the corresponding lattices cannot be synthesized as some coefficient k_i may be equal to 1.

following objective function:

$$\begin{aligned}\xi(k_1, k_2, \dots, k_M) = & \int_0^{\omega_p} \left(1 - |H_0(e^{j\omega})|\right)^2 d\omega + \int_{\omega_r}^{\pi} \left|H_0(e^{j\omega})\right|^2 d\omega \\ & + \int_{\omega_r}^{\pi} \left(1 - |H_1(e^{j\omega})|\right)^2 d\omega + \int_0^{\omega_p} \left|H_1(e^{j\omega})\right|^2 d\omega\end{aligned}\quad (12.33)$$

which corresponds essentially to a least-squares error function.

Example 12.2. (a) Implement the transfer function below using a lattice structure:

$$H_1(z) = z^{-5} + 2z^{-4} + 0.5z^{-3} - 0.5z^{-2} - 2z^{-1} - 1. \quad (12.34)$$

(b) Determine $H_0(z)$ and the synthesis filters obtained.

Solution

(a) Using $M = 2$ in Equation (12.20), the polyphase matrix of the analysis filter bank is then given by

$$\begin{aligned}\mathbf{E}(z) &= \mathcal{K}_2 \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & \kappa_2 \\ \kappa_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & \kappa_1 \\ \kappa_1 & 1 \end{bmatrix} \\ &= \mathcal{K}_2 \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & \kappa_2 \\ \kappa_2 z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & \kappa_1 \\ \kappa_1 z^{-1} & 1 \end{bmatrix}\end{aligned}\quad (12.35)$$

such that, from Equation (12.15), the highpass filter becomes

$$H_1(z) = \mathcal{K}_2 \left[-1 - \kappa_1 z^{-1} + \kappa_2 (1 - \kappa_1) z^{-2} - \kappa_2 (1 - \kappa_1) z^{-3} + \kappa_1 z^{-4} + z^{-5} \right]. \quad (12.36)$$

Equating this expression to the given transfer function, one gets

$$\left. \begin{array}{l} \kappa_1 = 2 \\ \kappa_2 = 0.5 \end{array} \right\}. \quad (12.37)$$

Note that with this lattice structure the resulting filter banks are equal to the desired $H_0(z)$ and $H_1(z)$ up to a constant, since the value of \mathcal{K}_2 must be given by Equation (12.21), which, in this case, is

$$\mathcal{K}_2 = \left(\frac{1}{2}\right) \left(\frac{1}{1 - 2^2}\right) \left[\frac{1}{1 - (0.5)^2}\right] = -\frac{2}{9}. \quad (12.38)$$

Therefore, the filter $H_1(z)$ becomes

$$H_1(z) = -\frac{2}{9} \left(-1 - 2z^{-1} - 0.5z^{-2} - 0.5z^{-3} + 2z^{-4} + z^{-5} \right). \quad (12.39)$$

The polyphase components of the highpass filter of the analysis filter bank are such that

$$\left. \begin{aligned} E_{10}(z) &= \mathcal{K}_2 [-1 - \kappa_2(\kappa_1 - 1)z^{-1} + \kappa_1 z^{-2}] = -\frac{2}{9} (-1 - 0.5z^{-1} + 2z^{-2}) \\ E_{11}(z) &= \mathcal{K}_2 [-\kappa_1 - \kappa_2(1 - \kappa_1)z^{-1} + z^{-2}] = -\frac{2}{9} (-2 + 0.5z^{-1} + z^{-2}) \end{aligned} \right\}, \quad (12.40)$$

and for the lowpass filter, we have that

$$\left. \begin{aligned} E_{00}(z) &= \mathcal{K}_2 [1 + \kappa_2(1 + \kappa_1)z^{-1} + \kappa_1 z^{-2}] = -\frac{2}{9} (1 + 1.5z^{-1} + 2z^{-2}) \\ E_{01}(z) &= \mathcal{K}_2 [\kappa_1 + \kappa_2(1 + \kappa_1)z^{-1} + z^{-2}] = -\frac{2}{9} (2 + 1.5z^{-1} + z^{-2}) \end{aligned} \right\}. \quad (12.41)$$

(b) The lowpass filter of the analysis filter bank has the following expression:

$$\begin{aligned} H_0(z) &= E_{00}(z^2) + z^{-1}E_{01}(z^2) \\ &= -\frac{2}{9} (1 + 2z^{-1} + 1.5z^{-2} + 1.5z^{-3} + 2z^{-4} + z^{-5}). \end{aligned} \quad (12.42)$$

The determinant of the matrix $\mathbf{E}(z)$ has the following expression:

$$\begin{aligned} \det[\mathbf{E}(z)] &= E_{00}(z)E_{11}(z) - E_{10}(z)E_{01}(z) \\ &= \left(\frac{2}{9}\right)^2 \left[(1 + 1.5z^{-1} + 2z^{-2})(-2 + 0.5z^{-1} + z^{-2}) \right. \\ &\quad \left. - (-1 - 0.5z^{-1} + 2z^{-2})(2 + 1.5z^{-1} + z^{-2}) \right] \\ &= -\frac{2}{9}z^{-2}. \end{aligned} \quad (12.43)$$

As a result, from Equation (12.23), the polyphase matrix of the synthesis filter is given by

$$\begin{aligned} \mathbf{R}(z) &= \frac{z^{-2}}{\det[\mathbf{E}(z)]} \begin{bmatrix} E_{11}(z) & -E_{01}(z) \\ -E_{10}(z) & E_{00}(z) \end{bmatrix} \\ &= \begin{bmatrix} (-2 + 0.5z^{-1} + z^{-2}) & (-2 - 1.5z^{-1} - z^{-2}) \\ (1 + 0.5z^{-1} - 2z^{-2}) & (1 + 1.5z^{-1} + 2z^{-2}) \end{bmatrix} \end{aligned} \quad (12.44)$$

and the synthesis filters, from Equation (12.16), are

$$\left. \begin{aligned} G_0(z) &= 1 - 2z^{-1} + 0.5z^{-2} + 0.5z^{-3} - 2z^{-4} + z^{-5} \\ G_1(z) &= 1 - 2z^{-1} + 1.5z^{-2} - 1.5z^{-3} + 2z^{-4} - z^{-5} \end{aligned} \right\}. \quad (12.45)$$

△

12.3 Polyphase form

A nonrecursive transfer function of the form

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}, \quad (12.46)$$

when $N = K\bar{N}$ (as seen in Section 8.8), can also be expressed as

$$\begin{aligned} H(z) &= \sum_{n=0}^{\bar{N}-1} h(Kn)z^{-Kn} + \sum_{n=0}^{\bar{N}-1} h(Kn+1)z^{-Kn-1} + \dots \\ &\quad + \sum_{n=0}^{\bar{N}-1} h(Kn+K-1)z^{-Kn-K+1} \\ &= \sum_{n=0}^{\bar{N}-1} h(Kn)z^{-Kn} + z^{-1} \sum_{n=0}^{\bar{N}-1} h(Kn+1)z^{-Kn} + \dots \\ &\quad + z^{-K+1} \sum_{n=0}^{\bar{N}-1} h(Kn+K-1)z^{-Kn}. \end{aligned} \quad (12.47)$$

This last form of writing $H(z)$ can be directly mapped onto the realization shown in Figure 12.4, where each $H_i(z)$ is given by

$$H_i(z^K) = \sum_{n=0}^{\bar{N}-1} h(Kn+i)z^{-Kn} \quad (12.48)$$

for $i = 0, 1, \dots, (K-1)$. Such a realization is referred to as the polyphase realization and it finds a large range of applications in the study of multirate systems and filter banks, as seen in Chapter 9.

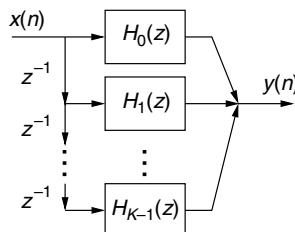


Fig. 12.4. Block diagram of realization of Equation (12.47).

12.4 Frequency-domain form

The output of an FIR filter corresponds to the linear convolution of the input signal with the finite-length impulse response of the filter $h(n)$. Therefore, from Section 3.4 we have that, if the input signal $x(n)$ of a nonrecursive digital filter is known for all n , and null for $n < 0$ and $n > L$, an alternative approach to computing the output $y(n)$ can be derived using the FFT. If the filter length is N , thereby completing these sequences with the necessary number of zeros (zero-padding procedure) and determining the resulting $(N + L)$ -element FFTs of $h(n)$, $x(n)$, and $y(n)$ we have that

$$\text{FFT}\{y(n)\} = \text{FFT}\{h(n)\}\text{FFT}\{x(n)\} \quad (12.49)$$

and then

$$y(n) = \text{FFT}^{-1}\{\text{FFT}\{h(n)\}\text{FFT}\{x(n)\}\}. \quad (12.50)$$

Using this approach, we are able to compute the entire sequence $y(n)$ with a number of arithmetic operations per output sample proportional to $\log_2(L + N)$. In the case of direct evaluation, the number of operations per output sample is of the order of L . Clearly, for large values of L and not too large values of N , the FFT method is the most efficient one.

In the above approach, the entire input sequence must be available to allow one to compute the output signal. In this case, if the input is extremely long, the complete computation of $y(n)$ can result in a long computational delay, which is undesirable in several applications. For such cases, the input signal can be sectioned and each data block processed separately using the so-called overlap-and-save and overlap-and-add methods, as described in Chapter 3.

12.5 Recursive running sum form

The direct realization of the transfer function

$$H(z) = \sum_{i=0}^M z^{-i}, \quad (12.51)$$

where all the multiplier coefficients are equal to one, requires a large number of additions for large M . An alternative way to implement such a transfer function results from interpreting it as the sum of the terms of a geometric series. This yields

$$H(z) = \frac{1 - z^{-M-1}}{1 - z^{-1}}. \quad (12.52)$$

This equation leads to the realization in Figure 12.5, widely known as the recursive running sum (RRS) (Adams & Willson, 1983).

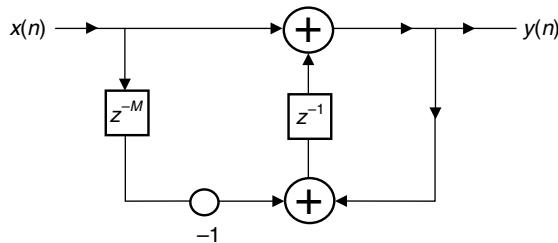


Fig. 12.5. Block diagram of the RRS realization.

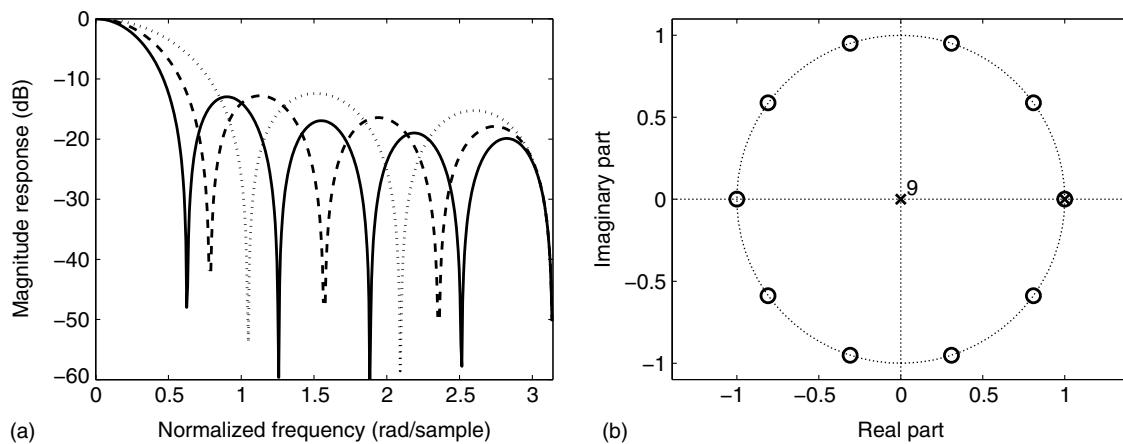


Fig. 12.6. RRS characteristics: (a) magnitude response for $M = 5$ (dotted line), $M = 7$ (dashed line), and $M = 9$ (solid line); (b) pole-zero constellation for $M = 9$.

The RRS corresponds to a very simple lowpass filter, comprising $(M + 1)$ delays and only two additions per output sample. The RRS bandwidth can be controlled by appropriately choosing the value of M , as illustrated in Example 12.3.

Example 12.3. Determine the magnitude responses and pole-zero constellations for the RRS blocks with $M = 5, 7, 9$.

Solution

The RRS has a pole at $z = 1$ that is canceled by a zero at the same position, leading to an FIR filter with a recursive realization. By examining the numerator polynomial of the RRS filter its zeros are equally spaced on the unit circle, placed at $z = e^{j2\pi/(M+1)}$. The magnitude responses of the RRS for $M = 5, 7, 9$ are depicted in Figure 12.6a, whereas the pole-zero constellation when $M = 9$ is seen in Figure 12.6b. \triangle

The RRS DC gain is equal to $(M + 1)$. To compensate for that, a scaling factor of $1/(M + 1)$ should be employed at the filter input, which generates an output roundoff noise with variance of about $[(M + 1)q]^2/12$. To reduce this effect, one may eliminate the input scaling and perform the RRS internal computations with higher dynamic range, by

increasing the internal binary wordlength accordingly. In this case, the output-noise variance is reduced to $q^2/12$, since the signal is quantized only at the output. To guarantee this, the number of extra bits must be the smallest integer larger than or equal to $\log_2(M + 1)$.

12.6 Modified-sinc filter

The RRS filter is certainly one of the most widely used lowpass filters with very efficient implementation. The main drawback of the RRS filter is its low stopband attenuation, which cannot be increased in a straightforward manner. A simple and yet efficient extension of the RRS is the modified-sinc filter proposed by Le Presti (2000) and further discussed by Lyons (2007). The modified-sinc filter has the following general transfer function:

$$\begin{aligned} H(z) &= \frac{1}{(M+1)^3} \left[\frac{1 - bz^{-(M+1)} + bz^{-2(M+1)} - z^{-3(M+1)}}{1 - az^{-1} + az^{-2} - z^{-3}} \right] \\ &= \frac{1}{(M+1)^3} \left[\frac{1 - 2 \cos(M+1)\omega_0 z^{-(M+1)} + z^{-2(M+1)}}{1 - 2 \cos\omega_0 z^{-1} + z^{-2}} \right] \left[\frac{1 - z^{-(M+1)}}{1 - z^{-1}} \right], \end{aligned} \quad (12.53)$$

where $a = 1 + 2 \cos\omega_0$ and $b = 1 + 2 \cos(M+1)\omega_0$. Figure 12.7 shows a canonic structure for the modified-sinc filter; that is, utilizing the minimum number of multipliers, delays,

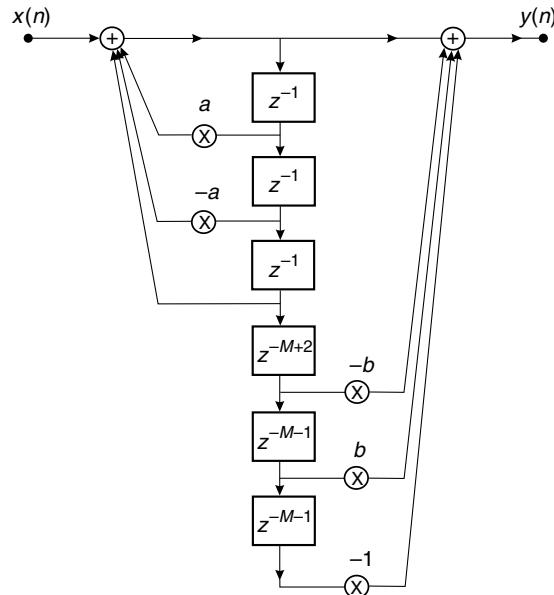


Fig. 12.7. Canonic realization of the modified-sinc filter.

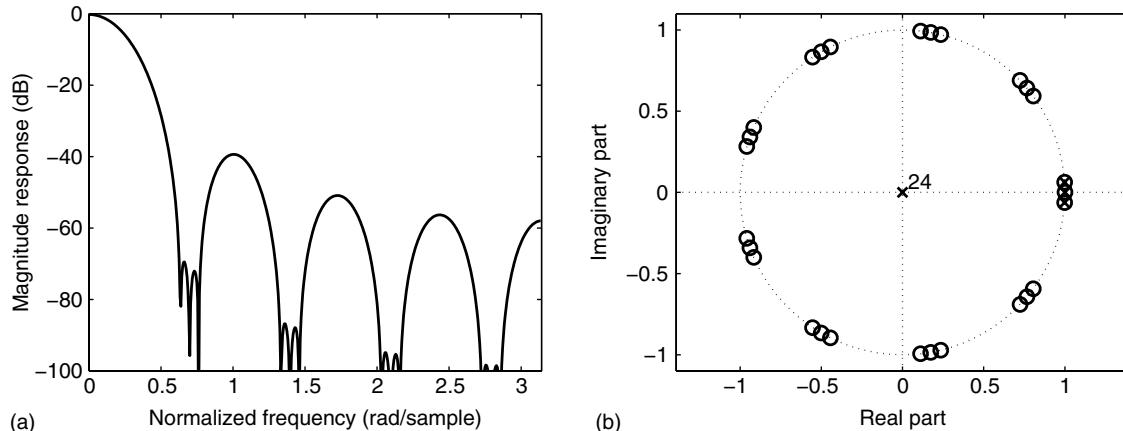


Fig. 12.8. Characterization of modified-sinc filter for $M = 8$ and $\omega_0 = \pi/50$: (a) magnitude response; (b) pole-zero constellation.

and adders. The modified-sinc filter places equally spaced triplets of zeros on the unit circle. The first triplet is located around DC (one zero at DC and two at $\pm\omega_0$), and is canceled by the filter poles, generating an improved lowpass filter, compared with the RRS filter, as long as $\omega_0 < \pi/(M + 1)$. Figure 12.8 depicts the frequency response of the modified-sinc filter for $M = 8$ and $\omega_0 = \pi/50$.

12.7 Realizations with reduced number of arithmetic operations

The main drawback of FIR filters is the large number of arithmetic operations required to satisfy practical specifications. However, especially in the case of filters with narrow passbands or transition bands, there is a correlation among the values of filter multiplier coefficients. This fact can be exploited in order to reduce the number of arithmetic operations required to satisfy such specifications (Van Gerwen *et al.*, 1975; Adams & Willson, 1983, 1984; Benvenuto *et al.*, 1984; Neüvo *et al.*, 1984; Lim, 1986; Saramäki *et al.*, 1988). This section presents some of the most widely used methods for designing FIR filters with reduced computational complexity.

12.7.1 Prefilter approach

The main idea of the prefilter method consists of generating a simple FIR filter, with reduced multiplication and addition counts, whose frequency response approximates the desired response as far as possible. Then this simple filter is cascaded with an amplitude equalizer, designed so that the overall filter satisfies the prescribed specifications (Adams

& Willson, 1983). The reduction in the computational complexity results from the fact that the prefilter greatly relieves the specifications for the equalizer. This happens because the equalizer has wider transition bands to approximate, thus requiring a lower order.

Several prefilter structures are given in the literature (Adams & Willson, 1984), and choosing the best one for a given specification is not an easy task. A very simple lowpass prefilter is the RRS filter, seen in Section 12.5. From Equation (12.52), the frequency response of the M th-order RRS filter is given by

$$H(e^{j\omega}) = \frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}. \quad (12.54)$$

This indicates that the RRS frequency response has several ripples at the stopband with decreasing magnitudes as ω approaches π . The first zero in the RRS frequency response occurs at

$$\omega_{z1} = \frac{2\pi}{M+1}. \quad (12.55)$$

Using the RRS as a prefilter, this first zero must be placed above and as close as possible to the stopband edge ω_r . In order to achieve this, the RRS order M must be such that

$$M = \left\lfloor \frac{2\pi}{\omega_r} - 1 \right\rfloor, \quad (12.56)$$

where $\lfloor x \rfloor$ represents the largest integer less than or equal to x .

More efficient prefilters can be generated by cascading several RRS sections. For example, if we cascade two prefilters, so that the first one satisfies Equation (12.56) and the second is designed to cancel the secondary ripples of the first, then we could expect a higher stopband attenuation for the resulting prefilter. This would relieve the specifications for the equalizer even further. In particular, the first stopband ripple belonging to the first RRS must be attenuated by the second RRS, without introducing zeros in the passband. Although the design of prefilters by cascading several RRS sections is always possible, practice has shown that there is little to gain in cascading more than three sections. The modified-sinc filter is also a very efficient prefilter.

We show now that the Chebyshev (minimax) approach for designing optimal FIR filters presented in Chapter 5 can be adapted to design the equalizer, by modifying the error function definition, in the following way. The response obtained by cascading the prefilter with the equalizer is given by

$$H(z) = H_p(z)H_e(z), \quad (12.57)$$

where $H_p(z)$ is the prefilter transfer function and only the coefficients of $H_e(z)$ are to be optimized.

The error function from Equation (5.127) can then be rewritten as

$$\begin{aligned}|E(\omega)| &= \left| W(\omega) \left(D(\omega) - H_p(e^{j\omega})H_e(e^{j\omega}) \right) \right| \\&= \left| W(\omega)H_p(e^{j\omega}) \left(\frac{D(\omega)}{H_p(e^{j\omega})} - H_e(e^{j\omega}) \right) \right| \\&= \left| W'(\omega) \left(D'(\omega) - H_e(e^{j\omega}) \right) \right|,\end{aligned}\quad (12.58)$$

where

$$D'(\omega) = \begin{cases} \frac{1}{H_p(e^{j\omega})}, & \omega \in \text{passbands} \\ 0, & \omega \in \text{stopbands} \end{cases} \quad (12.59)$$

$$W'(\omega) = \begin{cases} |H_p(e^{j\omega})|, & \omega \in \text{passbands} \\ \frac{\delta_p}{\delta_r} |H_p(e^{j\omega})|, & \omega \in \text{stopbands}. \end{cases} \quad (12.60)$$

It is worth mentioning that $H_p(e^{j\omega})$ often has zeros at some frequencies, which causes problems to the optimization algorithm. One way to circumvent this is to replace $|H_p(e^{j\omega})|$ in the neighborhoods of its zeros by a small number, such as 10^{-6} .

Example 12.4. Design a highpass filter using the standard minimax and the prefilter methods satisfying the following specifications:

$$\left. \begin{array}{l} A_r = 40 \text{ dB} \\ \Omega_r = 6600 \text{ Hz} \\ \Omega_p = 7200 \text{ Hz} \\ \Omega_s = 16000 \text{ Hz} \end{array} \right\}. \quad (12.61)$$

Solution

The prefilter approach described above applies only to narrowband lowpass filters. However, it requires only a slight modification in order to be applied to narrowband highpass filter design. The modification consists of designing the lowpass filter approximating $D(\pi - \omega)$ and then replacing z^{-1} by $-z^{-1}$ in the realization. Therefore, the lowpass specifications are

$$\Omega'_p = \frac{\Omega_s}{2} - \Omega_p = 8000 - 7200 = 800 \text{ Hz} \quad (12.62)$$

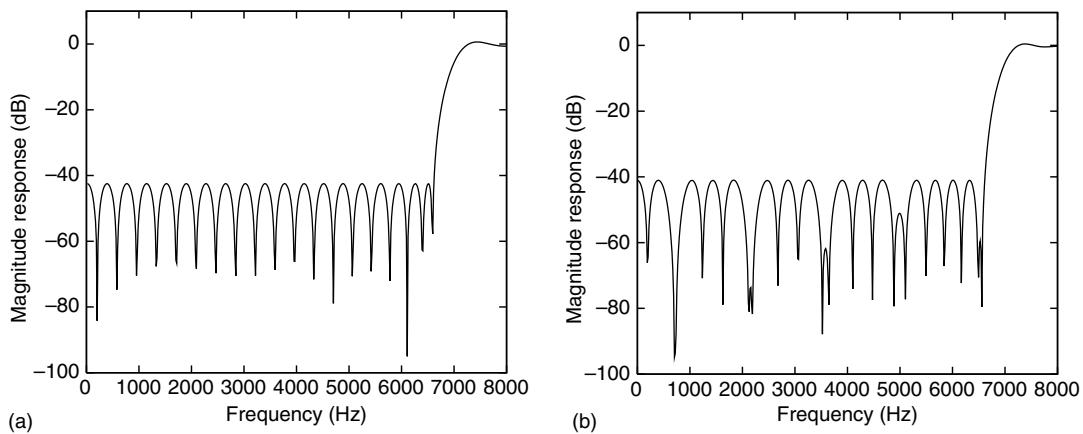
$$\Omega'_r = \frac{\Omega_s}{2} - \Omega_r = 8000 - 6600 = 1400 \text{ Hz}. \quad (12.63)$$

Using the minimax approach, the resulting direct-form filter has order 42, thus requiring 22 multiplications per output sample. Using the prefilter approach, with an RRS of order 10, the resulting equalizer has order 34, requiring only 18 multiplications per output sample. Only half of the equalizer coefficients are shown in Table 12.1, as the other coefficients can be obtained as $h(34 - n) = h(n)$.

The magnitude responses of the direct-form and the prefilter-equalizer filters are depicted in Figure 12.9. \triangle

Table 12.1. Equalizer coefficients $h(0)$ to $h(17)$.

$h(0) = -5.7525\text{E}-03$	$h(6) = 3.9039\text{E}-03$	$h(12) = -2.9552\text{E}-03$
$h(1) = 1.4791\text{E}-04$	$h(7) = -5.3685\text{E}-03$	$h(13) = 7.1024\text{E}-03$
$h(2) = -1.4058\text{E}-03$	$h(8) = 6.1928\text{E}-03$	$h(14) = -1.1463\text{E}-02$
$h(3) = 6.0819\text{E}-04$	$h(9) = -5.9842\text{E}-03$	$h(15) = 1.5271\text{E}-02$
$h(4) = 6.3692\text{E}-04$	$h(10) = 4.4243\text{E}-03$	$h(16) = -1.7853\text{E}-02$
$h(5) = -2.2099\text{E}-03$	$h(11) = 9.1634\text{E}-04$	$h(17) = 1.8756\text{E}-02$

**Fig. 12.9.** Magnitude responses: (a) direct-form minimax approach; (b) prefilter approach.

With the prefilter approach we can also design bandpass filters centered at $\omega_0 = \pi/2$ and with band edges at $((\pi/2) - (\omega_p/2))$, $((\pi/2) + (\omega_p/2))$, $((\pi/2) - (\omega_r/2))$, and $((\pi/2) + (\omega_r/2))$. This can be done by noting that such bandpass filters can be obtained from a lowpass filter with band edges at ω_p and ω_r , by applying the transformation $z^{-1} \rightarrow -z^{-2}$.

There are also generalizations of the prefilter approach that allow the design of narrow bandpass filters with central frequency away from $\pi/2$, as well as narrow stopband filters (Neüvo *et al.*, 1987; Cabezas & Diniz, 1990).

Owing to the reduced number of multipliers, filters designed using the prefilter method tend to generate less roundoff noise at the output than minimax filters implemented in direct form. Their sensitivity to coefficient quantization is also reduced when compared with direct-form minimax designs, as shown in Adams & Willson (1983).

12.7.2 Interpolation approach

FIR filters with narrow passband and transition bands tend to have adjacent multiplier coefficients, representing their impulse responses, with very close magnitude. This means there is a correlation among them that could be exploited to reduce computational complexity. Indeed, we could think of removing some samples of the impulse response, by

replacing them with zeros, and approximating their values by interpolating the remaining nonzero samples. That is, using the terminology of Chapter 8, we would decimate and then interpolate the filter coefficients. This is the main idea behind the interpolation approach.

Consider an initial filter with frequency and impulse responses given by $H_i(e^{j\omega})$ and $h_i(n)$ respectively. If $h_i(n)$ is interpolated by L (see Equation (8.17)), then $(L - 1)$ null samples are inserted after each sample of $h_i(n)$, and the resulting sequence $h'_i(n)$ is given by

$$h'_i(n) = \begin{cases} h_i\left(\frac{n}{L}\right), & \text{for } n = kL, \text{ with } k = 0, 1, 2, \dots \\ 0, & \text{for } n \neq kL \end{cases}. \quad (12.64)$$

The corresponding frequency response, $H'_i(e^{j\omega})$, is periodic with period $2\pi/L$. For example, Figure 12.10 illustrates the form of $H'_i(e^{j\omega})$ generated from a lowpass filter with frequency response $H_i(e^{j\omega})$, using $L = 3$.

The filter with frequency response $H'_i(e^{j\omega})$, commonly referred to as the interpolated filter, is then connected in cascade with an interpolator $G(e^{j\omega})$, resulting in a transfer function of the form

$$H(z) = H'_i(z)G(z). \quad (12.65)$$

The function of the interpolator is to eliminate undesirable bands of $H'_i(z)$ (see Figure 12.10), while leaving its lowest frequency band unaffected.

As can be observed, the initial filter has passband and stopband which are L times larger than those of the interpolated filter (in Figure 12.10, $L = 3$). As a consequence, the number of multiplications in the initial filter tends to be approximately L times smaller than the number of multiplications of a filter directly designed with the minimax approach to satisfy the narrowband specifications. An intuitive explanation for this is that a filter with larger passbands, stopbands, and transition bands is easier to implement than one with narrow bands. Indeed, this can be verified by analyzing the formula in Exercise 5.25, which predicts the order of the minimax FIR filter required to satisfy a given specification.

For lowpass filters, the maximum value of L such that the initial filter satisfies the specifications in the passband and in the stopband is given by

$$L_{\max} = \left\lfloor \frac{\pi}{\omega_r} \right\rfloor, \quad (12.66)$$

where ω_r is the lowest rejection band frequency of the desired filter. This value for L assures that $\omega_{r_i} < \pi$.

For highpass filters, L_{\max} is given by

$$L_{\max} = \left\lfloor \frac{\pi}{\pi - \omega_r} \right\rfloor, \quad (12.67)$$

whereas L_{\max} for bandpass filters, is the largest L such that $\pi k/L \leq \omega_{r_1} < \omega_{r_2} \leq \pi(k + 1)/L$, for some k . In practice, L is chosen smaller than L_{\max} in order to relieve the interpolator specifications.

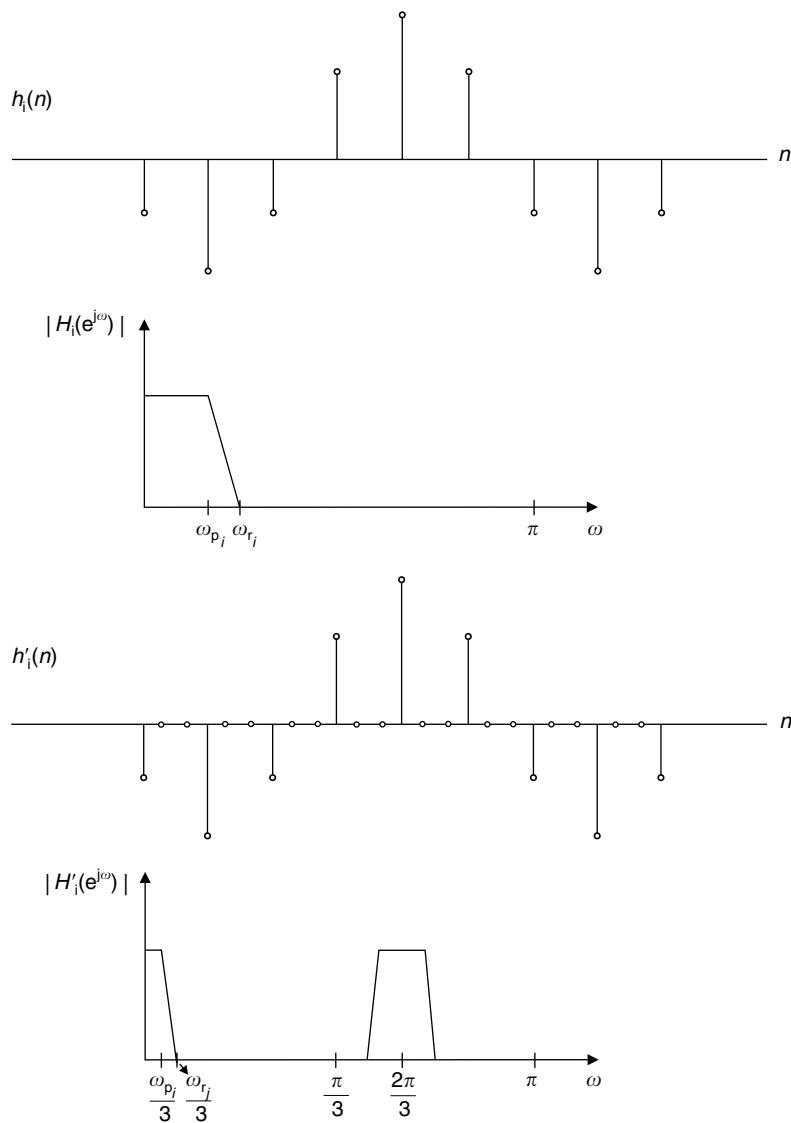
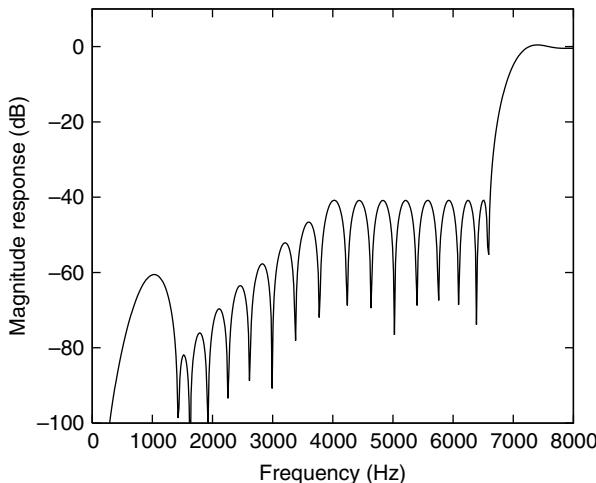


Fig. 12.10. Effects of inserting $(L - 1) = 2$ zeros in a discrete-time impulse response.

Naturally, to achieve reduction in the computational requirements of the final filter, the interpolator must be as simple as possible, not requiring too many multiplications. For instance, the interpolator can be designed as a cascade of subsections, in which each subsection places zeros on an undesirable passband. For the example in Figure 12.10, if we wish to design a lowpass filter, $G(e^{j\omega})$ should have zeros at $e^{\pm j2\pi/3}$. Alternatively, we can use the minimax method to design the interpolator, with the passband of $G(e^{j\omega})$ coinciding with the specified passband, and with the stopbands located in the frequency ranges of the undesired passband replicas of the interpolated filter (Saramäki *et al.*, 1988).

Table 12.2.Initial filter coefficients $h(0)$ to $h(10)$.

$h(0) = 1.0703E-03$	$h(4) = -2.8131E-03$	$h(8) = 9.5809E-03$
$h(1) = 7.3552E-04$	$h(5) = -3.3483E-03$	$h(9) = 1.4768E-02$
$h(2) = 3.9828E-04$	$h(6) = -1.2690E-03$	$h(10) = 1.6863E-02$
$h(3) = -1.2771E-03$	$h(7) = 3.3882E-03$	

**Fig. 12.11.** Response of the interpolated model filter in cascade with the interpolator.

Once the value of L and the interpolator are chosen, the interpolated filter can be designed such that

$$\left. \begin{aligned} (1 - \delta_p) &\leq \left| H_i(e^{j\omega})G(e^{j\omega/L}) \right| \leq (1 + \delta_p), & \text{for } \omega \in [0, L\omega_p] \\ \left| H_i(e^{j\omega})G(e^{j\omega/L}) \right| &\leq \delta_r, & \text{for } \omega \in [L\omega_r, \pi] \end{aligned} \right\}, \quad (12.68)$$

where the minimax method to design optimal FIR filters can be directly used.

Example 12.5. Design the filter specified in Example 12.1 using the interpolation method.

Solution

Using $L = 2$, we obtain the initial filter of order 20, thus requiring 11 multiplications per output sample, whose coefficients are listed in Table 12.2. The required interpolator is given by $G(z) = (1 - z^{-1})^4$, and the resulting magnitude response of the cascade of the initial filter and the interpolator is seen in Figure 12.11. \triangle

It is worth observing that the prefilter and the interpolation methods were initially described as effective methods to design narrowband filters of the types lowpass, high-pass, and bandpass. However, we can also design both wideband and narrow stopband filters with the interpolation method by noting they can be obtained from a narrowband

filter $H(z)$ which is complementary to the desired filter, using

$$H_{FL}(z) = z^{-M/2} - H(z), \quad (12.69)$$

where M is the even order of $H(z)$.

12.7.3 Frequency-response masking approach

Another interesting application of interpolation appears in the design of wideband sharp cutoff filters using the so-called frequency-response masking approach (Lim, 1986). A brief introduction to it was given in Section 8.10.2. Such an approach makes use of the concept of complementary filters, which constitute a pair of filters, $H_a(z)$ and $H_c(z)$, whose frequency responses add to a constant delay; that is:

$$\left| H_a(e^{j\omega}) + H_c(e^{j\omega}) \right| = 1. \quad (12.70)$$

If $H_a(z)$ is a linear-phase FIR filter of even order M , its frequency response can be written as

$$H_a(e^{j\omega}) = e^{-j(M/2)\omega} A(\omega), \quad (12.71)$$

where $A(\omega)$ is a trigonometric function of ω , as given in Section 5.6, Equation (5.126). Therefore, the frequency response of the complementary filter must be of the form

$$H_c(e^{j\omega}) = e^{-j(M/2)\omega} (1 - A(\omega)) \quad (12.72)$$

and the corresponding transfer functions are such that

$$H_a(z) + H_c(z) = z^{-M/2}. \quad (12.73)$$

Hence, given the realization of $H_a(z)$, its complementary filter $H_c(z)$ can easily be implemented by subtracting the output of $H_a(z)$ from the $(M/2)$ th delayed version of its input, as seen in Figure 12.12. For an efficient implementation of both filters, the tapped delay line of $H_a(z)$ can be used to form $H_c(z)$, as indicated in Figure 12.13, in which either the symmetry or antisymmetry of $H_a(z)$ is exploited, as we are assuming that this filter has linear phase.

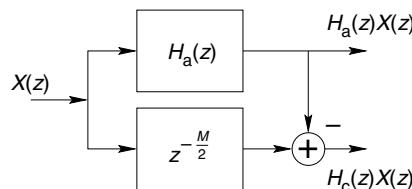


Fig. 12.12.

Realization of the complementary filter $H_c(z)$.

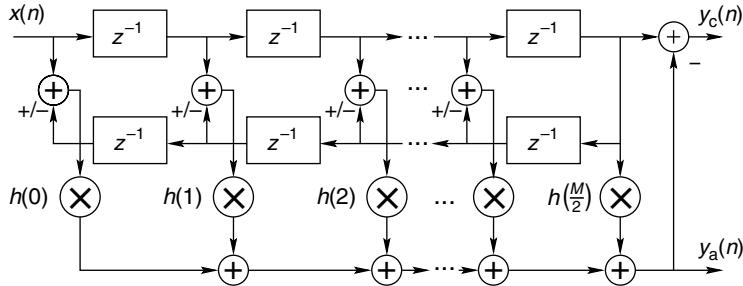


Fig. 12.13. Efficient realization of the complementary filter $H_c(z)$.

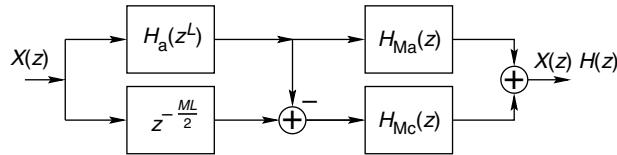


Fig. 12.14. Block diagram of the frequency-response masking approach.

The overall structure of the filter designed with the frequency-response masking approach is seen in Figure 12.14. The basic idea is to design a wideband lowpass filter and compress its frequency response by using an interpolation operation. A complementary filter is obtained, following the development seen above. We then use masking filters, $H_{Ma}(z)$ and $H_{Mc}(z)$, to eliminate the undesired bands in the interpolated and complementary filters, respectively. The corresponding outputs are added together to form the desired lowpass filter.

To understand the overall procedure in the frequency domain, consider Figure 12.15. Suppose that $H_a(z)$ corresponds to a lowpass filter of even order M , designed with the standard minimax approach, with passband edge θ and stopband edge ϕ , as seen in Figure 12.15a. We can then form $H_c(z)$, corresponding to a highpass filter, with θ and ϕ being the respective stopband and passband edges. By interpolating both filters by L , two complementary multiband filters are generated, as shown in Figures 12.15b and 12.15c respectively. We can then use two masking filters, $H_{Ma}(z)$ and $H_{Mc}(z)$, characterized as in Figures 12.15d and 12.15e, to generate the magnitude responses shown in Figures 12.15f and 12.15g. Adding these two components, the resulting desired filter seen in Figure 12.15h can have a passband of arbitrary width with a very narrow transition band. In Figure 12.15, the positions of the transition band edges are dictated by the $H_{Ma}(z)$ masking filter. An example of the frequency response masking approach where the transition band edges are determined by the $H_{Mc}(z)$ masking filter, is shown in Figure 12.16.

From Figure 12.14, it is easy to see that the product ML must be even to avoid a half-sample delay. This is commonly satisfied by forcing M to be even, as above, thus freeing the parameter L from any constraint. In addition, $H_{Ma}(z)$ and $H_{Mc}(z)$ must have the same group delay, so that they complement each other appropriately in the resulting passband when added together to form the desired filter $H(z)$. This means that they must be both of

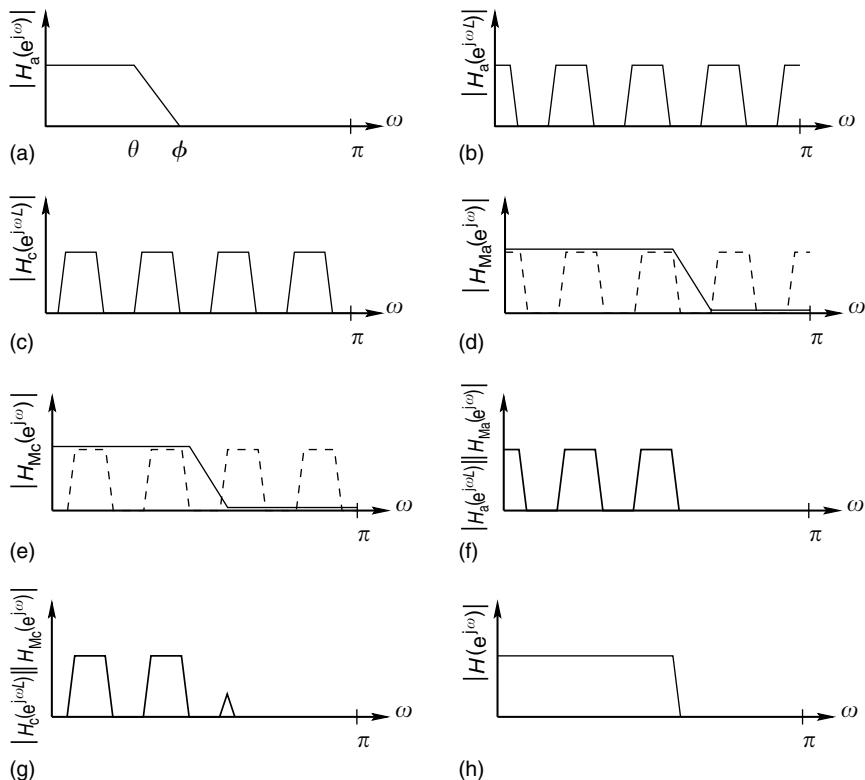


Fig. 12.15.

Frequency-response masking design of a lowpass filter with the $H_{Ma}(z)$ mask determining the passband: (a) base filter; (b) interpolated filter; (c) complementary to the interpolated filter; (d) masking filter $H_{Ma}(z)$; (e) masking filter $H_{Mc}(z)$; (f) cascade of $H_a(z^L)$ with masking filter $H_{Ma}(z)$; (g) cascade of $H_c(z^L)$ with masking filter $H_{Mc}(z)$; (h) frequency-response masking filter.

even order or both of odd order, and that a few delays may be appended before and after either $H_{Ma}(z)$ or $H_{Mc}(z)$, if necessary, to equalize their group delays.

For a complete description of the frequency-response masking approach, we must characterize the filters $H_a(z)$, $H_{Ma}(z)$, and $H_{Mc}(z)$. When the resulting magnitude response is determined mainly by the masking filter $H_{Ma}(z)$, as exemplified in Figure 12.15, then we can conclude that the desired band edges are such that

$$\omega_p = \frac{2m\pi + \theta}{L} \quad (12.74)$$

$$\omega_r = \frac{2m\pi + \phi}{L}, \quad (12.75)$$

where m is an integer less than L . Therefore, a solution for the values of m , θ , and ϕ , such that $0 < \theta < \phi < \pi$, is given by

$$m = \left\lfloor \frac{\omega_p L}{2\pi} \right\rfloor \quad (12.76)$$

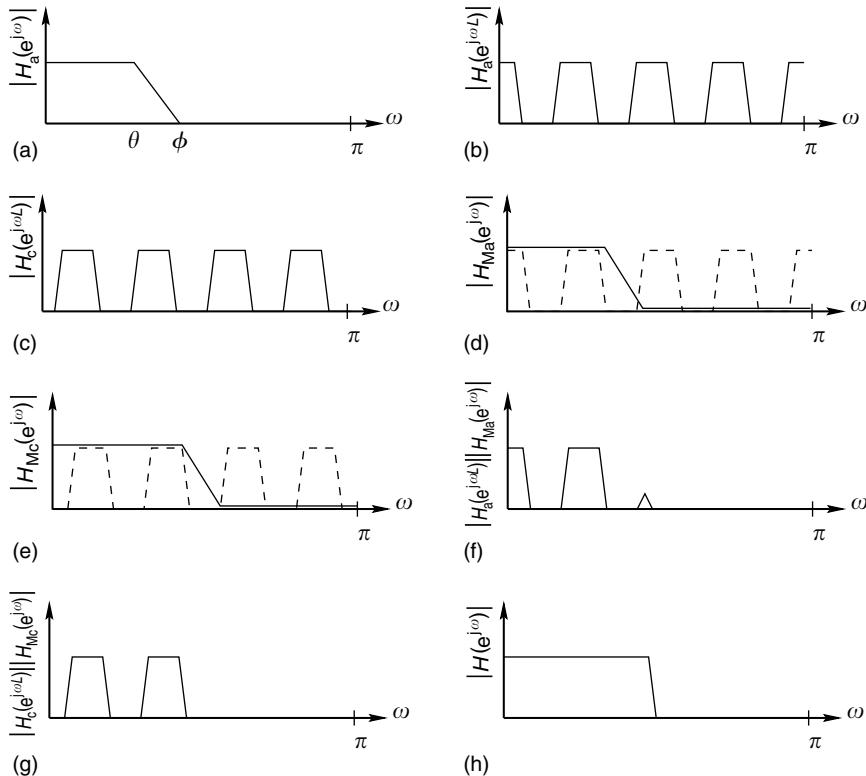


Fig. 12.16.

Frequency response masking design of a lowpass filter with the $H_{Mc}(z)$ mask determining the passband: (a) base filter; (b) interpolated filter; (c) complementary to the interpolated filter; (d) masking filter $H_{Ma}(z)$; (e) masking filter $H_{Mc}(z)$; (f) cascade of $H_a(z^L)$ with masking filter $H_{Ma}(z)$; (g) cascade of $H_c(z^L)$ with masking filter $H_{Mc}(z)$; (h) frequency response masking filter.

$$\theta = \omega_p L - 2m\pi \quad (12.77)$$

$$\phi = \omega_r L - 2m\pi, \quad (12.78)$$

where $\lfloor x \rfloor$ indicates the largest integer less than or equal to x . With these values, from Figure 12.15 we can determine the band edges for the masking filters as given by

$$\omega_{p, Ma} = \frac{2m\pi + \theta}{L} \quad (12.79)$$

$$\omega_{r, Ma} = \frac{2(m+1)\pi - \phi}{L} \quad (12.80)$$

$$\omega_{p, Mc} = \frac{2m\pi - \theta}{L} \quad (12.81)$$

$$\omega_{r, Mc} = \frac{2m\pi + \phi}{L}, \quad (12.82)$$

where $\omega_{p,Ma}$ and $\omega_{r,Ma}$ are respectively, the passband and stopband edges for the $H_{Ma}(z)$ masking filter and $\omega_{p,Mc}$ and $\omega_{r,Mc}$ are respectively the passband and stopband edges for the $H_{Mc}(z)$ masking filter.

When $H_{Mc}(z)$ is the dominating masking filter, as seen in Figure 12.16, we have that

$$\omega_p = \frac{2m\pi - \phi}{L} \quad (12.83)$$

$$\omega_r = \frac{2m\pi - \theta}{L} \quad (12.84)$$

and a solution for m , θ , and ϕ , such that $0 < \theta < \phi < \pi$, is given by

$$m = \left\lceil \frac{\omega_r L}{2\pi} \right\rceil \quad (12.85)$$

$$\theta = 2m\pi - \omega_r L \quad (12.86)$$

$$\phi = 2m\pi - \omega_p L. \quad (12.87)$$

where $\lceil x \rceil$ indicates the smallest integer greater than or equal to x . In this case, from Figure 12.16, the band edges for the masking filters are given by

$$\omega_{p,Ma} = \frac{2(m-1)\pi + \phi}{L} \quad (12.88)$$

$$\omega_{r,Ma} = \frac{2m\pi - \theta}{L} \quad (12.89)$$

$$\omega_{p,Mc} = \frac{2m\pi - \phi}{L} \quad (12.90)$$

$$\omega_{r,Mc} = \frac{2m\pi + \theta}{L}. \quad (12.91)$$

Given the desired ω_p and ω_r , each value of L may allow one solution such that $\theta < \phi$, either in the form of Equations (12.76)–(12.78) or of Equations (12.85)–(12.87). In practice, the determination of the best L (that is, the one which minimizes the total number of multiplications per output sample), can be done empirically with the aid of the order estimation given in Exercise 5.25.

The passband ripples and attenuation levels used when designing $H_a(z)$, $H_{Ma}(z)$, and $H_{Mc}(z)$ are determined based on the specifications of the desired filter. As these filters are cascaded, their frequency responses will be added in decibels, thus requiring a certain margin to be used in their designs. For the base filter $H_a(z)$, one must keep in mind that its passband ripple δ_p corresponds to the stopband attenuation δ_r of its complementary filter $H_c(z)$, and vice versa. Therefore, when designing $H_a(z)$ one should use the smallest value between δ_p and δ_r , incorporating an adequate margin. In general, a margin of 50% in the values of the passband ripples and stopband attenuations should be used, as in the example that follows.

Table 12.3.Filter characteristics for several values of the interpolation factor L .

L	M_{H_a}	$M_{H_{Ma}}$	$M_{H_{Mc}}$	Π	M
2	368	29	0	200	765
3	246	11	49	155	787
4	186	21	29	120	773
5	150	582	16	377	1332
6	124	29	49	103	793
7	108	28	88	115	844
8	94	147	29	137	899
9	84	61	49	99	817
10	76	32	582	348	1342
11	68	95	51	109	843

Example 12.6. Design a lowpass filter using the frequency-response masking method satisfying the following specifications:

$$\left. \begin{array}{l} A_p = 0.2 \text{ dB} \\ A_r = 60 \text{ dB} \\ \Omega_p = 0.6\pi \text{ rad/s} \\ \Omega_r = 0.61\pi \text{ rad/s} \\ \Omega_s = 2\pi \text{ rad/s} \end{array} \right\}. \quad (12.92)$$

Compare your results with the filter obtained using the standard minimax scheme.

Solution

Table 12.3 shows the estimated orders for the base filter and the masking filters for several values of the interpolation factor L . Although these values are probably slight underestimates, they allow a quick decision as to the value of L that minimizes the total number of multiplications required. In this table M_{H_a} is the order of the base filter $H_a(z)$, $M_{H_{Ma}}$ is the order of the masking filter $H_{Ma}(z)$, and $M_{H_{Mc}}$ is the order of the masking filter $H_{Mc}(z)$. Also:

$$\Pi = f(M_{H_a}) + f(M_{H_{Ma}}) + f(M_{H_{Mc}}) \quad (12.93)$$

indicates the total number of multiplications required to implement the overall filter, where

$$f(x) = \begin{cases} \frac{x+1}{2}, & \text{if } x \text{ is odd} \\ \frac{x}{2} + 1, & \text{if } x \text{ is even} \end{cases} \quad (12.94)$$

and

$$M = LM_{H_a} + \max\{M_{H_{Ma}}, M_{H_{Mc}}\} \quad (12.95)$$

is the effective order of the overall filter designed with the frequency-response masking approach. From this table, we predict that $L = 9$ should yield the most efficient filter with respect to the total number of multiplications per output sample.

Using $L = 9$ in Equations (12.76)–(12.91), the corresponding band edges for all filters are given by

$$\left. \begin{array}{l} \theta = 0.5100\pi \\ \phi = 0.6000\pi \\ \omega_{p,Ma} = 0.5111\pi \\ \omega_{r,Ma} = 0.6100\pi \\ \omega_{p,Mc} = 0.6000\pi \\ \omega_{r,Mc} = 0.7233\pi \end{array} \right\}. \quad (12.96)$$

From the filter specifications, we have that $\delta_p = 0.0115$ and $\delta_r = 0.001$. Therefore, the value of δ_r with a margin of 50% was used as the passband ripple and the stopband gain for the base filter to generate Table 12.3; that is:

$$\delta_{p,a} = \delta_{r,a} = \min\{\delta_p, \delta_r\} \times 50\% = 0.0005, \quad (12.97)$$

corresponding to a passband ripple of 0.0087 dB and a stopband attenuation of 66.0206 dB. As $\delta_{p,a} = \delta_{r,a}$ the relative weights for the passband and stopband in the minimax design of the base filter are both equal to 1.0000.

For the masking filters, we use

$$\delta_{p,Ma} = \delta_{p,Mc} = \delta_p \times 50\% = 0.00575 \quad (12.98)$$

$$\delta_{r,Ma} = \delta_{r,Mc} = \delta_r \times 50\% = 0.0005, \quad (12.99)$$

corresponding to a passband ripple of 0.0996 dB and a stopband attenuation of 66.0206 dB. In this case, the relative weights for the minimax design of both masking filters were made equal to 1.0000 and 11.5124 in the passband and stopband respectively.

The magnitude responses of the resulting filters for $L = 9$ are depicted in Figures 12.17 and 12.18. In Figures 12.17a and 12.17b, the base filter and its complementary filter are depicted, and Figures 12.17c and 12.17d show the corresponding interpolated filters. Similarly, Figures 12.18a and 12.18b depict the two masking filters, $H_{Ma}(z)$ and $H_{Mc}(z)$ respectively, and Figures 12.18c and 12.18d show the results at the outputs of these filters, which are added together to form the desired filter. The overall frequency-response masking filter is characterized in Figure 12.19 and presents a passband ripple equal to $A_p = 0.0873$ dB and a stopband attenuation of $A_r = 61.4591$ dB.

The resulting minimax filter is of order 504, thus requiring 253 multiplications per output sample. Therefore, in this case, the frequency-response masking design represents a saving of about 60% of the number of multiplications required by the standard minimax filter. Δ

Example 12.6 above shows that a constant ripple margin throughout the whole frequency range $\omega \in [0, \pi]$ is not required. In fact, with the ripple margin of 50% the passband ripple was considerably smaller than necessary, as seen in Figure 12.19a, and the attenuation was

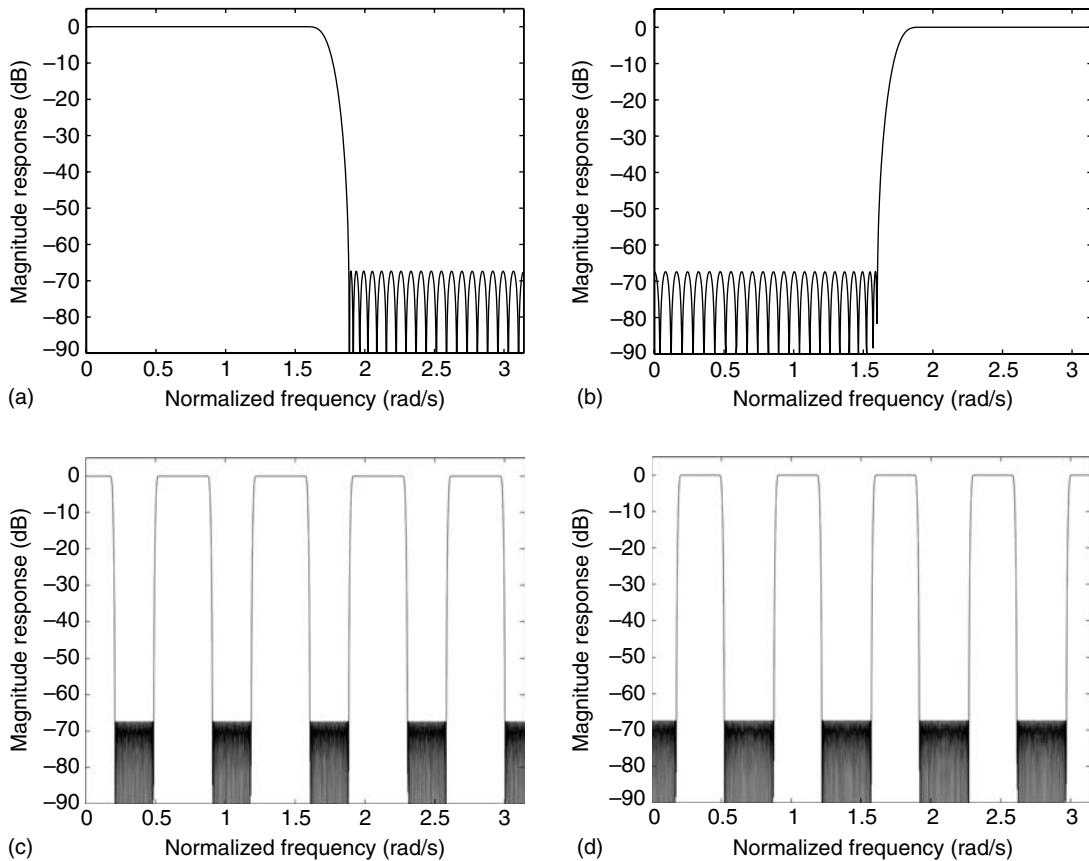


Fig. 12.17. Magnitude responses: (a) base filter $H_a(z)$; (b) complementary to the base filter $H_c(z)$; (c) interpolated base filter $H_a(z^L)$; (d) complementary to the interpolated base filter $H_c(z^L)$.

higher than required through most of the stopband, as seen in Figure 12.19b. A detailed analysis of the required margins in each band was performed in Lim (1986), where it was concluded that:

- the ripple margin must be of the order of 50% at the beginning of the stopbands of each masking filter;
- for the remaining frequency values, the ripple margin can be set around 15–20%.

It can be verified that such a distribution of the ripple margins results in a more efficient design, yielding an overall filter with a smaller group delay and fewer multiplications per output sample, as illustrated in the following example.

Example 12.7. Design the lowpass filter specified in Example 12.6 using the frequency-response masking method with an efficient assignment of the ripple margin. Compare the results with the filter obtained in Example 12.6.

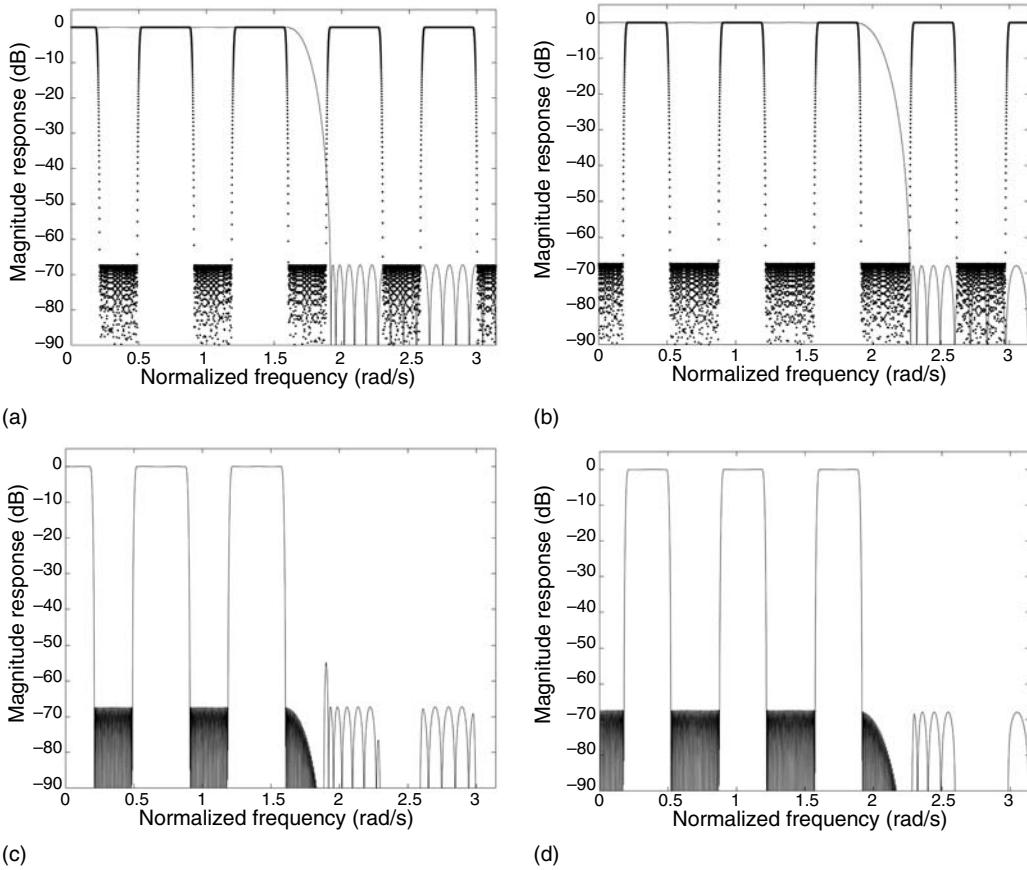


Fig. 12.18. Magnitude responses: (a) masking filter $H_{Ma}(z)$; (b) masking filter $H_{Mc}(z)$; (c) combination of $H_a(z^L)H_{Ma}(z)$; (d) combination of $H_c(z^L)H_{Mc}(z)$.

Solution

The design follows the same procedure as before, except that the relative weights at the beginning of the stopbands of the masking filters is set to 2.5, whereas for the remaining frequency the weight is set to 1.0. That corresponds to ripple margins proportional to 50% and 20% respectively in these two distinct frequency ranges.

Table 12.4 shows the filter characteristics for several values of the interpolation factor L . As in Example 12.6, the minimum number of multiplications per output sample is obtained when $L = 9$, and the band edges for the base and frequency-response masking filters are as given in Equation (12.96). In this case, however, only 91 multiplications are required, as opposed to 99 multiplications in Example 12.6.

Figures 12.20a and 12.20b depict the magnitude responses of the two masking filters, $H_{Ma}(z)$ and $H_{Mc}(z)$ respectively, and Figures 12.20c and 12.20d show the magnitude responses which are added together to form the desired filter. From these figures, one can clearly see the effects of the more efficient ripple margin distribution. The overall

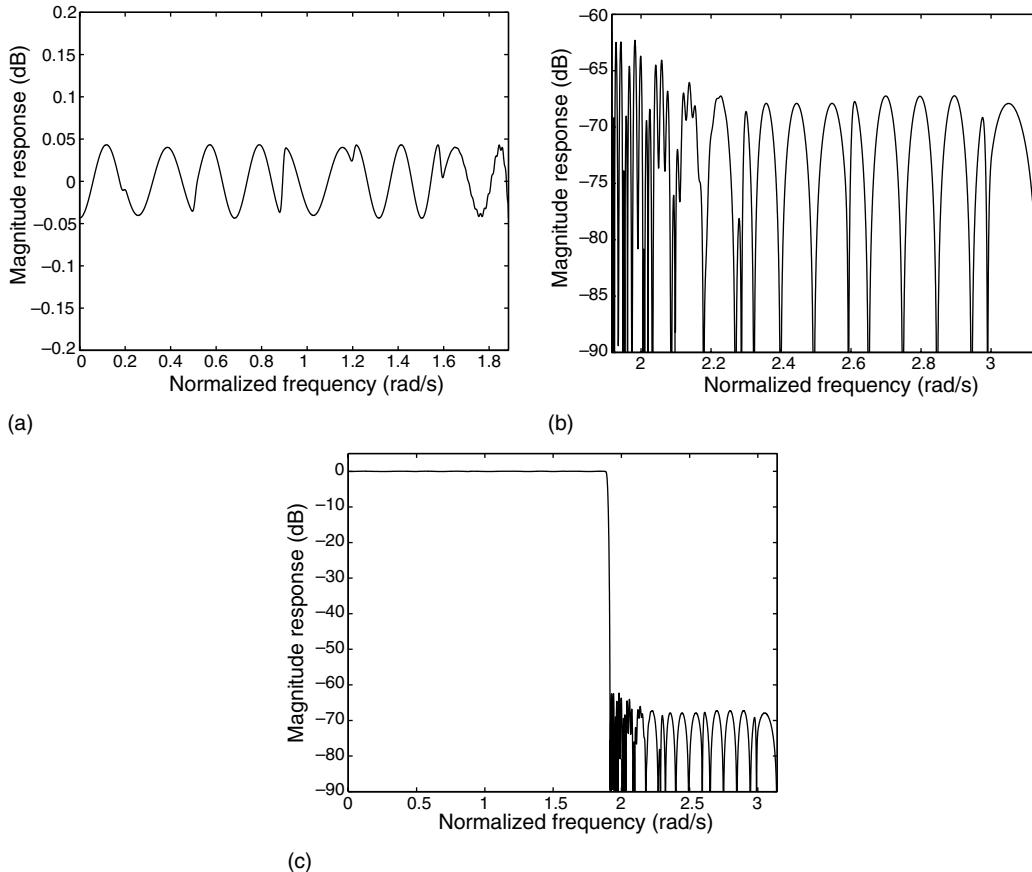


Fig. 12.19. Magnitude response of the frequency-response masking filter: (a) passband detail; (b) stopband detail; (c) overall response.

frequency-response masking filter is characterized in Figure 12.21, and presents a passband ripple equal to $A_p = 0.1502$ dB and a stopband attenuation of $A_r = 60.5578$ dB. Notice how these values are closer to the specifications than the values of the filter obtained in Example 12.6.

Table 12.5 presents the first half of the base filter coefficients before interpolation, and the coefficients of the masking filters $H_{Ma}(z)$ and $H_{Mc}(z)$ are given in Tables 12.6 and 12.7 respectively. It must be noted that, as stated before, for a smooth composition of the outputs of the masking filters when forming the filter $H(z)$, both these filters must have the same group delay. To achieve that in this design example, we must add five delays before and five delays after the masking filter $H_{Mc}(z)$, which has a smaller number of coefficients. \triangle

So far, we have discussed the use of the frequency-response masking filters to design wideband lowpass filters. The design of narrowband lowpass filters can also be performed by considering that only one masking filter is necessary. Usually, we consider the branch formed by the base filter $H_a(z)$ and its corresponding masking filter $H_{Ma}(z)$, greatly reducing

Table 12.4.Filter characteristics for several values of the interpolation factor L .

L	M_{H_a}	$M_{H_{Ma}}$	$M_{H_{Mc}}$	Π	M
2	342	26	0	186	710
3	228	10	44	144	728
4	172	20	26	112	714
5	138	528	14	343	1218
6	116	26	44	96	740
7	100	26	80	106	780
8	88	134	26	127	838
9	78	55	45	91	757
10	70	30	528	317	1228
11	64	86	46	101	790

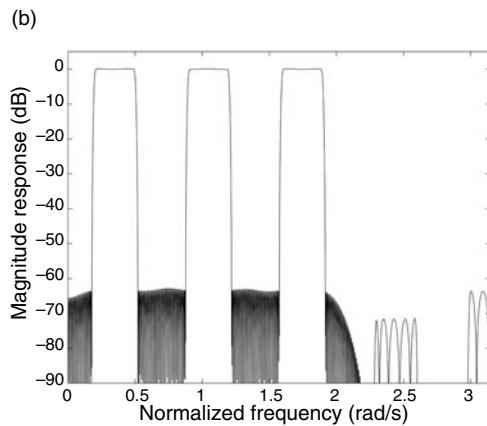
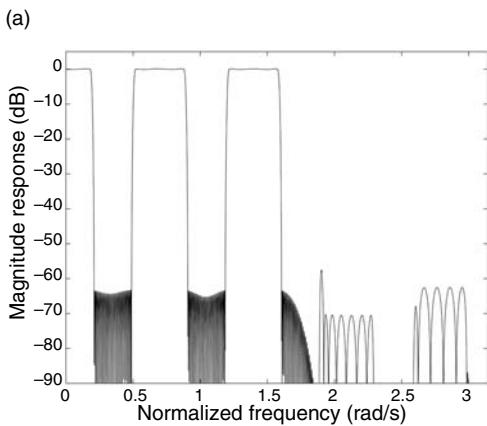
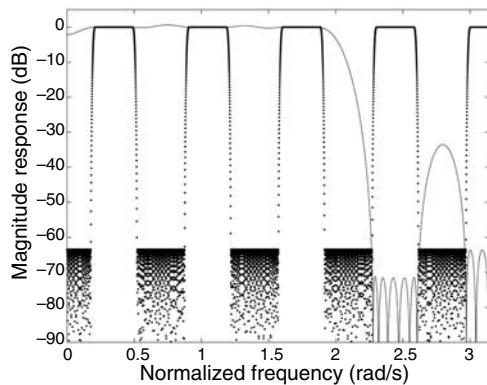
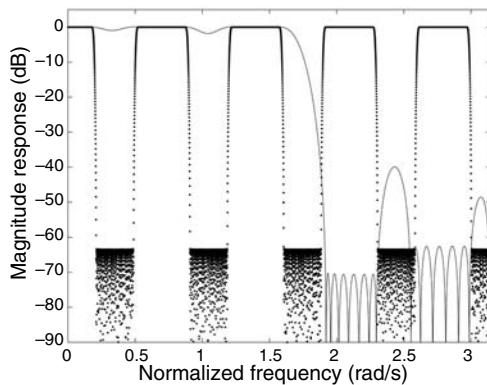


Fig. 12.20.

Magnitude responses: (a) masking filter $H_{Ma}(z)$; (b) masking filter $H_{Mc}(z)$; (c) combination of $H_a(z^L)H_{Ma}(z)$; (d) combination of $H_c(z^L)H_{Mc}(z)$.

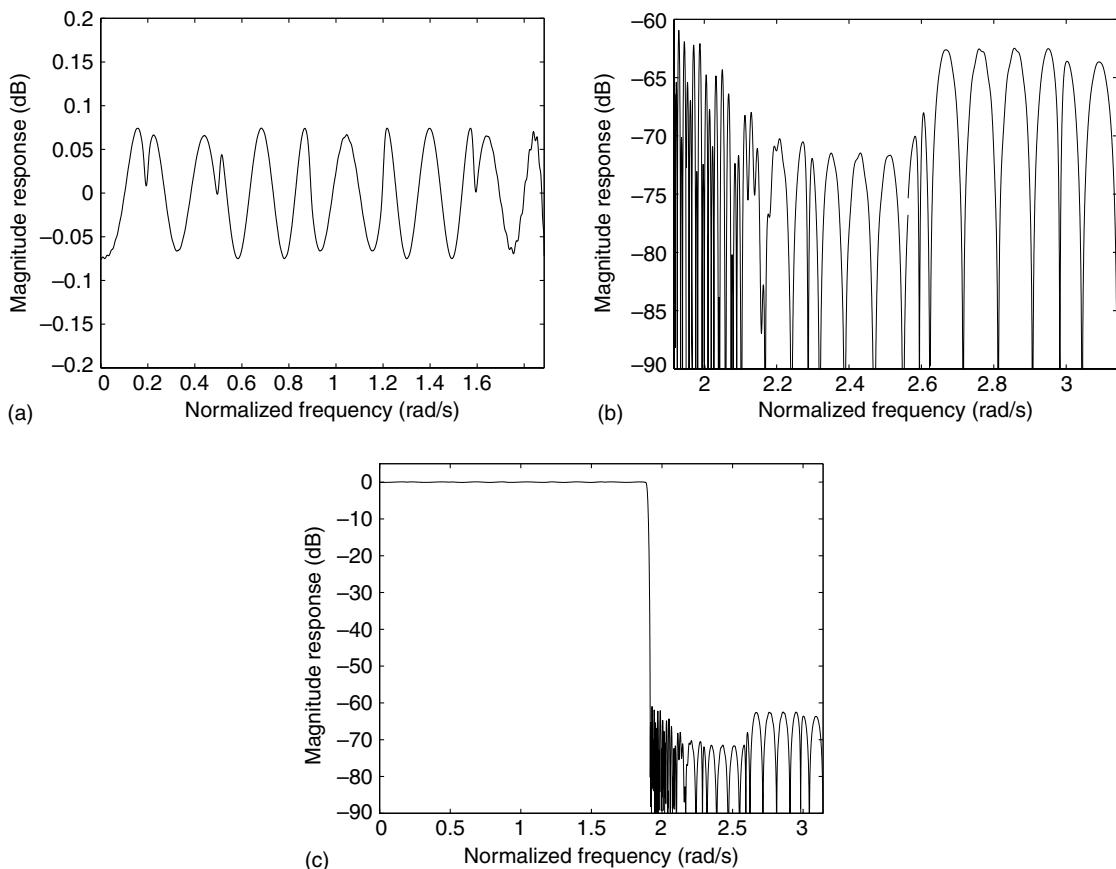


Fig. 12.21. Magnitude response of the frequency response masking filter: (a) passband detail; (b) stopband detail; (c) overall response.

the overall complexity of the designed filter. In such cases, the frequency-response masking approach becomes similar to the prefilter and interpolation approaches seen in previous subsections. The design of highpass filters can be inferred from the design for lowpass filters, or can be performed using the concept of complementary filters seen at the beginning of this subsection. The design of bandpass and bandstop filters with reduced arithmetic complexity is addressed in the next subsection.

12.7.4 Quadrature approach

In this subsection, a method for designing symmetric bandpass and bandstop filters is introduced. For narrowband filters, the so-called quadrature approach uses an FIR prototype of the form (Neüvo *et al.*, 1987)

$$H_p(z) = H_a(z^L)H_M(z) \quad (12.100)$$

Table 12.5.

Base filter $H_a(z)$ coefficients $h_a(0)$ to $h_a(39)$.

$h_a(0) = -3.7728E-04$	$h_a(14) = -1.7275E-03$	$h_a(28) = 7.7616E-03$
$h_a(1) = -2.7253E-04$	$h_a(15) = -4.3174E-03$	$h_a(29) = -2.6954E-02$
$h_a(2) = 6.7027E-04$	$h_a(16) = 3.9192E-03$	$h_a(30) = 5.0566E-04$
$h_a(3) = -1.1222E-04$	$h_a(17) = 4.0239E-03$	$h_a(31) = 3.5429E-02$
$h_a(4) = -8.2895E-04$	$h_a(18) = -6.5698E-03$	$h_a(32) = -1.4927E-02$
$h_a(5) = 4.1263E-04$	$h_a(19) = -2.5752E-03$	$h_a(33) = -4.3213E-02$
$h_a(6) = 1.1137E-03$	$h_a(20) = 9.3182E-03$	$h_a(34) = 3.9811E-02$
$h_a(7) = -1.0911E-03$	$h_a(21) = -3.4385E-04$	$h_a(35) = 4.9491E-02$
$h_a(8) = -1.1058E-03$	$h_a(22) = -1.1608E-02$	$h_a(36) = -9.0919E-02$
$h_a(9) = 1.9480E-03$	$h_a(23) = 4.9074E-03$	$h_a(37) = -5.3569E-02$
$h_a(10) = 7.4658E-04$	$h_a(24) = 1.2712E-02$	$h_a(38) = 3.1310E-01$
$h_a(11) = -2.9427E-03$	$h_a(25) = -1.1084E-02$	$h_a(39) = 5.5498E-01$
$h_a(12) = 1.7063E-04$	$h_a(26) = -1.1761E-02$	
$h_a(13) = 3.8315E-03$	$h_a(27) = 1.8604E-02$	

Table 12.6.

Masking filter $H_{Ma}(z)$ coefficients $h_{Ma}(0)$ to $h_{Ma}(27)$.

$h_{Ma}(0) = 3.9894E-03$	$h_{Ma}(10) = -1.5993E-02$	$h_{Ma}(20) = 1.8066E-02$
$h_{Ma}(1) = 5.7991E-03$	$h_{Ma}(11) = -4.4088E-03$	$h_{Ma}(21) = -4.8343E-02$
$h_{Ma}(2) = 9.2771E-05$	$h_{Ma}(12) = 1.6123E-02$	$h_{Ma}(22) = -1.2214E-02$
$h_{Ma}(3) = -6.1430E-03$	$h_{Ma}(13) = 4.5664E-03$	$h_{Ma}(23) = 6.7391E-02$
$h_{Ma}(4) = -2.5059E-03$	$h_{Ma}(14) = -1.5292E-02$	$h_{Ma}(24) = -1.3277E-02$
$h_{Ma}(5) = 3.1213E-03$	$h_{Ma}(15) = 1.7599E-03$	$h_{Ma}(25) = -1.1247E-01$
$h_{Ma}(6) = -8.6700E-04$	$h_{Ma}(16) = 1.5389E-02$	$h_{Ma}(26) = 1.0537E-01$
$h_{Ma}(7) = -3.8008E-03$	$h_{Ma}(17) = -1.1324E-02$	$h_{Ma}(27) = 4.7184E-01$
$h_{Ma}(8) = 2.1950E-03$	$h_{Ma}(18) = -7.2774E-03$	
$h_{Ma}(9) = -3.8907E-03$	$h_{Ma}(19) = 3.7826E-02$	

Table 12.7.

Masking filter $H_{Mc}(z)$ coefficients $h_{Mc}(0)$ to $h_{Mc}(22)$.

$h_{Mc}(0) = 1.9735E-04$	$h_{Mc}(8) = -1.3031E-02$	$h_{Mc}(16) = 2.3092E-02$
$h_{Mc}(1) = -7.0044E-03$	$h_{Mc}(9) = 3.5921E-03$	$h_{Mc}(17) = -5.8850E-02$
$h_{Mc}(2) = -7.3774E-03$	$h_{Mc}(10) = -4.7280E-03$	$h_{Mc}(18) = 7.3208E-03$
$h_{Mc}(3) = 1.9310E-03$	$h_{Mc}(11) = -2.5730E-02$	$h_{Mc}(19) = 5.5313E-02$
$h_{Mc}(4) = -3.0938E-04$	$h_{Mc}(12) = 6.5528E-03$	$h_{Mc}(20) = -1.2326E-01$
$h_{Mc}(5) = -7.1047E-03$	$h_{Mc}(13) = 1.1745E-02$	$h_{Mc}(21) = 9.7698E-03$
$h_{Mc}(6) = 3.0039E-03$	$h_{Mc}(14) = -3.2147E-02$	$h_{Mc}(22) = 5.4017E-01$
$h_{Mc}(7) = 5.8004E-04$	$h_{Mc}(15) = 7.8385E-03$	

where $H_a(z)$ is the base filter and $H_M(z)$ is the masking filter or interpolator, which attenuates the undesired spectral images of the passband of $H_a(z^L)$, commonly referred to as the shaping filter. Such a prototype can be designed using the prefilter, interpolation, or simplified one-branch frequency-response masking approaches seen above. The main idea of the quadrature approach is to shift the frequency response of the base filter to the desired central frequency ω_0 , and then apply the masking filter (interpolator) to eliminate any other undesired passbands.

Consider a linear-phase lowpass filter $H_a(z)$ with impulse response $h_a(n)$, such that

$$H_a(z) = \sum_{n=0}^M h_a(n)z^{-n}. \quad (12.101)$$

Let the passband ripple and stopband gain be equal to δ'_p and δ'_r respectively, and let the passband and stopband edges be ω'_p and ω'_r , respectively. If $h_a(n)$ is interpolated by a factor L and the resulting sequence is multiplied by $e^{j\omega_0 n}$, then we generate an auxiliary $H_1(z^L)$ as

$$H_1(z^L) = \sum_{n=0}^M h_a(n)e^{j\omega_0 n} z^{-nL}. \quad (12.102)$$

This implies that the passband of $H_a(z)$ is squeezed by a factor of L , and becomes centered at ω_0 . Analogously, using the interpolation operation followed by the modulating sequence $e^{-j\omega_0 n}$, we have another auxiliary function such that

$$H_2(z^L) = \sum_{n=0}^M h_a(n)e^{-j\omega_0 n} z^{-nL} \quad (12.103)$$

with the corresponding squeezed passband centered at $-\omega_0$. We can then use two masking filters, $H_{M1}(z)$ and $H_{M2}(z)$, appropriately centered at ω_0 and $-\omega_0$ to eliminate the undesired bands in $H_1(z^L)$ and $H_2(z^L)$ respectively. The addition of the two resulting sequences yields a symmetric bandpass filter centered at ω_0 .

Clearly, although the two-branch overall bandpass filter will have real coefficients, each branch in this case will present complex coefficients. To overcome this problem, first note that $H_1(z^L)$ and $H_2(z^L)$ have complex conjugate coefficients. If we design $H_{M1}(z)$ and $H_{M2}(z)$ so that their coefficients are complex conjugates of each other, then it is easy to verify that

$$\begin{aligned} H_1(z^L)H_{M1}(z) &= \left(H_{1,R}(z^L) + jH_{1,I}(z^L) \right) \left(H_{M1,R}(z) + jH_{M1,I}(z) \right) \\ &= \left(H_{1,R}(z^L)H_{M1,R}(z) - H_{1,I}(z^L)H_{M1,I}(z) \right) \\ &\quad + j \left(H_{1,R}(z^L)H_{M1,I}(z) + H_{1,I}(z^L)H_{M1,R}(z) \right) \end{aligned} \quad (12.104)$$

$$\begin{aligned}
H_2(z^L)H_{M2}(z) &= \left(H_{2,R}(z^L) + jH_{2,I}(z^L) \right) \left(H_{M2,R}(z) + jH_{M2,I}(z) \right) \\
&= \left(H_{1,R}(z^L) - jH_{1,I}(z^L) \right) \left(H_{M1,R}(z) - jH_{M1,I}(z) \right) \\
&= \left(H_{1,R}(z^L)H_{M1,R}(z) - H_{1,I}(z^L)H_{M1,I}(z) \right) \\
&\quad - j \left(H_{1,R}(z^L)H_{M1,I}(z) + H_{1,I}(z^L)H_{M1,R}(z) \right), \tag{12.105}
\end{aligned}$$

where the subscripts R and I indicate the parts of the corresponding transfer function with real and imaginary coefficients respectively. Therefore:

$$H_1(z^L)H_{M1}(z) + H_2(z^L)H_{M2}(z) = 2 \left(H_{1,R}(z^L)H_{M1,R}(z) - H_{1,I}(z^L)H_{M1,I}(z) \right) \tag{12.106}$$

and the structure seen in Figure 12.22 can be used for the real implementation of the quadrature approach for narrowband filters. Disregarding the effects of the masking filters, the resulting quadrature filter is characterized by

$$\left. \begin{array}{l} \delta_p = \delta'_p + \delta'_r \\ \delta_r = 2\delta'_r \\ \omega_{r_1} = \omega_0 - \omega'_r \\ \omega_{p_1} = \omega_0 - \omega'_p \\ \omega_{p_2} = \omega_0 + \omega'_p \\ \omega_{r_2} = \omega_0 + \omega'_r \end{array} \right\}. \tag{12.107}$$

For wideband filters, the prototype filter should be designed with the frequency-response masking approach. In that case, we have two complete masking filters, and the quadrature implementation involving solely real filters is seen in Figure 12.23, with $H_1(z^L)$ as defined in Equation (12.102), and $H_{Ma}(z)$ and $H_{Mc}(z)$ corresponding to the two masking filters, appropriately centered at ω_0 and $-\omega_0$, respectively.

For bandstop filters, we may start with a highpass prototype and apply the quadrature design, or design a bandpass filter and then determine its complementary filter (Rajan *et al.*, 1988).

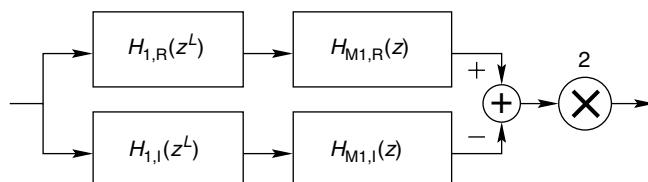


Fig. 12.22. Block diagram of the quadrature approach for narrowband filters.

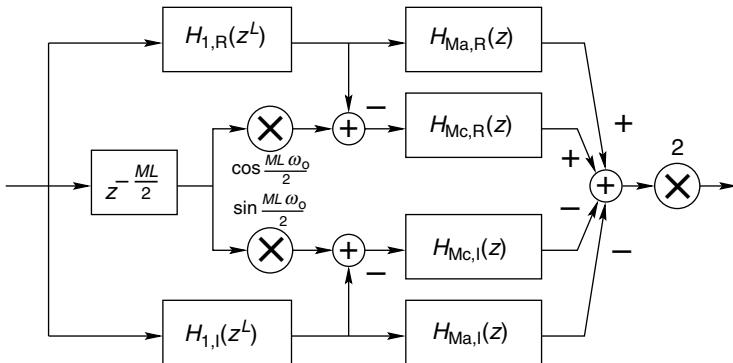


Fig. 12.23. Block diagram of the quadrature approach for wideband filters.

Example 12.8. Design a bandpass filter using the quadrature method satisfying the following specifications:

$$\left. \begin{array}{l} A_p = 0.2 \text{ dB} \\ A_r = 40 \text{ dB} \\ \Omega_{r_1} = 0.09\pi \text{ rad/s} \\ \Omega_{p_1} = 0.1\pi \text{ rad/s} \\ \Omega_{p_2} = 0.7\pi \text{ rad/s} \\ \Omega_{r_2} = 0.71\pi \text{ rad/s} \\ \Omega_s = 2\pi \text{ rad/s} \end{array} \right\}. \quad (12.108)$$

Solution

Given the bandpass specifications, the lowpass prototype filter must have a passband half the size of the desired passband width and a transition bandwidth equal to the minimum transition bandwidth for the bandpass filter. For the passband ripple and stopband attenuation, the values specified for the bandpass filter can be used with a margin of about 40%. Therefore, in this example, the lowpass prototype is characterized by

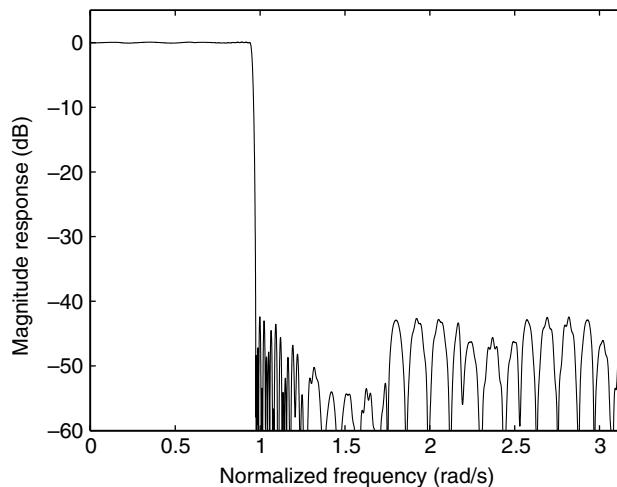
$$\left. \begin{array}{l} \delta'_p = 0.0115 \times 40\% = 0.0046 \\ \delta'_r = 0.01 \times 40\% = 0.004 \\ \omega'_p = \frac{\omega_{p_2} - \omega_{p_1}}{2} = 0.3\pi \\ \omega'_r = \omega'_p + \min \{(\omega_{p_1} - \omega_{r_1}), (\omega_{r_2} - \omega_{p_2})\} = 0.31\pi \end{array} \right\}. \quad (12.109)$$

This filter can be designed using the frequency-response masking approach with an efficient ripple margin assignment, seen in the previous subsection. In this case, the interpolation factor that minimizes the total number of multiplications is $L = 8$ and the corresponding filter characteristics are given in Table 12.8, with the resulting magnitude response in Figure 12.24.

The resulting bandpass filter using the quadrature method is shown in Figure 12.25.

Table 12.8.Filter characteristics for the interpolation factor $L = 8$.

L	M_{H_a}	$M_{H_{Ma}}$	$M_{H_{Mc}}$	Π	M
8	58	34	42	70	506

**Fig. 12.24.**

Lowpass prototype designed with the frequency-response masking approach for the quadrature design of a bandpass filter.

For the complete quadrature realization, the total number of multiplications is 140, twice the number of multiplications necessary for the prototype lowpass filter. For this example, the minimax filter would be of order 384, thus requiring 193 multiplications per output sample. Therefore, in this case, the quadrature design represents a saving of about 30% of the number of multiplications required by the standard minimax filter. \triangle

12.8 Do-it-yourself: efficient FIR structures

Experiment 12.1

A base-band telephone speech signal occupies the frequency range around 300–3600 Hz. Let us emulate such a signal as a sum of four sinusoids as given by

```
Fs = 40000; Ts = 1/Fs; time = 0:Ts:(1-Ts);
f1 = 300; f2 = 1000; f3 = 2500; f4 = 3600;
s1 = sin(2*pi*f1*time); s2 = sin(2*pi*f2*time);
s3 = sin(2*pi*f3*time); s4 = sin(2*pi*f4*time);
x = s1 + s2 + s3 + s4;
```

Using the modulation theorem, we can shift the spectrum of x of f_C by multiplying this signal by a cosine function; that is:

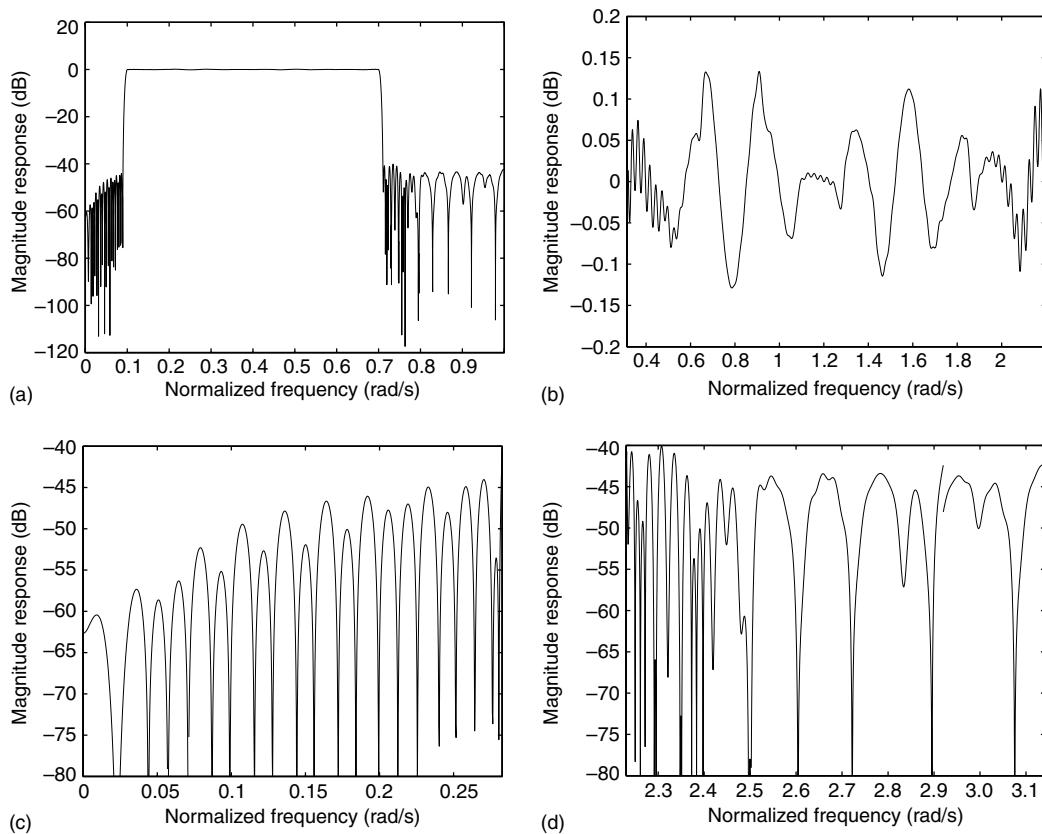


Fig. 12.25. Magnitude responses of the bandpass filter designed with the quadrature approach: (a) overall filter; (b) passband detail; (c) lower stopband detail; (d) upper stopband detail.

```
fc = 10000;
xDSB = x.*cos(2*pi*fc*time);
```

In this way, several speech signals can fit in a single communication channel by using distinct values of f_c for each one. The spectral representations of x before and after modulation are seen in Figure 12.26.

In this figure, one clearly notices that the spectrum of x_{DSB} occupies twice the band of the spectrum of x , thus originating the double sideband (DSB) nomenclature for that signal. For a single speech signal, this does not seem much of a problem, as the additional 4 kHz can be easily dealt with in most communications systems. However, when you think of millions of users of the telephone system, this doubling effect can generate an unwelcome overload on the given channel. Therefore, we consider here the elimination of the so-called upper part of the spectrum of x_{DSB} , within 10.3–13.6 kHz.

In this illustrative experiment, one can design a Chebyshev filter using the `firpm` command in MATLAB; for instance, using the frequency range between $\omega_p = 2\pi 9700$ rad/s and $\omega_r = 2\pi 10300$ rad/s as the filter transition band. However, in more demanding cases, of higher sampling frequency or even narrower transition bands, an efficient FIR structure must be employed, as exemplified here.

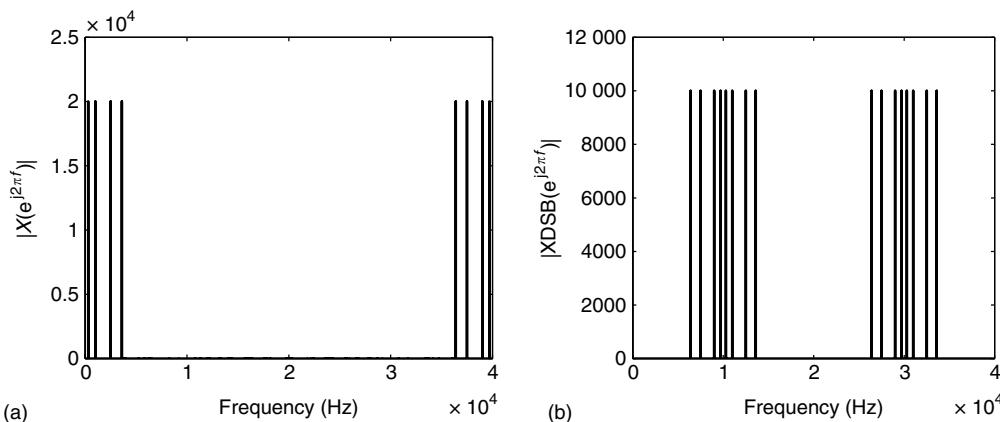


Fig. 12.26. Spectra of emulated speech signal: (a) base-band x ; (b) modulated x_{DSB} .

A frequency-response masking filter for this application can be designed in MATLAB, such as

$$wp = (fc-f1)*2*pi/Fs; wr = (fc+f1)*2*pi/Fs;$$

Using $L = 5$, the $H_{Ma}(z)$ masking filter defines the transition band; therefore, from Equations (12.76)–(12.78), we have that

$$\begin{aligned} L &= 5; m = \text{floor}(wp*L/(2*pi)); \\ \theta &= wp*L-m*2*pi; \phi = wr*L-m*2*pi; \end{aligned}$$

It must be emphasized that different specifications may force the reader to employ Equations (12.85)–(12.87) instead.

Setting the passband ripple and attenuation levels to $A_p = 0.1$ dB and $A_r = 50$ dB, respectively, we get

$$\begin{aligned} Ap &= 0.1; \text{delta_p} = (10^{(Ap/20)}-1)/(10^{(Ap/20)}+1); \\ Ar &= 50; \text{delta_r} = 10^{(-Ar/20)}; \end{aligned}$$

and then the `firpmord` and `firpm` commands can be used in tandem to design the frequency-response masking base filter, as given by

$$\begin{aligned} Fvec_b &= [\theta \phi]./pi; \\ [M, f_b, m_b, w_b] &= \text{firpmord}(Fvec_b, [1 0], [\text{delta_p} \text{ delta_r}]); \\ hb &= \text{firpm}(M, f_b, m_b, w_b); \end{aligned}$$

In this stage, as discussed above, one must remember to enforce an even base-filter order M , increasing it by 1 if necessary. In the script above, $M=32$ and no order increment is required.

The interpolated base filter can be formed as

$$\begin{aligned} hbL &= [hb; \text{zeros}(L-1, M+1)]; \\ hbL &= \text{reshape}(hbL, 1, L*(M+1)); \end{aligned}$$

and the corresponding complementary filter as

$$\begin{aligned} \text{hbLc} &= -\text{hbL}; \\ \text{hbLc} (\text{M}^* \text{L}/2 + 1) &= 1 - \text{hbL} (\text{M}^* \text{L}/2 + 1); \end{aligned}$$

The positive masking filter, as specified in Equations (12.79) and (12.80), is designed as

```
wp_p = (2*m*pi+theta)/L; wr_p = (2*(m+1)*pi-phi)/L;
Fvec_p = [wp_p wr_p]./pi;
[M_p, f_p, m_p, w_p] = firpmord(Fvec_p, [1 0], [delta_p delta_r]);
hp = firpm(M_p, f_p, m_p, w_p);
```

and the negative masking filter, as given by Equations (12.81) and (12.80), may be determined as

```
wp_n = (2*m*pi-theta)/L; wr_n = (2*m*pi+phi)/L;
Fvec_n = [wp_n wr_n]./pi;
[M_n, f_n, m_n, w_n] = firpmord(Fvec_n, [1 0], [delta_p delta_r]);
hn = firpm(M_n, f_n, m_n, w_n);
```

The magnitude responses for the interpolated base filter hbL , complementary filter hbLc , positive masking filter hp , and negative masking filter hn are shown in Figure 12.27 for the entire $0-F_s$ frequency range.

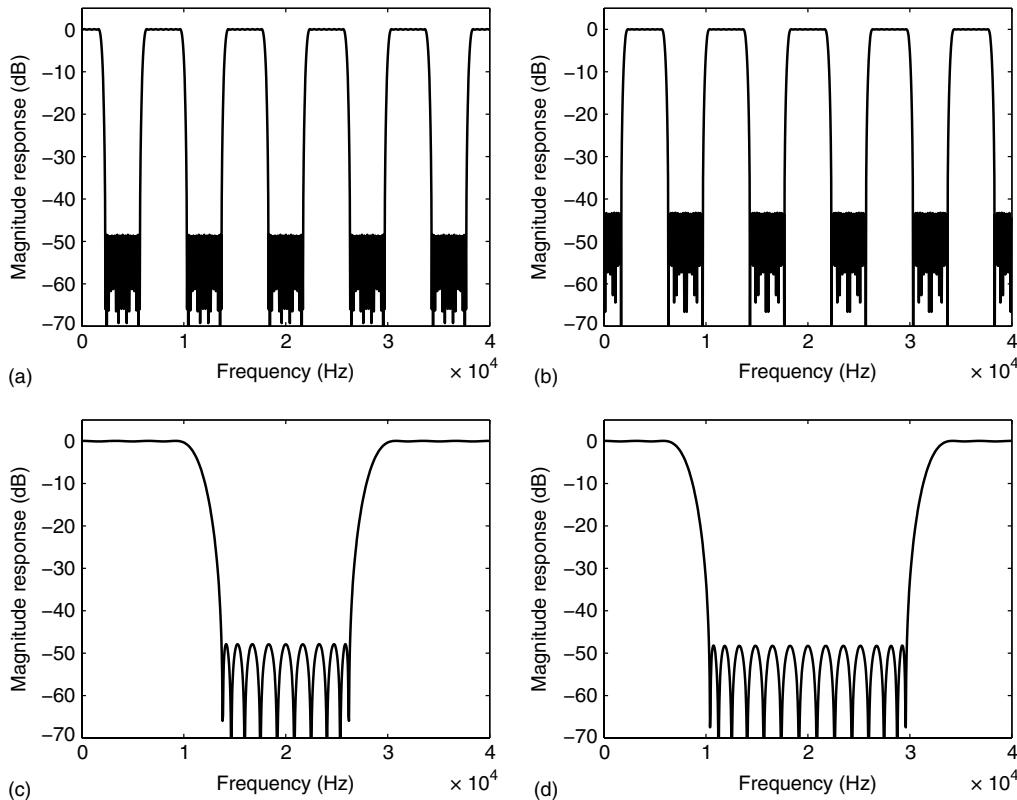


Fig. 12.27. Magnitude responses of frequency-response masking subfilters: (a) interpolated base filter hbL ; (b) complementary filter hbLc ; (c) positive masking filter hp ; (d) negative masking filter hn .

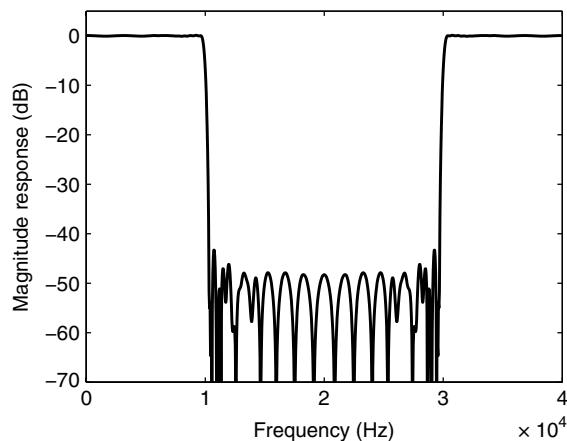


Fig. 12.28. Magnitude response of frequency-response masking filter.

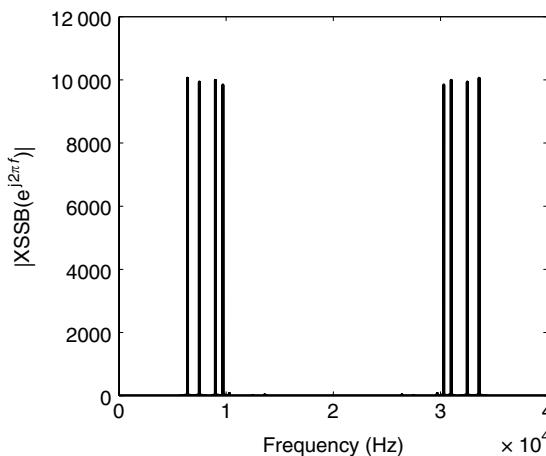


Fig. 12.29. Spectrum of xSSB signal.

The impulse response for the general frequency-response masking filter can be obtained as

$$hFRM = \text{conv}(hbL, hp) + \text{conv}(hbLc, hm);$$

which corresponds to the magnitude response shown in Figure 12.28.

Applying the xDSB to the input of the frequency-response masking filter designed above yields the single sideband (SSB) xSSB modulated signal

$$xSSB = \text{filter}(hFRM, 1, xDSB);$$

which presents the same bandwidth as the original base-band signal x , as shown in Figure 12.29.

In practical real-time applications, $hFRM$ should not be employed to perform the desired filtering operation, since it does not benefit from the modular frequency-response masking internal structure. In such a case, signal processing should be based directly on the individual

hbL, hp, and hm filters to reduce the computational effort in determining each output sample. △

12.9 Efficient FIR structures with MATLAB

Most of the realizations described in this chapter are somewhat advanced and do not have any related commands in the standard MATLAB versions, with the exception of the lattice and the frequency-domain forms, as seen below.

- **tf2latc:** Converts the direct form into the lattice form, inverting the operation of latc2tf. Notice that this command applies to FIR and IIR lattices, according to the transfer function format provided by the user. IIR lattices, however, as will be seen in Chapter 13, may also include a set of ladder parameters.

Input parameters: two vectors, one each for the numerator and denominator coefficients of the direct-form transfer function.

Output parameters:

- a vector k with the reflection coefficients;
- a vector with the ladder coefficients, for IIR lattice filters.

Example:

```
num=[1 0.6 -0.16]; den=[1 0.7 0.12];
[k,v]=tf2latc(num,den);
```

- **latc2tf:** Converts the lattice form into the direct form, inverting the operation of tf2latc.

Input parameters:

- a vector k with the reflection coefficients;
- a vector with the ladder coefficients, for IIR lattice filters;
- a string 'fir' (default) or 'iir' that determines whether the desired lattice is correspondingly FIR or IIR, if only one input vector is provided.

Output parameters: two vectors, one each for the numerator and denominator coefficients of the direct-form transfer function.

Examples:

```
k=[0.625 0.12];
num=latc2tf(k,'fir'); % FIR case
[num,den]=latc2tf(k,'iir'); % IIR case
```

- **latcfilt:** Performs signal filtering with lattice form.

Input parameters:

- A vector k of reflection coefficients.
- For IIR lattice filters, a vector of ladder coefficients. For FIR lattice filters, use a scalar 1.
- The input-signal vector x.

Output parameter: the filtered signal y .

Example:

```
k=[0.9 0.8 0.7]; x=0:0.1:10;
y=latcfilt(k,x);
```

- **fftfilt**: Performs signal filtering with the FIR frequency-domain form. See Section 3.9.

For all other realizations described in this chapter, MATLAB can be used as a friendly environment for implementing efficient design procedures, and the interested reader is encouraged to do so.

12.10 Summary

In this chapter, several efficient FIR structures were presented as alternatives to the direct form seen in Chapter 5.

The lattice structure was introduced and its applications to the implementation of two-band biorthogonal filter banks with linear phase was discussed. Other structures included the polyphase form, which is suitable for parallel processing, the frequency-domain form, which is suitable for offline filtering operations, and the recursive form, which constitutes a basic block for several algorithms in digital signal processing.

The design of linear-phase FIR filters with reduced arithmetic complexity was presented, including the prefilter, the interpolation, and the frequency-response masking approaches. The frequency-response masking approach can be seen as a generalization of the first two approaches, allowing the design of wideband filters with very narrow transition bands. The complete frequency-response masking design was presented, including the use of an efficient ripple margin assignment. Extensions of the frequency-response masking approach include the use of multiple levels of masking, where the frequency-response masking concept is applied to the design of one or both of the masking filters (Lim, 1986). The computational complexity can also be reduced by exploiting the similarities of the two masking filters; that is, the same passband ripple and stopband attenuation levels. If these two filters also have similar cutoff frequencies, then a single masking filter with intermediate characteristics can be used. Such a filter is then followed by simple equalizers that transform the frequency response of this unique masking filter into the required responses of the original frequency masks (Lim & Lian, 1994). The chapter ended with a description of the quadrature method for bandpass and bandstop filters, which reduces the computational complexity.

12.11 Exercises

12.1 Prove Equations (12.6) and (12.7).

12.2 Using Equations (12.3)–(12.7), derive the order-recursive relationship given in Equation (12.8).

- 12.3 Write down a MATLAB command that determines the FIR lattice coefficients from the FIR direct-form coefficients.
- 12.4 For $\mathbf{E}(z)$ and $\mathbf{R}(z)$ defined as in Equations (12.20) and (12.22), show that:
- $\mathbf{E}(z)\mathbf{R}(z) = z^{-M}$.
 - The synthesis filter bank has linear phase.
- 12.5 Synthesize a two-band lattice filter bank for the case such that

$$H_0(z) = z^{-5} + 2z^{-4} + 4z^{-3} + 4z^{-2} + 2z^{-1} + 1.$$

Determine the corresponding $H_1(z)$.

- 12.6 Design a two-band QMF filter bank satisfying the following specifications:

$$\begin{aligned}\delta_p &= 0.5 \text{ dB} \\ \delta_r &= 40 \text{ dB} \\ \Omega_p &= 0.45 \frac{\Omega_s}{2} \\ \Omega_r &= 0.55 \frac{\Omega_s}{2} \\ \Omega_s &= 2\pi \text{ rad/s.}\end{aligned}$$

- Design the filter using the standard direct-form parametrization.
 - Design the filter by optimizing the parameters of the appropriate lattice FIR structure.
- 12.7 Design a two-band perfect-reconstruction filter bank with linear-phase subfilters satisfying the following specifications:

$$\begin{aligned}\delta_p &= 1 \text{ dB} \\ \delta_r &= 60 \text{ dB} \\ \Omega_p &= 0.40 \frac{\Omega_s}{2} \\ \Omega_r &= 0.60 \frac{\Omega_s}{2} \\ \Omega_s &= 2\pi \text{ rad/s.}\end{aligned}$$

- Design the filter using the standard direct-form parametrization.
 - Design the filter by optimizing the parameters of the appropriate lattice FIR structure.
- 12.8 A lattice-like realization with second-order section allows the design of linear-phase two-band filter banks with even order where both $H_0(z)$ and $H_1(z)$ are symmetric. In this case

$$\mathbf{E}(z) = \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \left[\prod_{i=1}^L \begin{pmatrix} 1+z^{-1} & 1 \\ 1+\beta_i z^{-1}+z^{-2} & 1+z^{-1} \end{pmatrix} \begin{pmatrix} \gamma_i & 0 \\ 0 & 1 \end{pmatrix} \right].$$

- Design an analysis filter bank such that $H_1(z) = -1 + z^{-1} + 2z^{-2} + z^{-3} - z^{-4}$.
- Determine the corresponding $H_0(z)$.
- Derive the general expression for the polyphase matrix of the synthesis filter.

- 12.9 Design a highpass filter using the minimax method satisfying the following specifications:

$$\begin{aligned}A_p &= 0.8 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_r &= 5000 \text{ Hz} \\\Omega_p &= 5200 \text{ Hz} \\\Omega_s &= 12\,000 \text{ Hz.}\end{aligned}$$

Determine the corresponding lattice coefficients for the resulting filter using the command you created in Exercise 12.3. Compare the results obtained using the standard command `tf2latt`.

- 12.10 Use the MATLAB commands `filter` and `filtlatt` with the coefficients obtained in Exercise 12.9 to filter a given input signal. Verify that the output signals are identical. Compare the processing time and the total number of floating-point operations used by each command, using `tic`, `toc`, and `flops`.
- 12.11 Show that the linear-phase relations in Equation (12.24) hold for the analysis filters when their polyphase components are given by Equation (12.20).
- 12.12 Use the MATLAB commands `filter`, `conv`, and `fft` to filter a given input signal, with the impulse response obtained in Exercise 12.9. Verify what must be done to the input signal in each case to force all output signals to be identical. Compare the processing time and the total number of floating-point operations used by each command, using `tic`, `toc`, and `flops` (see the MATLAB `help` for the correct usage of these commands).
- 12.13 Discuss the distinct feature of an RRS filter with even and odd values of M .
- 12.14 By replacing z with $-z$ in an RRS filter, where will its pole and zeros be located? What type of magnitude response will result?
- 12.15 Replace z by z^2 in an RRS filter and discuss the properties of the resulting filter.
- 12.16 Plot the magnitude response of the modified-sinc filter with $M = 2, 4, 6, 8$. Choose an appropriate ω_0 and compare the resulting passband widths.
- 12.17 Plot the magnitude response of the modified-sinc filter with $M = 7$. Choose four values for ω_0 and discuss the effect on the magnitude response.
- 12.18 Given the transfer function below

$$H(z) = -z^{-2N} + \dots - z^{-6} + z^{-4} - z^{-2} + 1,$$

where M is an odd number:

- (a) Show a realization with the minimum number of adders.
- (b) For a fixed-point implementation, determine the scaling factors of the filter for the L_2 and L_∞ norms.
- (c) Discuss the advantages and disadvantages of the given structure with respect to its transposed version.

- 12.19 Design a lowpass filter using the prefilter and interpolation methods satisfying the following specifications:

$$\begin{aligned}A_p &= 0.8 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_p &= 4500 \text{ Hz} \\\Omega_r &= 5200 \text{ Hz} \\\Omega_s &= 12\,000 \text{ Hz.}\end{aligned}$$

Compare the required number of multiplications per output sample in each design.

- 12.20 Repeat Exercise 12.19 using the modified-sinc structure as building block.
 12.21 Demonstrate quantitatively that FIR filters based on the prefilter and the interpolation methods have lower sensitivity and reduced output roundoff noise than the minimax filters implemented using the direct form, when they satisfy the same specifications.
 12.22 Design a lowpass filter using the frequency-response masking method satisfying the following specifications:

$$\begin{aligned}A_p &= 2.0 \text{ dB} \\A_r &= 40 \text{ dB} \\\omega_p &= 0.33\pi \text{ rad/sample} \\\omega_r &= 0.35\pi \text{ rad/sample.}\end{aligned}$$

Compare the results obtained with and without an efficient ripple margin distribution, with respect to the total number of multiplications per output sample, with the maximum passband ripple and with the resulting minimum stopband attenuation.

- 12.23 Design a highpass filter using the frequency-response masking method satisfying the specifications in Exercise 12.9. Compare the results obtained with and without an efficient ripple margin distribution, with respect to the total number of multiplications per output sample, with the results obtained in Exercise 12.9.
 12.24 Design a bandpass filter using the quadrature method satisfying the following specifications:

$$\begin{aligned}A_p &= 0.02 \text{ dB} \\A_r &= 40 \text{ dB} \\\Omega_{r_1} &= 0.068\pi \text{ rad/s} \\\Omega_{p_1} &= 0.07\pi \text{ rad/s} \\\Omega_{p_2} &= 0.95\pi \text{ rad/s} \\\Omega_{r_2} &= 0.952\pi \text{ rad/s} \\\Omega_s &= 2\pi \text{ rad/s.}\end{aligned}$$

Compare your results with the standard minimax filter.

- 12.25 Design a bandstop filter using the quadrature method satisfying the following specifications:

$$A_p = 0.2 \text{ dB}$$

$$A_r = 60 \text{ dB}$$

$$\omega_{p_1} = 0.33\pi \text{ rad/sample}$$

$$\omega_{r_1} = 0.34\pi \text{ rad/sample}$$

$$\omega_{r_2} = 0.46\pi \text{ rad/sample}$$

$$\omega_{p_2} = 0.47\pi \text{ rad/sample.}$$

Compare your results with the standard minimax filter.

- 12.26 Create a command in MATLAB that processes an input signal x with a frequency-response masking filter, taking advantage of the internal structure of this device. The command receives x , the frequency-response masking interpolating factor L , and the impulse responses for the base filter hb , positive masking filter hp , and negative masking filter hn . Use this command in Experiment 12.1 to process x_{DSB} with the designed frequency-response masking filter, generating the corresponding x_{SSB} signal.
- 12.27 Revisit Experiment 12.1 performing the following tasks:

- (a) Design a direct-form FIR filter using the following specifications:

$$A_p = 0.1 \text{ dB}$$

$$A_r = 45 \text{ dB}$$

$$\Omega_p = 2\pi 9700 \text{ Hz}$$

$$\Omega_r = 2\pi 10300 \text{ Hz.}$$

Compare the number of multiplications required by this filter with that required by the frequency-response masking designed in the experiment.

- (b) Design a new frequency-response masking filter suitable for $F_s = 100 \text{ kHz}$.
- (c) Design a new frequency-response masking filter to remove the upper baseband when $f_1 = 100$.

13.1 Introduction

The most widely used realizations for IIR filters are the cascade and parallel forms of second-order, and, sometimes, first-order, sections. The main advantages of these realizations come from their inherent modularity, which leads to efficient VLSI implementations, to simplified noise and sensitivity analyses, and to simple limit-cycle control. This chapter presents high-performance second-order structures, which are used as building blocks in high-order realizations. The concept of section ordering for the cascade form, which can reduce roundoff noise in the filter output, is introduced. Then we present a technique to reduce the output roundoff-noise effect known as error spectrum shaping. This is followed by consideration of some closed-form equations for the scaling coefficients of second-order sections for the design of parallel-form filters.

We also deal with other interesting realizations, such as the doubly complementary filters, made from allpass blocks, and IIR lattice structures, whose synthesis method is presented. A related class of realizations is the wave digital filters, which have very low sensitivity and also allow the elimination of zero-input and overflow limit cycles. The wave digital filters are derived from analog filter prototypes, employing the concepts of incident and reflected waves. The detailed design of these structures is presented in this chapter.

13.2 IIR parallel and cascade filters

The N th-order direct forms seen in Chapter 4, Figures 4.11–4.13, have roundoff-noise transfer functions $G_i(z)$ (see Figure 11.16) and scaling transfer functions $F_i(z)$ (see Figure 11.20) whose L_2 or L_∞ norms assume significantly high values. This occurs because these transfer functions have the same denominator polynomial as the filter transfer function, but without the zeros to attenuate the gain introduced by the poles close to the unit circle (see Exercise 11.17).

Also, we have that an N th-order direct-form filter is in general implemented as the ratio of two high-order polynomials. Since a variation in a single coefficient can cause significant variation on all the polynomial roots, such a filter tends to present high sensitivity to coefficient quantization.

In order to deal with the above problems, it is wise to implement high-order transfer functions through the cascade or parallel connection of second-order building blocks, instead

of using the direct-form realization. The second-order building blocks can be selected from a large collection of possibilities (Jackson, 1969; Szczupak & Mitra, 1975; Diniz, 1984). In this section, we deal with cascade and parallel realizations whose sub-filters are direct-form second-order sections. These realizations are canonic in the sense that they require the minimum number of multipliers, adders, and delays to implement a given transfer function.

Another advantage inherent to the cascade and parallel forms is their modularity, which makes them suitable for VLSI implementation.

13.2.1 Parallel form

A canonic parallel realization is shown in Figure 13.1, where the corresponding transfer function is given by

$$H(z) = h_0 + \sum_{i=1}^m H_i^p(z) = h_0 + \sum_{i=1}^m \frac{\gamma_{0i}z^2 + \gamma_{1i}z}{z^2 + m_{1i}z + m_{2i}}. \quad (13.1)$$

Many alternative parallel realizations can be generated by choosing different second-order sections, as seen, for instance, in Jackson (1969), Szczupak & Mitra (1975), and Diniz (1984). Another example is considered in Figure 13.56, Exercise 13.1. In general, the performances of the structures presented in Figures 13.1 and 13.56 are among the best for parallel forms employing canonic second-order sections.

It is easy to show that the scaling coefficients to avoid internal signal overflow in the form seen in Figure 13.1 are given by (see Exercise 11.17)

$$\lambda_i = \frac{1}{\|F_i(z)\|_p}, \quad (13.2)$$

where

$$F_i(z) = \frac{1}{D_i(z)} = \frac{1}{z^2 + m_{1i}z + m_{2i}}. \quad (13.3)$$

Naturally, the numerator coefficients of each section must be divided by λ_i , so that the overall filter transfer function remains unchanged.

Using Equation (11.60), one can show that the PSD of the output roundoff noise for the structure in Figure 13.1 is given by

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(2m + 1 + 3 \sum_{i=1}^m \frac{1}{\lambda_i^2} H_i^p(e^{j\omega}) H_i^p(e^{-j\omega}) \right) \quad (13.4)$$

when quantizations are performed before the additions. In this case, the output-noise variance, or the average power of the output noise, is then equal to

$$\sigma_o^2 = \sigma_e^2 \left(2m + 1 + 3 \sum_{i=1}^m \frac{1}{\lambda_i^2} \|H_i^p(e^{j\omega})\|_2^2 \right) \quad (13.5)$$

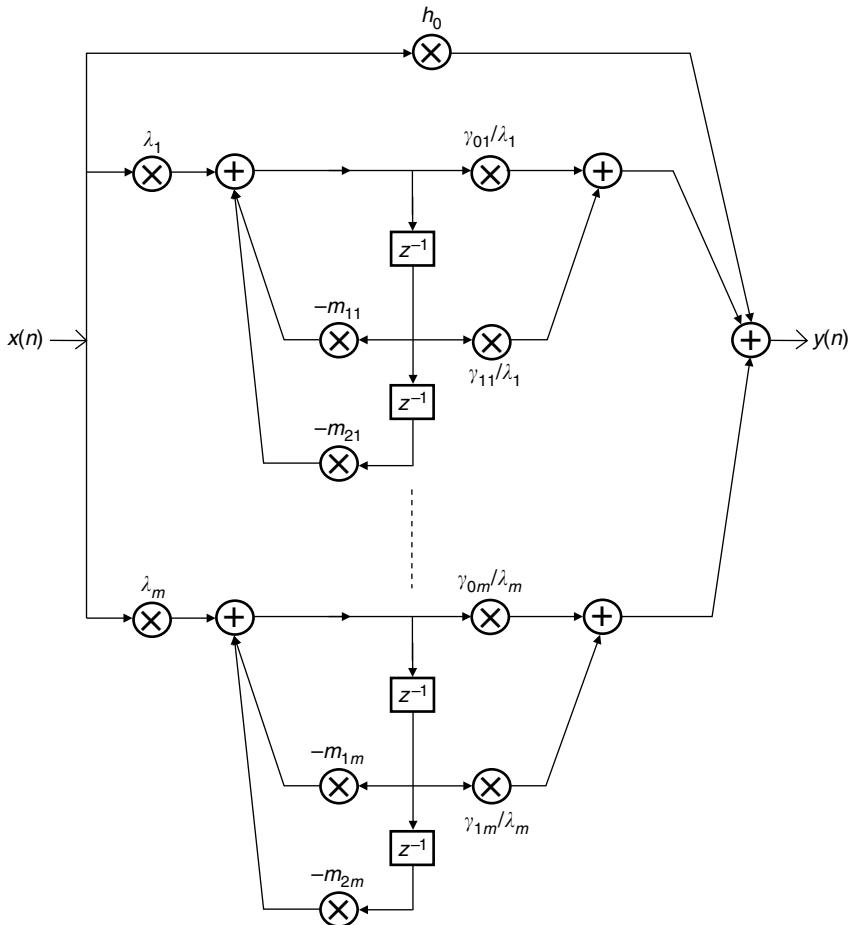


Fig. 13.1. Parallel structure with direct-form sections.

and the relative noise variance becomes

$$\sigma^2 = \frac{\sigma_o^2}{\sigma_e^2} = \left(2m + 1 + 3 \sum_{i=1}^m \frac{1}{\lambda_i^2} \|H_i^p(e^{j\omega})\|_2^2 \right). \quad (13.6)$$

For the cases where quantization is performed after the additions, the PSD becomes

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(1 + \sum_{i=1}^m \frac{1}{\lambda_i^2} H_i^p(e^{j\omega}) H_i^p(e^{-j\omega}) \right) \quad (13.7)$$

and then

$$\sigma^2 = \frac{\sigma_o^2}{\sigma_e^2} = 1 + \sum_{i=1}^m \frac{1}{\lambda_i^2} \|H_i^p(e^{j\omega})\|_2^2. \quad (13.8)$$

Although only even-order structures have been discussed so far, expressions for odd-order structures (which contain one first-order section) can be obtained in a very similar way.

In the parallel forms, as the positions of the zeros depend on the summation of several polynomials, which involves all coefficients in the realization, the precise positioning of the filter zeros becomes a difficult task. Such high sensitivity of the zeros to coefficient quantization constitutes the main drawback of the parallel forms for most practical implementations.

13.2.2 Cascade form

The cascade connection of direct-form second-order sections, depicted in Figure 13.2, has a transfer function given by

$$H(z) = \prod_{i=1}^m H_i(z) = \prod_{i=1}^m \frac{\gamma_{0i}z^2 + \gamma_{1i}z + \gamma_{2i}}{z^2 + m_{1i}z + m_{2i}}. \quad (13.9)$$

In this structure, the scaling coefficients are calculated as

$$\lambda_i = \frac{1}{\left\| \prod_{j=1}^{i-1} H_j(z) F_i(z) \right\|_p}, \quad (13.10)$$

with

$$F_i(z) = \frac{1}{D_i(z)} = \frac{1}{z^2 + m_{1i}z + m_{2i}} \quad (13.11)$$

as before. As illustrated in Figure 13.2, the scaling coefficient of each section can be incorporated with the output coefficients of the previous section. This strategy leads not only to a reduction in the multiplier count, but also to a possible decrease in the quantization noise at the filter output, since the number of nodes to be scaled is reduced.

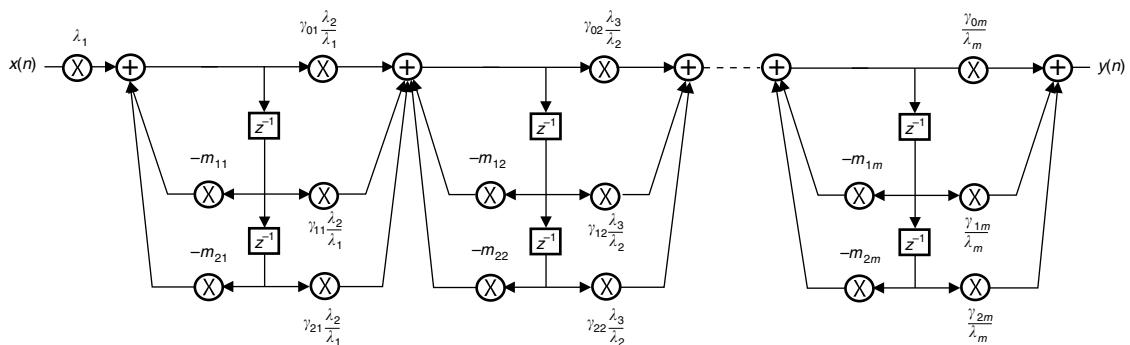


Fig. 13.2. Cascade of direct-form sections.

Assuming that the quantizations are performed before the additions, the output PSD for the cascade structure of Figure 13.2 is given by

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(3 + \frac{3}{\lambda_1^2} \prod_{i=1}^m H_i(e^{j\omega}) H_i(e^{-j\omega}) + 5 \sum_{j=2}^m \frac{1}{\lambda_j^2} \prod_{i=j}^m H_i(e^{j\omega}) H_i(e^{-j\omega}) \right). \quad (13.12)$$

The relative noise variance is then

$$\sigma^2 = \frac{\sigma_o^2}{\sigma_e^2} = \left(3 + \frac{3}{\lambda_1^2} \left\| \prod_{i=1}^m H_i(e^{j\omega}) \right\|_2^2 + 5 \sum_{j=2}^m \frac{1}{\lambda_j^2} \left\| \prod_{i=j}^m H_i(e^{j\omega}) \right\|_2^2 \right). \quad (13.13)$$

For the case where quantizations are performed after additions, $P_y(z)$ becomes

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(1 + \sum_{j=1}^m \frac{1}{\lambda_j^2} \prod_{i=j}^m H_i(e^{j\omega}) H_i(e^{-j\omega}) \right) \quad (13.14)$$

and then

$$\frac{\sigma_o^2}{\sigma_e^2} = 1 + \sum_{j=1}^m \frac{1}{\lambda_j^2} \left\| \prod_{i=j}^m H_i(e^{j\omega}) \right\|_2^2. \quad (13.15)$$

Two practical problems that need consideration in the design of cascade structures are:

- which pairs of poles and zeros will form each second-order section (the pairing problem);
- the ordering of the sections.

Both issues have a large effect on the output quantization noise. In fact, the roundoff noise and the sensitivity of cascade-form structures can be very high if an inadequate choice for the pairing and ordering is made (Jackson, 1969).

A rule of thumb for the pole–zero pairing in cascade form using second-order sections is to minimize the L_p norm of the transfer function of each section, for either $p = 2$ or $p = \infty$. The pairs of complex conjugate poles close to the unit circle, if not accompanied by zeros which are close to them, tend to generate sections whose norms of $H_i(z)$ are high. As a result, a natural rule is to pair the poles closest to the unit circle with the zeros that are closest to them. Then one should pick the poles second closest to the unit circle and pair them with the zeros, amongst the remaining, that are closest to them, and so on, until all sections are formed. Needless to say, when dealing with filters with real coefficients, most poles and zeros come in complex conjugate pairs, and in those cases the complex conjugate poles (and zeros) are jointly considered in the pairing process.

For section ordering, first we must notice that, for a given section of the cascade structure, the previous sections affect its scaling factor, whereas the following sections affect the noise gain. We then define a peaking factor that indicates how sharp the section frequency

response is:

$$P_i = \frac{\|H_i(z)\|_\infty}{\|H_i(z)\|_2}. \quad (13.16)$$

We now consider two separate cases (Jackson, 1970b):

- If we scale the filter using the L_2 norm, then the scaling coefficients tend to be large, and thus the signal-to-noise ratio at the output of the filter is in general not problematic. In these cases, it is interesting to choose the section ordering so that the maximum value of the output-noise PSD, $\|\text{PSD}\|_\infty$, is minimized. Section i amplifies the $\|\text{PSD}\|_\infty$ originally at its input by $(\lambda_i \|H_i(e^{j\omega})\|_\infty)^2$. Since, in the L_2 scaling, $\lambda_i = 1/\|H_i(e^{j\omega})\|_2$, then each section amplifies the $\|\text{PSD}\|_\infty$ by $(\|H_i(e^{j\omega})\|_\infty/\|H_i(e^{j\omega})\|_2)^2 = P_i^2$, the square of the peaking factor. Since the first sections affect the least number of noise sources, one should order the sections in decreasing order of peaking factors so as to minimize the maximum value of output-noise PSD.
- If we scale the filter using the L_∞ norm, then the scaling coefficients tend to be small, and thus the maximum peak value of the output-noise PSD is in general not problematic. In these cases, it is interesting to choose the section ordering so that the output signal-to-noise ratio is maximized; that is, the output-noise variance σ_o^2 is minimized. Section i amplifies the output-noise variance at its input by $(\lambda_i \|H_i(e^{j\omega})\|_2)^2$. Since, in the L_∞ scaling, $\lambda_i = 1/\|H_i(e^{j\omega})\|_\infty$, then each section amplifies the σ_o^2 by $(\|H_i(e^{j\omega})\|_2/\|H_i(e^{j\omega})\|_\infty)^2 = 1/P_i^2$, the inverse of the square of the peaking factor. Since the first sections affect the least number of noise sources, one should order the sections in increasing order of peaking factors so as to minimize σ_o^2 .

For other types of scaling, both ordering strategies are considered equally efficient.

Example 13.1. Design an elliptic bandpass filter satisfying the following specifications:

$$\left. \begin{array}{l} A_p = 0.5 \text{ dB} \\ A_r = 65 \text{ dB} \\ \Omega_{r1} = 850 \text{ rad/s} \\ \Omega_{p1} = 980 \text{ rad/s} \\ \Omega_{p2} = 1020 \text{ rad/s} \\ \Omega_{r2} = 1150 \text{ rad/s} \\ \Omega_s = 10\,000 \text{ rad/s} \end{array} \right\}. \quad (13.17)$$

Realize the filter using the parallel and cascade forms of second-order direct-form sections. Then scale the filters using L_2 norm and quantize the resulting coefficients to 9 bits, including the sign bit, and verify the results.

Solution

Using the `ellipord` and `ellip` commands in tandem, one can readily obtain the direct-form filter in MATLAB. We may then use the `residuez` command, and combine the resulting first-order sections, to determine the parallel structure, whose coefficients are shown in Table 13.1.

Table 13.1.

Parallel structure using direct-form second-order sections. Feedforward coefficient:
 $h_0 = -0.00015$.

Coefficient	Section 1	Section 2	Section 3
γ_0	-0.0077	-0.0079	0.0159
γ_1	0.0049	0.0078	-0.0128
m_1	-1.6268	-1.5965	-1.6054
m_2	0.9924	0.9921	0.9843

Table 13.2.

Scaled parallel structure using direct-form second-order sections. Feedforward coefficient:
 $h_0 = -0.00015$.

Coefficient	Section 1	Section 2	Section 3
λ	0.0711	0.0750	0.1039
γ_0/λ	-0.1077	-0.1055	0.1528
γ_1/λ	0.0692	0.1036	-0.1236
m_1	-1.6268	-1.5965	-1.6054
m_2	0.9924	0.9921	0.9843

Using L_2 norm, each block can be scaled by

$$\lambda_i = \frac{1}{\| F_i(z) \|_2} = \frac{1}{\| \frac{1}{D_i(z)} \|_2}, \quad (13.18)$$

which can be determined in MATLAB using the command lines

```
D_i = [1 m1i m2i];
F_i = freqz(1,D_i,npoints);
lambda_i = 1/sqrt(sum(abs(F_i).^2)/npoints);
```

where npoints is the number of points used in the freqz command. Scaling the second-order blocks using these factors, the resulting γ_0 and γ_1 coefficients are as given in Table 13.2, whereas the denominator coefficients m_1 and m_2 for each block remain unchanged.

Quantization of a given coefficient x using B bits (including the sign bit), can be performed in MATLAB using the command line

```
xQ = quant(x, 2^(-(B-1)));
```

Using this approach with $(B-1) = 8$ results in the coefficients shown in Table 13.3.

The cascade form can be obtained from the direct form in MATLAB using the tf2sos command. This yields the coefficients shown in Table 13.4. For the cascade form, we must perform the pole-zero pairing and the section ordering leading to a practical realization with low roundoff noise at the filter output. We may then scale all blocks according to some

Table 13.3.

Parallel structure using direct-form second-order sections quantized with 9 bits. Feedforward coefficient: $[h_0]_Q = 0.0000$.

Coefficient	Section 1	Section 2	Section 3
$[\lambda]_Q$	0.0703	0.0742	0.1055
$[\frac{v_0}{\lambda}]_Q$	-0.1094	-0.1055	0.1523
$[\frac{v_1}{\lambda}]_Q$	0.0703	0.1055	-0.1250
$[m_1]_Q$	-1.6250	-1.5977	-1.6055
$[m_2]_Q$	0.9922	0.9922	0.9844

Table 13.4.

Cascade structure using direct-form second-order sections. Gain constant: $h_0 = 1.4362 - 04$.

Coefficient	Section 1	Section 2	Section 3
γ_0	1.0000	1.0000	1.0000
γ_1	0.0000	-1.4848	-1.7198
γ_2	-1.0000	1.0000	1.0000
m_1	-1.6054	-1.5965	-1.6268
m_2	0.9843	0.9921	0.9924

Table 13.5.

Reordered cascade structure after coefficient scaling. Gain constant: $h'_0 = h_0 \lambda_2 = 0.0750$.

Coefficient	Section 1'	Section 2'	Section 3'
γ'_0	0.1605	0.1454	0.0820
γ'_1	-0.2383	-0.2501	0.0000
γ'_2	0.1605	0.1454	-0.0820
m'_1	-1.5965	-1.6268	-1.6054
m'_2	0.9921	0.9924	0.9843

predetermined norm. All these procedures for the cascade realization in this example are detailed in Experiment 13.1 included in Section 13.7.

After section reordering and coefficient scaling, the cascade realization is characterized as given in Table 13.5, and the quantized coefficients are as shown in Table 13.6. Notice that in this case the structure gain is not quantized to avoid it becoming zero.

The magnitude responses for the ideal filter and the quantized parallel and cascade realizations are depicted in Figure 13.3. Note that, despite the reasonably large number of bits used to represent the coefficients, the magnitude responses moved notably away from the ideal ones. This occurred, in part, because the designed elliptic filter has poles with high selectivity; that is, poles very close to the unit circle. In particular, the performance of

Table 13.6. Reordered cascade structure after coefficient quantization. Gain constant: $[h'_0]_Q = 0.0742$.

Coefficient	Section 1'	Section 2'	Section 3'
$[\gamma'_0]_Q$	0.1602	0.1445	0.0820
$[\gamma'_1]_Q$	-0.2383	-0.2500	0.0000
$[\gamma'_2]_Q$	0.1602	0.1445	-0.0820
$[m'_1]_Q$	-1.5977	-1.6250	-1.6055
$[m'_2]_Q$	0.9922	0.9922	0.9844

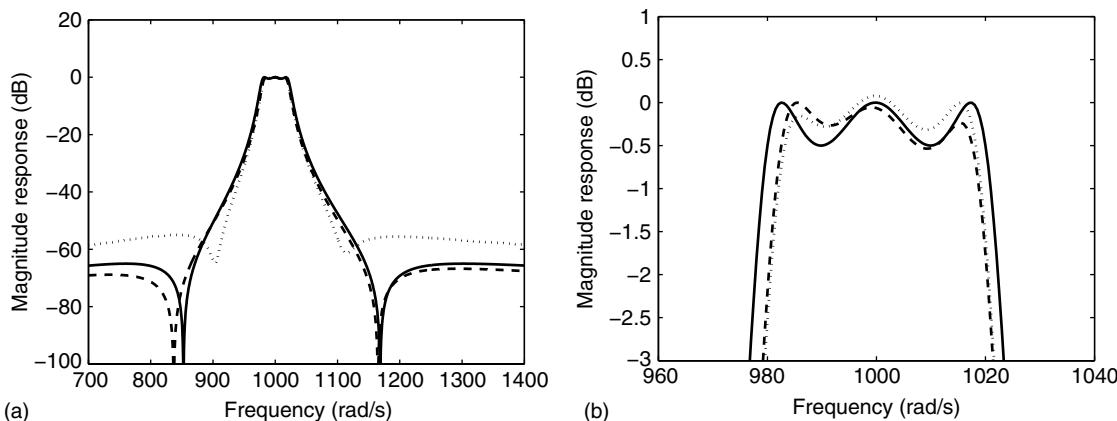


Fig. 13.3. Coefficient-quantization effects in the cascade and parallel forms, using direct-form second-order sections: (a) overall magnitude response; (b) passband detail. (Solid line – initial design; dashed line – cascade of direct-form sections (9 bits); dotted line – parallel of direct-form sections (9 bits).)

the quantized parallel realization became quite poor over the stopbands, as discussed in Section 11.2.1. \triangle

13.2.3 Error spectrum shaping

This subsection introduces a technique to reduce the quantization noise effects on digital filters by feeding back the quantization error. This technique is known as error spectrum shaping (ESS) or error feedback.

Consider every adder whose inputs include at least one nontrivial product which is followed by a quantizer. The ESS consists of replacing all these adders by a recursive structure, as illustrated in Figure 13.4, whose purpose is to introduce zeros in the output-noise PSD. Although Figure 13.4 depicts a second-order feedback network for the error signal, the order of this network can assume any value in practice. The ESS coefficients are chosen to minimize the output-noise PSD (Higgins & Munson, 1984; Laakso & Hartimo, 1992). In some

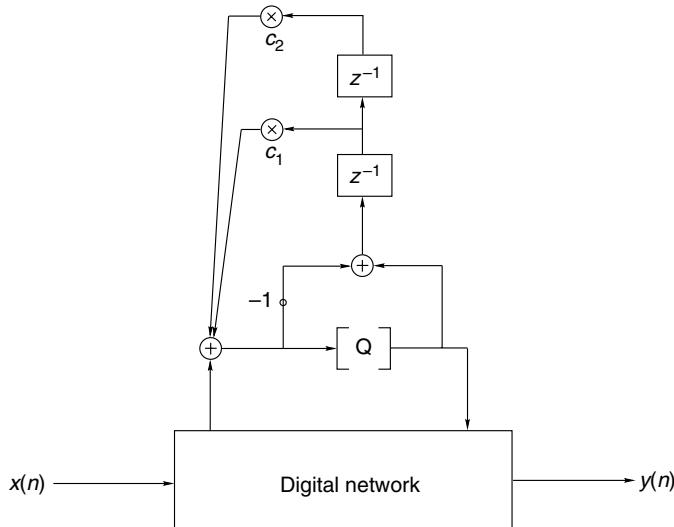


Fig. 13.4. Error spectrum shaping structure (Q denotes quantizer).

cases, these coefficients can be made trivial, such as 0, ± 1 , ± 2 , and still achieve sufficient noise reduction. Overall, the ESS approach can be interpreted as a form of recycling the quantization error signal, thus reducing the effects of signal quantization after a particular adder.

ESS can be applied to any digital filter structure and to any internal quantization node. However, since its implementation implies a cost overhead, ESS should be applied only at selected internal nodes, whose noise gains to the filter output are high. Structures having reduced number of quantization nodes (Diniz & Antoniou, 1985), for instance, are particularly suitable for ESS implementation. For example, the direct-form structure of Figure 4.11 requires a single ESS substitution for the whole filter.

For the cascade structure of direct-form second-order sections, as in Figure 13.2, each section j requires an ESS substitution. Let each feedback network be of second order. The values of $c_{1,j}$ and $c_{2,j}$ that minimize the output noise are calculated by solving the following optimization problem (Higgins & Munson, 1984):

$$\min_{c_{1,j}, c_{2,j}} \left\{ \left\| \left(1 + c_{1,j}z^{-1} + c_{2,j}z^{-2} \right) \prod_{i=j}^m H_i(e^{j\omega}) \right\|_2^2 \right\}. \quad (13.19)$$

The optimal values of $c_{1,j}$ and $c_{2,j}$ are given by

$$c_{1,j} = \frac{t_1 t_2 - t_1 t_3}{t_3^2 - t_1^2} \quad (13.20)$$

$$c_{2,j} = \frac{t_1^2 - t_2 t_3}{t_3^2 - t_1^2}, \quad (13.21)$$

where

$$t_1 = \int_{-\pi}^{\pi} \left| \prod_{i=j}^m H_i(e^{j\omega}) \right|^2 \cos \omega d\omega \quad (13.22)$$

$$t_2 = \int_{-\pi}^{\pi} \left| \prod_{i=j}^m H_i(e^{j\omega}) \right|^2 \cos(2\omega) d\omega \quad (13.23)$$

$$t_3 = \int_{-\pi}^{\pi} \left| \prod_{i=j}^m H_i(e^{j\omega}) \right|^2 d\omega. \quad (13.24)$$

The complete algebraic development behind Equations (13.20) and (13.21) is left as an exercise to the reader.

Using ESS, with the quantization performed after summation, the output RPSD, which is independent of σ_e^2 for the cascade design, is given by

$$\text{RPSD} = 10 \log \left\{ 1 + \sum_{j=1}^m \left| \frac{1}{\lambda_j} \left(1 + c_{1,j}z^{-1} + c_{2,j}z^{-2} \right) \prod_{i=j}^m H_i(e^{j\omega}) \right|^2 \right\}, \quad (13.25)$$

where λ_j is the scaling factor of section j . This expression explicitly shows how the ESS technique introduces zeros in the RPSD, thus allowing its subsequent reduction.

For a first-order ESS, the optimal value of $c_{1,j}$ would be

$$c_{1,j} = \frac{-t_1}{t_3}, \quad (13.26)$$

with t_1 and t_3 as above.

For the cascade realization with ESS, the most appropriate strategy for section ordering is to place the sections with poles closer to the unit circle at the beginning. This is because the poles close to the unit circle contribute to peaks in the noise spectrum, so this ordering strategy forces the noise spectrum to have a narrower band, making the action of the ESS zeros more effective.

As seen in Chang (1981), Singh (1985), Laakso *et al.* (1992), and Laakso (1993), the ESS technique can also be used to eliminate limit cycles. In such cases, absence of limit cycles can only be guaranteed at the quantizer output; hidden limit cycles may still exist in the internal loops (Butterweck *et al.*, 1984).

13.2.4 Closed-form scaling

In most types of digital filter implemented with fixed-point arithmetic, scaling is based on the L_2 and L_∞ norms of the transfer functions from the filter inputs to the inputs of the multipliers. Usually, the L_2 norm is computed through summation of a large number of

sample points (of the order of 200 or more) of the squared magnitude of the scaling transfer function. For the L_∞ norm, a search for the maximum magnitude of the scaling transfer function is performed over about the same number of sample points. It is possible, however, to derive simple closed-form expressions for the L_2 and L_∞ norms of second-order transfer functions. Such expressions are useful for scaling the sections independently, and greatly facilitate the design of parallel and cascade realizations of second-order sections (Bomar & Joseph, 1987; Laakso, 1992), and also of noncanonic structures with respect to the number of multipliers (Bomar, 1989).

For example, consider

$$H(z) = \frac{\gamma_1 z + \gamma_2}{z^2 + m_1 z + m_2}. \quad (13.27)$$

Using, for instance, the pole-residue approach for solving circular integrals, the corresponding L_2 norm is given by

$$\|H(e^{j\omega})\|_2^2 = \frac{\gamma_1^2 + \gamma_2^2 - 2\gamma_1\gamma_2[m_1/(m_2 + 1)]}{(1 - m_2^2)\{1 - [m_1/(m_2 + 1)]^2\}}. \quad (13.28)$$

For the L_∞ norm, we have to find the maximum of $|H(z)|^2$. By noticing that $|H(e^{j\omega})|^2$ is a function of $\cos \omega$, and $\cos \omega$ is limited to the interval $[-1, 1]$, we have that the maximum is either at the extrema $\omega = 0$ ($z = 1$), $\omega = \pi$ ($z = -1$), or at ω_0 such that $-1 \leq \cos \omega_0 \leq 1$. Therefore, the L_∞ norm is given by (Bomar & Joseph, 1987; Laakso, 1992)

$$\|H(e^{j\omega})\|_\infty^2 = \max \left\{ \left(\frac{\gamma_1 + \gamma_2}{1 + m_1 + m_2} \right)^2, \left(\frac{-\gamma_1 + \gamma_2}{1 - m_1 + m_2} \right)^2, \frac{\gamma_1^2 + \gamma_2^2 + 2\gamma_1\gamma_2\zeta}{4m_2[(\zeta - \eta)^2 + \nu]} \right\}, \quad (13.29)$$

where

$$\eta = \frac{-m_1(1 + m_2)}{4m_2} \quad (13.30)$$

$$\nu = \left(1 - \frac{m_1^2}{4m_2} \right) \frac{(1 - m_2)^2}{4m_2} \quad (13.31)$$

$$\zeta = \begin{cases} \text{sat}(\eta), & \text{for } \gamma_1\gamma_2 = 0 \\ \text{sat} \left\{ \nu \left[\sqrt{\left(1 + \frac{\eta}{\nu} \right)^2 + \frac{\nu}{\nu^2}} - 1 \right] \right\}, & \text{for } \gamma_1\gamma_2 \neq 0, \end{cases} \quad (13.32)$$

with

$$\nu = \frac{\gamma_1^2 + \gamma_2^2}{2\gamma_1\gamma_2} \quad (13.33)$$

and the function $\text{sat}(\cdot)$ being defined as

$$\text{sat}(x) = \begin{cases} 1, & \text{for } x > 1 \\ -1, & \text{for } x < -1 \\ x, & \text{for } -1 \leq x \leq 1. \end{cases} \quad (13.34)$$

The derivations of the L_2 and L_∞ norms of $H(z)$ are left as an exercise for the interested reader.

Example 13.2. Consider the transfer function

$$H(z) = \frac{0.5z^2 - z + 1}{(z^2 - z + 0.5)(z + 0.5)}. \quad (13.35)$$

- (a) Show cascade and parallel decompositions using Type 1 direct-form sections.
- (b) Scale the filters using L_2 norm.
- (c) Calculate the output noise variances.

Solution

The cascade decomposition is

$$H(z) = \frac{0.5z^2 - z + 1}{z^2 - z + 0.5} \cdot \frac{1}{z + 0.5}, \quad (13.36)$$

whereas the parallel decomposition is

$$H(z) = \frac{-\frac{8}{5}z + \frac{7}{5}}{2z^2 - 2z + 1} + \frac{\frac{13}{10}}{z + 0.5}. \quad (13.37)$$

The second-order section of the cascade design is an allpass, so that we only have to scale the internal nodes of the section. The result is obtained by employing Equation (13.28); that is:

$$\|F_1(z)\|_2^2 = \left\| \frac{1}{D(z)} \right\|_2^2 = \frac{1}{(1 - 0.25) \left[1 - \left(\frac{-1}{1.5} \right)^2 \right]} = 1.44, \quad (13.38)$$

so that

$$\lambda_1 = \sqrt{\frac{1}{1.44}} = \frac{1}{1.2} = 0.8333. \quad (13.39)$$

For the second section the scaling factor should be

$$\lambda_2 = \sqrt{0.75} = \sqrt{1 - (0.5)^2}. \quad (13.40)$$

Similarly the scaling factors for the parallel realization are given by

$$\lambda_1 = \sqrt{\frac{1}{1.44}} = \frac{1}{1.2} = 0.8333 \quad (13.41)$$

and

$$\lambda_2 = \frac{10}{13} \sqrt{0.75} = 0.6667 \quad (13.42)$$

respectively.

The relative output noise variance for the cascade design is given by

$$\frac{\sigma_y^2}{\sigma_e^2} = 3 \frac{1}{\lambda_1^2} \frac{1}{0.75} + 4 \frac{1}{\lambda_2^2} \frac{1}{0.75} + 1 = 13.88. \quad (13.43)$$

Using the result of Equation (13.28) we can compute the L_2 norm of the second-order section in the parallel solution:

$$\|H_1(z)\|_2^2 = \frac{1}{4} \left[\frac{\frac{64}{25} + \frac{49}{25} + \frac{2 \times 8 \times 7}{25} \times \frac{-1}{1.5}}{(1 - 0.25)(1 - (-1/1.5)^2)} \right] = \frac{1}{4} \left[\frac{38.333/25}{0.41666} \right] = 0.92, \quad (13.44)$$

so that the relative output noise variance for the parallel design is given by

$$\frac{\sigma_y^2}{\sigma_e^2} = 3 \frac{1}{\lambda_1^2} \|H_1(z)\|_2^2 + \frac{1}{\lambda_2^2} \frac{1}{0.75} + 4 = 10.98. \quad (13.45)$$

△

13.3 State-space sections

The state-space approach allows the formulation of a design method for IIR digital filters with minimum roundoff noise. The theory behind this elegant design method was originally proposed by Mullis and Roberts (1976a, b) and Hwang (1977). For a filter of order N , the minimum noise method leads to a realization entailing $(N + 1)^2$ multiplications. This multiplier count is very high for most practical implementations, which induced investigators to search for realizations which could approach the minimum noise performance while employing a reasonable number of multiplications. A good trade-off is achieved if we realize high-order filters using parallel or cascade forms, where the second-order sections are minimum-noise state-space structures. In this section, we study two commonly used second-order state-space sections suitable for such approaches.

13.3.1 Optimal state-space sections

The second-order state-space structure shown in Figure 13.5 can be described by

$$\left. \begin{array}{l} \mathbf{x}(n+1) = \mathbf{Ax}(n) + \mathbf{Bu}(n) \\ y(n) = \mathbf{C}^T \mathbf{x}(n) + \mathbf{Du}(n) \end{array} \right\}, \quad (13.46)$$

where $\mathbf{x}(n)$ is a column vector representing the outputs of the delays, $y(n)$ is a scalar, and

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (13.47)$$

$$\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (13.48)$$

$$\mathbf{C}^T = [c_1 \ c_2] \quad (13.49)$$

$$\mathbf{D} = [d]. \quad (13.50)$$

The overall transfer function, described as a function of the matrix elements related to the state-space formulation, is given by

$$H(z) = \mathbf{C}^T [\mathbf{I}z - \mathbf{A}]^{-1} \mathbf{B} + \mathbf{D}. \quad (13.51)$$

The second-order state-space structure can realize transfer functions described by

$$H(z) = d + \frac{\gamma_1 z + \gamma_2}{z^2 + m_1 z + m_2}. \quad (13.52)$$

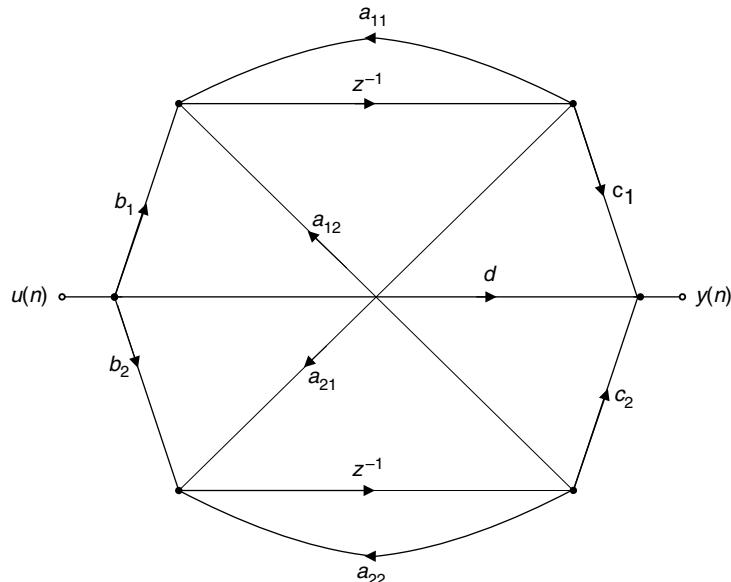


Fig. 13.5. Second-order state-space structure.

Given $H(z)$ in the form of Equation (13.52), an optimal design in the sense of minimizing the output roundoff noise can be derived, since the state-space structure has more coefficients than the minimum required. To explore this feature, we examine, without proof, a theorem first proposed by Mullis and Roberts (1976a,b). The design procedure resulting from the theorem generates realizations with a minimum-variance output noise, provided that the L_2 norm is employed to determine the scaling factor. It is interesting to notice that, despite being developed for filters using L_2 scaling, the minimum-noise design also leads to low-noise filters scaled using the L_∞ norm.

Note that, in the remainder of this subsection, primed variables will indicate filter parameters after scaling.

Theorem 13.1. *The necessary and sufficient conditions to obtain an output noise with minimum variance in a state-space realization are given by*

$$\mathbf{W}' = \mathbf{R}\mathbf{K}'\mathbf{R} \quad (13.53)$$

$$K'_{ii} W'_{ii} = K'_{jj} W'_{jj} \quad (13.54)$$

for $i, j = 1, 2, \dots, N$, where N is the filter order, \mathbf{R} is an $N \times N$ diagonal matrix, and

$$\mathbf{K}' = \sum_{k=0}^{\infty} \mathbf{A}'^k \mathbf{B}' \mathbf{B}'^H (\mathbf{A}'^k)^H \quad (13.55)$$

$$\mathbf{W}' = \sum_{k=0}^{\infty} (\mathbf{A}'^k)^H \mathbf{C}'^H \mathbf{C}' \mathbf{A}'^k, \quad (13.56)$$

where the H indicates the conjugate and transpose operations. \diamond

It can be shown (see Exercise 13.6) that

$$K'_{ii} = \|F'_i(e^{j\omega})\|_2^2 \quad (13.57)$$

$$W'_{ii} = \|G'_i(e^{j\omega})\|_2^2 \quad (13.58)$$

for $i = 1, 2, \dots, N$, where $F'_i(z)$ is the transfer function from the scaled filter input to the state variable $x_i(k+1)$ and $G'_i(z)$ is the transfer function from the state variable $x_i(k)$ to the scaled filter output (see Figures 11.20 and 11.16).

Then, from Equations (13.57) and (13.58), we have that, in the frequency domain, Equation (13.54) is equivalent to

$$\|F'_i(e^{j\omega})\|_2^2 \|G'_i(e^{j\omega})\|_2^2 = \|F'_j(e^{j\omega})\|_2^2 \|G'_j(e^{j\omega})\|_2^2. \quad (13.59)$$

In the case of second-order filters, if the L_2 scaling is performed, then

$$K'_{11} = K'_{22} = \|F'_1(e^{j\omega})\|_2^2 = \|F'_2(e^{j\omega})\|_2^2 = 1 \quad (13.60)$$

and then, from Theorem 13.1, the following equality must hold:

$$W'_{11} = W'_{22}. \quad (13.61)$$

Similarly, we can conclude that we must have

$$\|G'_1(e^{j\omega})\|_2^2 = \|G'_2(e^{j\omega})\|_2^2, \quad (13.62)$$

indicating that the contributions of the internal noise sources to the output-noise variance are identical.

The conditions $K'_{ii} = \|F'_i(e^{j\omega})\|_2^2 = 1$ and $W'_{ii} = W'_{jj}$, for all i and j , show that Equation (13.53) can only be satisfied if

$$\mathbf{R} = \alpha \mathbf{I}; \quad (13.63)$$

as a consequence, the optimality condition of Theorem 13.1 is equivalent to

$$\mathbf{W}' = \alpha^2 \mathbf{K}'. \quad (13.64)$$

For a second-order filter, since \mathbf{W}' and \mathbf{K}' are symmetric and their respective diagonal elements are identical, Equation (13.64) remains valid if we rewrite it as

$$\mathbf{W}' = \alpha^2 \mathbf{J} \mathbf{K}' \mathbf{J}, \quad (13.65)$$

where \mathbf{J} is the reverse identity matrix defined as

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (13.66)$$

By employing the definitions of \mathbf{W}' and \mathbf{K}' in Equations (13.55) and (13.56), Equation (13.65) is satisfied when

$$\mathbf{A}'^T = \mathbf{J} \mathbf{A}' \mathbf{J} \quad (13.67)$$

$$\mathbf{C}'^T = \alpha \mathbf{J} \mathbf{B}', \quad (13.68)$$

or, equivalently:

$$a'_{11} = a'_{22} \quad (13.69)$$

$$\frac{b'_1}{b'_2} = \frac{c'_2}{c'_1}. \quad (13.70)$$

Then, the following procedure can be derived for designing optimal second-order state-space sections.

(i) For filters with complex conjugate poles, choose an antisymmetric matrix \mathbf{A} such that

$$\left. \begin{aligned} a_{11} &= a_{22} = \text{real part of the poles} \\ -a_{12} &= a_{21} = \text{imaginary part of the poles} \end{aligned} \right\}. \quad (13.71)$$

Note that the first optimality condition (Equation (13.69)) is satisfied by this choice of **A**. The coefficients of matrix **A** can be calculated as a function of the coefficients of the transfer function $H(z)$ using

$$\left. \begin{array}{l} a_{11} = -\frac{m_1}{2} \\ a_{12} = -\sqrt{m_2 - \frac{m_1^2}{4}} \\ a_{21} = -a_{12} \\ a_{22} = a_{11} \end{array} \right\}. \quad (13.72)$$

Also, compute the parameters b_1 , b_2 , c_1 , and c_2 , using

$$\left. \begin{array}{l} b_1 = \sqrt{\frac{\sigma + \gamma_2 + a_{11}\gamma_1}{2a_{21}}} \\ b_2 = \frac{\gamma_1}{2b_1} \\ c_1 = b_2 \\ c_2 = b_1 \end{array} \right\}, \quad (13.73)$$

where

$$\sigma = \sqrt{\gamma_2^2 - \gamma_1\gamma_2 m_1 + \gamma_1^2 m_2}. \quad (13.74)$$

For real poles, the matrix **A** must be of the form

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ a_2 & a_1 \end{bmatrix}, \quad (13.75)$$

where

$$\left. \begin{array}{l} a_1 = \frac{1}{2}(p_1 + p_2) \\ a_2 = \pm \frac{1}{2}(p_1 - p_2) \end{array} \right\}, \quad (13.76)$$

with p_1 and p_2 denoting the real poles. The elements of vectors **B** and **C** are given by

$$\left. \begin{array}{l} b_1 = \pm \sqrt{\frac{\pm\sigma + \gamma_2 + a_1\gamma_1}{2a_2}} \\ b_2 = \frac{\gamma_2}{2b_1} \\ c_1 = b_2 \\ c_2 = b_1 \end{array} \right\}, \quad (13.77)$$

with σ as before.

This synthesis approach is only valid if $\gamma_2^2 - \gamma_1\gamma_2m_1 + \gamma_1^2m_2 > 0$, which may not occur for real poles in some designs. Solutions to this problem have been proposed by Kim (1980). However, it is worth observing that real poles can be implemented separately in first-order sections. In practice, we very seldom find more than one real pole in filter approximations.

- (ii) Scale the filter using L_2 norm, through the following similarity transformation:

$$(\mathbf{A}', \mathbf{B}', \mathbf{C}', d) = (\mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \mathbf{T}^{-1}\mathbf{B}, \mathbf{C}\mathbf{T}, d), \quad (13.78)$$

where

$$\mathbf{T} = \begin{bmatrix} \|F_1(e^{j\omega})\|_2 & 0 \\ 0 & \|F_2(e^{j\omega})\|_2 \end{bmatrix}. \quad (13.79)$$

For the L_∞ -norm scaling, use the following scaling matrix:

$$\mathbf{T} = \begin{bmatrix} \|F_1(e^{j\omega})\|_\infty & 0 \\ 0 & \|F_2(e^{j\omega})\|_\infty \end{bmatrix}. \quad (13.80)$$

In this case the resulting second-order section is not optimal in the L_∞ sense. Nevertheless, practical results indicate that the solution is close to the optimal one.

Defining the vectors $\mathbf{f}(z) = [F_1(z), F_2(z)]^T$ and $\mathbf{g}(z) = [G_1(z), G_2(z)]^T$, the effects of the scaling matrix on these vectors are

$$\mathbf{f}'(z) = [z\mathbf{I} - \mathbf{A}']^{-1} \mathbf{B}' = \mathbf{T}^{-1} \mathbf{f}(z) \quad (13.81)$$

$$\mathbf{g}'(z) = [z\mathbf{I} - \mathbf{A}'^T]^{-1} \mathbf{C}'^T = (\mathbf{T}^{-1})^T \mathbf{g}(z). \quad (13.82)$$

The transfer functions $F_i(z)$ from the input node, where $u(n)$ is inserted, to the state variables $x_i(n)$ of the system $(\mathbf{A}, \mathbf{B}, \mathbf{C}, d)$ are given by

$$F_1(z) = \frac{b_1 z + (b_2 a_{12} - b_1 a_{22})}{z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21})} \quad (13.83)$$

$$F_2(z) = \frac{b_2 z + (b_1 a_{21} - b_2 a_{11})}{z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21})}. \quad (13.84)$$

The expressions for the transfer functions from the internal nodes, namely from the signals $x_i(n+1)$ to the section output node, are

$$G_1(z) = \frac{c_1 z + (c_2 a_{21} - c_1 a_{22})}{z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21})} \quad (13.85)$$

$$G_2(z) = \frac{c_2 z + (c_1 a_{12} - c_2 a_{11})}{z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21})}. \quad (13.86)$$

The output roundoff-noise PSD, considering quantization before the adders, for the section-optimal state-space structure in cascade form can be expressed as

$$\Gamma_Y(e^{j\omega}) = 3\sigma_e^2 \sum_{j=1}^m \prod_{l=j+1}^m H_l(e^{j\omega})H_l(e^{-j\omega}) \left(1 + \sum_{i=1}^2 G'_{ij}(e^{j\omega})G'_{ij}(e^{-j\omega}) \right), \quad (13.87)$$

where G'_{ij} , for $i = 1, 2$, are the noise transfer functions of the j th scaled section, and we consider $\prod_{l=m+1}^m H_l(z)H_l(z^{-1}) = 1$.

The scaling in the state-space sections of the cascade form is performed internally, using the transformation matrix \mathbf{T} . In order to calculate the elements of matrix \mathbf{T} , we can use the same procedure as in the cascade direct form, taking the effect of previous blocks into consideration.

In the case of the parallel form, the expression for the output roundoff-noise PSD, assuming quantization before additions, is

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left(2m + 1 + 3 \sum_{j=1}^m \sum_{i=1}^2 G'_{ij}(e^{j\omega})G'_{ij}(e^{-j\omega}) \right). \quad (13.88)$$

The expressions for the output roundoff-noise PSD assuming quantization after additions can be easily derived.

13.3.2 State-space sections without limit cycles

This section presents a design procedure for a second-order state-space section that is free from constant-input limit cycles.

The transition matrix related to the section-optimal structure, described in the previous subsection (see Equation (13.69)), has the following general form:

$$\mathbf{A} = \begin{bmatrix} a & -\zeta/\sigma \\ \zeta\sigma & a \end{bmatrix}, \quad (13.89)$$

where a , ζ , and σ are constants. This form is the most general for \mathbf{A} that allows the realization of complex conjugate poles and the elimination of zero-input limit cycles.

As studied in Section 7.6.3, one can eliminate zero-input limit cycles on a recursive structure if there is a positive-definite diagonal matrix \mathbf{G} , such that $(\mathbf{G} - \mathbf{A}^T \mathbf{G} \mathbf{A})$ is positive semidefinite. For second-order sections, this condition is satisfied if

$$a_{12}a_{21} \geq 0 \quad (13.90)$$

or

$$a_{12}a_{21} < 0 \text{ and } |a_{11} - a_{22}| + \det(\mathbf{A}) \leq 1. \quad (13.91)$$

In the section-optimal structures, the elements of matrix \mathbf{A} automatically satisfy Equation (13.90), since $a_{11} = a_{22}$ and $\det(\mathbf{A}) \leq 1$, for stable filters.

Naturally, the quantization performed at the state variables still must be such that

$$|[x_i(k)]_Q| \leq |x_i(k)|, \text{ for all } k, \quad (13.92)$$

where $[x]_Q$ denotes the quantized value of x . This condition can be easily guaranteed by using, for example, magnitude truncation and saturation arithmetic to deal with overflow.

If we also want to eliminate constant-input limit cycles, according to Theorem 11.3, then the values of the elements of $\mathbf{p}u_0$, where $\mathbf{p} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$, must be machine representable. In order to guarantee this condition independently of u_0 , the column vector \mathbf{p} must assume one of the forms

$$\mathbf{p} = \begin{cases} [\pm 1 \ 0]^T & \text{(Case I)} \\ [0 \ \pm 1]^T & \text{(Case II)} \\ [\pm 1 \ \pm 1]^T & \text{(Case III).} \end{cases} \quad (13.93)$$

For each case above, vector \mathbf{B} of the state-space structure must be appropriately chosen to ensure the elimination of constant-input limit cycles, as given by:

- *Case I*

$$\left. \begin{array}{l} b_1 = \pm(1 - a_{11}) \\ b_2 = \mp a_{21} \end{array} \right\}; \quad (13.94)$$

- *Case II*

$$\left. \begin{array}{l} b_1 = \mp a_{12} \\ b_2 = \pm(1 - a_{22}) \end{array} \right\}; \quad (13.95)$$

- *Case III*

$$\left. \begin{array}{l} b_1 = \mp a_{12} \pm (1 - a_{11}) \\ b_2 = \mp a_{21} \pm (1 - a_{22}) \end{array} \right\}. \quad (13.96)$$

Based on the values of b_1 and b_2 , for each case, it is possible to generate three structures (Diniz & Antoniou, 1986), henceforth referred to as Structures I, II, and III. Figure 13.6 depicts Structure I, where it can be seen that b_1 and b_2 are formed without actual multiplications. As a consequence, the resulting structure is more economical than the optimal second-order state-space structure. Similar results apply to all three structures. In fact, Structures I and II have the same complexity, whereas Structure III requires five extra additions, if we consider the adders needed for the elimination of constant-input limit cycles. For that reason, in what follows, we present the design for Structure I. If desired, the complete design for Structure III can be found in Sarcinelli Filho and Camponêz (1997, 1998).

For Structure I, we have that

$$\left. \begin{array}{l} a_{11} = a \\ a_{12} = -\zeta/\sigma \\ a_{21} = \sigma\zeta \\ a_{22} = a_{11} \end{array} \right\} \quad (13.97)$$

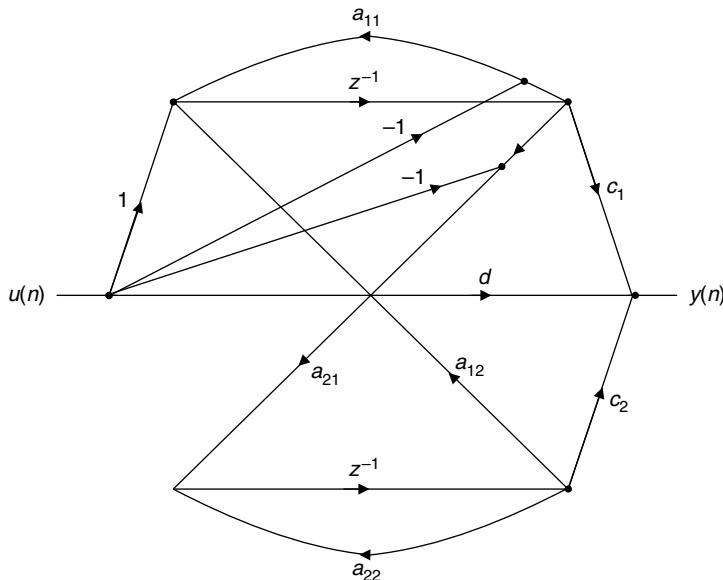


Fig. 13.6. State-space structure free from limit cycles.

and

$$\left. \begin{aligned} b_1 &= 1 - a_{11} \\ b_2 &= -a_{21} \\ c_1 &= \frac{\gamma_1 + \gamma_2}{1 + m_1 + m_2} \\ c_2 &= -\frac{(m_1 + 2m_2)\gamma_1 + (2 + m_1)\gamma_2}{2\sigma\zeta(1 + m_1 + m_2)} \end{aligned} \right\}, \quad (13.98)$$

where

$$a = -\frac{m_1}{2}, \quad (13.99)$$

$$\zeta = \sqrt{\left(m_2 - \frac{m_1^2}{4}\right)}, \quad (13.100)$$

and σ is a free parameter whose choice is explained below. From the above equations:

$$\frac{b_1}{b_2} = -\frac{2 + m_1}{2\sigma\zeta} \quad (13.101)$$

$$\frac{c_2}{c_1} = -\frac{(m_1 + 2m_2)\gamma_1 + (2 + m_1)\gamma_2}{2\sigma\zeta(\gamma_1 + \gamma_2)}. \quad (13.102)$$

Therefore, in this case, the optimality condition derived from Theorem 13.1 (Equations (13.69) and (13.70)) is only met if

$$\frac{\gamma_1}{\gamma_2} = \frac{m_1 + 2}{m_2 - 1}. \quad (13.103)$$

Usually, this condition is violated, showing that the state-space structure which is free from constant-input limit cycles does not lead to minimum output noise. In practice, however, it is found that the performance of Structure I is very close to the optimal. More specifically, in the case where the zeros of $H(z)$ are placed at $z = 1$, the values of γ_1 and γ_2 are

$$\left. \begin{aligned} \gamma_1 &= -\gamma_0(2 + m_1) \\ \gamma_2 &= \gamma_0(1 - m_2) \end{aligned} \right\}, \quad (13.104)$$

which satisfy Equation (13.103). Hence, in the special case of filters with zeros at $z = 1$, Structure I also leads to minimum output noise.

The signal-to-noise ratio in a digital filter implemented using fixed-point arithmetic increases by increasing the internal signal dynamic range, which in turn increases by equalizing the maximum signal levels at the input of the quantizers. For the three structures, the maximum signal levels must be equalized at the state variables. Parameter σ is usually used to optimize the dynamic range of the state variables.

For Structure I, the transfer functions from the input node $u(k)$ to the state variables $x_i(k)$ are given by

$$F_1(z) = \frac{(1 - a)z + (\zeta^2 - a + a^2)}{z^2 - 2az + (a^2 + \zeta^2)} \quad (13.105)$$

$$F_2(z) = \sigma F_2''(z), \quad (13.106)$$

where

$$F_2''(z) = \frac{-\zeta z + \zeta}{z^2 - 2az + (a^2 + \zeta^2)}. \quad (13.107)$$

Equalization of the maximum signal level at the state variables is then achieved by forcing

$$\|F_1(z)\|_p = \|\sigma F_2''(z)\|_p, \quad (13.108)$$

where $p = \infty$ or $p = 2$. Consequently, we must have

$$\sigma = \frac{\|F_1(z)\|_p}{\|F_2''(z)\|_p}. \quad (13.109)$$

The transfer functions from the state variables $x_i(k + 1)$ to the output in the structure of Figure 13.6 can be expressed as

$$G_1(z) = \frac{c_1}{2} \frac{2z + (m_1 + 2\xi)}{z^2 + m_1 z + m_2} \quad (13.110)$$

$$G_2(z) = \frac{c_2}{2} \frac{2z + [\alpha_1 + (2\xi^2/\xi)]}{z^2 + \alpha_1 z + \alpha_2}, \quad (13.111)$$

where

$$\xi = \frac{-(\alpha_1 + 2\alpha_2)\beta_1 + (2 + \alpha_1)\beta_2}{2(\beta_1 + \beta_2)}. \quad (13.112)$$

The RPSD expression for Structure I is then

$$\begin{aligned} \text{RPSD} &= 10 \log \left\{ 2 \left| G'_1(e^{j\omega}) \right|^2 + 2 \left| G'_2(e^{j\omega}) \right|^2 + 3 \right\} \\ &= 10 \log \left\{ 2 \frac{1}{\lambda^2} \left| G''_1(e^{j\omega}) \right|^2 + 2 \frac{1}{\lambda^2 \sigma^2} \left| G''_2(e^{j\omega}) \right|^2 + 3 \right\}, \end{aligned} \quad (13.113)$$

where $G'_1(e^{j\omega})$ and $G'_2(e^{j\omega})$ are the noise transfer functions for the scaled filter, λ is the scaling factor, and $G''_1(e^{j\omega})$ and $G''_2(e^{j\omega})$ are functions generated from $G'_1(e^{j\omega})$ and $G'_2(e^{j\omega})$, when we remove the parameters σ and λ from them.

We now show that choosing σ according to Equation (13.109) leads to the minimization of the output noise. From Equation (13.113), we can infer that the output noise can be minimized when σ and λ are maximized, with the scaling coefficient given by

$$\lambda = \frac{1}{\max \{ \|F_1(z)\|_p, \|F_2(z)\|_p \}}. \quad (13.114)$$

However, $F_1(z)$ is not a function of σ , and, as a consequence, the choice of $\|F_2(z)\|_p = \|F_1(z)\|_p$ leads to a maximum value for λ . On the other hand, the maximum value that σ can assume, without reducing the value of λ , is given by

$$\sigma = \frac{\|F_1(e^{j\omega})\|_p}{\|F''_2(e^{j\omega})\|_p}, \quad (13.115)$$

from which we can conclude that this choice for σ minimizes the roundoff noise at the filter output.

In order to design a cascade structure without limit cycles which realizes

$$H(z) = \prod_{i=1}^m H_i(z) = H_0 \prod_{i=1}^m \frac{z^2 + \gamma'_{1i}z + \gamma'_{2i}}{z^2 + \alpha_{1i}z + \alpha_{2i}} = \prod_{i=1}^m (d_i + H'_i(z)), \quad (13.116)$$

with $H'_i(z)$ described in the form of the first term of the right side of Equation (13.51), one must adopt the following procedure for Structure I:

- (i) Calculate σ_i and λ_i for each section using

$$\sigma_i = \frac{\left\| F_{1i}(z) \prod_{j=1}^{i-1} H_j(z) \right\|_p}{\left\| F''_{2i}(z) \prod_{j=1}^{i-1} H_j(z) \right\|_p} \quad (13.117)$$

$$\lambda_i = \frac{1}{\left\| F_{2i}(z) \prod_{j=1}^{i-1} H_j(z) \right\|_p}. \quad (13.118)$$

- (ii) Determine a and ζ from Equations (13.99) and (13.100).
- (iii) Compute the coefficients of \mathbf{A} , \mathbf{B} , and \mathbf{C} using Equations (13.97) and (13.98).
- (iv) Calculate the multiplier coefficients d_i by

$$d_i = \begin{cases} \frac{1}{\left| \prod_{j=1}^i H_j(z) \right|_p}, & \text{for } i = 1, 2, \dots, (m-1) \\ \frac{H_0}{\prod_{j=1}^{m-1} d_j}, & \text{for } i = m \end{cases} \quad (13.119)$$

in order to satisfy overflow constraints at the output of each section.

- (v) Incorporate the scaling multipliers of sections $2, 3, \dots, m$ into the output multipliers of sections $1, 2, \dots, (m-1)$, generating

$$\left. \begin{array}{l} c'_{1i} = c_{1i} \frac{\lambda_{i+1}}{\lambda_i} \\ c'_{2i} = c_{2i} \frac{\lambda_{i+1}}{\lambda_i} \\ d'_i = d_i \frac{\lambda_{i+1}}{\lambda_i} \end{array} \right\}. \quad (13.120)$$

The cascade-form design procedures employing the section-optimal and the limit-cycle-free state-space structures use the same strategy for pairing the poles and zeros as those employing direct-form sections. The section ordering depends on the definition of a parameter u_j , given by

$$u_j = \sum_{i=1}^2 \frac{\max \{|F_{ij}(e^{j\omega})|\}}{\min \{|F_{ij}(e^{j\omega})|\}}, \quad (13.121)$$

where the maximum is computed for all ω , while the minimum is calculated solely within the passband. According to Figure 13.7, for an odd value of sections m the ordering consists of placing the section with highest value for u_j to be the central section. For an even number of sections, the two with largest u_j are placed in the central positions; these are called first and second middle sections. For odd m , the sections prior to and following the central block are chosen from the remaining sections so as to minimize the summation of u_a and u_b (see Figure 13.7), one referred to the combination of the central section and all previous ones and the other referred to the combination of the central section and all the following ones. For even m , the sections before and after the central sections are chosen, among the remaining ones, in order to minimize the summation of u_a and u_b (see Figure 13.7), one referred to the combination of the first middle section and all previous sections and the other referred to the combination of the second middle section and the sections following it (Kim, 1980). This approach is continuously employed until all second-order blocks have been ordered.

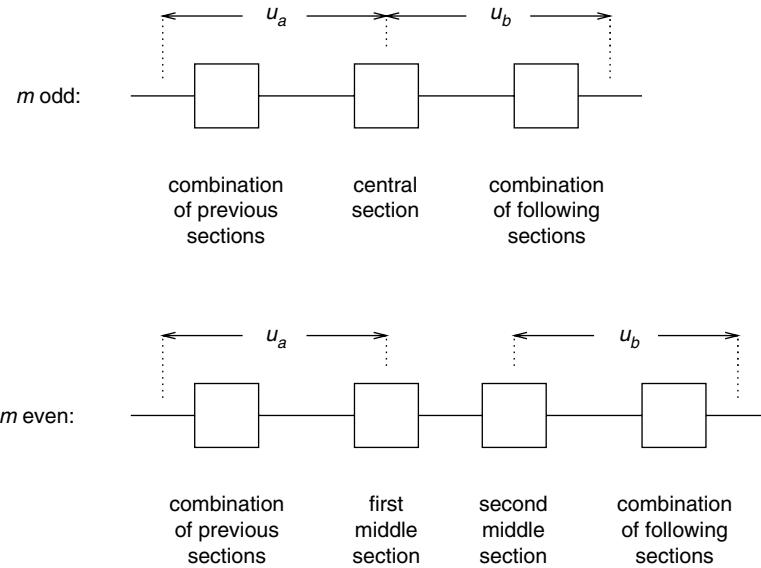


Fig. 13.7. Ordering of state-space sections.

The output roundoff-noise PSD of the state-space structure without limit cycles in cascade form is expressed by

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \sum_{i=1}^m \frac{1}{\lambda_{i+1}^2} \left(2G'_{1i}(e^{j\omega})G'_{1i}(e^{-j\omega}) + 2G'_{2i}(e^{j\omega})G'_{2i}(e^{-j\omega}) + 3 \right), \quad (13.122)$$

where $G'_{1i}(e^{j\omega})$ and $G'_{2i}(e^{j\omega})$ are the noise frequency responses of the scaled sections and $\lambda_{m+1} = 1$.

The design procedure for the state-space structure without limit cycles in the parallel form, which is rather simple, is provided in Diniz and Antoniou (1986). In this case, the expression for the roundoff-noise PSD at the output is

$$\Gamma_Y(e^{j\omega}) = \sigma_e^2 \left[1 + \sum_{i=1}^m \left(2G'_{1i}(e^{j\omega})G'_{1i}(e^{-j\omega}) + 2G'_{2i}(e^{j\omega})G'_{2i}(e^{-j\omega}) + 2 \right) \right]. \quad (13.123)$$

Example 13.3. Repeat Example 13.1 using the cascade of optimal and limit-cycle-free state-space sections. Quantize the coefficients to 9 bits, including the sign bit, and verify the results.

Solution

In each case, all sections but the last one are scaled first to guarantee unit L_2 norm at its output. After this initial scaling, for the optimal state-space structure, the transformation matrix \mathbf{T} is determined as given in Equation (13.79) considering also the effect of the cumulative transfer function from previous blocks. Tables 13.7–13.10 list the coefficients

Table 13.7.

Cascade structure using optimal state-space second-order sections.

Coefficient	Section 1	Section 2	Section 3
a_{11}	8.0271E-01	8.1339E-01	7.9823E-01
a_{12}	-5.9094E-01	-5.7910E-01	-6.0685E-01
a_{21}	5.7520E-01	5.7117E-01	5.8489E-01
a_{22}	8.0271E-01	8.1339E-01	7.9823E-01
b_1	8.0236E-02	6.4821E-03	2.9027E-02
b_2	1.5745E-01	-1.8603E-02	8.8313E-03
c_1	8.8747E-01	-8.8929E-01	2.5127E-02
c_2	4.5225E-01	3.0987E-01	8.2587E-02
d	8.8708E-02	1.2396E-01	1.3061E-02

Table 13.8.

Cascade structure using optimal state-space second-order sections quantized with 9 bits.

Coefficient	Section 1	Section 2	Section 3
$[a_{11}]_Q$	8.0078E-01	8.1250E-01	7.9688E-01
$[a_{12}]_Q$	-5.8984E-01	-5.7813E-01	-6.0547E-01
$[a_{21}]_Q$	5.7422E-01	5.7031E-01	5.8594E-01
$[a_{22}]_Q$	8.0078E-01	8.1250E-01	7.9688E-01
$[b_1]_Q$	8.2031E-02	7.8125E-03	2.7344E-02
$[b_2]_Q$	1.5625E-01	-1.9531E-02	7.8125E-03
$[c_1]_Q$	8.8672E-01	-8.9063E-01	2.3438E-02
$[c_2]_Q$	4.5313E-01	3.0859E-01	8.2031E-02
$[d]_Q$	8.9844E-02	1.2500E-01	1.1719E-02

Table 13.9.

Cascade structure without limit cycles using optimal state-space second-order sections.

$$\lambda = 2.7202E-01.$$

Coefficient	Section 1	Section 2	Section 3
a_{11}	8.0272E-01	8.1339E-01	7.9822E-01
a_{12}	-5.8289E-01	-5.7823E-01	-5.8486E-01
a_{21}	5.8316E-01	5.7204E-01	6.0688E-01
a_{22}	8.0272E-01	8.1339E-01	7.9822E-01
b_1	1.9728E-01	1.8661E-01	2.0178E-01
b_2	-5.8316E-01	-5.7204E-01	-6.0688E-01
c_1	-9.2516E-03	-4.4228E-02	9.1891E-02
c_2	-2.8600E-02	1.6281E-02	-2.5557E-02
d	9.2516E-03	1.8323E-01	2.9191E-01

Table 13.10. Cascade structure without limit cycles using optimal state-space second-order sections quantized with 9 bits. $[\lambda]_Q = 2.7344E-01$.

Coefficient	Section 1	Section 2	Section 3
$[a_{11}]_Q$	8.0078E-01	8.1250E-01	7.9688E-01
$[a_{12}]_Q$	-5.8203E-01	-5.7812E-01	-5.8594E-01
$[a_{21}]_Q$	5.8203E-01	5.7031E-01	6.0547E-01
$[a_{22}]_Q$	8.0078E-01	8.1250E-01	7.9688E-01
$[b_1]_Q$	1.9922E-01	1.8750E-01	2.0313E-01
$[b_2]_Q$	-5.8203E-01	-5.7031E-01	-6.0547E-01
$[c_1]_Q$	-7.8125E-03	-4.2969E-02	9.3750E-02
$[c_2]_Q$	-2.7344E-02	1.5625E-02	-2.7344E-02
$[d]_Q$	7.8125E-03	1.8359E-01	2.9297E-01

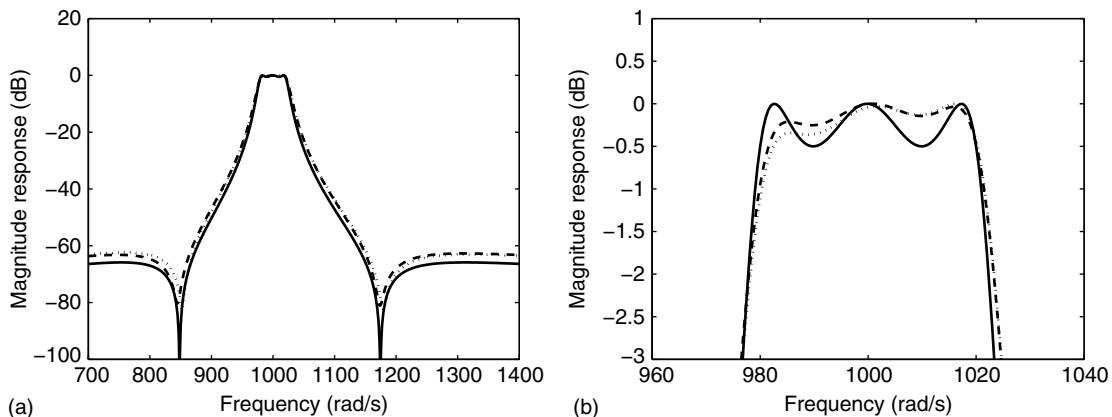


Fig. 13.8. Coefficient-quantization effects in the cascade forms, using state-space second-order sections: (a) overall magnitude response; (b) passband detail. (Solid line – initial design; dashed line – cascade of optimal state-space sections (9 bits); dotted line – cascade of limit-cycle-free state-space sections (9 bits).)

of all the designed filters. Figure 13.8 depicts the magnitude responses obtained by the cascade of optimal state-space sections and of state-space sections without limit cycles. In all cases the coefficients were quantized to 9 bits including the sign bit.

Note that, with the same given number of bits used for quantization, the magnitude responses did not move away as much as in Example 13.1. This indicates that the present state-space sections have better sensitivity properties than the second-order direct-form sections used in Example 13.1. \triangle

13.4 Lattice filters

Consider a general IIR transfer function written in the form

$$H(z) = \frac{N_M(z)}{D_N(z)} = \frac{\sum_{i=0}^M b_{i,M} z^{-i}}{1 + \sum_{i=1}^N a_{i,N} z^{-i}}. \quad (13.124)$$

In the lattice construction, we concentrate first on the realization of the denominator polynomial through an order-reduction strategy. For that, we define the auxiliary N th-order polynomial, obtained by reversing the order of the coefficients of the denominator $D_N(z)$, as given by

$$zB_N(z) = D_N(z^{-1})z^{-N} = z^{-N} + \sum_{i=1}^N a_{i,N} z^{i-N}. \quad (13.125)$$

We can then calculate a reduced-order polynomial as

$$\begin{aligned} (1 - a_{N,N}^2)D_{N-1}(z) &= D_N(z) - a_{N,N}zB_N(z) \\ &= (1 - a_{N,N}^2) + \cdots + (a_{N-1,N} - a_{N,N}a_{1,N})z^{-N+1}. \end{aligned} \quad (13.126)$$

where we can also express $D_{N-1}(z)$ as $1 + \sum_{i=1}^{N-1} a_{i,N-1} z^{-i}$. Note that the first and last coefficients of $D_N(z)$ are 1 and $a_{N,N}$, whereas for the polynomial $zB_N(z)$ they are $a_{N,N}$ and 1 respectively. This strategy to achieve the order reduction guarantees a monic $D_{N-1}(z)$; that is, $D_{N-1}(z)$ having the coefficient of $z^0 = 1$. By induction, this order-reduction procedure can be performed repeatedly, thus yielding

$$zB_j(z) = D_j(z^{-1})z^{-j} \quad (13.127)$$

$$D_{j-1}(z) = \frac{1}{1 - a_{j,j}^2} (D_j(z) - a_{j,j}zB_j(z)), \quad (13.128)$$

for $j = N, (N-1), \dots, 1$, with $zB_0(z) = D_0(z) = 1$. It can be shown that the above equations are equivalent to the following expression:

$$\begin{bmatrix} D_{j-1}(z) \\ B_j(z) \end{bmatrix} = \begin{bmatrix} 1 & -a_{j,j} \\ a_{j,j}z^{-1} & (1 - a_{j,j}^2)z^{-1} \end{bmatrix} \begin{bmatrix} D_j(z) \\ B_{j-1}(z) \end{bmatrix}. \quad (13.129)$$

The above equation can be implemented, for example, by the two-port network TP_j shown in Figure 13.9.

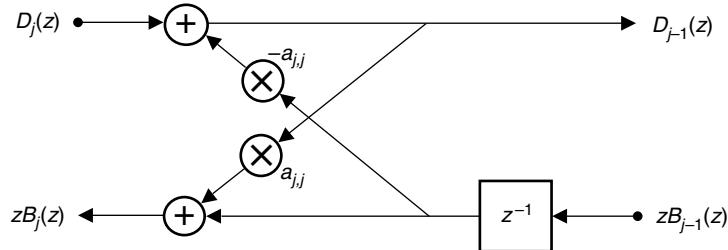


Fig. 13.9. Two-multiplier network TP_j implementing Equation (13.129).

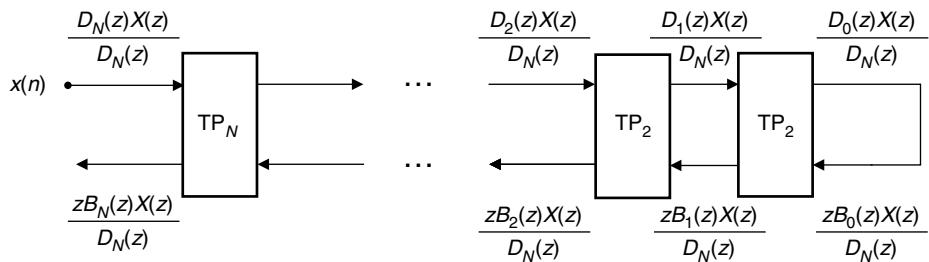


Fig. 13.10. Generation of the denominator of the IIR lattice digital filter structure.

The advantage of this representation arises from the fact that, by cascading two-port networks TP_j as in Figure 13.10, for $j = N, (N - 1), \dots, 1$, we can implement $1/D_N(z)$, where $D_N(z)$ is the denominator of the transfer function. This can be easily understood by looking at the input $X(z)$ as $(X(z)D_N(z))/D_N(z)$. If we do so, then at the right of TP_1 we will end up having $(X(z)D_0(z))/D_N(z) = X(z)/D_N(z)$.

Since at the lower branches of the two-port networks in Figure 13.10 we have the signals $zB_j(z)/D_N(z)$ available, then a convenient way to form the desired numerator is to apply weights to the polynomials $zB_j(z)$, such that

$$N_M(z) = \sum_{j=0}^M v_j zB_j(z), \quad (13.130)$$

where the tap coefficients v_j are calculated through the following order-reduction recursion:

$$N_{j-1}(z) = N_j(z) - zv_j B_j(z), \quad (13.131)$$

for $j = M, (M - 1), \dots, 1$, with $v_M = b_{M,M}$ and $v_0 = b_{0,0}$.

Then, a way of implementing the overall IIR transfer function $H(z) = N(z)/D(z) = (\sum_{j=0}^M v_j zB_j(z))/D_N(z)$ is to use the structure in Figure 13.11, which is called the IIR lattice realization.

From the above, we have a simple procedure for obtaining the lattice network given the direct-form transfer function, $H(z) = N_M(z)/D_N(z)$:

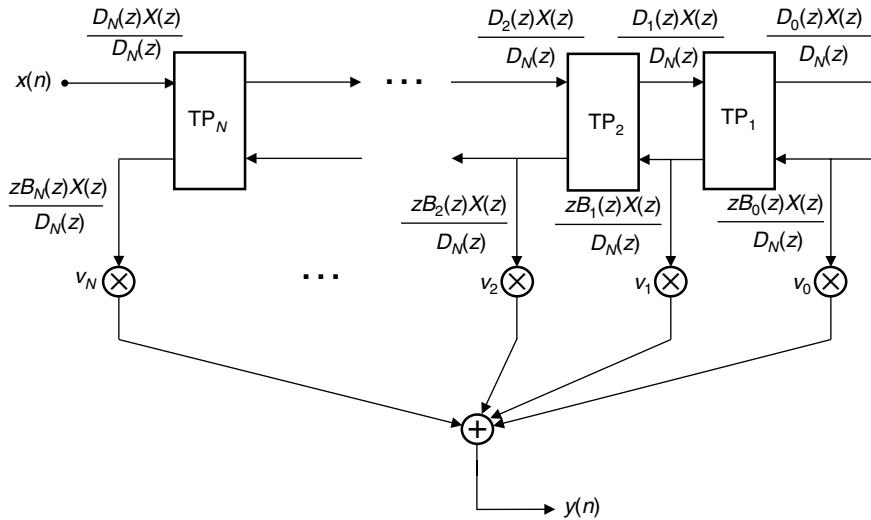


Fig. 13.11. General IIR lattice digital filter structure.

- Obtain, recursively, the polynomials $B_j(z)$ and $D_j(z)$, as well as the lattice coefficient $a_{j,j}$, for $j = N, (N - 1), \dots, 1$, using Equations (13.127) and (13.128).
- Compute the coefficients v_j , for $j = N, (N - 1), \dots, 1$, using the recursion in Equation (13.131).

Conversely, if given the lattice realization we want to compute the direct-form transfer function, we can use the following procedure:

- Start with $zB_0(z) = D_0(z)=1$.
- Compute recursively $B_j(z)$ and $D_j(z)$ for $j = 1, 2, \dots, N$, using the following relation, which can be derived from Equations (13.127) and (13.128):

$$\begin{bmatrix} D_j(z) \\ B_j(z) \end{bmatrix} = \begin{bmatrix} 1 & a_{j,j} \\ a_{j,j}z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} D_{j-1}(z) \\ B_{j-1}(z) \end{bmatrix}. \quad (13.132)$$

- Compute $N_M(z)$ using Equation (13.130).
- The direct-form transfer function is then $H(z) = N_M(z)/D_N(z)$.

There are some important properties related to the lattice realization which should be mentioned. If $D_N(z)$ has all the roots inside the unit circle then the lattice structure will have all coefficients $a_{j,j}$ with magnitude less than one. Otherwise, $H(z) = N(z)/D(z)$ represents an unstable system. This straightforward stability condition makes the lattice realizations useful for implementing time-varying filters. In addition, the polynomials $zB_j(z)$, for $j = 0, 1, \dots, M$, form an orthogonal set. This property justifies the choice of these polynomials to form the desired numerator polynomial $N_M(z)$, as described in Equation (13.131).

Since, in Figure 13.10, the two-port system consisting of section TP_j and all the sections to its right, which relates the output signal $(zB_j(z)X(z))/D_N(z)$ to the input signal $(D_j(z)X(z))/D_N(z)$ is linear, its transfer function remains unchanged if we multiply its

input signal by λ_j and divide its output by the same amount. Therefore, $zB_N(z)/D_N(z)$ will not change if we multiply the signal entering the upper-left branch of section TP_j by λ_j and divide the signal leaving the lower-left branch by λ_j . This is equivalent to scaling the section TP_j by λ_j . If we do this for every branch j , the signals entering and leaving at the left of section N remain unchanged, the signals entering and leaving at the left of section $(N - 1)$ will be scaled by λ_N , the signals entering and leaving at the left of section $(N - 2)$ will be multiplied by $\lambda_N \lambda_{N-1}$, and so on, leading to the scaled signals $(\bar{D}_j(z)X(z))/D_N(z)$ and $(z\bar{B}_j(z)X(z))/D_N(z)$ at the left of section TP_j , such that

$$\bar{D}_j(z) = \left(\prod_{i=N}^{j+1} \lambda_i \right) D_j(z) \quad (13.133)$$

$$\bar{B}_j(z) = \left(\prod_{i=N}^{j+1} \lambda_i \right) B_j(z), \quad (13.134)$$

for $j = (N - 1), (N - 2), \dots, 1$, with $\bar{D}_N(z) = D_N(z)$ and $\bar{B}_N(z) = B_N(z)$. Therefore, in order to maintain the transfer function of the scaled lattice realization unchanged, we must make

$$\bar{v}_j = \frac{v_j}{\prod_{i=N}^{j+1} \lambda_i}, \quad (13.135)$$

for $j = (N - 1), (N - 2), \dots, 1$, with $\bar{v}_N = v_N$.

Based on the above property, we can derive a more economical two-port network using a single multiplier, as shown in Figure 13.12, where the plus-or-minus signs indicate that two different realizations are possible. The choice of these signs can vary from section to section, aiming at the reduction of the quantization noise at the filter output. Note that this network is equivalent to the one in Figure 13.9 scaled using $\lambda_j = 1 \pm a_{jj}$; therefore, the coefficients \bar{v}_j should be computed using Equation (13.135). The plus sign in the computation

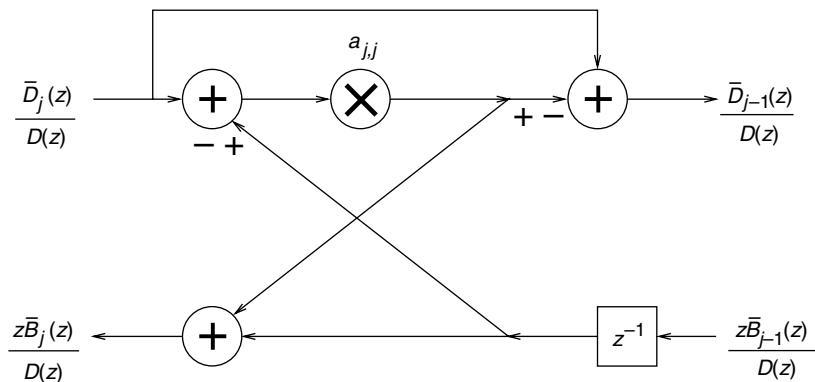


Fig. 13.12. The one-multiplier network for Equation (13.129).

of λ_j corresponds to summation being performed on the adder on the right and subtraction performed on the adder on the left, and vice versa for the minus sign.

Another important realization for the two-port network results when the scaling parameters λ_i are chosen such that all the internal nodes of the lattice network have a transfer function from the input of unit L_2 norm. The appropriate scaling can be derived by first noting that, at the left of section TP_j , the norms of the corresponding transfer functions are given by

$$\left\| \frac{\bar{D}_j(z)}{\bar{D}_N(z)} \right\|_2 = \left\| \frac{z\bar{B}_j(z)}{\bar{D}_N(z)} \right\|_2, \quad (13.136)$$

since, from Equation (13.127), $z\bar{B}_j(z) = D_j(z^{-1})z^{-j}$.

From the above equations, if we want L_2 norm at the internal nodes of the lattice network, we must have

$$\left\| \frac{z\bar{B}_0(z)}{\bar{D}_N(z)} \right\|_2 = \dots = \left\| \frac{z\bar{B}_{N-1}(z)}{\bar{D}_N(z)} \right\|_2 = \left\| \frac{z\bar{B}_N(z)}{\bar{D}_N(z)} \right\|_2 = \left\| \frac{\bar{D}_N(z)}{\bar{D}_N(z)} \right\|_2 = 1. \quad (13.137)$$

Then, using λ_j from Equations (13.132) to (13.134), it can be derived that (Gray & Markel, 1973, 1975)

$$\lambda_j = \frac{\left\| \frac{z\bar{B}_j(z)}{D_N(z)} \right\|_2}{\left\| \frac{z\bar{B}_{j-1}(z)}{D_N(z)} \right\|_2} = \sqrt{1 - a_{jj}^2}. \quad (13.138)$$

It is easy to show that section TP_j of the normalized lattice can be implemented as depicted in Figure 13.13. The most important feature of the normalized lattice realization is that, since all its internal nodes have a transfer function with unit L_2 norm, it presents an automatic scaling in the L_2 -norm sense. This explains the low roundoff noise generated by

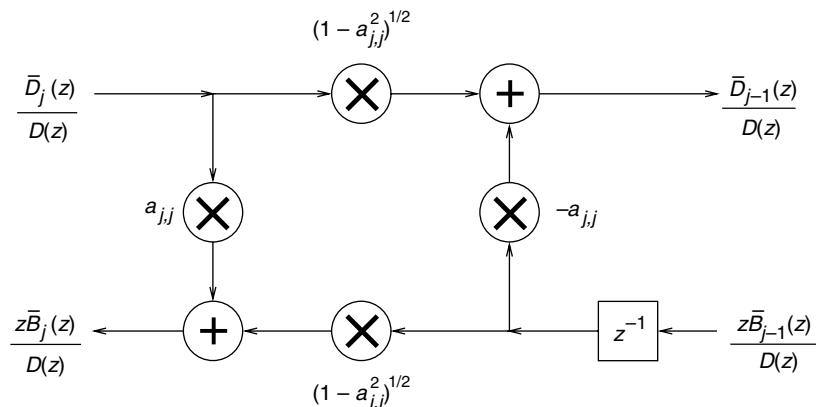


Fig. 13.13. The normalized network for Equation (13.129).

Table 13.11. Coefficients of a two-multiplier lattice.

Section j	$a_{j,j}$	v_j
0		-2.1521E-06
1	8.0938E-01	-1.1879E-06
2	-9.9982E-01	9.3821E-06
3	8.0903E-01	3.4010E-06
4	-9.9970E-01	8.8721E-05
5	8.0884E-01	-2.3326E-04
6	-9.6906E-01	-1.4362E-04

Table 13.12. Coefficients of a one-multiplier lattice.

Section j	$a_{j,j}$	\bar{v}_j
0		-2.1371E+02
1	8.0938E-01	-2.1342E+02
2	-9.9982E-01	3.0663E-01
3	8.0903E-01	2.0108E-01
4	-9.9970E-01	1.5850E-03
5	8.0884E-01	-7.5376E-03
6	-9.6905E-01	-1.4362E-04

the normalized lattice realization compared with the other forms of the lattice realization. Note that the coefficients \bar{v}_j have to be computed using Equation (13.135).

Example 13.4. Repeat Example 13.1 using the one-multiplier, two-multiplier, and normalized lattice forms. Quantize the coefficients of the normalized lattice using 9 bits, including the sign bit, and verify the results.

Solution

The two-multiplier IIR lattice can be determined from the direct form using the MATLAB command `tf2latc`. For the one-multiplier, we use $\lambda_j = (1 + a_{j,j})$ in Equation (13.135) to determine the feedforward coefficients, whereas for the normalized lattice we use $\lambda_j = \sqrt{1 - a_{j,j}^2}$. The resulting coefficients in each case are seen in Tables 13.11–13.14.

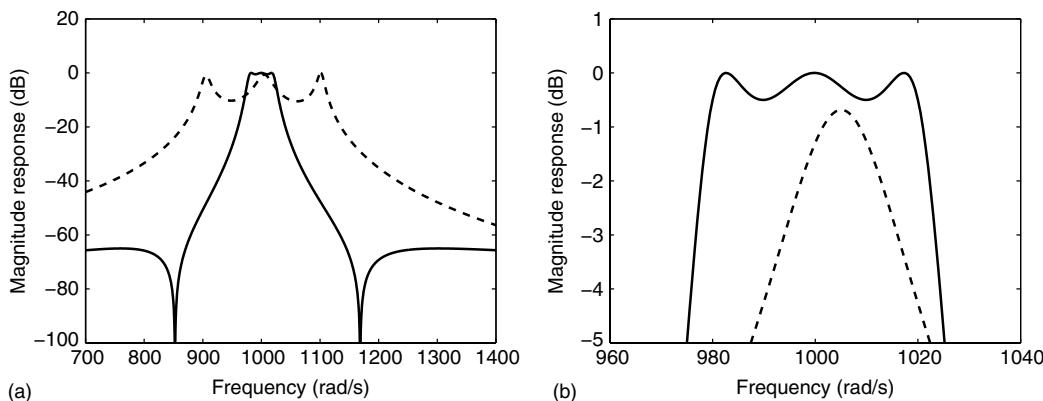
During the quantization procedure, we must guarantee that the absolute value of all feedback coefficients $a_{j,j}$ remain below one to guarantee stability of the resulting filter. From Tables 13.11–13.14, one observes that the three lattice forms have serious quantization issues due to the wide range covered by their coefficients. It must be added that the normalized lattice performs much better than the two- and one-multiplier lattices with respect to quantization effects. It is also worth mentioning that, in the two-multiplier structure, the feedforward coefficients assume very small values, forcing the use of more than 9 bits for

Table 13.13. Coefficients of a normalized lattice.

Section j	$a_{j,j}$	\bar{v}_j
0		-9.1614E-02
1	8.0938E-01	-2.9697E-02
2	-9.9982E-01	4.4737E-03
3	8.0903E-01	9.5319E-04
4	-9.9970E-01	6.1121E-04
5	8.0884E-01	-9.4494E-04
6	-9.6905E-01	-1.4362E-04

Table 13.14. Coefficients of a normalized lattice quantized with 9 bits.

Section j	$a_{j,j}$	\bar{v}_j
0		-8.9844E-02
1	8.0938E-01	-3.1250E-02
2	-9.9982E-01	3.9063E-03
3	8.0903E-01	0.0000E+00
4	-9.9970E-01	0.0000E+00
5	8.0884E-01	0.0000E+00
6	-9.6905E-01	0.0000E+00

**Fig. 13.14.** Coefficient-quantization effects in the normalized lattice form: (a) overall magnitude response; (b) passband detail. (Solid line – initial design; dashed line – normalized lattice (9 bits).)

their representation. This normally happens when designing a filter having poles very close to the unit circle, as is the case in this example.

Figure 13.14 depicts the magnitude responses obtained by the original and quantized normalized lattices. Note that the magnitude responses of the normalized lattice structure

are significantly different to the ideal one, especially when compared with the results shown in Examples 13.1 and 13.2. \triangle

13.5 Doubly complementary filters

In this section the class of the doubly complementary filter is discussed, since it plays an important role in alias-free two-band filter banks and some audio applications (Regalia *et al.*, 1988).

Theorem 13.2. Two transfer functions $H_0(z)$ and $H_1(z)$ are referred to as doubly complementary if their frequency responses are allpass complementary, namely

$$|H_0(e^{j\omega}) + H_1(e^{j\omega})|^2 = 1, \quad (13.139)$$

and also power complementary, such that

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1 \quad (13.140)$$

for all ω . For doubly complementary filters, we can write that

$$H_0(z) + H_1(z) = F_0(z) \quad (13.141)$$

$$H_0(z) - H_1(z) = F_1(z), \quad (13.142)$$

where $F_0(z)$ and $F_1(z)$ are stable allpass transfer functions, and then

$$H_0(z) = \frac{1}{2} (F_0(z) + F_1(z)) \quad (13.143)$$

$$H_1(z) = \frac{1}{2} (F_0(z) - F_1(z)), \quad (13.144)$$

whose implementation can be as shown in Figure 13.15. \diamond

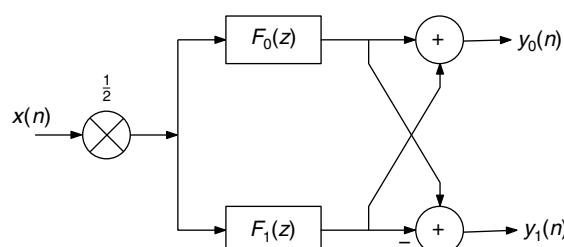


Fig. 13.15. Doubly complementary filters.

Proof The doubly complementary frequency responses can be described in polar form as follows:

$$H_0(e^{j\omega}) = r_0(\omega) e^{j\phi_0(\omega)} \quad (13.145)$$

$$H_1(e^{j\omega}) = r_1(\omega) e^{j\phi_1(\omega)}. \quad (13.146)$$

Using these expressions, the left-hand side L of Equation (13.139) can be written as

$$\begin{aligned} L &= |r_0(\omega) e^{j\phi_0(\omega)} + r_1(\omega) e^{j\phi_1(\omega)}|^2 \\ &= (r_0(\omega) e^{j\phi_0(\omega)} + r_1(\omega) e^{j\phi_1(\omega)}) (r_0(\omega) e^{-j\phi_0(\omega)} + r_1(\omega) e^{-j\phi_1(\omega)}) \\ &= r_0^2(\omega) + r_1^2(\omega) + r_0(\omega)r_1(\omega) e^{j(\phi_0(\omega)-\phi_1(\omega))} + r_0(\omega)r_1(\omega) e^{-j(\phi_0(\omega)-\phi_1(\omega))} \\ &= r_0^2(\omega) + r_1^2(\omega) + 2r_0(\omega)r_1(\omega) \cos(\phi_0(\omega) - \phi_1(\omega)). \end{aligned} \quad (13.147)$$

Since Equation (13.140) is equivalent to $r_0^2(\omega) + r_1^2(\omega) = 1$, then for allpass complementary $H_0(e^{j\omega})$ and $H_1(e^{j\omega})$ we must have that

$$2r_0(\omega)r_1(\omega) \cos(\phi_0(\omega) - \phi_1(\omega)) = 0. \quad (13.148)$$

By following the same procedure as the derivation of Equation (13.147), it is possible to show that

$$\begin{aligned} |H_0(e^{j\omega}) - H_1(e^{j\omega})|^2 &= r_0^2(\omega) + r_1^2(\omega) - 2r_0(\omega)r_1(\omega) \cos(\phi_0(\omega) - \phi_1(\omega)) \\ &= 1. \end{aligned} \quad (13.149)$$

By applying the expressions of Equations (13.143) and (13.144) in Equation (13.140) and using the polar representation, it is straightforward to show that

$$|F_0(e^{j\omega})|^2 + |F_1(e^{j\omega})|^2 = 2. \quad (13.150)$$

Also, applying Equations (13.143) and (13.144) along with (13.148) in Equation (13.139) it follows that

$$|F_0(e^{j\omega})|^2 = r_0^2(\omega) + r_1^2(\omega) = 1 \quad (13.151)$$

and then

$$|F_1(e^{j\omega})|^2 = r_0^2(\omega) + r_1^2(\omega) = 1. \quad (13.152)$$

Therefore, $F_0(z)$ and $F_1(z)$ are both allpass filters. \square

Allpass transfer functions have the following general form:

$$F_i(z) = \frac{\sum_{l=0}^{N_i} a_{N_i-l,i} z^{-l}}{\sum_{l=0}^{N_i} a_{l,i} z^{-l}} = \frac{D_i(z^{-1})}{z^{-N_i} D_i(z)} = z^{N_i} \frac{D_i(z^{-1})}{D_i(z)} \quad (13.153)$$

for $i = 0, 1$ and $D_i(z) = a_{0,i}z^{N_i} + a_{1,i}z^{N_i-1} + \dots + a_{N_i,i}$. The phase responses of the allpass filters are given by

$$\theta_i(\omega) = -N_i\omega + 2 \arctan \left[\frac{\sum_{l=0}^{N_i} a_{l,i} \sin(l\omega)}{\sum_{l=0}^{N_i} a_{l,i} \cos(l\omega)} \right]. \quad (13.154)$$

Given that $F_0(e^{j\omega})$ and $F_1(e^{j\omega})$ are allpass frequency responses, they can be expressed in polar form as

$$F_0(e^{j\omega}) = e^{j\theta_0(\omega)} \quad (13.155)$$

$$F_1(e^{j\omega}) = e^{j\theta_1(\omega)} \quad (13.156)$$

in such way that

$$|H_0(e^{j\omega})| = 1/2 \left| e^{j(\theta_0(\omega) - \theta_1(\omega))} + 1 \right| \quad (13.157)$$

$$|H_1(e^{j\omega})| = 1/2 \left| e^{j(\theta_0(\omega) - \theta_1(\omega))} - 1 \right|. \quad (13.158)$$

Assuming that, at frequency $\omega = 0$, both allpass filters have zero phase, as a result $|H_0(1)| = 1$ and $|H_1(1)| = 0$, which are typical features of lowpass and highpass filters, respectively. On the other hand, for $\omega = \pi$:

$$|H_0(e^{j\pi})| = 1/2 |e^{j(N_0-N_1)\pi} + 1| \quad (13.159)$$

$$|H_1(e^{j\pi})| = 1/2 |e^{j(N_0-N_1)\pi} - 1| \quad (13.160)$$

so that if the difference $(N_0 - N_1)$ is odd, then $|H_0(e^{j\pi})| = 0$ and $|H_1(e^{j\pi})| = 1$, again a typical property of lowpass and highpass filters, respectively.

Let us consider a simple, and yet useful, choice for the allpass transfer functions; that is:

$$F_0(z) = z^{-N_0} \quad (13.161)$$

$$F_1(z) = z^{-1} F'_1(z), \quad (13.162)$$

where $F'_1(z)$ is a standard allpass transfer function of order N_0 of the form in Equation (13.153). With an odd $(N_0 - N_1) = 1$, it is possible to generate doubly

complementary transfer functions with lowpass and highpass shapes given by

$$H_0(z) = z^{-N_0} + z^{-1}F'_1(z) \quad (13.163)$$

$$H_1(z) = z^{-N_0} - z^{-1}F'_1(z). \quad (13.164)$$

The difference in the phase response of the allpass filters are given by

$$\begin{aligned} \theta_0(\omega) - \theta_1(\omega) &= -N_0\omega + \theta_1(\omega) \\ &= (-N_0 - 1)\omega + \angle F'_1(e^{j\omega}) \\ &= (-N_0 - 1)\omega + N_0\omega - 2 \arctan \left[\frac{\sum_{l=0}^{N_0} a_{l,0} \sin(l\omega)}{\sum_{l=0}^{N_0} a_{l,0} \cos(l\omega)} \right] \\ &= -\omega - 2 \arctan \left[\frac{\sum_{l=0}^{N_0} a_{l,0} \sin(l\omega)}{\sum_{l=0}^{N_0} a_{l,0} \cos(l\omega)} \right]. \end{aligned} \quad (13.165)$$

Example 13.5. Design doubly complementary filters $H_0(z)$ and $H_1(z)$ satisfying the specification for the lowpass filter below:

$$\left. \begin{array}{l} A_r = 40 \text{ dB} \\ \Omega_p = 0.5\pi \text{ rad/s} \\ \Omega_r = 0.6\pi \text{ rad/s} \end{array} \right\}. \quad (13.166)$$

Solution

In this solution we employ the simple choice for the first allpass filter $F_0(z) = z^{-N_0}$. In this case, we start the solution by first designing an allpass filter $F_1(z)$ whose phase response follows as closely as possible the following specifications

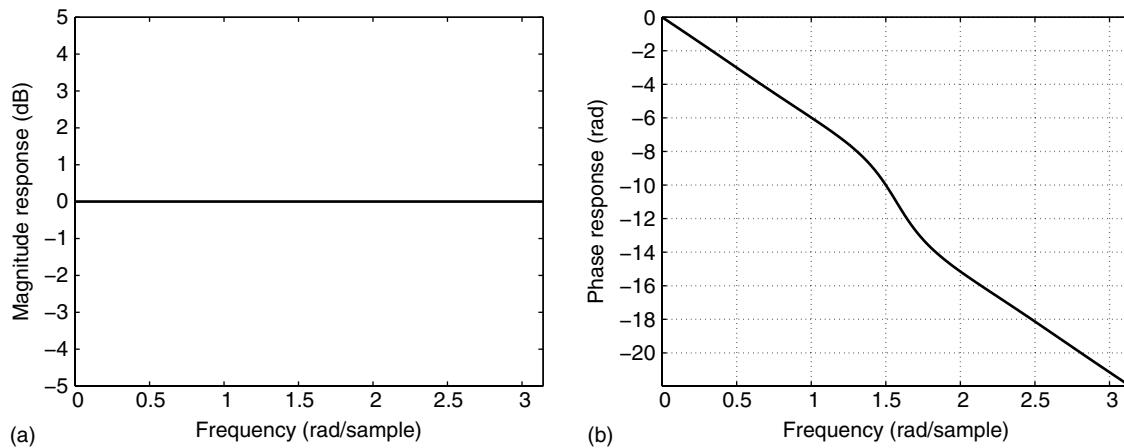
$$\theta_1(\omega) = \begin{cases} -N_0\omega, & \text{for } 0 \leq \omega \leq \Omega_p \\ -N_0\omega + \pi, & \text{for } \Omega_r \leq \omega \leq \pi, \end{cases} \quad (13.167)$$

considering the sampling frequency $\Omega_s = 2\pi$. With this strategy the phase difference $\theta_0(\omega) - \theta_1(\omega) = -N_0\omega - \theta_1(\omega)$ in Equation (13.165) will be approximately zero at low frequencies and approximately π after the frequency $\pi/2$, thus enforcing the doubly complementary property.

There are several ways to design allpass filters satisfying prescribed phase specifications, such as those based on the L_p -norm minimization criteria described in Section 6.5. Other specialized methods are described in Nguyen *et al.* (1994). We employed them to design an allpass filter $F_1(z)$ whose coefficients are shown in Table 13.15. A sixth-order allpass filter sufficed to generate a stopband attenuation of about 40 dB.

Table 13.15. $F_1(z)$ allpass filter coefficients $a_{j,1}$.

$a_{0,1} =$	1.0000
$a_{1,1} =$	0.0000
$a_{2,1} =$	0.4780
$a_{3,1} =$	0.0000
$a_{4,1} =$	-0.0941
$a_{5,1} =$	0.0000
$a_{6,1} =$	0.0283

**Fig. 13.16.** Allpass filter of order $N = 6$: (a) magnitude response of the allpass filter; (b) unwrapped phase response.

As can be observed, the even-order coefficients of the allpass filter are zero, a property originated from the fact that the allpass filter has symmetric response around the frequency $\pi/2$ as any half-band filter. Figure 13.16 shows the magnitude and phase responses of the allpass filter $F'_1(z)$, where from the phase response it is possible to observe the differences in the phase delays at the low- and high-frequency ranges.

Figure 13.17 shows the magnitude and phase responses of the doubly complementary filters $H_0(z)$ and $H_1(z)$ respectively generated according to Equations (13.163) and (13.164). \triangle

13.5.1 QMF filter bank implementation

We consider the case where $H_0(z)$ and $H_1(z)$ satisfy the doubly complementary conditions, and they are chosen as the lowpass and highpass analysis filters from a filter bank respectively. The synthesis filters are selected according to the QMF conditions of Equations (9.141) and (9.142), namely $G_0(z) = H_1(-z)$ and $G_1(z) = -H_0(-z)$, and the

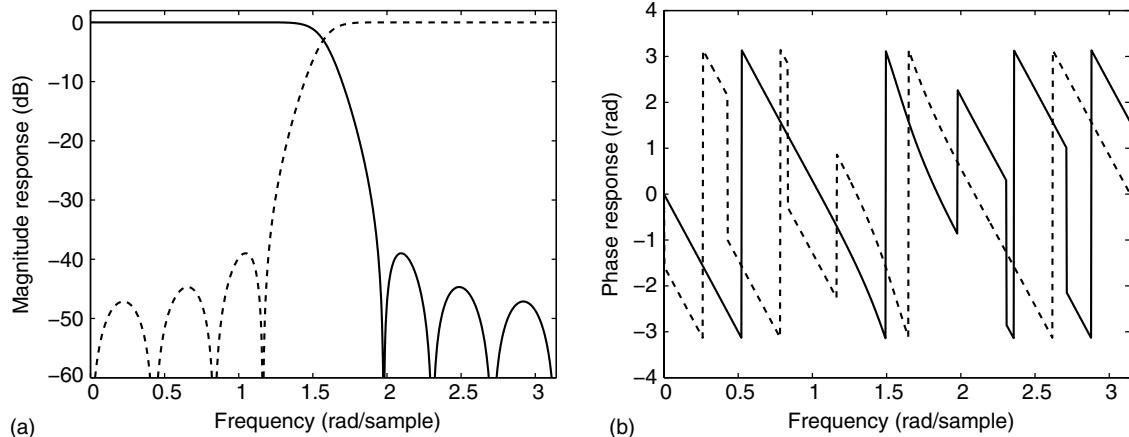


Fig. 13.17. Doubly complementary filter of order $N = 7$: (a) magnitude responses of the lowpass and highpass filters; (b) phase responses.

overall transfer function of the two-band QMF filter bank is given by Equation (9.143), repeated here for convenience:

$$H(z) = \frac{1}{2}(H_0(z)H_1(-z) - H_1(z)H_0(-z)). \quad (13.168)$$

If $H_0(z)$ and $H_1(z)$ are chosen according to Equations (13.143) and (13.144), then

$$\begin{aligned} H(z) &= \frac{1}{2} \left[\frac{1}{4} (F_0(z) + F_1(z)) (F_0(-z) - F_1(-z)) \right. \\ &\quad \left. - \frac{1}{4} (F_0(z) - F_1(z)) (F_0(-z) + F_1(-z)) \right] \\ &= \frac{1}{2} \left[\frac{1}{2} (F_0(z)F_1(-z) - F_0(-z)F_1(z)) \right]. \end{aligned} \quad (13.169)$$

Using the result of Equation (9.144), the overall transfer function of a two-band QMF filter bank whose analysis filters are $H_0(z)$ and $H_1(z)$ is given by

$$H(z) = -\frac{1}{2}z^{-1}\hat{F}_0(z^2)\hat{F}_1(z^2), \quad (13.170)$$

where $F_0(z) = \hat{F}_0(z^2)$ and $F_1(z) = z^{-1}\hat{F}_1(z^2)$, since $F_0(z)$ and $F_1(z)$ are half-band filters, given that the specifications of $H_0(z)$ and $H_1(z)$ are symmetrical with each other around $\pi/2$.

It is possible to observe that the filter bank transfer function is alias free and has no magnitude distortion, as $H(z)$ consists of a product of allpass functions.

13.6 Wave filters

In classical analog filter design, it is widely known that doubly terminated LC lossless filters have zero sensitivity of the transfer function, with respect to the lossless L and C components, at frequencies where the maximal power is transferred to the load. Filter transfer functions that are equiripple in the passband, such as Chebyshev and elliptic filters, have several frequencies of maximal power transfer. Since the ripple values are usually kept small within the passband, the sensitivities of the transfer function to variations in the filter components remain small over the frequency range consisting of the entire passband. This is the reason why several methods have been proposed to generate realizations that attempt to emulate the internal operations of the doubly terminated lossless filters.

In digital-filter design, the first attempt to derive a realization starting from an analog prototype consisted of applying the bilinear transformation to the continuous-time transfer function, establishing a direct correspondence between the elements of the analog prototype and of the resulting digital filter. However, the direct simulation of the internal quantities, such as voltages and currents, of the analog prototype in the digital domain leads to delay-free loops, as will be seen below. These loops cannot be computed sequentially, since not all node values in the loop are initially known (Antoniou, 1993).

An alternative approach results from the fact that any analog n -port network can be characterized using the concepts of incident and reflected waves' quantities known from distributed parameter theory (Belevitch, 1968). Through the application of the wave characterization, digital filter realizations without delay-free loops can be obtained from passive and active analog filters, using the bilinear transformation, as originally proposed by Fettweis (1971a, 1986). The realizations obtained using this procedure are known as wave digital filters (Fettweis, 1971a, 1986; Sedlmeyer & Fettweis, 1973; Fettweis *et al.*, 1974; Fettweis & Meerkötter, 1975a,b; Antoniou & Rezk, 1977, 1980; Diniz & Antoniou, 1988). The name wave digital filter derives from the fact that wave quantities are used to represent the internal analog signals in the digital-domain simulation. The possible wave quantities are voltage, current, and power quantities. The choice between voltage or current waves is irrelevant, whereas power waves lead to more complicated digital realizations. Traditionally, voltage wave quantities are the most widely used, and, therefore, we base our presentation on that approach.

Another great advantage of the wave digital filters, when imitating doubly terminated lossless filters, is their inherent stability under linear conditions (infinite-precision arithmetic), as well as in the nonlinear case, where the signals are subjected to quantization. Also, if the states of a wave digital filter structure, imitating a passive analog network, are quantized using magnitude truncation and saturation arithmetic, no zero-input or overflow limit cycles can be sustained.

The wave digital filters are also adequate to simulate certain analog systems, such as power systems, due to the topological equivalence with their analog counterparts (Roitman & Diniz, 1995, 1996).

13.6.1 Motivation

The transformation of a transfer function $T(s)$ representing a continuous-time system into a discrete-time transfer function $H(z)$ may be performed using the bilinear transformation in the following form:

$$H(z) = T(s)|_{s=\frac{2}{T} \frac{z-1}{z+1}}. \quad (13.171)$$

Given the doubly terminated LC network depicted in Figure 13.18, if we use voltage and current variables to simulate the analog components, we have that

$$\left. \begin{array}{l} I_1 = \frac{V_i - V_2}{R_1} \\ I_2 = \frac{V_2}{Z_C} \\ I_3 = \frac{V_2}{Z_L} \\ I_4 = \frac{V_2}{R_2} \\ I_1 = I_2 + I_3 + I_4 \end{array} \right\}. \quad (13.172)$$

The possible representations for an inductor in the z plane will be in one of the forms shown in Figure 13.19; that is:

$$\left. \begin{array}{l} V = sLI = \frac{2Lz - 1}{Tz + 1}I \\ I = \frac{V}{sL} = \frac{Tz + 1}{2Lz - 1}V \end{array} \right\}. \quad (13.173)$$

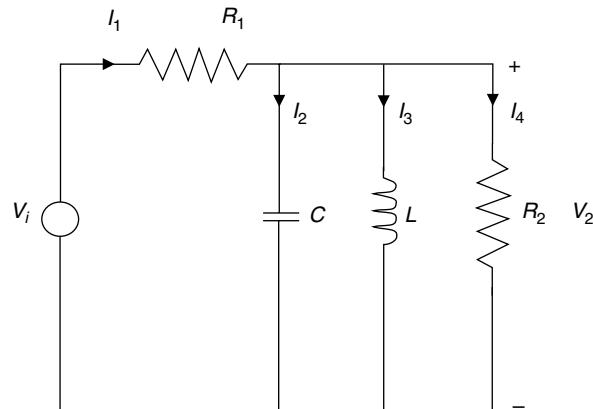


Fig. 13.18. Doubly terminated LC network.

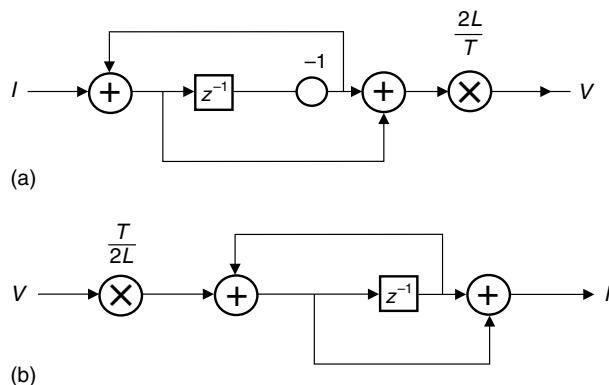


Fig. 13.19. Two possible inductor realizations.

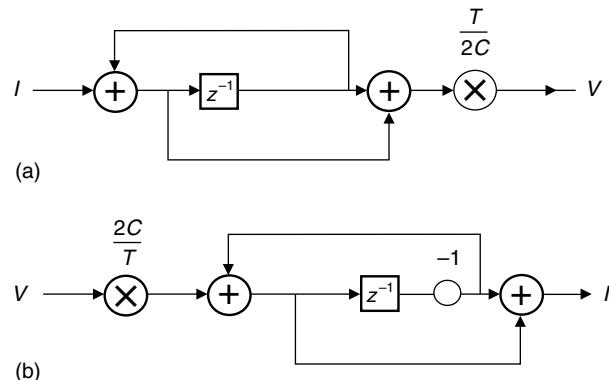


Fig. 13.20. Two possible capacitor realizations.

For a capacitor, the resulting possible representations are depicted in Figure 13.20, such that

$$\left. \begin{aligned} V &= \frac{I}{sC} = \frac{T}{2C} \frac{z+1}{z-1} I \\ I &= sCV = \frac{2C}{T} \frac{z-1}{z+1} V \end{aligned} \right\}. \quad (13.174)$$

The sources and the loads are represented in Figure 13.21. Therefore, using Figures 13.19–13.21, the digital simulation of the doubly terminated LC network of Figure 13.18 leads to the digital network shown in Figure 13.22, where we notice the existence of delayless loops. In the next subsection, we show how these loops can be avoided using the concept of wave digital filters.

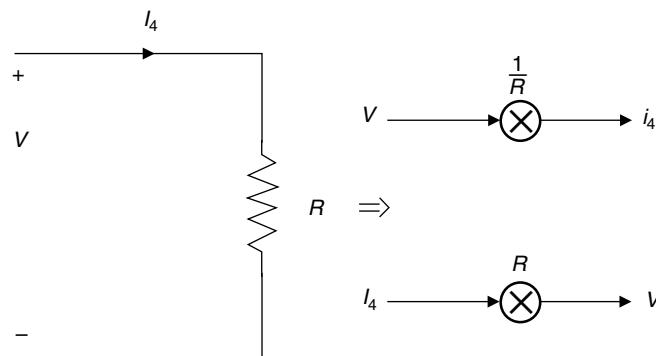
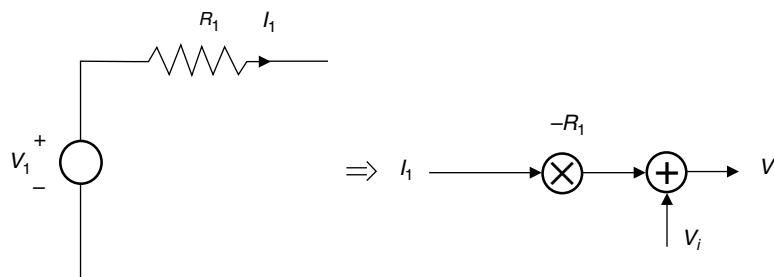


Fig. 13.21. Termination realizations.

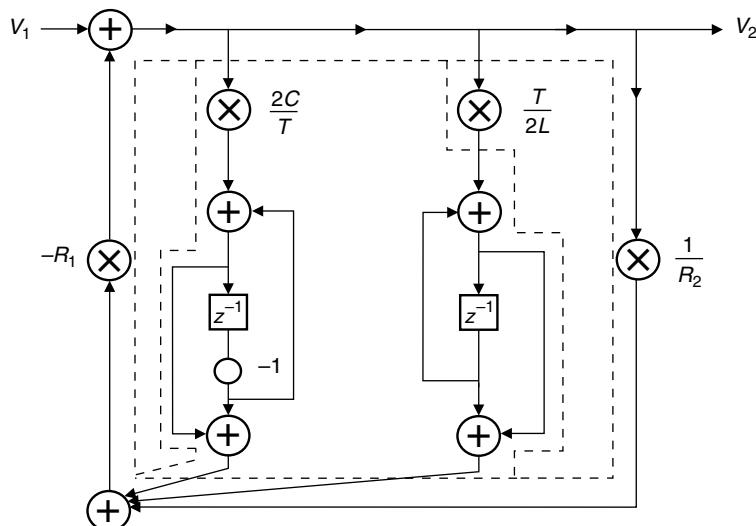


Fig. 13.22. Digital network with delay-free loops.

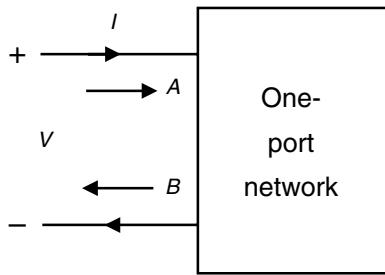


Fig. 13.23. Convention for the incident and reflected waves.

13.6.2 Wave elements

As discussed in the previous subsection, the direct simulation of the branch elements of the analog network introduces delayless loops, generating a noncomputable digital network. This problem can be circumvented by simulating the analog network using the wave equations that represent multiport networks instead of representing the voltages and currents in a straightforward manner.

As shown in Figure 13.23, an analog one-port network can be described in terms of a wave characterization as a function of the variables

$$\left. \begin{aligned} a &= v + Ri \\ b &= v - Ri \end{aligned} \right\}, \quad (13.175)$$

where a and b are the incident and reflected voltage wave quantities respectively and R is the port resistance assigned to the one-port network. In the frequency domain the wave quantities are A and B , such that

$$\left. \begin{aligned} A &= V + RI \\ B &= V - RI \end{aligned} \right\}. \quad (13.176)$$

Notice how the voltage waves consist of linear combinations of the voltage and current of the one-port network.

The value of R is a positive parameter called port resistance. A proper choice of R leads to simple multiport network realizations. In the following, we examine how to represent several analog elements using the incident and reflected waves.

13.6.2.1 One-port elements

For a capacitor, the following equations apply:

$$\left. \begin{aligned} V &= \frac{1}{sC}I \\ B &= A \frac{V - RI}{V + RI} \end{aligned} \right\}. \quad (13.177)$$

thus

$$B = A \frac{(1/sC) - R}{(1/sC) + R}. \quad (13.178)$$

By applying the bilinear transformation, we find

$$B = \frac{(T/2C)(z + 1) - R(z - 1)}{(T/2C)(z + 1) + R(z - 1)} A. \quad (13.179)$$

The value of R that leads to a significant simplification in the implementation of B as a function of A is

$$R = \frac{T}{2C}, \quad (13.180)$$

and then

$$B = z^{-1} A. \quad (13.181)$$

The realization of B as a function of A is done as shown in Figure 13.24.

Following a similar reasoning, the digital representation of several other one-port elements can be derived, as shown in Figure 13.25, along with the respective wave equations.

13.6.2.2 Voltage generalized immittance converter

The voltage generalized immittance converter (VGIC) (Diniz & Antoniou, 1988), depicted in Figure 13.26, is a two-port network characterized by

$$\left. \begin{array}{l} V_1(s) = r(s)V_2(s) \\ I_1(s) = -I_2(s) \end{array} \right\}, \quad (13.182)$$

where $r(s)$ is the so-called conversion function and the pairs (V_1, I_1) and (V_2, I_2) are the VGIC voltages and currents at ports 1 and 2 respectively.

The VGICs are not employed in the design of analog circuits due to difficulties in implementation when using conventional active devices such as transistors and operational amplifiers. However, there is no difficulty in utilizing VGICs in the design of digital filters.

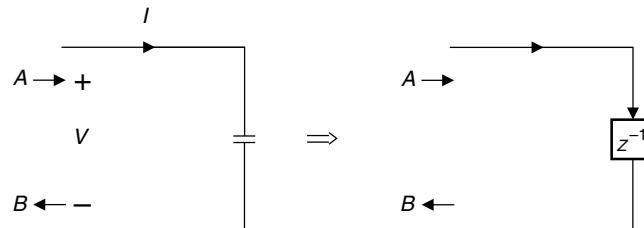


Fig. 13.24. Wave realization for a capacitor.

The VGIC of Figure 13.26 may be described in terms of wave equations by

$$\left. \begin{aligned} A_1 &= V_1 + \frac{I_1}{G_1} \\ A_2 &= V_2 + \frac{I_2}{G_2} \\ B_1 &= V_1 - \frac{I_1}{G_1} \\ B_2 &= V_2 - \frac{I_2}{G_2} \\ V_1(s) &= r(s)V_2(s) \\ I_1(s) &= -I_2(s) \end{aligned} \right\}, \quad (13.183)$$

where A_i and B_i are the incident and reflected waves of each port respectively and G_i represents the conductance of port i , for $i = 1, 2$.

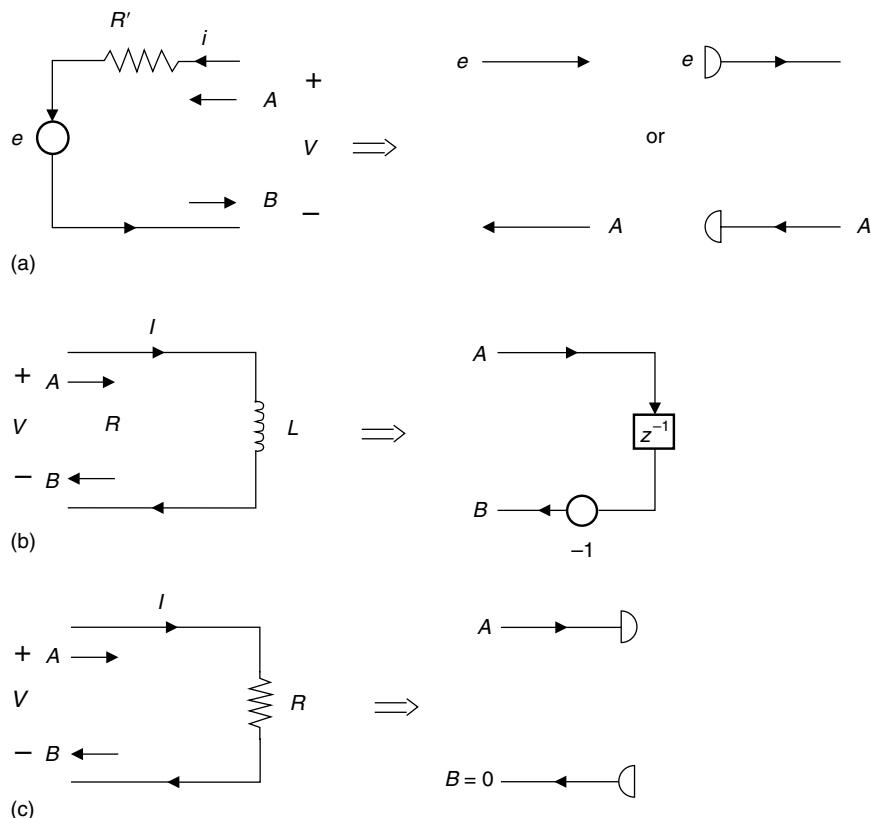


Fig. 13.25.

Wave realization for several one-port networks: (a) series connection of a voltage source with resistor: $e = V - R'I = V - RI = B$, for $R' = R$; (b) inductor: $R = 2L/T$; (c) resistor: $B = 0$, $A = 2RI$.

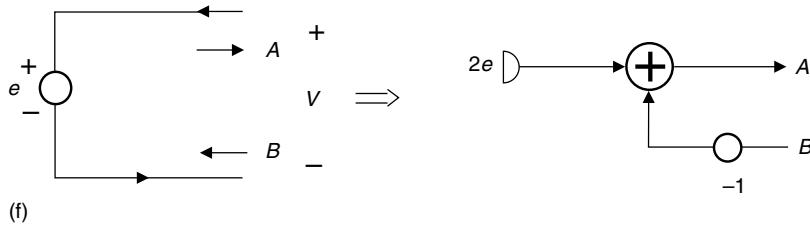
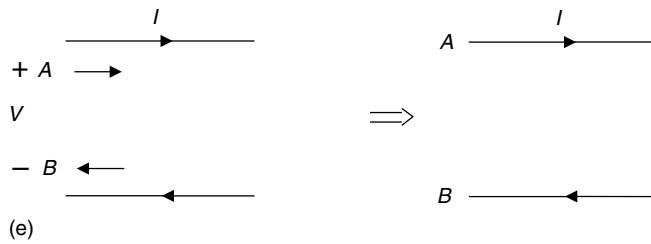
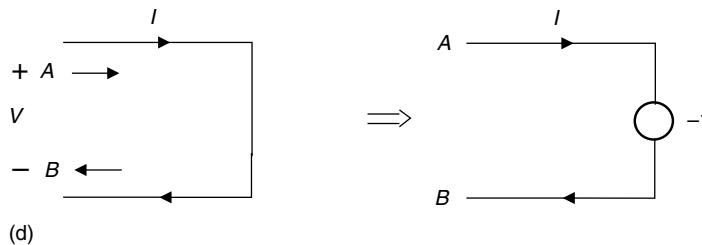


Fig. 13.25. (cont.) Wave realization for several one-port networks: (d) short circuit: $A = RI$, $B = -RI$; (e) open circuit: $A = V$, $B = V$; (f) voltage source: $A = 2e - B$.

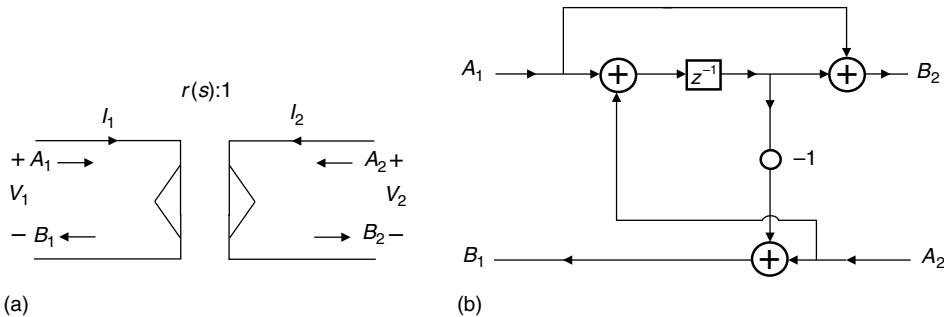


Fig. 13.26. VGIC: (a) analog symbol; (b) digital realization: $r(s) = Ts/2$.

After some algebraic manipulation, we can calculate the values of B_1 and B_2 , as functions of A_1, A_2, G_1, G_2 , and $r(s)$, as

$$\left. \begin{aligned} B_1 &= \frac{r(s)G_1 - G_2}{r(s)G_1 + G_2} A_1 + \frac{2r(s)G_2}{r(s)G_1 + G_2} A_2 \\ B_2 &= \frac{2G_1}{r(s)G_1 + G_2} A_1 + \frac{G_2 - r(s)G_1}{r(s)G_1 + G_2} A_2 \end{aligned} \right\}. \quad (13.184)$$

By applying the bilinear transformation, and by choosing $G_2 = G_1$ and $r(s) = (T/2)s$, which lead to a simple digital realization, as seen in Figure 13.26b, the following relations result:

$$\left. \begin{array}{l} B_1 = -z^{-1}A_1 + (1 - z^{-1})A_2 \\ B_2 = (1 + z^{-1})A_1 + z^{-1}A_2 \end{array} \right\}. \quad (13.185)$$

13.6.2.3 Current generalized immittance converter

The current generalized immittance converter (CGIC) (Antoniou & Rezk, 1977) is described by

$$\left. \begin{array}{l} V_1 = V_2 \\ I_1 = -h(s)I_2 \end{array} \right\}. \quad (13.186)$$

Choosing $G_1 = 2G_2/T$ and $h(s) = s$, a simple realization for the CGIC results, as illustrated in Figure 13.27.

13.6.2.4 Transformer

A transformer with a turn ratio of $n : 1$ and with port resistances R_1 and R_2 , with $R_2/R_1 = 1/n^2$, has a digital representation as shown in Figure 13.28.

13.6.2.5 Gyrator

A gyrator is a lossless two-port element described by

$$\left. \begin{array}{l} V_1 = -RI_2 \\ V_2 = RI_1 \end{array} \right\}. \quad (13.187)$$

It can be easily shown in this case that $B_2 = A_1$ and $B_1 = -A_2$, with $R_1 = R_2 = R$. The digital realization of a gyrator is depicted in Figure 13.29.

We are now equipped with the digital representations of the main analog elements which serve as the basic building blocks for the realization of wave digital filters. However, to

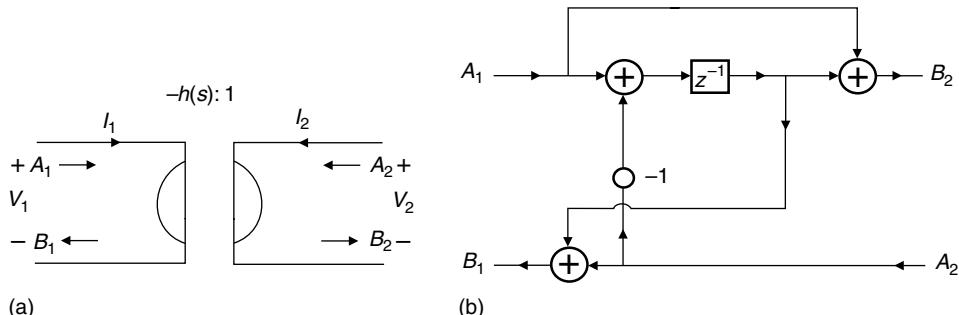


Fig. 13.27. CGIC: (a) analog symbol; (b) digital realization: $h(s) = s$.

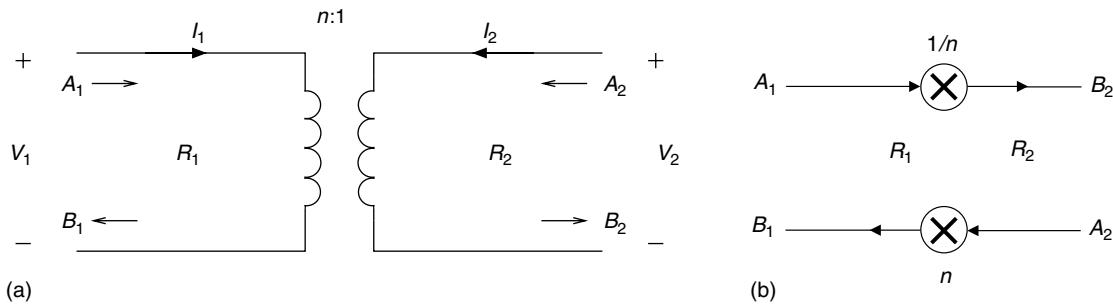


Fig. 13.28. Transformer digital representation: (a) analog symbol; (b) digital realization: $R_2/R_1 = 1/n^2$.

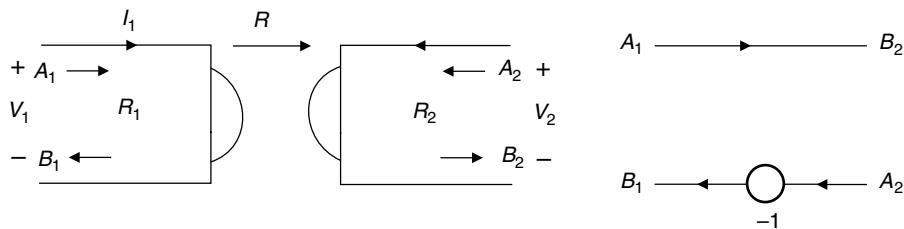


Fig. 13.29. Gyrator digital representation.

achieve our goal, we still have to learn how to interconnect these building blocks. To avoid delay-free loops, this must be done in the same way, as these blocks are interconnected in the reference analog filter. Since the port resistances of the various elements are different, there is also a need to derive the so-called adaptors to allow the interconnection. Such adaptors guarantee that the current and voltage Kirchoff laws are satisfied at all series and parallel interconnections of ports with different port resistances.

13.6.2.6 Two-port adaptors

Consider the parallel interconnection of two elements with port resistances given by R_1 and R_2 , as shown in Figure 13.30. The wave equations in this case are given by

$$\left. \begin{array}{l} A_1 = V_1 + R_1 I_1 \\ A_2 = V_2 + R_2 I_2 \\ B_1 = V_1 - R_1 I_1 \\ B_2 = V_2 - R_2 I_2 \end{array} \right\}. \quad (13.188)$$

Since $V_1 = V_2$ and $I_1 = -I_2$, we have that

$$\left. \begin{array}{l} A_1 = V_1 + R_1 I_1 \\ A_2 = V_1 - R_2 I_1 \\ B_1 = V_1 - R_1 I_1 \\ B_2 = V_1 + R_2 I_1 \end{array} \right\}. \quad (13.189)$$

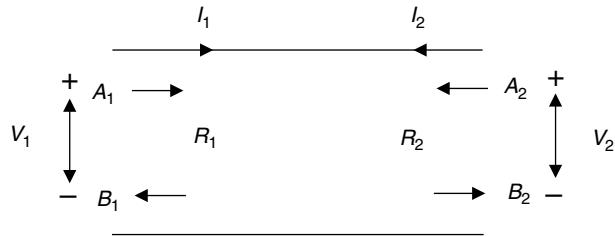


Fig. 13.30. Two-port adaptor.

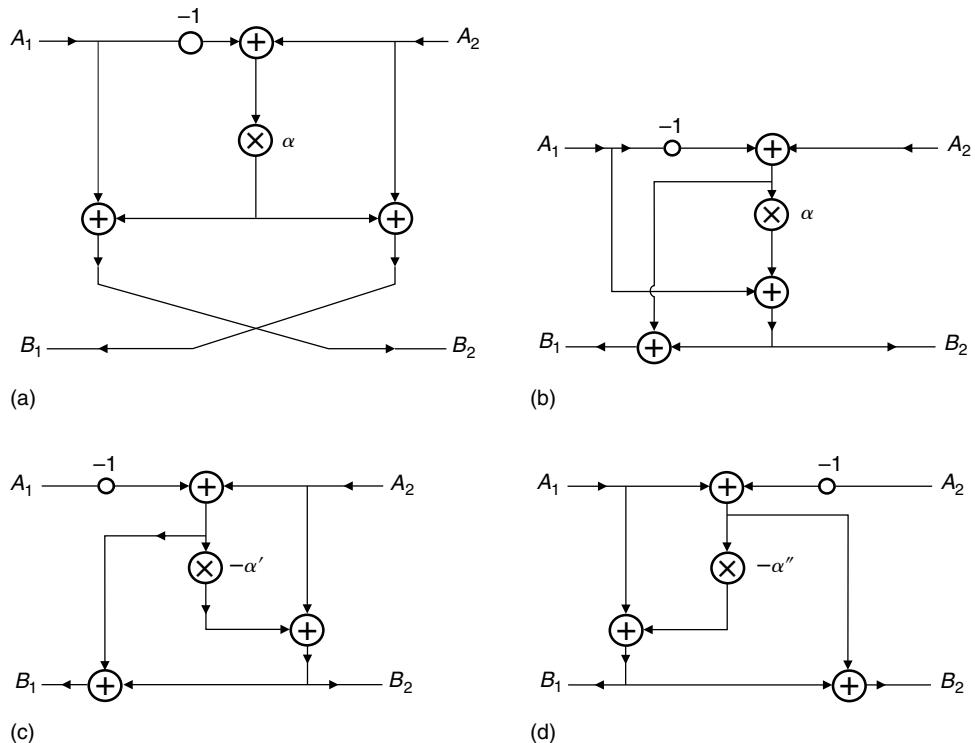


Fig. 13.31. Possible digital realizations of the general two-port adaptors based on: (a) Equation (13.190); (b) Equation (13.191); (c) Equation (13.192); (d) Equation (13.193).

Eliminating V_1 and I_1 from the above equations, we obtain

$$\left. \begin{aligned} B_1 &= A_2 + \alpha (A_2 - A_1) \\ B_2 &= A_1 + \alpha (A_2 - A_1) \end{aligned} \right\}, \quad (13.190)$$

where $\alpha = (R_1 - R_2)/(R_1 + R_2)$. A realization for this general two-port adaptor is shown in Figure 13.31a.

Expressing B_1 as a function of B_2 in Equation (13.190), we get

$$\left. \begin{aligned} B_1 &= B_2 - A_1 + A_2 \\ B_2 &= A_1 + \alpha (A_2 - A_1) \end{aligned} \right\}, \quad (13.191)$$

leading to a modified version of the two-port adaptor, as shown in Figure 13.31b.

Other alternative forms of two-port adaptors are generated by expressing the equations for B_1 and B_2 in different ways, such as

$$\left. \begin{aligned} B_1 &= B_2 - A_1 + A_2 \\ B_2 &= A_2 - \alpha' (A_2 - A_1) \end{aligned} \right\}, \quad (13.192)$$

where $\alpha' = 2R_2/(R_1 + R_2)$, generating the structure seen in Figure 13.31c, or

$$\left. \begin{aligned} B_1 &= A_1 - \alpha'' (A_1 - A_2) \\ B_2 &= B_1 + A_1 - A_2 \end{aligned} \right\}, \quad (13.193)$$

where $\alpha'' = 2R_1/(R_1 + R_2)$, leading to the structure seen in Figure 13.31d.

It is worth observing that the incident and reflected waves in any port could be expressed in the time domain – that is, through the instantaneous signal values ($a_i(k)$ and $b_i(k)$) – or in the frequency domain ($A_i(z)$ and $B_i(z)$), corresponding to the steady-state description of the wave signals.

13.6.2.7 The n -port parallel adaptor

In cases where we need to interconnect n elements in parallel, with port resistances given by R_1, R_2, \dots, R_n , it is necessary to use an n -port parallel adaptor. The symbol to represent the n -port parallel adaptor is shown in Figure 13.32. Figure 13.33 illustrates a three-port parallel adaptor.

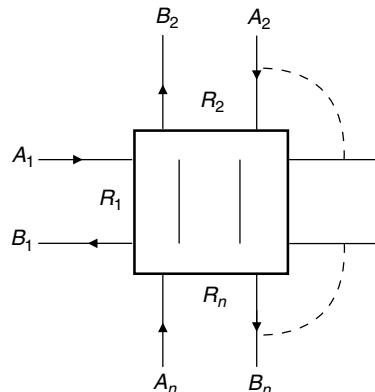


Fig. 13.32. Symbol of the n -port parallel adaptor.

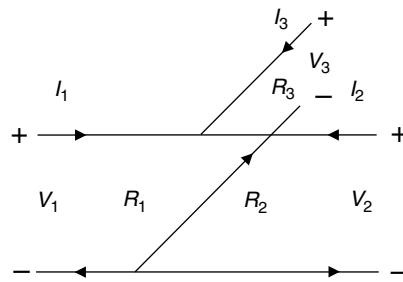


Fig. 13.33. The three-port parallel adaptor.

The wave equation on each port of a parallel adaptor is given by

$$\left. \begin{aligned} A_k &= V_k + R_k I_k \\ B_k &= V_k - R_k I_k \end{aligned} \right\}, \quad (13.194)$$

for $k = 1, 2, \dots, n$. As all ports are connected in parallel, we then have that

$$\left. \begin{aligned} V_1 &= V_2 = \dots = V_n \\ I_1 + I_2 + \dots + I_n &= 0 \end{aligned} \right\}. \quad (13.195)$$

After some algebraic manipulation to eliminate V_k and I_k , we have that

$$B_k = A_0 - A_k, \quad (13.196)$$

where

$$A_0 = \sum_{k=1}^n \alpha_k A_k, \quad (13.197)$$

with

$$\alpha_k = \frac{2G_k}{G_1 + G_2 + \dots + G_n} \quad (13.198)$$

and

$$G_k = \frac{1}{R_k}. \quad (13.199)$$

From Equation (13.198), we note that

$$\alpha_1 + \alpha_2 + \dots + \alpha_n = 2; \quad (13.200)$$

thus, we can eliminate an internal multiplication of the adaptor, since the computation related to one of the α_i is not required. If we calculate α_n as a function of the remaining α_i ,

we can express A_0 as

$$\begin{aligned} A_0 &= \sum_{k=1}^{n-1} \alpha_k A_k + \alpha_n A_n \\ &= \sum_{k=1}^{n-1} \alpha_k A_k + [2 - (\alpha_1 + \alpha_2 + \cdots + \alpha_{n-1})] A_n \\ &= 2A_n + \sum_{k=1}^{n-1} \alpha_k (A_k - A_n), \end{aligned} \quad (13.201)$$

where only $(n - 1)$ multipliers are required to calculate A_0 . In this case, port n is called the dependent port. It is also worth observing that if we have several port resistances R_k with the same value, the number of multiplications can be reduced further. If, however, $\sum_{k=1}^{n-1} \alpha_k \approx 2$, the error in computing α_n may be very high due to quantization effects. In this case, it is better to choose another port k , with α_k as large as possible, to be the dependent one.

In practice, the three-port adaptors are the most widely used in wave digital filters. A possible implementation for a three-port parallel adaptor is shown in Figure 13.34a, which corresponds to the direct realization of Equation (13.196), with A_0 calculated using Equation (13.201).

Substituting Equation (13.201) into Equation (13.196), with $k = n$, then

$$\left. \begin{aligned} B_n &= A_0 - A_n = A_n + \sum_{k=1}^{n-1} \alpha_k (A_k - A_n) \\ B_k &= A_0 - A_k = B_n + A_n - A_k \end{aligned} \right\}, \quad (13.202)$$

for $k = 1, 2, \dots, (n - 1)$, and we end up with an alternative realization for the three-port parallel adaptor, as seen in Figure 13.34b.

Analyzing Equation (13.202), we observe that the reflection wave B_i is directly dependent on the incident wave A_i at the same port. Hence, if two arbitrary adaptors are directly interconnected, a delay-free loop will appear between the two adaptors, as shown in Figure 13.35. A solution to this problem is to choose one of the α in the adaptor to be equal to one. For example, $\alpha_n = 1$. In this case, according to Equations (13.196), (13.198), and (13.201), the equations describing the parallel adaptor become

$$\left. \begin{aligned} G_n &= G_1 + G_2 + \cdots + G_{n-1} \\ B_n &= \sum_{k=1}^{n-1} \alpha_k A_k \end{aligned} \right\} \quad (13.203)$$

and the expression for B_n becomes independent of A_n , thus eliminating the delay-free loops at port n . In this case, Equation (13.200) becomes

$$\alpha_1 + \alpha_2 + \cdots + \alpha_{n-1} = 1 \quad (13.204)$$

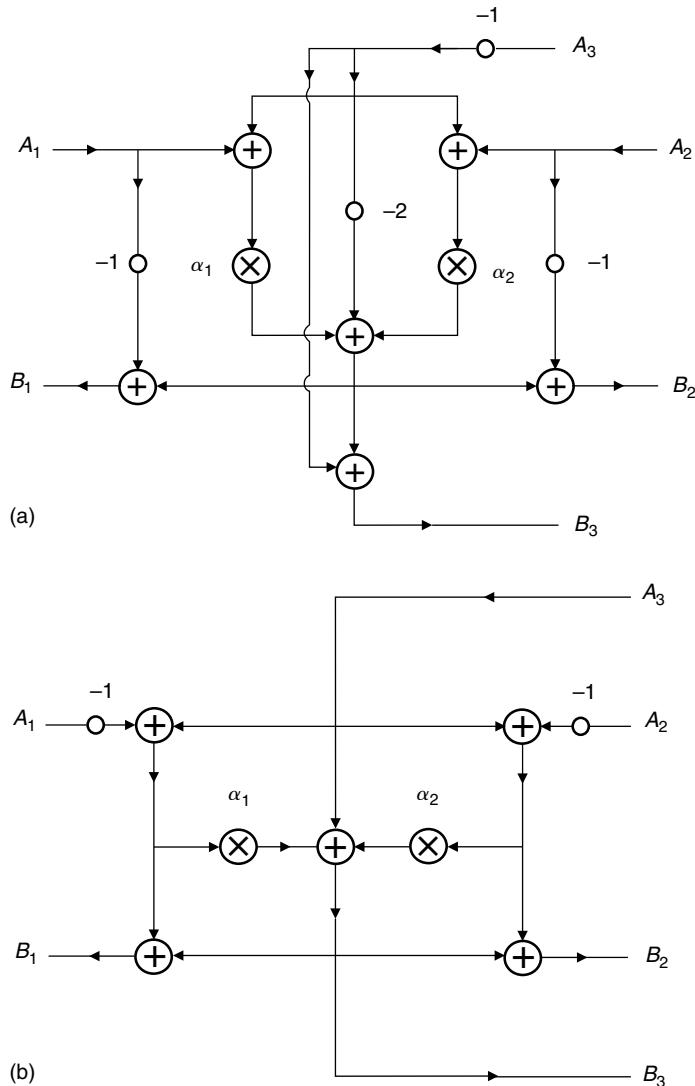


Fig. 13.34. Possible digital realizations of the three-port parallel adaptor based on: (a) Equation (13.201); (b) Equation (13.202).

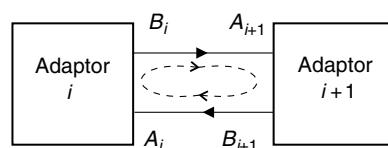


Fig. 13.35. Adaptor interconnection.

which still allows one of the α_i , for $i = 1, 2, \dots, (n - 1)$, to be expressed as a function of the others, thus eliminating one multiplication.

It is worth observing that choosing one of the $\alpha = 1$ does not imply any loss of generality in the wave digital filter design. In fact, at the ports corresponding to these coefficients, the port resistances can be chosen arbitrarily, since they are used only for interconnection and, therefore, they do not depend on any component value of the analog prototype. For example, the resistance of the ports common to two interconnected adaptors must be the same but can have arbitrary values. For instance, in the case of a three-port parallel adaptor, the describing equations would be

$$\left. \begin{aligned} \alpha_2 &= 1 - \alpha_1 \\ B_3 &= \alpha_1 A_1 + (1 - \alpha_1) A_2 \\ B_2 &= A_0 - A_2 = \alpha_1 A_1 + (1 - \alpha_1) A_2 + A_3 - A_2 = \alpha_1 (A_1 - A_2) + A_3 \\ B_1 &= \alpha_1 A_1 + (1 - \alpha_1) A_2 + A_3 - A_1 = (1 - \alpha_1) (A_2 - A_1) + A_3 \end{aligned} \right\}, \quad (13.205)$$

the realization of which is shown in Figure 13.36. Note that the port with no possibility of delay-free loops is marked with (\vdash), and is known as the reflection-free port.

A parallel adaptor, as illustrated in Figure 13.37, can be interpreted as a parallel connection of n ports with $(n - 2)$ auxiliary ports, which are introduced to provide separation among several external ports. The same figure also shows the symbolic representation of the n -port parallel adaptor consisting of several three-port adaptors.

13.6.2.8 The n -port series adaptor

In the situation where we need to interconnect n elements in series with distinct port resistances R_1, R_2, \dots, R_n , we need to use an n -port series adaptor, whose symbol is shown in

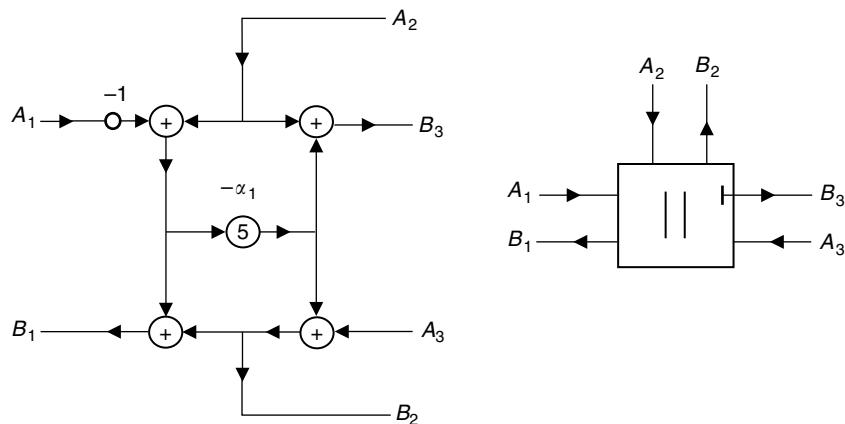


Fig. 13.36. Reflection-free parallel adaptor at port 3.

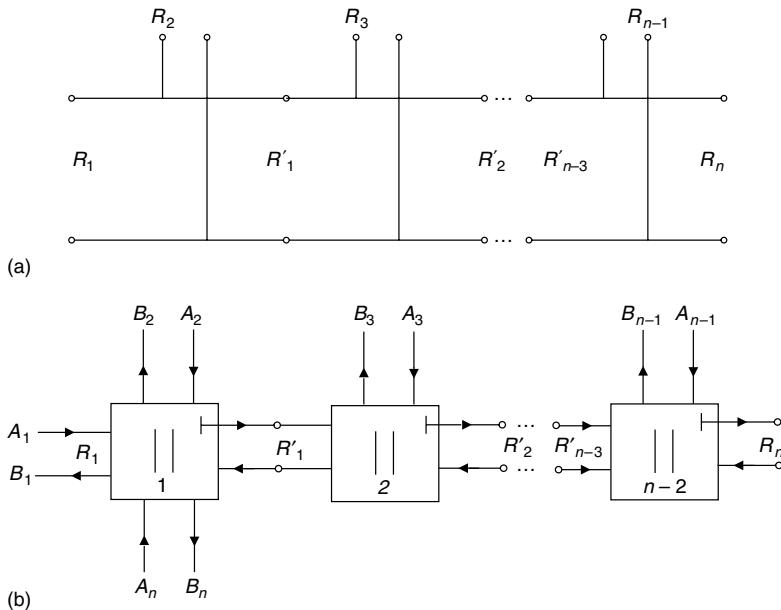


Fig. 13.37. The n -port parallel adaptor: (a) equivalent connection; (b) interpretation as several three-port parallel adaptors.

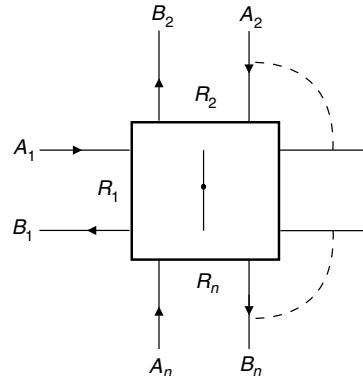


Fig. 13.38. Symbol of the n -port series adaptor.

Figure 13.38. In this case, the wave equations for each port are

$$\left. \begin{aligned} A_k &= V_k + R_k I_k \\ B_k &= V_k - R_k I_k \end{aligned} \right\}, \quad (13.206)$$

for $k = 1, 2, \dots, n$. We then must have that

$$\left. \begin{aligned} V_1 + V_2 + \dots + V_n &= 0 \\ I_1 = I_2 = \dots = I_n &= I \end{aligned} \right\}. \quad (13.207)$$

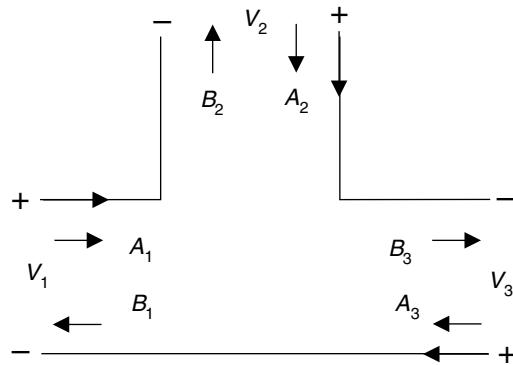


Fig. 13.39. The three-port series adaptor.

Figure 13.39 shows a possible three-port series adaptor.

Since

$$\sum_{i=1}^n A_i = \sum_{i=1}^n V_i + \sum_{i=1}^n R_i I_i = I \sum_{i=1}^n R_i, \quad (13.208)$$

it follows that

$$B_k = A_k - 2R_k I_k = A_k - \frac{2R_k}{\sum_{i=1}^n R_i} \sum_{i=1}^n A_i = A_k - \beta_k A_0, \quad (13.209)$$

where

$$\left. \begin{aligned} \beta_k &= \frac{2R_k}{\sum_{i=1}^n R_i} \\ A_0 &= \sum_{i=1}^n A_i \end{aligned} \right\}. \quad (13.210)$$

From Equation (13.210), we observe that

$$\sum_{k=1}^n \beta_k = 2. \quad (13.211)$$

Therefore, it is possible to eliminate a multiplication, as previously done for the parallel adaptor, by expressing β_n as a function of the remaining β , and then

$$B_n = A_n + (-2 + \beta_1 + \beta_2 + \cdots + \beta_{n-1}) A_0. \quad (13.212)$$

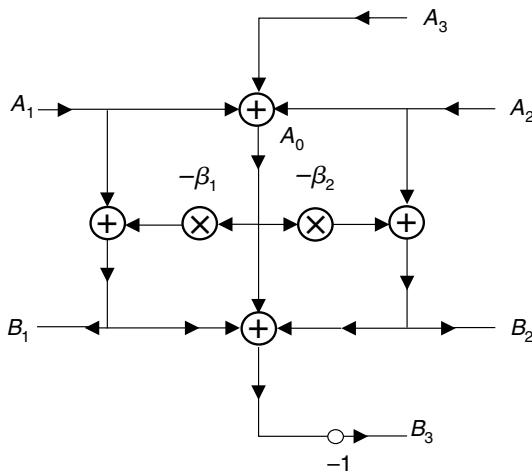


Fig. 13.40. Digital realization of the three-port series adaptor.

Since from Equation (11.165)

$$\sum_{k=1}^{n-1} B_k = \sum_{k=1}^{n-1} A_k - A_0 \sum_{k=1}^{n-1} \beta_k, \quad (13.213)$$

then

$$B_n = A_n - 2A_0 + \sum_{k=1}^{n-1} A_k - \sum_{k=1}^{n-1} B_k = -A_0 - \sum_{k=1}^{n-1} B_k, \quad (13.214)$$

where port n is the so-called dependent port.

The three-port adaptor realized from the equations above is shown in Figure 13.40. The specific equations describing this adaptor are given by

$$\left. \begin{aligned} B_1 &= A_1 - \beta_1 A_0 \\ B_2 &= A_2 - \beta_2 A_0 \\ B_3 &= -(A_0 + B_1 + B_2) \end{aligned} \right\}, \quad (13.215)$$

where $A_0 = A_1 + A_2 + A_3$.

For the series adaptors, we avoid the delay-free loops by choosing one of the $\beta = 1$. For example, if we choose $\beta_n = 1$, we have that

$$\left. \begin{aligned} R_n &= R_1 + R_2 + \cdots + R_{n-1} \\ B_n &= (A_n - \beta_n A_0) = -A_1 + A_2 + \cdots + A_{n-1} \end{aligned} \right\}. \quad (13.216)$$

Equation (13.211) can now be replaced by

$$\beta_1 + \beta_2 + \cdots + \beta_{n-1} = 1, \quad (13.217)$$

which allows one of the β_i to be calculated from the remaining ones.

A three-port series adaptor having $\beta_3 = 1$ and β_2 described as a function of β_1 is described by

$$\left. \begin{aligned} \beta_2 &= 1 - \beta_1 \\ B_1 &= A_1 - \beta_1 A_0 \\ B_2 &= A_2 - (1 - \beta_1) A_0 = A_2 - (A_1 + A_2 + A_3) + \beta_1 A_0 = \beta_1 A_0 - (A_1 + A_3) \\ B_3 &= A_3 - A_0 = -(A_1 + A_2) \end{aligned} \right\} \quad (13.218)$$

and its implementation is depicted in Figure 13.41. Note again that the port which avoids delay-free loops is marked with (−).

Consider now a series connection with inverted odd-port orientations, starting from port 3 and with $(n - 2)$ auxiliary ports used for separation, as depicted in Figure 13.42a. We can easily show that such a series connection can be implemented through several elementary series adaptors, as shown in Figure 13.42b.

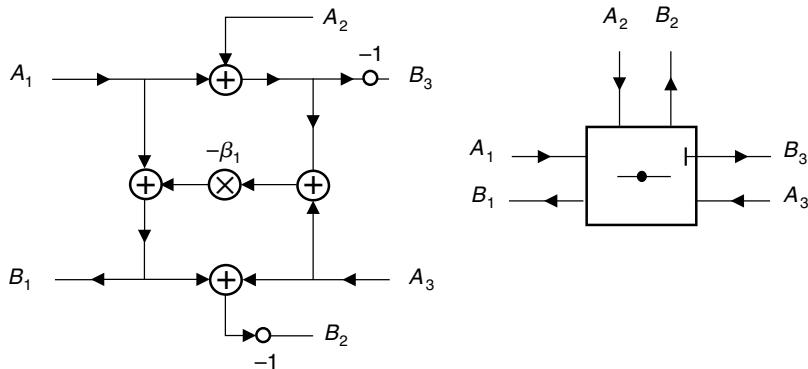


Fig. 13.41. Reflection-free series adaptor at port 3.

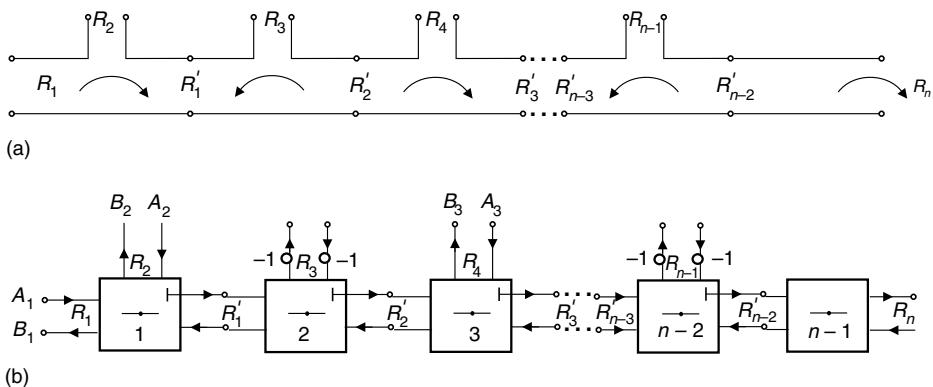


Fig. 13.42. The n -port series adaptor: (a) equivalent connection; (b) interpretation as several three-port adaptors.

13.6.3 Lattice wave digital filters

In some designs, it is preferable to implement a wave digital filter from a lattice analog network rather than from a ladder analog network. This is because, when implemented digitally, the lattice structures have low sensitivity in the filter passband and high sensitivity in the stopband. There are two explanations for the sensitivity properties of the lattice realization. First, any changes in the coefficients belonging to the adaptors of the lattice structure do not destroy its symmetry, whereas in the symmetric ladder this symmetry can be lost. The second reason applies to the design of filters with zeros on the unit circle. In the lattice structures, quantization usually moves these zeros away from the unit circle, whereas in the ladder structures the zeros always move around the unit circle.

Fortunately, all symmetric ladder networks can be transformed into lattice networks by applying the so-called Bartlett bisection theorem (Van Valkenburg, 1974). This subsection deals with the implementation of lattice wave digital filters.

Given the symmetric analog lattice network of Figure 13.43, where Z_1 and Z_2 are the lattice impedances, it can be shown that the incident and reflected waves are related by

$$\left. \begin{aligned} B_1 &= (S_1/2)(A_1 - A_2) + (S_2/2)(A_1 + A_2) \\ B_2 &= -(S_1/2)(A_1 - A_2) + (S_2/2)(A_1 + A_2) \end{aligned} \right\}, \quad (13.219)$$

where

$$\left. \begin{aligned} S_1 &= \frac{Z_1 - R}{Z_1 + R} \\ S_2 &= \frac{Z_2 - R}{Z_2 + R} \end{aligned} \right\}, \quad (13.220)$$

with S_1 and S_2 being the reflectances of the impedances Z_1 and Z_2 respectively. That is, S_1 and S_2 correspond to the ratio between the reflected and incident waves at ports Z_1 and Z_2 with port resistance R , since

$$\frac{B}{A} = \frac{V - RI}{V + RI} = \frac{Z - R}{Z + R}. \quad (13.221)$$

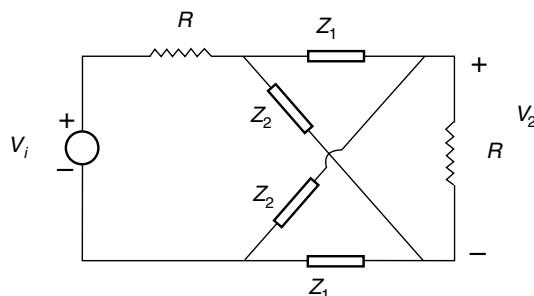


Fig. 13.43. Analog lattice network.

The lattice realization then consists solely of impedance realizations, as illustrated in Figure 13.44. Note that, in this figure, since the network is terminated by a resistance, then $A_2 = 0$.

Example 13.6. Realize the lowpass filter represented in Figure 13.45, using a wave lattice network.

Solution

The circuit in Figure 13.45 is a symmetric ladder network, as is made clear when it is redrawn as in Figure 13.46.

Bartlett's bisection theorem states that, when you have a two-port network composed of two equal half networks connected by any number of wires, as illustrated in Figure 13.47, then it is equivalent to a lattice network as in Figure 13.43. Figure 13.46 is a good example of a symmetric ladder network.

The impedance Z_1 is equal to the input impedance of any half network when the connection wires to the other half are short circuited, and Z_2 is equal to the input impedance of any half network when the connection wires to the other half are open. This is illustrated in Figure 13.48, where the determinations of the impedances Z_1 and Z_2 of the equivalent lattice are shown. Figure 13.49 shows the computation of Z_1 and Z_2 for this example.

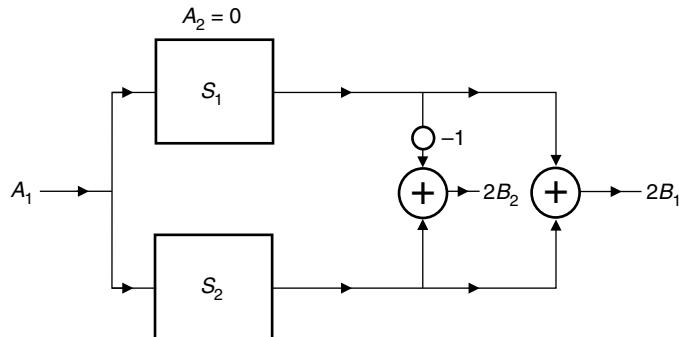


Fig. 13.44. Wave digital lattice representation.

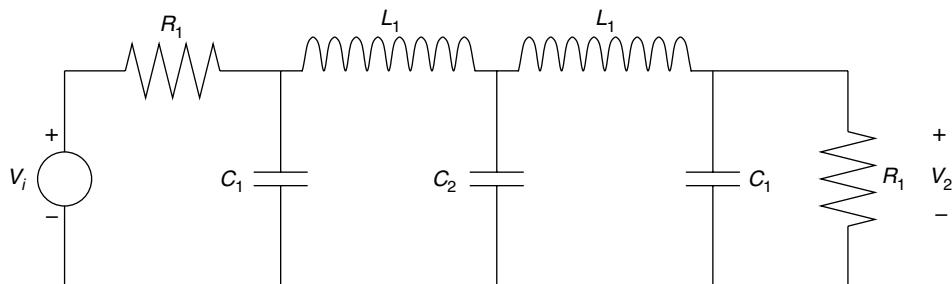


Fig. 13.45. Lowpass RLC network.

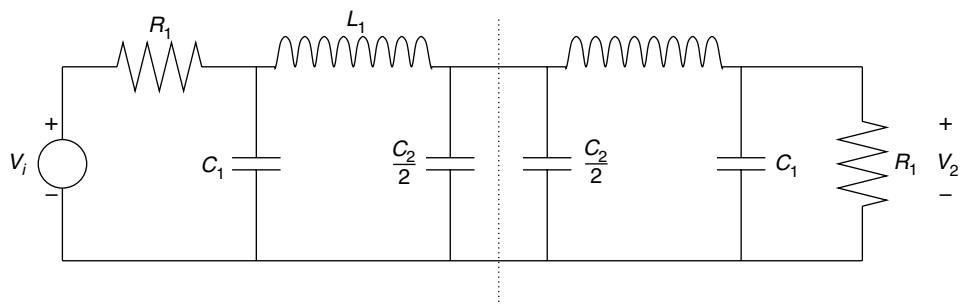


Fig. 13.46. Symmetric ladder network.

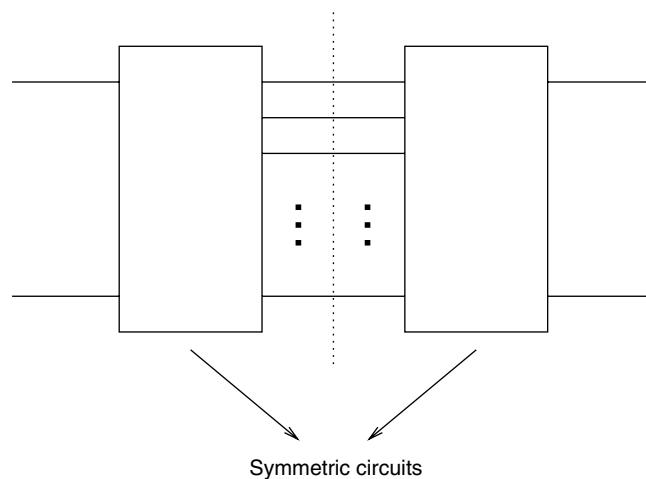


Fig. 13.47. Generic symmetric ladder network.

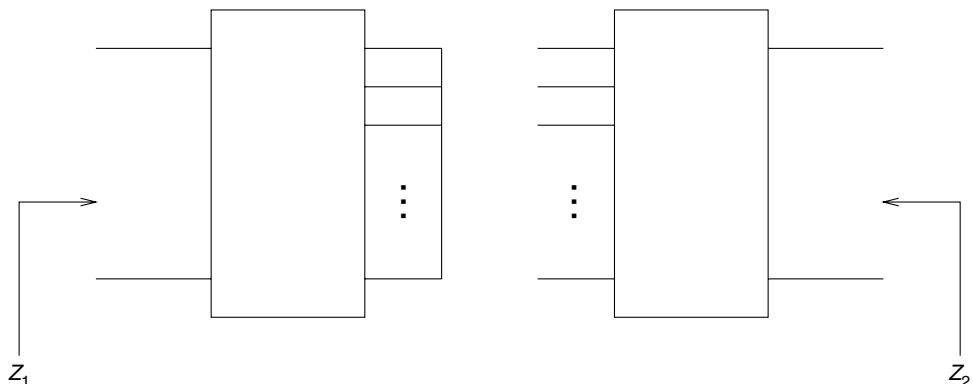


Fig. 13.48. Computation of the lattice's impedances Z_1 and Z_2 for the generic case.

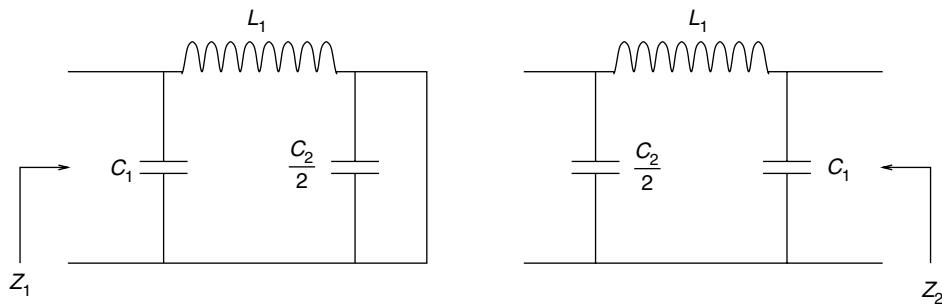


Fig. 13.49. Computation of the lattice's impedances Z_1 and Z_2 for Example 13.6.

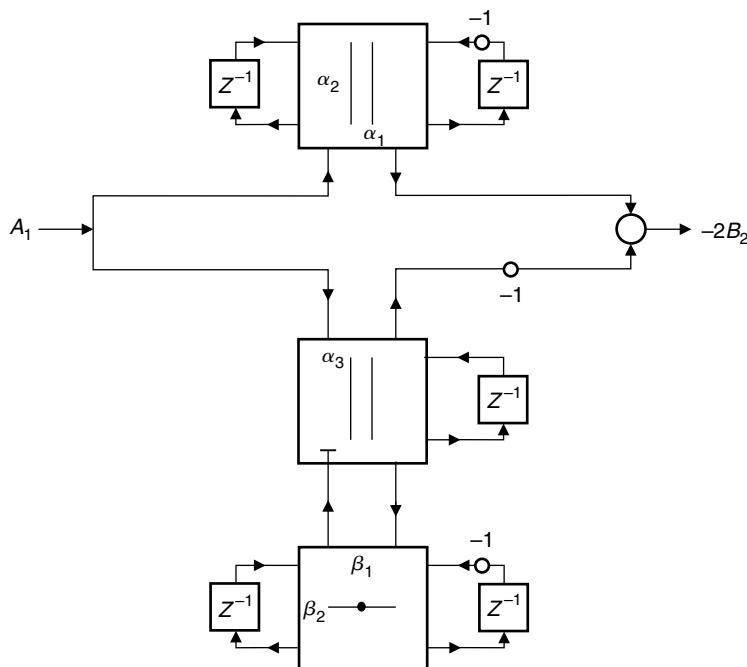


Fig. 13.50. Resulting digital lattice network.

From the resulting lattice network, the final wave filter is then as represented in Figure 13.50, where

$$\alpha_1 = \frac{2G_1}{G_1 + (2C_1/T) + (T/2L_1)} \quad (13.222)$$

$$\alpha_2 = \frac{2(2C_1/T)}{G_1 + (2C_1/T) + (T/2L_1)} \quad (13.223)$$

$$\alpha_3 = \frac{2G_1}{G_1 + (2C_1/T) + G_3} + \frac{2G_1}{2G_1 + (4C_1/T)} = \frac{G_1}{G_1 + (2C_1/T)} \quad (13.224)$$

$$G_3 = G_1 + (2C_1/T) \quad (13.225)$$

$$\beta_1 = \frac{2R_3}{R_3 + (T/C_2) + (2L_1/T)} \quad (13.226)$$

$$\beta_2 = \frac{(2T/C_2)}{R_3 + (T/C_2) + (2L_1/T)}. \quad (13.227)$$

One should note that, since we want a network that generates the output voltage at the load, then B_2 is the only variable of interest to us; therefore, B_1 does not need to be computed. \triangle

Example 13.7. Realize the ladder filter represented in Figure 13.51 using a wave network.

Solution

The connections among the elements can be interpreted as illustrated in Figure 13.52. The resulting wave filter should be represented as in Figure 13.53, where the choice of the reflection-free ports is arbitrary.

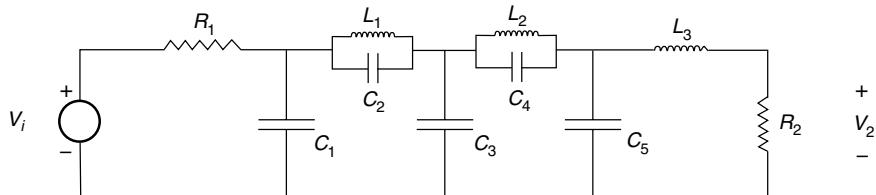


Fig. 13.51.

Ladder RLC network.

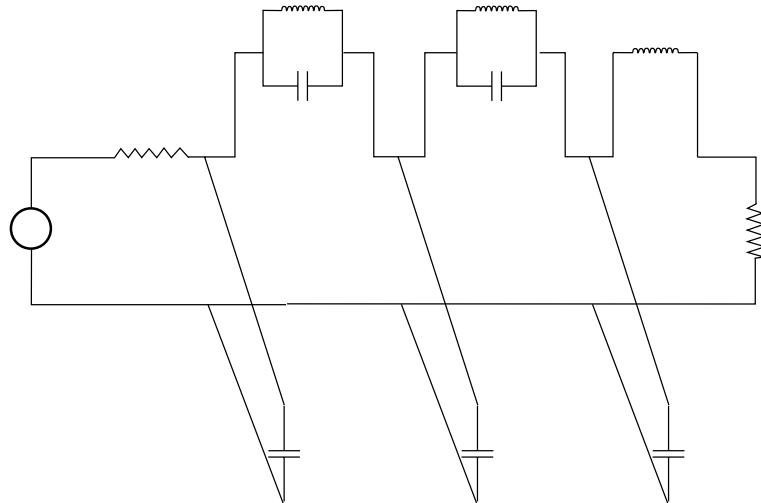


Fig. 13.52.

Component connections.

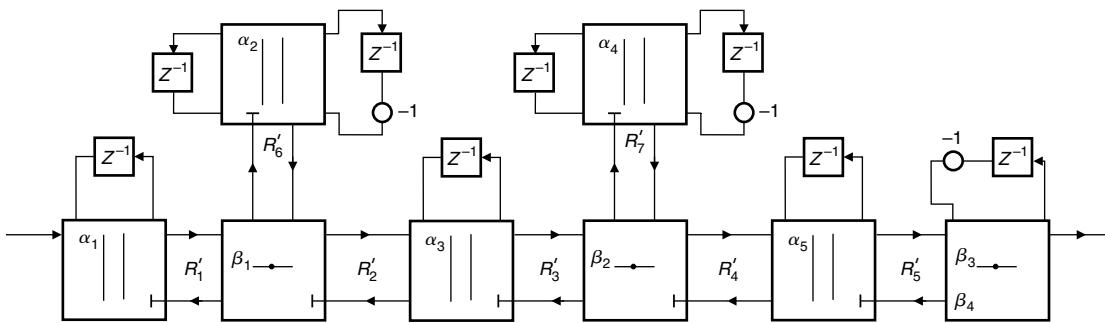


Fig. 13.53. Resulting digital wave network.

The equations below describe how to calculate the multiplier coefficient of each adaptor, and Figure 13.54 depicts the resulting wave digital filter realization.

$$G'_1 = G_1 + \frac{2C_1}{T} \quad (13.228)$$

$$\alpha_1 = \frac{2G_1}{G_1 + (2C_1/T) + G'_1} = \frac{2G_1}{2G_1 + (4C_1/T)} = \frac{G_1}{G_1 + (2C_1/T)} \quad (13.229)$$

$$G'_6 = \frac{2C_2}{T} + \frac{T}{2L_1} \quad (13.230)$$

$$\alpha_2 = \frac{2(2C_2/T)}{(2C_2/T) + (T/2L_1) + G'_6} = \frac{2C_2/T}{(2C_2/T) + (T/2L_1)} \quad (13.231)$$

$$R'_2 = R'_1 + R'_6 \quad (13.232)$$

$$\beta_1 = \frac{R'_1}{R'_1 + R'_6} = \frac{1}{1 + \frac{G_1 + (2C_1/T)}{(2C_2/T) + (T/2L_1)}} \quad (13.233)$$

$$G'_3 = G'_2 + \frac{2C_3}{T} \quad (13.234)$$

$$\alpha_3 = \frac{G'_2}{G'_2 + (2C_3/T)} \quad (13.235)$$

$$G'_7 = \frac{2C_4}{T} + \frac{T}{2L_2} \quad (13.236)$$

$$\alpha_4 = \frac{2C_4/T}{(2C_4/T) + (T/2L_2)} \quad (13.237)$$

$$R'_4 = R'_3 + R'_7 \quad (13.238)$$

$$\beta_2 = \frac{R'_3}{R'_3 + R'_7} \quad (13.239)$$

$$G'_5 = G'_4 + \frac{2C_5}{T} \quad (13.240)$$

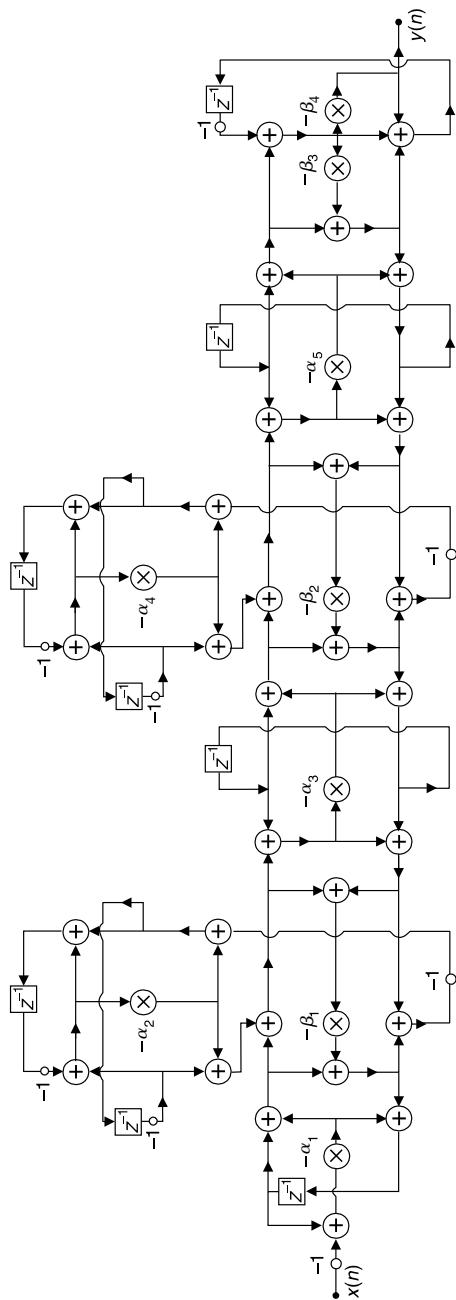


Fig. 13.54. Resulting wave filter realization.

$$\alpha_5 = \frac{2G'_4}{G'_4 + (2C_5/T)} \quad (13.241)$$

$$\beta_3 = \frac{2R'_5}{R'_5 + (2L_3/T) + R_2} \quad (13.242)$$

$$\beta_4 = \frac{2R_2}{R'_5 + (2L_3/T) + R_2}. \quad (13.243)$$

△

13.7 Do-it-yourself: efficient IIR structures

Experiment 13.1

Consider the cascade filter designed in Example 13.1 and described in Table 13.4 with the gain constant h_0 . Given the transfer function of each section

$$H_i(z) = \frac{N_i(z)}{D_i(z)} = \frac{\text{gamma0i}z^2 + \text{gamma1i}z + \text{gamma2i}}{z^2 + \text{m1i}z + \text{m2i}}, \quad (13.244)$$

the corresponding peaking factor, P_i , as defined in Equation (13.16), can be determined in MATLAB as

```
N_i = [gamma0i gamma1i gamma2i]; D_i = [1 m1i m2i];
npoints = 1000;
Hi = freqz(N_i,D_i,npoints);
Hi_infty = max(abs(Hi));
Hi_2 = sqrt(sum(abs(Hi).^2)/npoints);
Pi = Hi_infty/Hi_2;
```

Using the coefficient values provided in Table 13.4, one gets

$$\left. \begin{aligned} P1 &= 11.2719 \\ P2 &= 13.4088 \\ P3 &= 12.5049 \end{aligned} \right\}. \quad (13.245)$$

If we scale the filter with the L_2 norm, then, in order to minimize the maximum value of the output-noise PSD, one should change the section order to get a decreasing P_i sequence, as seen in Figure 13.55.

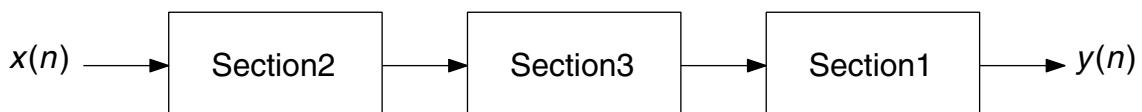


Fig. 13.55. New section ordering in cascade filter to minimize peak value of output-noise PSD.

Scaling the old Section 2 using L_2 norm, we get

$$\lambda_2 = \frac{1}{\| F_2(z) \|_2} = \frac{1}{\| \frac{h_0}{D_2(z)} \|_2}, \quad (13.246)$$

such that

```
D_2 = [1 m12 m22];
F_2 = freqz(h_0,D_2,npoints);
lambda_2 = 1/sqrt(sum(abs(F_2).^2)/npoints);
```

Scaling the old Section 3, taking into account that it now comes after the old Section 2, we get

$$\lambda_3 = \frac{1}{\| H_2(z)F_3(z) \|_2} = \frac{1}{\| (h_0N_2(z))/(D_2(z)D_3(z)) \|_2} \quad (13.247)$$

such that

```
N_2 = [gamma02 gamma12 gamma22];
D_3 = [1 m13 m23];
D_2D_3 = conv(D_2,D_3);
H_2F_3 = freqz(h_0N_2,D_2D_3,npoints);
lambda_3 = 1/sqrt(sum(abs(H_2F_3).^2)/npoints);
```

Finally, scaling the old Section 1, taking into account that it comes after the old Sections 2 and 3, we get

$$\lambda_1 = \frac{1}{\| H_2(z)H_3(z)F_1(z) \|_2} = \frac{1}{\| (h_0N_2(z)N_3(z))/(D_2(z)D_3(z)D_1(z)) \|_2} \quad (13.248)$$

such that

```
N_3 = [gamma03 gamma13 gamma23];
D_1 = [1 m11 m21];
N_2N_3 = conv(N_2,N_3);
D_2D_3D_1 = conv(D_2D_3,D_1);
H_2H_3F_1 = freqz(h_0N_2N_3,D_2D_3D_1,npoints);
lambda_1 = 1/sqrt(sum(abs(H_2H_3F_1).^2)/npoints);
```

yielding

$$\left. \begin{array}{l} \lambda_2 = \text{lambda_2} = 522.2077 \\ \lambda_3 = \text{lambda_3} = 83.8126 \\ \lambda_1 = \text{lambda_1} = 12.1895 \end{array} \right\}. \quad (13.249)$$

△

13.8 Efficient IIR structures with MATLAB

MATLAB has several functions related to the structures described in this chapter. These functions have already been described in Chapters 4 and 12. In order to make reference easier, we present below a list of these functions.

- `sos2tf`: Converts the cascade form into the direct form. See Chapter 4 for a list of parameters. Notice that there is no direct command that reverses this operation.
- `residuez`: Performs the partial-fraction expansion in the z domain, if there are two input parameters. This command considers complex roots. To obtain a parallel expansion of a given transfer function, one should combine such roots in complex conjugate pairs with the `cplxpairs` command (see Chapter 4) to form second-order sections with solely real coefficients. The `residuez` command also converts the partial-fraction expansion back to the original direct form. See Chapter 4 for a list of parameters.
- `tf2ss` and `ss2tf`: Convert the direct form into the state-space form, and vice versa. See Chapter 4 for a list of parameters.
- `sos2ss` and `ss2sos`: Convert the cascade form into the state-space form, and vice versa. See Chapter 4 for a list of parameters.
- `tf2latc` and `latc2tf`: Convert the direct form into the lattice form, and vice versa. See Chapter 12 for a list of parameters.

13.9 Summary

This chapter introduced several efficient structures for the realization of IIR digital filters.

For IIR filters, the designs of cascade and parallel forms were presented in detail. Emphasis was given to the cases where the second-order sections were comprised of direct-form sections, section-optimal state-space sections, and state-space structures which are free from limit cycles. These structures are modular and should always be considered as one of the main candidates for practical implementations.

We then addressed the issue of reducing the roundoff error at the filter output by introducing zeros to the noise transfer function by feeding back the quantization error. Such a technique is widely known as ESS. We also introduced closed-form equations to calculate the scaling factor of second-order sections facilitating the design of parallel-form filters of noncanonic structures with respect to the number of multipliers.

The lattice structures were presented in some detail. In particular, the structures discussed here utilize two-port sections with one, two, and four multipliers. These structures can be shown to be free from zero-input limit cycles, and their design methods rely on a polynomial-reduction strategy. We then presented doubly complementary filters built from allpass blocks, with an example of their application to filter bank design.

The last class of structures presented in this chapter was the one comprising the wave digital filters, which are designed from analog prototype filters. This wave strategy yields digital filters with very low sensitivity which are free from zero-input limit cycles.

13.10 Exercises

13.1 Consider the alternative parallel structure seen in Figure 13.56, whose transfer function is

$$H(z) = h'_0 + \sum_{k=1}^m \frac{\gamma'_{1k}z + \gamma'_{2k}}{z^2 + m_{1k}z + m_{2k}}.$$

Discuss the scaling procedure and determine the relative noise variance for this realization.

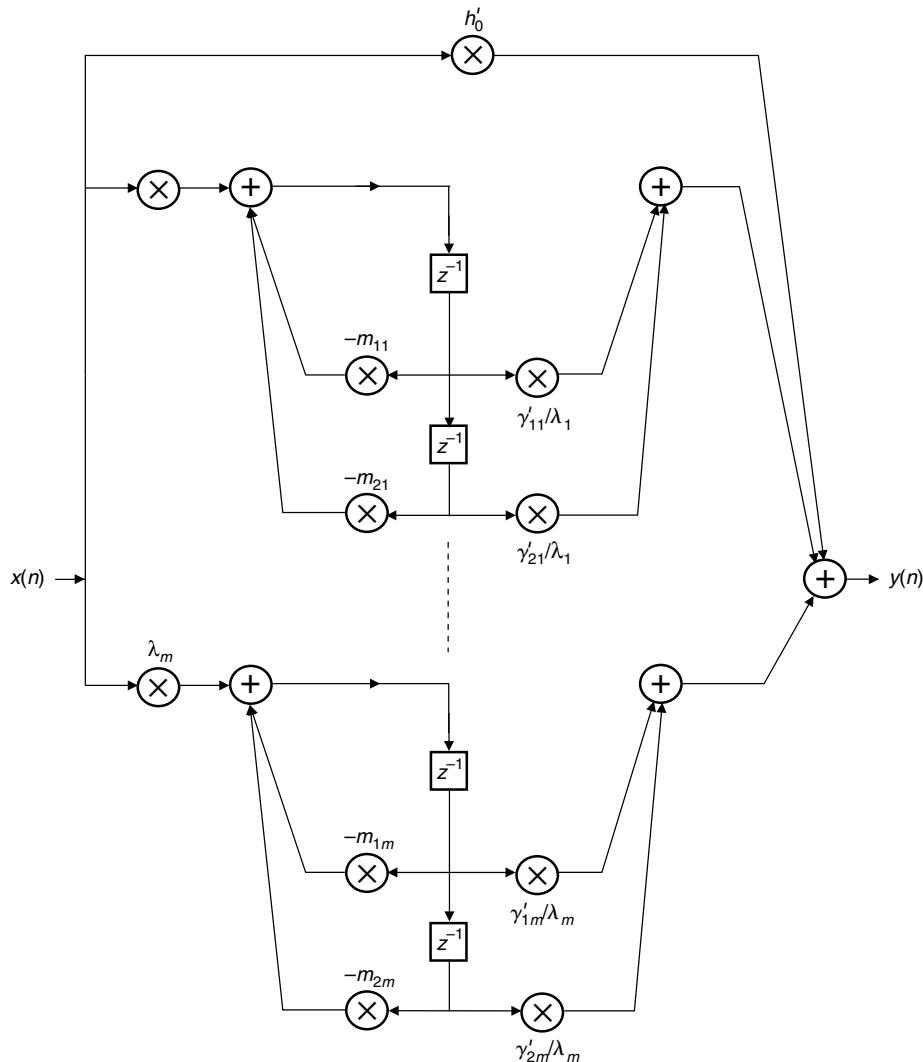


Fig. 13.56. Alternative parallel structure with direct-form sections.

13.2 Derive the expressions in Equations (13.20) and (13.21).

13.3 Show that the L_2 and L_∞ norms of a second-order $H(z)$ given by

$$H(z) = \frac{\gamma_1 z + \gamma_2}{z^2 + m_1 z + m_2}$$

are as given in Equations (13.28) and (13.29), respectively.

13.4 Derive the expressions for the output-noise PSD, as well as their variances, for odd-order cascade filters realized with the following second-order sections:

- (a) direct-form;
- (b) optimal state space;
- (c) state-space free of limit cycles.

Assume fixed-point arithmetic.

13.5 Repeat Exercise 13.4 assuming floating-point arithmetic.

13.6 From Equations (13.55), (13.56), and the definitions of $F'_i(z)$ and $G'_i(z)$ in Figures 11.20 and 11.16 respectively, derive Equations (13.57) and (13.58).

13.7 Verify that Equations (13.72)–(13.73) correspond to a state-space structure with minimum output noise.

13.8 Show that when the poles of a second-order section with transfer function

$$H(z) = d + \frac{\gamma_1 z + \gamma_2}{z^2 + m_1 z + m_2}$$

are complex conjugate, then the parameter σ defined in Equation (13.74) is necessarily real.

13.9 Show that the state-space section free of limit cycles, Structure I, as determined by Equations (13.97) and (13.98), has minimum roundoff noise only if

$$\frac{\gamma_1}{\gamma_2} = \frac{m_1 + 2}{m_2 - 1},$$

as indicated by Equation (13.103).

13.10 Design an elliptic filter satisfying the specifications below:

$$A_p = 0.4 \text{ dB}$$

$$A_r = 50 \text{ dB}$$

$$\Omega_{r1} = 1000 \text{ rad/s}$$

$$\Omega_{p1} = 1150 \text{ rad/s}$$

$$\Omega_{p2} = 1250 \text{ rad/s}$$

$$\Omega_{r2} = 1400 \text{ rad/s}$$

$$\Omega_s = 10000 \text{ rad/s.}$$

Realize the filter using:

- (a) parallel direct-form sections;
- (b) cascade of direct-form sections;

- (c) cascade of optimal state-space sections;
- (d) cascade of state-space sections without limit cycles.

The specifications must be satisfied when the coefficients are quantized in the range of 18 to 20 bits, including the sign bit.

- 13.11 Repeat Exercise 13.10 with the specifications below:

$$A_p = 1.0 \text{ dB}$$

$$A_r = 70 \text{ dB}$$

$$\omega_p = 0.025\pi \text{ rad/sample}$$

$$\omega_r = 0.04\pi \text{ rad/sample.}$$

- 13.12 Design an elliptic filter satisfying the specifications of Exercise 13.10, using a cascade of direct-form structures with the ESS technique.
- 13.13 Design an elliptic filter satisfying the specifications of Exercise 13.10, using parallel connection of second-order direct-form structures, with L_2 and L_∞ norms for scaling, employing the closed-form equations of Section 13.2.4.
- 13.14 Derive Equation (13.132).
- 13.15 Derive the scaling factor utilized in the two-multiplier lattice of Figure 13.9, in order to generate the one-multiplier lattice of Figure 13.12.
- 13.16 Derive the scaling factor to be used in the normalized lattice of Figure 13.13, in order to generate a three-multiplier lattice.
- 13.17 Design a two-multiplier lattice structure to implement the filter described in Exercise 13.10.
- 13.18 The coefficients of an allpass filter designed to generate doubly complementary filters of order $N_1 = 5$ are given in Table 13.16. Describe what happens to the properties of these filters when we replace z by z^2 .

Table 13.16. $F_1(z)$ allpass filter coefficients $a_{j,1}$.

$a_{0,1} = 1.0000$
$a_{1,1} = 0.0000$
$a_{2,1} = 0.4698$
$a_{3,1} = 0.0000$
$a_{4,1} = -0.0829$

- 13.19 The coefficients of an allpass filter designed to generate doubly complementary filters of order $N_1 = 9$ are given in Table 13.17.
- (a) Plot the phase response of the allpass filter and verify the validity of the specifications described in Equation (13.167).
- (b) Compute the difference in phase between $H_0(z)$ and $H_1(z)$ and comment on the results.

Table 13.17. $F_1(z)$ allpass filter coefficients $a_{j,1}$.

$a_{0,1} =$	1.0000
$a_{1,1} =$	0.0000
$a_{2,1} =$	0.4829
$a_{3,1} =$	0.0000
$a_{4,1} =$	-0.1007
$a_{5,1} =$	0.0000
$a_{6,1} =$	0.0352
$a_{7,1} =$	0.0000
$a_{8,1} =$	-0.0117

- (c) Propose the implementation of a QMF filter using the designed filters with minimum number of multiplications per sample.
- 13.20 Derive a realization for the circuit of Figure 13.45, using the standard wave digital filter method, and compare it with the wave lattice structure obtained in Figure 13.50, with respect to overall computational complexity.
- 13.21 Table 13.18 shows the multiplier coefficients of a scaled Chebyshev filter designed with the following specifications

$$A_p = 0.6 \text{ dB}$$

$$A_r = 48 \text{ dB}$$

$$\Omega_r = 3300 \text{ rad/s}$$

$$\Omega_p = 4000 \text{ rad/s}$$

$$\Omega_s = 10\,000 \text{ rad/s.}$$

Table 13.18. Coefficients of a Chebyshev filter. Scaling factor: $\lambda = 2.6167 \times 10^{-2}$.

Coefficient	Section 1	Section 2	Section 3
γ_0	0.0252	0.0883	1.0000
γ_1	-0.0503	-0.1768	-2.0000
γ_2	0.0252	0.0883	1.0000
m_0	1.5751	1.6489	1.5465
m_1	0.7809	0.7012	0.9172

The structure used is the cascade of second-order direct-form structures.

- (a) Plot the output-noise RPSD in decibels.
- (b) Determine the filter coefficient wordlength, using the sign-magnitude representation, that obtains at most 1.0 dB as the passband ripple.

- (c) Plot the filter magnitude response when the filter coefficients are quantized with the number of bits obtained in item (b). Compare the results with the nonquantized magnitude response.
- 13.22 Revisit Experiment 13.1 and determine:
- The peaking factor and the scaling factors for the 9-bit quantized cascade filter, as described in Table 13.5.
 - The relative output-noise variance for the nonquantized and 9-bit quantized versions of the scaled cascade filter.

References

- Adams, J. W. (1991a). FIR digital filters with least-squares stopbands subject to peak-gain constraints. *IEEE Transactions on Circuits and Systems*, 39, 376–88.
- Adams, J. W. (1991b). A new optimal window. *IEEE Transactions on Signal Processing*, 39, 1753–69.
- Adams, J. W. & Willson, Jr., A. N. (1983). A new approach to FIR digital filters with fewer multipliers and reduced sensitivity. *IEEE Transactions on Circuits and Systems*, CAS-30, 277–83.
- Adams, J. W. & Willson, Jr., A. N. (1984). Some efficient digital prefilter structures. *IEEE Transactions on Circuits and Systems*, CAS-31, 260–5.
- Ahmed, N., Natarajan, T. & Rao, K. R. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, C-23, 90–3.
- Akansu, A. N. & Medley, M. J. (1999). *Wavelets, Subband and Block Transforms in Communications and Multimedia*. Boston, MA: Kluwer Academic Publishers.
- Analog Devices, Inc. (December 2004a). *ADSP-TS201 TigerSHARC Processor – Hardware Reference*. Analog Devices, Inc.
- Analog Devices, Inc. (March 2004b). *ADSP-TS206X SHARC Processor – User’s Manual*. Analog Devices, Inc.
- Analog Devices, Inc. (April 2005). *ADSP-TS201 TigerSHARC Processor – Programming Reference*. Analog Devices, Inc.
- Analog Devices, Inc. (February 2009). *ADSP-BF538/ADSP-BF538F Blackfin Processor – Hardware Reference*. Analog Devices, Inc.
- Ansari, R. & Liu, B. (1983). Efficient sampling rate alteration using recursive (IIR) digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31, 1366–73.
- Antonini, M., Barlaud, M., Mathieu, P. & Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1, 205–20.
- Antoniou, A. (1982). Accelerated procedure for the design of equiripple nonrecursive digital filters. *IEE Proceedings – Part G*, 129, 1–10.
- Antoniou, A. (1983). New improved method for design of weighted-Chebyschev, nonrecursive, digital filters. *IEEE Transactions on Circuits and Systems*, CAS-30, 740–50.
- Antoniou, A. (1993). *Digital Filters: Analysis, Design, and Applications*, 2nd edn. New York, NY: McGraw-Hill.
- Antoniou, A. (2006). *Digital Signal Processing: Signals, Systems, and Filters*. New York, NY: McGraw-Hill.
- Antoniou, A. & Lu, W.-S. (2007). *Practical Optimization: Algorithms and Engineering Applications*. New York, NY: Springer.

- Antoniou, A. & Rezk, M. G. (1977). Digital filters synthesis using concept of generalized-imittance converter. *IEE Journal of Electronics Circuits*, 1, 207–16.
- Antoniou, A. & Rezk, M. G. (1980). A comparison of cascade and wave fixed-point digital-filter structures. *IEEE Transactions on Circuits and Systems*, CAS-27, 1184–94.
- Apostol, T. M. (1967). *Calculus*, 2nd edn., volume I. Toronto: Xerox College Publishing.
- Avenhaus, E. (1972). On the design of digital filters with coefficients of limited wordlength. *IEEE Transactions on Audio and Electroacoustics*, AU-20, 206–12.
- Bauer, P. H. & Wang, J. (1993). Limit cycle bounds for floating-point implementations of second-order recursive digital filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39, 493–501. Corrections on 41, 176, February 1994.
- Belevitch, V. (1968). *Classical Network Theory*. San Francisco, CA: Holden-Day.
- Benvenuto, N., Franks, L. E. & Hill, Jr., F. S. (1984). On the design of FIR filters with power-of-two coefficients. *IEEE Transactions on Communications*, COM-32, 1299–307.
- Bhaskaran, V. & Konstantinides, K. (1997). *Image and Video Compression Standards: Algorithms and Architectures*. Boston, MA: Kluwer Academic Publishers.
- Bomar, B. W. (1989). On the design of second-order state-space digital filter sections. *IEEE Transactions on Circuits and Systems*, 36, 542–52.
- Bomar, B. W. & Joseph, R. D. (1987). Calculation of L_∞ norms in second-order state-space digital filter sections. *IEEE Transactions on Circuits and Systems*, CAS-34, 983–4.
- Bomar, B. W., Smith, L. M. & Joseph, R. D. (1997). Roundoff noise analysis of state-space digital filters implemented on floating-point digital signal processors. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 44, 952–5.
- Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge, UK: Cambridge University Press.
- Bracewell, R. N. (1984). The fast Hartley transform. *Proceedings of the IEEE*, 72, 1010–18.
- Bracewell, R. N. (1994). Aspects of the Hartley transform. *Proceedings of the IEEE*, 82, 381–6.
- Burrus, C. S. & Parks, T. W. (1970). Time domain design of recursive digital filters. *IEEE Transactions on Audio and Electroacoustics*, AU-18, 137–41.
- Butterweck, H. J. (1975). Suppression of parasitic oscillations in second-order digital filters by means of a controlled-rounding arithmetic. *Archiv Elektrotechnik und Übertragungstechnik*, 29, 371–4.
- Butterweck, H. J., van Meer, A. C. P. & Verkroost, G. (1984). New second-order digital filter sections without limit cycles. *IEEE Transactions on Circuits and Systems*, CAS-31, 141–6.
- Cabezas, J. C. E. & Diniz, P. S. R. (1990). FIR filters using interpolated prefilters and equalizers. *IEEE Transactions on Circuits and Systems*, 37, 17–23.
- Chang, T.-L. (1981). Suppression of limit cycles in digital filters designed with one magnitude-truncation quantizer. *IEEE Transactions on Circuits and Systems*, CAS-28, 107–11.
- Charalambous, C. & Antoniou, A. (1980). Equalisation of recursive digital filters. *IEE Proceedings – Part G*, 127, 219–25.

- Chen, W.-H., Smith, C. H. & Fralick, S. C. (1977). A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, COM-25, 1004–9.
- Cheney, E. W. (1966). *Introduction to Approximation Theory*. New York, NY: McGraw-Hill.
- Churchill, R. V. (1975). *Complex Variables and Applications*. New York, NY: McGraw-Hill.
- Claasen, T. A. C. M., Mecklenbräuker, W. F. G. & Peek, J. B. H. (1975). On the stability of the forced response of digital filters with overflow nonlinearities. *IEEE Transactions on Circuits and Systems*, CAS-22, 692–6.
- Cochran, W., Cooley, J., Favin, D. et al. (1967). What is the fast Fourier transform? *IEEE Transactions on Audio and Electroacoustics*, AU-15, 45–55.
- Cohen, A., Daubechies, I. & Feauveau, J. C. (1992). Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, XLV, 485–560.
- Constantinides, A. G. (1970). Spectral transformations for digital filters. *IEE Proceedings*, 117, 1585–90.
- Cooley, J. W. & Tukey, J. W. (1965). An algorithm for the machine computation of complex Fourier series. *Mathematics of Computation*, 19, 297–301.
- Cortelazzo, G. & Lightner, M. R. (1984). Simultaneous design of both magnitude and group delay of IIR and FIR filters based on multiple criterion optimization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32, 9949–67.
- Cover, T. M. & Thomas, J. A. (2006). *Elements of Information Theory*, 2nd edn. Hoboken, NJ: Wiley-Interscience.
- Crochiere, R. E. & Oppenheim, A. V. (1975). Analysis of linear digital network. *Proceedings of the IEEE*, 63, 581–93.
- Crochiere, R. E. & Rabiner, L. R. (1983). *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.
- Croisier, A., Esteban, D. & Galand, C. (1976). Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques. In *Proceedings of International Symposium on Information, Circuits and Systems*, Patras, Greece.
- Croisier, A., Esteban, D. J., Levilion, M. E. & Riso, V. (1973). Digital filter for PCM encoded signals. U.S. Patent no. 3777130.
- Daniels, R. W. (1974). *Approximation Methods for Electronic Filter Design*. New York, NY: McGraw-Hill.
- Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, XLI, 909–96.
- Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36, 961–1005.
- Daubechies, I. (1991). *Ten Lectures on Wavelets*. Pennsylvania, PA: Society for Industrial and Applied Mathematics.
- Daubechies, I. (1993). Orthonormal bases of compactly supported wavelets II. Variations on a theme. *SIAM Journal on Mathematical Analysis*, 24, 499–519.
- De la Vega, A. S., Diniz, P. S. R., Mesquita, A. C. & Antoniou, A. (1995). A modular distributed arithmetic implementation of the inner product with application to digital filters. *Journal of VLSI Signal Processing*, 10, 93–106.

- De Queiroz, R. L., Nguyen, T. Q. & Rao, K. R. (1996). The GenLOT: generalized linear-phase lapped orthogonal transform. *IEEE Transactions on Signal Processing*, 44, 497–507.
- Deczky, A. G. (1972). Synthesis of recursive digital filters using the minimum p error criterion. *IEEE Transactions on Audio and Electroacoustics*, AU-20, 257–263.
- Deller, Jr., J. R., Hansen, J. H. L. & Proakis, J. G. (2000). *Discrete-Time Processing of Speech Signals*. Piscataway, NJ: IEEE Press.
- Diniz, P. S. R. (1984). New improved structures for recursive digital filters. PhD thesis, Concordia University, Montreal, Canada.
- Diniz, P. S. R. (1988). Elimination of constant-input limit cycles in passive lattice digital filters. *IEEE Transactions on Circuits and Systems*, 35, 1188–90.
- Diniz, P. S. R. (2008). *Adaptive Filtering: Algorithms and Practical Implementation*, 3rd edn. New York, NY: Springer.
- Diniz, P. S. R. & Antoniou, A. (1985). Low sensitivity digital filter structures which are amenable to error-spectrum shaping. *IEEE Transactions on Circuits and Systems*, CAS-32, 1000–7.
- Diniz, P. S. R. & Antoniou, A. (1986). More economical state-space digital-filter structures which are free of constant-input limit cycles. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34, 807–15.
- Diniz, P. S. R. & Antoniou, A. (1988). Digital filter structures based on the concept of voltage generalized admittance converter. *Canadian Journal of Electrical and Computer Engineering*, 13, 90–8.
- Diniz, P. S. R. & Netto, S. L. (1999). On WLS–Chebyshev FIR digital filters. *Journal on Circuits, Systems, and Computers*, 9, 155–68.
- Diniz, P. S. R., Barcellos, L. C. & Netto, S. L. (2004). Design of high-complexity cosine-modulated transmultiplexers with sharp transition band. *IEEE Transactions on Signal Processing*, 52, 1278–88.
- Einstein, A. (1987). Method for the determination of the statistical values of observations concerning quantities subject to irregular fluctuations. *IEEE ASSP Magazine*, 4, 6.
- Elliott, D. F. & Rao, K. R. (1982). *Fast Transforms: Algorithms, Analyses, and Applications*. New York, NY: Academic Press.
- Evans, A. G. & Fischel, R. (1973). Optimal least squares time domain synthesis of recursive digital filters. *IEEE Transactions on Audio and Electroacoustics*, AU-21, 61–5.
- Fettweis, A. (1971a). Digital filters structures related to classical filters networks. *Archiv Elektrotechnik und Übertragungstechnik*, 25, 79–89.
- Fettweis, A. (1971b). A general theorem for signal-flow networks. *Archiv Elektrotechnik und Übertragungstechnik*, 25, 557–61.
- Fettweis, A. (1986). Wave digital filters: theory and practice. *Proceedings of the IEEE*, 74, 270–327.
- Fettweis, A. & Meerkötter, K. (1975a). On adaptors for wave digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, 516–25.
- Fettweis, A. & Meerkötter, K. (1975b). Suppression of parasitic oscillations in wave digital filters. *IEEE Transactions on Circuits and Systems*, CAS-22, 239–46.

- Fettweis, A., Levin, H. & Sedlmeyer, A. (1974). Wave digital lattice filters. *International Journal of Circuit Theory and Applications*, 2, 203–11.
- Field, D. J. (1993). Scale-invariance and self-similar wavelet transforms: an analysis of natural scenes and mammalian visual systems. In M. Farge, J. C. R. Hunt & J. C. Vassilicos, eds., *Wavelets, Fractals and Fourier Transforms*. Oxford: Clarendon Press, pp. 151–93.
- Fletcher, R. (1980). *Practical Methods of Optimization*. Chichester, UK: John Wiley.
- Fliege, N. J. (1994). *Multirate Digital Signal Processing*. Chichester, UK: John Wiley.
- Freeny, S. L. (1975). Special-purpose hardware for digital filtering. *Proceedings of the IEEE*, 63, 633–48.
- Freescale Semiconductor (July 2007). *DSP56374 24-Bit Digital Signal Processor — User Guide*. Freescale Semiconductor.
- Furtado, Jr., M. B., Diniz, P. S. R. & Netto, S. L. (2003). Optimized prototype filter based on the FRM approach for cosine-modulated filter banks. *Circuits Systems and Signal Processing*, 22, 193–210.
- Furtado, Jr., M. B., Diniz, P. S. R. & Netto, S. L. (2005a). Numerically efficient optimal design of cosine-modulated filter banks with peak-constrained least-squares behavior. *IEEE Transactions on Circuits and Systems I: Regular Paper*, 52, 597–608.
- Furtado, Jr., M. B., Diniz, P. S. R., Netto, S. L. & Saramäki, T. (2005b). On the design of high-complexity cosine-modulated transmultiplexers based on the frequency-response masking approach. *IEEE Transactions on Circuits and Systems I: Regular Paper*, 52, 2413–26.
- Gabel, R. A. & Roberts, R. A. (1980). *Signals and Linear Systems*, 2nd edn. New York: John Wiley and Sons.
- Gabor, D. (1946). Theory of communication. *Journal of the Institute of Electrical Engineering*, 93, 429–57.
- Gardner, W. A. (1987). Introduction to Einstein's contribution to time-series analysis. *IEEE ASSP Magazine*, 4, 4–5.
- Gersho, A. & Gray, R. M. (1992). *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers.
- Godsill, S. J. & Rayner, J. W. (1998). *Digital Audio Restoration*. London, UK: Springer Verlag.
- Gold, B. & Jordan, Jr., K. L. (1969). A direct search procedure for designing finite impulse response filters. *IEEE Transactions on Audio and Electroacoustics*, 17, 33–6.
- Gray, Jr., A. H. & Markel, J. D. (1973). Digital lattice and ladder filter synthesis. *IEEE Transactions on Audio and Electroacoustics*, AU-21, 491–500.
- Gray, Jr., A. H. & Markel, J. D. (1975). A normalized digital filter structure. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, 268–77.
- Ha, Y. H. & Pearce, J. A. (1989). A new window and comparison to standard windows. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37, 298–301.
- Harmuth, H. F. (1970). *Transmission of Information by Orthogonal Signals*. New York, NY: Springer Verlag.
- Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modeling*. Hoboken, NJ: John Wiley & Sons.
- Haykin, S. (1996). *Adaptive Filter Theory*, 4th edn. Englewood-Cliffs, NJ: Prentice-Hall.

- Herrmann, O. (1971). On the approximation problem in nonrecursive digital filter design. *IEEE Transactions on Circuit Theory, CT-19*, 411–13.
- Hettling, K. J., Saulnier, G. J., Akansu, A. N. & Lin, X. (1999). Transmultiplexers: a unifying time-frequency tool for TDMA, FDMA and CDMA communications. In A. N. Akansu and M. J. Medley, eds., *Wavelet, Subband and Block Transforms in Communications and Multimedia*. Boston: Kluwer Academic Publishers, chapter 1, pp. 1–24.
- Higgins, W. E. & Munson, Jr., D. C. (1984). Optimal and suboptimal error spectrum shaping for cascade-form digital filters. *IEEE Transactions on Circuits and Systems, CAS-31*, 429–37.
- Holford, S. & Agathoklis, P. (1996). The use of model reduction techniques for designing IIR filters with linear phase in the passband. *IEEE Transactions on Signal Processing, 44*, 2396–403.
- Horng, B.-R., Samueli, H. & Willson, Jr., A. N. (1991). The design of low-complexity linear-phase FIR filter banks using powers-of-two coefficients with an application to subband image coding. *IEEE Transactions on Circuits and Systems for Video Technology, 1*, 318–24.
- Hwang, K. (1979). *Computer Arithmetic: Principles, Architecture and Design*. New York, NY: John Wiley.
- Hwang, S. Y. (1977). Minimum uncorrelated unit noise in state space digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-25*, 273–81.
- Ifeachor, E. C. & Jervis, B. W. (1993). *Digital Signal Processing: A Practical Approach*. Reading, MA: Addison-Wesley.
- Itakura, F. & Saito, S. (1971). Digital filtering techniques for speech analysis and synthesis. In *Proceedings of International Congress on Acoustics*, Budapest, Hungary, pp. 261–264.
- Jackson, L. B. (1969). An analysis of roundoff noise in digital filters. PhD thesis, Stevens Institute of Technology, Hoboken, NJ.
- Jackson, L. B. (1970a). On the interaction of roundoff noise and dynamic range in digital filters. *Bell Systems Technical Journal, 49*, 159–85.
- Jackson, L. B. (1970b). Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form. *IEEE Transactions on Audio and Electroacoustics, AU-18*, 107–22.
- Jackson, L. B. (1996). *Digital Filters and Signal Processing*, 3rd edn. Boston, MA: Kluwer Academic Publishers.
- Jackson, L. B., Kaiser, J. F. & McDonald, H. S. (1968). An approach to the implementation of digital filters. *IEEE Transactions on Audio and Electroacoustics, AU-16*, 413–21.
- Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall.
- Jayant, N. S. & Noll, P. (1984). *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall.
- Johnston, J. D. (1980). A filter family designed for use in quadrature mirror filter banks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 291–4.
- Jury, E. I. (1973). *Theory and Application of the Z-Transform Method*. Huntington, NY: R. E. Krieger.

- Kahrs, M. & Brandenburg, K. (1998). *Applications of Digital Signal Processing to Audio and Acoustics*. Boston, MA: Kluwer Academic Publishers.
- Kailath, T., Sayed, A. H. & Hassibi, B. (2000). *Linear Estimation*. Englewood Cliffs, NJ: Prentice-Hall.
- Kaiser, J. F. (1974). Nonrecursive digital filter design using the $I_0 - \sinh$ window function. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 20–3.
- Kalliojärvi, K. & Astola, J. (1996). Roundoff errors in block-floating-point systems. *IEEE Transactions on Signal Processing*, 44, 783–91.
- Kay, S. & Eldar, Y. C. (2008). Rethinking bias estimation. *IEEE Signal Processing Magazine*, 25, 133–6.
- Kay, S. & Smith, D. (1999). An optimal sidelobeless window. *IEEE Transactions on Signal Processing*, 47, 2542–6.
- Kay, S. M. (1988). *Modern Spectral Estimation: Theory and Application*. Englewood Cliffs, NJ: Prentice-Hall.
- Kha, H. H., Tuan, H. D. & Nguyen, T. Q. (2009). Efficient design of cosine-modulated filter bank via convex optimization. *IEEE Transactions on Signal Processing*, 57, 966–76.
- Kim, Y. (1980). State space structures for digital filters. PhD thesis, University of Rhode Island, Kingston.
- Koilpillai, R. D. & Vaidyanathan, P. P. (1992). Cosine-modulated FIR filter banks satisfying perfect reconstruction. *IEEE Transactions on Signal Processing*, 40, 770–83.
- Kolmogorov, A. N. & Fomin, S. V. (1962). *Measure, Lebesgue Integrals, and Hilbert Space*. London: Academic Press.
- Kovačević, J. (1991). Filter banks and wavelets – extensions and applications. PhD thesis, Columbia University.
- Kreider, D. L., Kuller, R. G., Ostberg, D. R. & Perkins, F. W. (1966). *An Introduction to Linear Analysis*. Reading, MA: Addison-Wesley.
- Kreyszig, E. (1979). *Advanced Engineering Mathematics*, 4th edn. New York, NY: Wiley.
- Laakso, T. I. (1992). Comments on ‘Calculation of L_∞ norms in second-order state-space digital filter sections’. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39, 256.
- Laakso, T. I. (1993). Elimination of limit cycles in direct form digital filters using error feedback. *International Journal of Circuit Theory and Applications*, 21, 141–63.
- Laakso, T. I. & Hartimo, I. (1992). Noise reduction in recursive digital filters using high-order error feedback. *IEEE Transactions on Signal Processing*, 40, 141–63.
- Laakso, T. I., Diniz, P. S. R., Hartimo, I. & Macedo, Jr., T. C. (1992). Elimination of zero-input limit cycles in single-quantizer recursive filter structures. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39, 638–45.
- Laakso, T. I., Zeng, B., Hartimo, I. & Neüvo, Y. (1994). Elimination of limit cycles in floating-point implementation of recursive digital filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41, 308–12.
- Lawson, C. L. (1968). Contribution to the theory of linear least maximum approximations. PhD thesis, University of California, Los Angeles, CA.
- Le Gall, D. (1992). The MPEG video compression algorithm. *Image Communication*, 4, 129–40.

- Le Gall, D. & Tabatabai, A. (1988). Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, New York, NY, pp. 761–4.
- Le Presti, L. (2000). Efficient modified-sinc filters for sigma-delta A/D converters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47, 1204–13.
- Lim, Y. C. (1986). Frequency-response masking approach for synthesis of sharp linear phase digital filters. *IEEE Transactions on Circuits and Systems, CAS-33*, 357–64.
- Lim, Y. C. & Lian, Y. (1994). Frequency-response masking approach for digital filter design: complexity reduction via masking filter factorization. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41, 518–25.
- Lim, Y. C., Lee, J. H., Chen, C. K. & Yang, R. H. (1992). A weighted least squares algorithm for quasi-equiripple FIR and IIR digital filter design. *IEEE Transactions Signal Processing*, 40, 551–8.
- Lin, I.-S. & Mitra, S. K. (1996). Overlapped block digital filtering. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43, 586–96.
- Liu, B. & Peled, A. (1975). A new hardware realization of high-speed fast Fourier transformers. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23*, 543–7.
- Liu, K. J. R., Chiu, C. T., Kolagotla, R. K. & JáJá, J. F. (1994). Optimal unified architectures for the real-time computation of time-recursive discrete sinusoidal transforms. *IEEE Transactions on Circuits and Systems for Video Technology*, 4, 168–80.
- Liu, K. S. & Turner, L. E. (1983). Stability, dynamic-range and roundoff noise in a new second-order recursive digital filter. *IEEE Transactions on Circuits and Systems, CAS-30*, 815–21.
- Lu, W.-S. (1999). Design of stable digital filters with equiripple passbands and peak-constrained least-squares stopbands. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46, 1421–6.
- Lu, W.-S., Saramäki, T. & Bregovic, R. (2004). Design of practically perfect-reconstruction cosine-modulated filter banks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51, 552–63.
- Luenberger, D. G. (1984). *Introduction to Linear and Nonlinear Programming*, 2nd edn. Boston, MA: Addison-Wesley.
- Lyon, R. F. (1976). Two's complement pipeline multipliers. *IEEE Transactions on Communications, COM-24*, 418–25.
- Lyons, R. G. (2007). *Streamlining Digital Signal Processing: A Trick of the Trade Guidebook*. Hoboken, NJ: Wiley Interscience.
- Mallat, S. G. (1989a). Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37, 2091–110.
- Mallat, S. G. (1989b). A theory of multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, 674–93.
- Mallat, S. G. (1999). *A Wavelet Tour of Signal Processing*, 2nd edn. San Diego, CA: Academic Press.

- Malvar, H. S. (1986). Fast computation of discrete cosine transform through fast Hartley transform. *Electronics Letters*, 22, 352–3.
- Malvar, H. S. (1987). Fast computation of discrete cosine transform and the discrete Hartley transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35, 1484–5.
- Malvar, H. S. (1992). *Signal Processing with Lapped Transforms*. Norwood, MA: Artech House.
- Manolakis, D. G., Ingle, V. & Kogon, S. (2000). *Statistical and Adaptive Signal Processing: Spectral Estimation, Signal Modeling, Adaptive Filtering and Array Processing*. New York, NY: McGraw-Hill.
- Martinez, H. G. & Parks, T. W. (1979). A class of infinite duration impulse response digital filters for sampling rate reduction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27, 154–62.
- McClellan, J. H. & Parks, T. W. (1973). A unified approach to the design of optimum FIR linear-phase digital filters. *IEEE Transactions on Circuit Theory*, CT-20, 190–6.
- McClellan, J. H. & Rader, C. M. (1979). *Number Theory in Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.
- McClellan, J. H., Parks, T. W. & Rabiner, L. R. (1973). A computer program for designing optimum FIR linear-phase digital filters. *IEEE Transactions on Audio and Electroacoustics*, AU-21, 506–26.
- Meerkötter, K. (1976). Realization of limit cycle-free second-order digital filters. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 295–8.
- Meerkötter, K. & Wegener, W. (1975). A new second-order digital filter without parasitic oscillations. *Archiv Elektrotechnik und Übertragungstechnik*, 29, 312–4.
- Merchant, G. A. & Parks, T. W. (1982). Efficient solution of a Toeplitz-plus-Hankel coefficient matrix system of equations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-30, 40–4.
- Mersereau, R. M. & Dudgeon, D. E. (1984). *Multidimensional Digital Signal Processing*, Prentice-Hall Signal Processing Series. Englewood Cliffs, NJ: Prentice-Hall.
- Mills, W. L., Mullis, C. T. & Roberts, R. A. (1978). Digital filters realizations without overflow oscillations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26, 334–8.
- Mitra, S. K. (2006). *Digital Signal Processing: A Computer Based Approach*, 3rd edn. New York, NY: McGraw-Hill.
- Mou, Z.-J. & Duhamel, P. (1991). Short-length FIR filters and their use in fast nonrecursive filtering. *IEEE Transactions on Signal Processing*, 39, 1322–32.
- Mullis, C. T. & Roberts, R. A. (1976a). Roundoff noise in digital filters: frequency transformations and invariants. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-24, 538–50.
- Mullis, C. T. & Roberts, R. A. (1976b). Synthesis of minimum roundoff noise fixed point digital filters. *IEEE Transactions on Circuits and Systems*, CAS-23, 551–62.
- Munson, D. C., Strickland, J. H. & Walker, T. P. (1984). Maximum amplitude zero-input limit cycles in digital filters. *IEEE Transactions on Circuits and Systems*, CAS-31, 266–75.

- Neüvo, Y., Cheng-Yu, D. & Mitra, S. K. (1984). Interpolated finite impulse response filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-32*, 563–70.
- Neüvo, Y., Rajan, G. & Mitra, S. K. (1987). Design of narrow-band FIR filters with reduced arithmetic complexity. *IEEE Transactions on Circuits and Systems, 34*, 409–19.
- Nguyen, T. Q. & Koilpillai, R. D. (1996). The theory and design of arbitrary-length cosine-modulated FIR filter banks and wavelets, satisfying perfect reconstruction. *IEEE Transactions on Signal Processing, 44*, 473–83.
- Nguyen, T. Q., Laakso, T. I. & Koilpillai, R. D. (1994). Eigenfilter approach for the design of allpass filters approximating a given phase response. *IEEE Transactions on Signal Processing, 42*, 2257–63.
- Nussbaumer, H. J. (1982). *Fast Fourier Transform and Convolution Algorithms*. Berlin, Germany: Springer Verlag.
- Nuttall, A. H. (1981). Some windows with very good sidelobe behavior. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-29*, 84–91.
- Olejniczak, K. J. & Heydt, G. T. (1994). Scanning the special section on the Hartley transform. *Proceedings of the IEEE, 82*, 372–80.
- Oppenheim, A. V. & Schafer, R. W. (1975). *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.
- Oppenheim, A. V. & Schafer, R. W. (1989). *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.
- Oppenheim, A. V., Willsky, A. S. & Young, I. T. (1983). *Signals and Systems*. Englewood Cliffs, NJ: Prentice-Hall.
- Papoulis, A. (1965). *Probability, Random Variables, and Stochastic Processes*. New York, NY: McGraw-Hill.
- Papoulis, A. (1977). *Signal Analysis*. New York, NY: McGraw-Hill.
- Peebles, Jr. P. Z. (2000). *Probability, Random Variables, and Random Signal Principles*, 4th edn. New York, NY: McGraw-Hill.
- Peled, A. & Liu, B. (1974). A new hardware realization of digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-22*, 456–62.
- Peled, A. & Liu, B. (1985). *Digital Signal Processing*. Malabar, FL: R. E. Krieger.
- Pope, S. P. (1985). Automatic generation of signal processing integrated circuits. PhD thesis, University of California, Berkeley, CA.
- Poularikas, A. D. & Seely, S. (1988). *Elements of Signals and Systems*. Boston: PWS-KENT Publishing Company.
- Proakis, J. G. & Manolakis, D. G. (2007). *Digital Signal Processing: Principles, Algorithms and Applications*, 4th edn. Upper Saddle River, NJ: Prentice-Hall.
- Pšenička, B., Garcia-Ugalde, F. & Herrera-Camacho, A. (2002). The bilinear Z transform by Pascal matrix and its application in the design of digital filters. *IEEE Signal Processing Letters, 9*, 368–70.
- Rabiner, L. R. (1979). *Programs for Digital Signal Processing*. New York, NY: IEEE Press.
- Rabiner, L. R. & Gold, B. (1975). *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.

- Rabiner, L. R., Gold, B. & McGonegal, G. A. (1970). An approach to the approximation problem for nonrecursive digital filters. *IEEE Transactions on Audio and Electroacoustics, AU-18*, 83–106.
- Rabiner, L. R., McClellan, J. H. & Parks, T. W. (1975). FIR digital filter design techniques using weighted Chebyshev approximation. *Proceedings of the IEEE, 63*, 595–610.
- Rajan, G., Neuvo, Y. & Mitra, S. K. (1988). On the design of sharp cutoff wide-band FIR filters with reduced arithmetic complexity. *IEEE Transactions on Circuits and Systems, 35*, 1447–54.
- Ralev, K. R. & Bauer, P. H. (1999). Realization of block floating-point digital filters and application to block implementations. *IEEE Transactions on Signal Processing, 47*, 1076–87.
- Regalia, P. A., Mitra, S. K. & Vaidyanathan, P. P. (1988). The digital all-pass filter: a versatile signal processing building block. *Proceedings of the IEEE, 76*, 19–37.
- Reitwiesner, R. W. (1960). *Binary Arithmetics*. New York, NY: Academic Press, pp. 231–308.
- Rice, J. R. & Usow, K. H. (1968). The Lawson algorithm and extensions. *Mathematics of Computation, 22*, 118–27.
- Rioul, O. (1992). Simple regularity criteria for subdivision schemes. *SIAM Journal on Mathematical Analysis, 23*, 1544–76.
- Rioul, O. & Vetterli, M. (1991). Wavelets and signal processing. *IEEE Signal Processing Magazine, 8*, 14–38.
- Roberts, R. A. & Mullis, C. T. (1987). *Digital Signal Processing*. Reading, MA: Addison-Wesley.
- Roitman, M. & Diniz, P. S. R. (1995). Power system simulation based on wave digital filters. *IEEE Transactions on Power Delivery, 11*, 1098–104.
- Roitman, M. & Diniz, P. S. R. (1996). Simulation of non-linear and switching elements for transient analysis based on wave digital filters. *IEEE Transactions on Power Delivery, 12*, 2042–8.
- Saramäki, T. (1993). Finite-impulse response filter design. In S. K. Mitra and J. F. Kaiser, eds., *Handbook for Digital Signal Processing*. New York, NY: Wiley, Chapter 4, pp. 155–277.
- Saramäki, T. & Neuvo, Y. (1984). Digital filters with equiripple magnitude and group delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-32*, 1194–200.
- Saramäki, T., Neuvo, Y. & Mitra, S. K. (1988). Design of computationally efficient interpolated FIR filters. *IEEE Transactions on Circuits and Systems, 35*, 70–88.
- Sarcinelli Filho, M. & Camponêz, M. de O. (1997). A new low roundoff noise second order digital filter section which is free of constant-input limit cycles. In *Proceedings of IEEE Midwest Symposium on Circuits and Systems*, volume 1, Sacramento, CA, pp. 468–71.
- Sarcinelli Filho, M. & Camponêz, M. de O. (1998). Design strategies for constant-input limit cycle-free second-order digital filters. In *Proceedings of IEEE Midwest Symposium on Circuits and Systems*, Notre Dame, IN, pp. 464–467.
- Sarcinelli Filho, M. & Diniz, P. S. R. (1990). Tridiagonal state-space digital filter structures. *IEEE Transactions on Circuits and Systems, CAS-36*, 818–24.
- Sathe, V. P. & Vaidyanathan, P. P. (1993). Effects of multirate systems in the statistical properties of random signals. *IEEE Transactions on Signal Processing, 41*, 131–46.

- Sayood, K. (2005). *Introduction to Data Compression*, 3rd edn. Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann Publishers.
- Sedlmeyer, A. & Fettweis, A. (1973). Digital filters with true ladder configuration. *International Journal of Circuit Theory and Applications*, 1, 5–10.
- Sedra, A. S. & Brackett, P. O. (1978). *Filter Theory and Design: Active and Passive*. Champaign, IL: Matrix Publishers.
- Selesnick, I. W., Lang, M. & Burrus, C. S. (1996). Constrained least square design of FIR filters without specified transition bands. *IEEE Transactions on Signal Processing*, 44, 1879–92.
- Selesnick, I. W., Lang, M. & Burrus, C. S. (1998). A modified algorithm for constrained least square design of multiband FIR filters without specified transition bands. *IEEE Transactions on Signal Processing*, 46, 497–501.
- Singh, V. (1985). Formulation of a criterion for the absence of limit cycles in digital filters designed with one quantizer. *IEEE Transactions on Circuits and Systems*, CAS-32, 1062–4.
- Singleton, R. C. (1969). An algorithm for computing the mixed radix fast Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, AU-17, 93–103.
- Skahill, K. (1996). *VHDL for Programmable Logic*. Reading, MA: Addison-Wesley.
- Smith, L. M., Bomar, B. W., Joseph, R. D. & Yang, G. C.-J. (1992). Floating-point roundoff noise analysis of second-order state-space digital filter structures. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39, 90–8.
- Smith, M. J. T. & Barnwell, T. P. (1986). Exact reconstruction techniques for tree-structured subband coders. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34, 434–41.
- Sorensen, H. V. & Burrus, C. S. (1993). Fast DFT and convolution algorithms. In S. K. Mitra and J. F. Kaiser, eds., *Handbook of Digital Signal Processing*. New York, NY: Wiley, chapter 8, pp. 491–610.
- Stark, H. & Woods, J. W. (2002). *Probability and Random Processes with Applications to Signal Processing*, 3rd edn. Upper Saddle River, NJ: Prentice-Hall.
- Steele, J. M. (2004). *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. MAA Problem Books Series. Cambridge, UK: Cambridge University Press.
- Stoica, P. & Moses, R. L. (1997). *Introduction to Spectral Analysis*. Upper Saddle River, NJ: Prentice-Hall.
- Strang, G. (1980). *Linear Algebra and Its Applications*, 2nd edn. Academic Press.
- Strang, G. & Nguyen, T. Q. (1996). *Wavelets and Filter Banks*. Wellesley, MA: Wellesley Cambridge Press.
- Stüber, G. L. (1996). *Principles of Mobile Communications*. Norwell, MA: Kluwer Academic Publishers.
- Sullivan, J. L. & Adams, J. W. (1998). PCLS IIR digital filters with simultaneous frequency response magnitude and group delay specifications. *IEEE Transactions on Signal Processing*, 46, 2853–62.
- Szczupak, J. & Mitra, S. K. (1975). Digital filter realization using successive multiplier-extraction approach. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, 235–9.

- Taubman, D. S. & Marcellin, M. W. (2001). *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers.
- Texas Instruments (November 2006). *TMS320C67X/C67x+ DSP CPU and Instruction Set – Reference Guide*. Texas Instruments.
- Texas Instruments (October 2008). *TMS320C64X/C64x+ DSP CPU and Instruction Set – Reference Guide*. Texas Instruments.
- Tran, T. D., de Queiroz, R. L. & Nguyen, T. Q. (2000). Linear phase perfect reconstruction filter bank: lattice structure, design, and application in image coding. *IEEE Transactions on Signal Processing*, 48, 133–47.
- Ulbrich, W. (1985). MOS digital filter design. In Y. Tsividis and P. Antognetti, eds., *Design of MOS VLSI Circuits for Telecommunications*. Englewood Cliffs, NJ: Prentice-Hall, chapter 8, pp. 236–71.
- Vaidyanathan, P. P. (1984). On maximally flat linear phase FIR filters. *IEEE Transactions on Circuits and Systems*, CAS-31, 830–2.
- Vaidyanathan, P. P. (1985). Efficient and multiplierless design of FIR filters with very sharp cutoff via maximally flat building blocks. *IEEE Transactions on Circuits and Systems*, CAS-32, 236–44.
- Vaidyanathan, P. P. (1987). Eigenfilters: a new approach to least-squares FIR filter design and applications including Nyquist filters. *IEEE Transactions on Circuits and Systems*, CAS-34, 11–23.
- Vaidyanathan, P. P. (1993). *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall.
- Van Gerwen, P. J., Mecklenbräuker, W. F., Verhoeckx, N. A. M., Snijders, F. A. M. & van Essen, H. A. (1975). A new type of digital filter for data transmission. *IEEE Transactions on Communications*, COM-23, 222–34.
- Van Trees, H. L. (1968). *Detection, Estimation and Modulation Theory*. New York, NY: John Wiley and Sons.
- Van Valkenburg, M. E. (1974). *Network Analysis*, 3rd edn. Englewood Cliffs, NJ: Prentice-Hall.
- Verdu, S. (1998). *Multiuser Detection*. Cambridge, UK: Cambridge University Press.
- Verkroost, G. (1977). A general second-order digital filter with controlled rounding to exclude limit cycles for constant input signals. *IEEE Transactions on Circuits and Systems*, CAS-24, 428–31.
- Verkroost, G. & Butterweck, H. J. (1976). Suppression of parasitic oscillations in wave digital filters and related structures by means of controlled rounding. *Archiv Elektrotechnik und Übertragungstechnik*, 30, 181–6.
- Vetterli, M. (1986). Filter banks allowing perfect reconstruction. *Signal Processing*, 10(3), 219–45.
- Vetterli, M. (1988). Running FIR and IIR filtering using multirate filter banks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36, 730–8.
- Vetterli, M. & Herley, C. (1992). Wavelets and filters banks: theory and design. *IEEE Transactions on Signal Processing*, 40, 2207–32.
- Vetterli, M. & Kovačević, J. (1995). *Wavelets and Subband Coding*. Englewood Cliffs, NJ: Prentice-Hall.

- Vetterli, M. & Le Gall, D. (1989). Perfect reconstruction FIR filter banks: some properties and factorizations. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-37*(7), 1057–71.
- Villemoes, L. F. (1992). Energy moments in time and frequency for two-scale difference equation solutions and wavelets. *SIAM Journal on Mathematical Analysis, 23*, 1519–43.
- Wanhammar, L. (1981). An approach to LSI implementation of wave digital filters. PhD thesis, Linköping University, Linköping, Sweden.
- Wanhammar, L. (1999). *DSP Integrated Circuis*. New York, NY: Academic Press.
- Webster, R. J. (1985). C^∞ -windows. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-33*, 753–60.
- Whitaker, J. (1999). *DTV: The Digital Video Revolution*, 2nd edn. New York, NY: McGraw-Hill.
- Widrow, B. & Stearns, S. D. (1985). *Adaptive Signal Processing*. Englewood-Cliffs, NJ: Prentice-Hall.
- Willems, J. L. (1970). *Stability Theory of Dynamical Systems*. London, UK: Thomas Nelson and Sons.
- Winston, W. L. (1991). *Operations Research – Applications and Algorithms*, 2nd edn. Boston, MA: PWS-Kent.
- Yaglom, A. M. (1987). Einstein's 1914 paper on the theory of irregularly fluctuating series of observations. *IEEE ASSP Magazine, 4*, 7–11.
- Yang, R. H. & Lim, Y. C. (1991). Grid density for design of one- and two-dimensional FIR filters. *IEE Electronics Letters, 27*, 2053–5.
- Yang, R. H. & Lim, Y. C. (1993). Efficient computational procedure for the design of FIR digital filters using WLS techniques. *IEE Proceedings - Part G, 140*, 355–9.
- Yang, R. H. & Lim, Y. C. (1996). A dynamic frequency grid allocation scheme for the efficient design of equiripple FIR filters. *IEEE Transactions on Signal Processing, 44*, 2335–9.
- Yang, S. & Ke, Y. (1992). On the three coefficient window family. *IEEE Transactions on Signal Processing, 40*, 3085–8.
- Yu, Y. J. (2003). Multiplierless multirate FIR filter design and implementation. PhD thesis, National University of Singapore, Singapore.
- Yu, Y. J. & Lim, Y. C. (2002). A novel genetic algorithm for the design of a signed power-of-two coefficient quadrature mirror filter lattice filter bank. *Circuits Systems and Signal Processing, 21*, 263–76.
- Yun, I. D. & Lee, S. U. (1995). On the fixed-point analysis of several fast IDCT algorithms. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 42*, 685–93.

Index

- abs, *see* MATLAB, command, abs
adaptive filters, 442
adder, 223
aliasing effect, 38, 41, 49, 50, 52, 369, 372, 456, 458, 475, 480, 504, 505, 508, 518, 540, 822
anti-, filter, 44, 50, 145, 466
allpass filter
 transfer function, 824
alternation theorem, 322
analog filters
 all-pole, 351, 356
 bandpass, 350
 bandstop, 350
 Butterworth, 351, 356
 computation, 363
 frequency transformation, 360
 in MATLAB, 399
 transfer function, 353
 Chebyshev, 353, 356, 828
 computation, 363
 frequency transformation, 360
 in MATLAB, 400
 transfer function, 356
 design, 361
 elliptic, 357, 828
 computation, 363
 frequency transformation, 360
 in MATLAB, 400
 transfer function, 358
 frequency transformation, 359, 360
 highpass, 350
 inverse Chebyshev, 406
 in MATLAB, 400
 ladder form, 828, 849, 852
 lattice realization, 848
 lowpass, 43, 350, 351
 specification, 350
analog signal processing, 5
analog-to-digital conversion, 46
AR model, *see* modeling theory, autoregressive model
ARMA model, *see* modeling theory, moving average
 autoregressive model
ASIC, *see* implementation, application-specific
 integrated circuit
autocorrelation method, *see* spectral estimation,
 parametric, autocorrelation method
autoregressive model, *see* modeling theory,
 autoregressive model
averaged periodogram, *see* spectral estimation,
 nonparametric, averaged periodogram
Bartlett bisection theorem, 848, 849
Bartlett window, 295, 338
Bessel function, 301
bias, *see* estimation theory, bias
bilinear transformation, *see* digital filters, IIR, bilinear
 transformation method
biorthogonal lapped transform, 573
bit-reversal ordering, 180, 183, 186, 203
Blackman window, *see* window functions, Blackman
Blackman–Tukey, *see* spectral estimation,
 nonparametric, Blackman-Tukey
block transforms, 548
butterfly computation, 181, 190, 192
Butterworth filter, *see* analog filters, Butterworth *and*
 digital filters, IIR, Butterworth approximation
canonic form networks, 223, 234, 235
canonic signed-digit representation, *see* digital
 representations, fixed-point, canonic signed-digit
cascade realization, *see* digital filters, FIR, cascade
 form *and* digital filters, IIR, cascade form
Cauer filters, *see* elliptic filters
causality
 classification, 12, 13, 19, 20
 definition, 12
CDF, *see* random variable, cumulative distribution
 function
Chebyshev filter, *see* analog filters, Chebyshev *and*
 digital filters, IIR, Chebyshev approximation
Chebyshev functions, 353
Chebyshev optimization, *see* digital filters, FIR,
 Chebyshev method
Chebyshev polynomials, 306, 353
Cholesky decomposition, 431
circulant matrix, 481
comb filter, 261
 FIR, 276
 IIR, 262, 276

- commutator models, 471, 509
 complementary filters, 343, 760
 convolution, continuous-time
 time domain, 43
 convolution, discrete-time
 circular, 212
 frequency domain, 293
 matrix form, 158
 time domain, 157, 158, 164, 166, 167, 205, 480
 frequency domain, 123, 147
 in MATLAB, *see* MATLAB, command, conv
 linear, 14, 21, 101, 211
 time domain, 157, 164, 165, 168, 169, 173
 notation, 15
 time domain, 14, 98, 123, 144
 using the discrete Fourier transform, 157
 z domain, 98, 100
 cosine function
 discrete-time, 7
 cosine transforms
 even, 204
 coupled form
 state-space structure, 707
 covariance method, *see* spectral estimation,
 parametric, covariance method
 Cramer–Rao lower bound, *see* estimation theory,
 Cramer–Rao lower bound
 CSD, *see* digital representations, fixed-point, canonic
 signed-digit
- DCT, *see* discrete cosine transform
 decimation, 456, 458, 460, 465–467, 475, 480, 492,
 494, 504, 505, 508, 624
 frequency response, 457
 in MATLAB, *see* MATLAB, command, decimate
 property
 computational complexity, 459
 periodic time invariance, 459
 time variance, 459
 decimation-in-frequency FFT algorithm, 184, 190
 decimation-in-time FFT algorithm, 179, 181, 190
 delay, 223
 DFT, *see* discrete Fourier transform
 DHT, *see* discrete Hartley transform
 difference equations, 6, 62
 annihilator polynomial, 26–28, 30
 auxiliary conditions, 17–20, 24, 29, 31, 32
 characteristic polynomial, 23–25, 27
 delay operator, 25
 Fibonacci equation, 23, 24
 homogeneous solution, 17–19, 22–27, 30
 nonrecursive, 21, 222
 particular solution, 17–19, 25–28
 recursive, 21
 representation, 17, 104
 solution, 17, 18
 steady-state solution, 30
 differential equations, 5, 17
 differentiator, 229, 231, 280, 283, 317, 333–336
 in MATLAB, *see* MATLAB, command, diff
 digital filters
 allpass section, 259, 384
 bandpass, 229, 278, 283, 303, 317
 bandstop, 228, 278, 283, 303, 317
 building blocks
 allpass, 257, 259
 bandpass, 257, 258
 comb filter, 261, 262
 highpass, 257
 highpass notch, 257, 259, 381
 lowpass, 257, 258
 lowpass notch, 257, 259
 oscillator, 260
 chrominance, 274
 complementary, 343, 760
 decimator, 460, 461, 465, 475, 476, 492, 509, 510
 decimators, 518
 delay equalizer, 395
 doubly complementary, 822, 825, 826
 FIR
 alternative direct form, 223
 cascade form, 224
 Chebyshev method, 321, 323, 325, 328, 331, 335,
 341, 461, 476, 546, 559, 754, 755, 766, 778
 coefficients, 278
 constrained weighted least-squares method, 319
 definition, 21
 difference equation, 21, 222
 direct form, 223
 eigenfilter method, 319
 frequency response, 278
 frequency-response masking design, 477,
 760–762, 765, 767
 frequency sampling design, 283, 286, 287,
 290, 310
 frequency-domain form, 750
 interpolation and decimation design, 474,
 475, 497
 interpolation design, 756, 759
 lattice form, 558, 740, 747
 lattice realization, 438
 linear-phase, 395
 maximally flat design, 309, 312
 minimum phase, 546
 modified-sinc filter, 753
 optimization methods, 313, 316
 polyphase form, 469, 507, 749
 prefilter design, 753, 755
 quadrature design, 771, 775
 recursive running sum, 750, 751, 754
 transfer function, 105

- weighted least-squares method, 317, 319, 320, 328, 331, 340
 window method, 291, 292, 297, 302, 304, 309, 313, 337
 WLS–Chebyshev method, 328–332, 561
 half-band, 545
 highpass, 228, 278, 283, 317
 IIR
 analog prototype, 350, 368
 bilinear transformation method, 372–377, 397, 402, 828
 Butterworth approximation, 376, 399
 cascade form, 236, 239, 263, 703, 787, 790, 792, 796, 799, 812, 855
 characteristics, 349
 Chebyshev approximation, 376, 400
 Chebyshev or minimax criterion, 383
 definition, 21
 difference equation, 21
 direct form, 234, 263, 787, 817
 elliptic approximation, 113, 376, 400, 709, 711, 792
 impulse-invariance method, 368, 370–372, 377, 402
 inverse Chebyshev approximation, 400, 406
 lattice realization, 255, 815–820
 numerical optimization methods, 382, 387–390, 403
 parallel form, 237, 239, 240, 264, 703, 787, 788, 792, 799, 812
 prewarping procedure, 374, 376
 second-order section, 687, 788, 791, 792, 796, 798
 state-space description, 244, 245, 253, 265, 704, 705, 719, 733
 state-space section, 689, 728, 800, 801, 803, 806, 807, 810, 812
 time-domain approximation, 391, 394
 transfer function, 105, 232
 variable cutoff-frequency method, 381
 wave filters, *see* wave digital filters
 implementation
 application-specific integrated circuit, *see* implementation, application-specific integrated circuit
 digital signal processors, *see* implementation, digital signal processors
 discrete hardware, *see* implementation, discrete hardware
 distributed arithmetic, *see* implementation, distributed arithmetic
 field programmable gate array, *see* implementation, field programmable gate array
 programmable logic devices, *see* implementation, programmable logic devices
 very large scale integration, *see* implementation, very large scale integration
 interpolator, 464, 465, 475, 476, 491, 509, 510, 756, 757, 759
 half-band filter, 465
 Lth-band filter, 464
 linear-phase, 225, 226
 amplitude response, 285, 286
 forms, 227–229, 280, 283, 285, 286, 314, 315, 317, 323
 phase response, 285, 286
 properties, 231
 realizations, 232
 zeros, 231
 lowpass, 113, 231, 278, 283, 317
 masking, 761, 762, 764, 773
 matrix representation, 241
 PCM, 325
 phase equalizer, 259, 384
 power complementary, 558
 prefilter, 753–755
 signal flowgraph, 223
 digital representations
 block-floating-point, 697
 fixed-point, 673, 719
 canonic signed-digit, 673, 732
 one's-complement, 670–672, 697, 732
 sign-magnitude, 670, 732
 signed power-of-two, 672, 673
 two's-complement, 670–676, 678, 683, 715, 732
 floating-point, 673
 pseudo-floating-point, 711
 digital signal processing, 5, 33, 46
 of continuous-time signals, 44
 digital-to-analog conversion, 46
 practical case, 72
 direct realization
 FIR filters, 223
 IIR filters, 234, 263
 discrete cosine transform, 548, 554, 563, 568
 application, 201, 206, 207
 butterfly, 202
 computational complexity, 202, 203
 definition, 199
 filter bank, 548, 550
 in MATLAB, *see* MATLAB, command, dct
 inverse, 200
 matrix notation, 554, 559, 566, 569, 570, 574
 factorization, 202
 discrete Fourier transform, 143, 194
 computation, 149, 150, 155, 159, 162
 computational complexity, 152, 175
 definition, 146, 148, 175, 210
 fast Fourier transform, *see* fast Fourier transform
 FFT, *see* fast Fourier transform
 frequency resolution, 147

- discrete Fourier transform, (*contd.*)
 implementation
 overlap-and-add, 168, 169, 480
 overlap-and-save, 171, 173
 interpretation, 145, 149, 153
 inverse, 145, 148, 175, 182, 210, 290
 computation, 155
 length, 144, 147, 153
 matrix notation, 151, 156, 183
 factorization, 184
 period, 144
 property
 circular convolution, 157
 complex conjugation, 159
 correlation, 158
 linearity, 153, 162
 modulation, 156
 of real and imaginary sequences, 159
 of symmetric and antisymmetric sequences, 160
 Parseval's theorem, 163
 real sequences, 159
 time-reversal, 153
 time-shift, 153, 156
 recursive computation, 220
 relation with
 Fourier transform, 147
 z transform, 163
 resolution, 147
 unitary, 196
 zero-padding, 147, 165, 168, 173, 750
- discrete Hartley transform
 application, 206
 definition, 205
 inverse, 205
 property, 205
- discrete transforms
 cosine transforms, 204
 discrete cosine transform, 199
 discrete Hartley transform, 205
 Hadamard transform, 206
 Karhunen–Loëve transform, 207
 unitary, 196, 199, 531
 Walsh–Hadamard transform, 206
 wavelet transform, 207
- Dolph–Chebyshev window, *see* window functions,
 Dolph–Chebyshev
- double sideband modulation, 777
- doubly complementary, 822
- downsampling, *see* decimation
- DSB, *see* double sideband modulation
- DSP, *see* implementation, digital signal processors
- elliptic filters, *see* analog filters, elliptic and digital filters, IIR, elliptic approximation
- entropy, 73, 437
- entropy, differential, 74
- equiripple approximation for FIR filters, *see* digital filters, FIR, Chebyshev method
- error spectrum shaping, 795
- ESS, *see* error spectrum shaping
- estimation theory, 410
 Bayesian estimation, 410
 bias, 410, 411, 413, 414
 classic estimation, 410
 consistency, 410, 413
 Cramer–Rao lower bound, 411
 efficiency, 411
 mean squared error, 410
 standard deviation, 410
 variance, 410, 411, 414
- even part of a sequence, 70
- exponential function
 discrete-time, complex, 75
 discrete-time, real, 8
- fast Fourier transform, 143, 175, 212, 213, 413
 basic cell, 181
 bit-reverse ordering, 180
 butterfly, 181, 186
 computational complexity, 175, 176, 178, 181, 190, 193
 decimation-in-frequency, 184, 190
 decimation-in-time, 179, 181, 190, 697
 FIR filtering, 750
 in MATLAB, *see* MATLAB, commands, `fft`, `ifft`, `fftshift`
 inverse, 182
 matrix notation
 factorization, 184
 overlap-and-add, 750
 overlap-and-save, 750
 quantization effects, 697
 radix-2, 190, 697
 algorithm, 176, 177
 butterfly, 179
 radix-3
 butterfly, 190, 192
 radix-4, 187, 190
 butterfly, 188
 radix- N , 192
 Winograd, 193
- FFT, *see* fast Fourier transform
- filter banks, 503
 wavelet transform relationship, 617, 628
- allpass decomposition, 822
- analysis
 M -band, 517
 analysis filters, 507, 742
 biorthogonality, 518, 530, 531
 block transforms, 548, 551
 conjugate quadrature, 543, 544, 547, 548, 581, 585, 633, 638
 design, 544–547

- example, 587
- cosine-modulated, 543, 554, 559, 561, 563
 - design, 559
 - implementation, 558
- CQF, *see* filter banks, conjugate quadrature
- critically decimated, 508, 599
 - M*-band, 506
 - using lattice form, 742
- Daubechies 4, 657
- doubly complementary, 822
- extended lapped transform, 563
- Haar filter bank, 513, 638
- hierarchical decomposition, 599
- Johnston filter bank, 542
- lapped orthogonal transform, 563, 566–570,
 - 588, 591
 - fast algorithm, 568, 570, 573
 - generalized, 563, 576, 577
 - implementation, 567
 - polyphase decomposition, 567
- lapped transform
 - biorthogonal, 573, 574
 - biorthogonal, generalized, 576
- maximally decimated
 - M*-band, 542
- modulation matrix representation, 518, 520
 - 2-band, 540
- near-perfect reconstruction, 540
- noncritically decimated, 508
- octave-band, 601, 605
- orthogonality, 518, 531, 533, 543, 544, 591, 620, 628, 633
 - z* domain, 533, 534
- paraunitary, 534, 544, 548, 563, 566
- perfect reconstruction, 467, 507, 509–511, 516–518, 520, 527, 529, 531, 534, 536, 537, 541, 543–545, 561, 563, 566, 574, 628, 640, 742, 746
 - analysis filters, 513, 514, 536
 - example, 513, 515, 516, 546
 - M*-band, 506, 509, 548, 556
 - M*-band, cascade, 534
 - practical case, 507
 - quadrature mirror, 542
 - synthesis filters, 513, 514, 536
 - transmultiplexer, 535
 - two-band, 506, 535, 638
 - two-band, design procedure, 537, 538
 - using lattice form, 742, 744
- polyphase decomposition, 513, 514, 517, 535, 536, 541, 554, 556, 566, 567, 573, 742, 749
 - M*-band, 507
- polyphase matrix, 743, 747
- power complementary, 822
- pseudo-quadrature mirror, 542
- QMF, *see* filter banks, quadrature mirror
 - quadrature mirror, 540–542
 - design, 542
 - implementation, 826
 - in MATLAB, *see* MATLAB, command, `qmf`
 - perfect reconstruction, 542
 - two-band, 540
 - symmetric short kernel, 539, 638
 - synthesis filters, 507, 742
 - time-domain analysis, 521
- finite impulsive-response filters, *see* digital filters, FIR
- FIR filters, *see* digital filters, FIR
- floating-point representation, *see* digital representations, floating-point
- Fortran, 325
- Fourier series, 36
 - coefficients, 34, 210
 - convergence, 292
 - definition, 34, 210
 - discrete-time, 125
 - periodic signal, 149
 - truncation, 294
- Fourier transform, 36, 46, 194
 - continuous-time
 - definition, 34, 116, 209
 - frequency sampling, 147
 - inverse, 34, 43, 116, 209
 - definition, 607
 - discrete-time, 115
 - computation, 117–119
 - definition, 76, 117, 143, 209
 - existence, 118
 - inverse, 117, 145, 209, 288
 - of impulse response, 43
 - periodic signal, 123
 - relation with the discrete Fourier transform, 147
 - of periodic sequence, 125
 - property
 - complex conjugation, 121
 - complex differentiation, 120
 - convolution, 34
 - frequency-domain convolution, 123
 - linearity, 120
 - modulation, 120, 777
 - of real and imaginary sequences, 121
 - of symmetric and antisymmetric sequences, 122, 123
 - Parseval's theorem, 123
 - time-domain convolution, 123
 - time-reversal, 120
 - time-shift, 120, 124
 - sampling of, 144
- PGA, *see* implementation, field programmable gate array
- frequency response
 - computation, 110, 111
 - computation from zeros and poles, 114

- frequency response (*contd.*)
 definition, 109
 generalized Hamming window, 296
 in MATLAB, *see* MATLAB, command, `freqz`
 Kaiser window, 301
 of decimated signal, 457, 458
 of ideal bandpass filter, 280
 of ideal bandstop filter, 279
 of ideal highpass filter, 280
 of ideal lowpass filter, 278
 of interpolated signal, 462
 rectangular window, 295
 frequency sampling, *see* digital filters, FIR, frequency sampling design
 frequency transformation
 continuous-time domain, 359, 360
 discrete-time domain, 378
 lowpass-to-bandpass, 380
 lowpass-to-bandstop, 381
 lowpass-to-highpass, 380, 381
 lowpass-to-lowpass, 379
 in MATLAB, *see* MATLAB, commands, `lp2lp`,
 `1p2hp`, `1p2bp`, `1p2bs`
- Gibbs' oscillations, 292, 319
 gradient vector, 387
 grid of poles, 707
 group delay, 109, 110, 225
 cascade IIR filters, 385
 computation, 110
 in MATLAB, *see* MATLAB, command, `grpdelay`
- Haar filter bank, 513, 638
 Haar wavelet, *see* wavelet transform, Haar
 Hadamard transform
 application, 206
 definition, 206
 fast algorithm, 207
 half-band filter, *see* digital filters, interpolator, half-band filter
 Hamming window, *see* window functions, Hamming
 Hann window, *see* window functions, Hann
 Hanning window, *see* window functions, Hann
 Hartley transform, *see* discrete Hartley transform
 Hessian matrix, 387, 388, 428
 Hilbert transformer, 229, 231, 281, 283, 317, 319,
 321, 336, 546, 581, 582
 impulse response, 283
 in MATLAB, *see* MATLAB, command, `hilbert`
 Hölder inequality, 701, 702
 Hölder regularity, *see* wavelet transform, regularity
 hyperbolic-sine function, 300
- IDFT, *see* discrete Fourier transform, inverse
 IIR filters, *see* digital filters, IIR
 implementation
 application-specific integrated circuit, 668
- digital signal processors, 669
 discrete hardware
 parallel adder, 684
 parallel multiplier, 684
 serial adder, 674
 serial multiplier, 676, 678, 679, 681, 683
 serial subtractor, 675
 distributed arithmetic, 685–687, 689, 728
 fixed-point, 693, 703, 709, 711
 floating-point, 696
 pipelining, 678, 681
 very large scale integration, 668
 implementation, field programmable gate array, 668
 implementation, programmable logic devices, 668
 impulse function, 7
 delayed, 7, 8, 14
 train of, 34, 35, 46, 47, 51, 52
 Fourier series of, 36
 Fourier transform of, 36
 impulse response, 14, 15, 21, 22, 31–33, 43, 44, 46,
 52, 53, 62, 84, 93, 126, 129, 211
 analysis filter, 521
 antisymmetric, 228, 229, 286
 finite-duration, 21
 linear phase, 226
 from state-space description, 244
 in MATLAB, *see* MATLAB, command, `impz`
 of differentiator, 334
 of ideal bandpass filter, 280
 of ideal bandstop filter, 279
 of ideal differentiator, 280
 of ideal highpass filter, 280
 of ideal Hilbert transformer, 283
 of ideal lowpass filter, 278
 symmetric, 227, 228, 285
 impulse-invariance method, *see* digital filters, IIR, impulse-invariance method
 infinite impulse-invariance filters, *see* digital filters, IIR
 interpolation, 462, 465–468, 475, 491, 505, 761
 formulas, 44, 148, 324
 frequency response, 462
 in MATLAB, *see* MATLAB, command, `interp`
 Lagrange, barycentric-form, 324
 property
 computational complexity, 464
 time variance, 464
 interreciprocity, 249, 250
- Kaiser window, *see* window functions, Kaiser
 Karhunen–Loëve transform, 207
 Kuhn–Tucker conditions, 561
- Lth-band filter, *see* digital filters, interpolator, Lth-band filter
 Lagrange interpolator, 324

-
- Lagrange multiplier, 417, 561
 Laplace transform, 75, 208
 lapped orthogonal transform, *see* filter banks, lapped orthogonal transform
 lattice realization, *see* digital filters, FIR, lattice form and digital filters, IIR, lattice form
 Laurent series, 77
 Lawson algorithm, 328
 least-squares filter design, *see* digital filters, FIR, weighted least-squares method
 Levinson–Durbin algorithm, 431, 432, 434, 435, 437, 438, 449
 Lim–Lee–Chen–Yang algorithm, 328
 limit cycles, *see* quantization errors, limit cycles
 linear prediction, *see* spectral estimation, parametric, linear prediction
 linearity
 classification, 12, 13, 20
 definition, 11
 of the discrete Fourier transform, 153
 of the Fourier transform, 120
 z transform, 94
 Lipschitz continuity, 633
 LU decomposition, 431
 Lyapunov function, 720, 725

 MA model, *see* modeling theory, moving average model
 magnitude response
 cascade IIR filters, 385
 computation, 110
 computation from zeros and poles, 114
 definition, 76, 109
 magnitude truncation, *see* quantization errors, arithmetic quantization, magnitude truncation
 MATLAB
 command
 abs, 130, 135
 angle, 130, 135
 appcoef, 654, 663
 appcoef2, 664
 aryule, 449
 bartlett, 338
 bi2de, 735
 bilinear, 402
 bin2num, 735
 binvec2dec, 735
 biorfilt, 661
 biorwavf, 661
 blackman, 298, 339
 boxcar, 338
 burg, 450
 buttap, 400
 butter, 395, 399
 butterord, 395
 buttord, 399
 ceil, 735
 cfirpm, 341
 cheb1ap, 400
 cheb1ord, 400
 cheb2ap, 400
 cheb2ord, 400
 chebwin, 339
 cheby, 395
 cheby1, 400
 cheby2, 400
 chebyord, 395
 coifwavf, 660
 conv, 67, 211, 212
 cov, 450
 cplxpair, 264, 268
 cremez, 341
 dbwavf, 660
 dct, 216
 dctmtx, 216
 de2bi, 735
 dec2bin, 735
 dec2binvec, 735
 decimate, 495, 496
 detcoef, 654, 664
 detcoef2, 664
 dfmtx, 215
 diff, 336
 dwt, 661
 dwt2, 664
 dwtmode, 662
 dyaddown, 594
 dyadup, 594
 ellip, 395, 400, 792
 ellipap, 401
 ellipord, 395, 401, 792
 fft, 215
 fftfilt, 215
 fftshift, 215
 filter, 211, 269
 fir1, 305, 337
 fir2, 338
 fircls, 340
 fircls1, 340
 fircls2, 340
 firpm, 325, 326, 331, 335, 336, 341, 778
 firpmord, 341, 778
 fix, 735
 floor, 735
 fminsearch, 591
 fminunc, 591
 freqspace, 136
 freqz, 130, 132, 135, 793
 grpdelay, 130, 136, 395
 hamming, 298, 338
 hanning, 298, 339
 hilbert, 337

- MATLAB (*contd.*)
- idct, 216
 - idwt, 662
 - idwt2, 664
 - ifft, 215
 - impinvar, 402
 - impz, 67
 - interp, 495, 496
 - intfilt, 497
 - invfreqs, 403
 - invfreqz, 403
 - kaiser, 339
 - kaiserord, 339
 - latc2tf, 781, 857
 - latcfilt, 781
 - levinson, 449
 - lp2bp, 401
 - lp2bs, 402
 - lp2hp, 401
 - lp2lp, 401
 - lpc, 403, 450
 - maxflat, 403
 - num2bin, 735
 - orthfilt, 661
 - periodogram, 450
 - poly, 267
 - prony, 403
 - psd, 446, 450
 - psdopts, 450
 - qmf, 594
 - quant, 735
 - rand, 129
 - randn, 65, 129
 - real, 212
 - remez, 341
 - remezord, 341
 - resample, 496
 - residue, 129, 264
 - residuez, 267, 792, 857
 - roots, 131, 267
 - round, 735
 - sos2ss, 269, 857
 - sos2tf, 267, 857
 - sos2zp, 269
 - spectrum, 446, 450
 - sptool, 341, 403
 - ss2sos, 269, 857
 - ss2tf, 268, 857
 - ss2zp, 269
 - stem, 67
 - stmcb, 403
 - symwavf, 660
 - tf2latc, 781, 857
 - tf2sos, 794
 - tf2ss, 268, 857
 - tf2zp, 131, 264, 266
 - tf2zpk, 585
 - triang, 338
 - unwrap, 136
 - upcoef, 664
 - upcoef2, 664
 - upfirdn, 495
 - uplev, 663
 - uplev2, 664
 - wavedec, 662
 - wavedec2, 664
 - wavedemo, 664
 - wavefun, 664
 - waveinfo, 660
 - wavemenu, 664
 - waverec, 663
 - waverec2, 664
 - welch, 450
 - wfilters, 660
 - wrcoef, 664
 - wrcoef2, 664
 - yulear, 450
 - yulewalk, 403
 - zp2sos, 264, 268
 - zp2ss, 269
 - zp2tf, 131, 132, 266
 - zplane, 137, 585
- toolbox
- Fixedpoint, 735
 - Optimization, 591
 - Signal Processing, 67, 135, 215, 266, 341, 495
 - Spectral Estimation, 449
 - Wavelet, 594, 656, 659, 660, 664
- maximally flat design, *see* digital filters, FIR, maximally flat design
- MIMO, *see* multiple-input, multiple-output system
- minimax approach, *see* digital filters, IIR, Chebyshev approximation
- minimum-phase stable ARMA model, 421
- minimum-variance, *see* spectral estimation, nonparametric, minimum-variance
- modeling theory, 419
- autoregressive model, 419–422, 424, 426, 437, 442, 589, 591
 - moving average autoregressive model, 419–422, 424, 426, 442
 - moving average model, 419, 420, 423, 426, 442
 - Wiener filter, *see* Wiener filter
 - Yule–Walker equations, 423, 426, 429, 432
 - autoregressive model, 424
 - moving average autoregressive model, 424
 - moving average model, 424
- modified-sinc filter, *see* digital filters, FIR, modified-sinc filter
- Moiré effect, 72
- moving average autoregressive model, *see* modeling theory, moving average autoregressive model

- moving average model, *see* modeling theory, moving average model
 MPEG2 standard, 548
 multiple-input, multiple-output system, 480, 486
 multiplier, 223

 Newton method, 388
 noble identities, 467, 469, 601
 non-Toeplitz matrix, 431
 number theoretic transform, 194
 Nyquist frequency, 37, 64, 369

 odd part of a sequence, 70
 one's-complement representation, *see* digital representations, fixed-point, one's-complement
 order estimate, 347
 overflow, 693, 697, 698, 700, 717, 718, 729
 overlapped block filtering, 479, 480, 483, 484, 487

 parallel multiplier, 684
 parallel realization, *see* digital filters, IIR, parallel form
 paraunitary matrix, 534, 544, 547, 548, 566
 Parks–McClellan algorithm, *see* digital filters, FIR, Chebyshev method
 Parseval's theorem, 195, 200, 610
 discrete Fourier transform, 163
 discrete transforms, 195
 Fourier transform, 123, 608, 612
 z transform, 100
 partial-fraction expansion, 88–90, 129, 238, 267
 Pascal matrix, 398
 PDF, *see* random variable, probability density function
 period, 9, 144
 periodic signals, 9
 periodogram, 411, *see* spectral estimation, nonparametric, periodogram
 phase response, 109
 computation, 110
 computation from zeros and poles, 114
 definition, 76, 109
 digital filters, linear, 225, 395
 FIR filters, linear
 conditions, 226
 pipelining, 678, 681
 PLD, *see* implementation, programmable logic devices
 poles, *see* transfer function, poles
 polynomial division, 90, 91
 polyphase decomposition, *see* digital filters, FIR, polyphase form
 prewarping procedure, *see* digital filters, IIR, prewarping procedure
 PSD, *see* random process, power spectral density
 pseudo-circulant matrix, 481, 483, 485, 486, 488, 489, 491

 quantization errors
 analog-to-digital conversion, 669
 arithmetic quantization, 669, 691, 694, 698, 789–791, 805, 810–812
 controlled rounding, 727
 magnitude truncation, 691, 692, 725, 727, 806
 overflow, 719
 rounding, 691, 698, 719
 saturation, 806
 truncation, 691, 719
 coefficient quantization, 669, 706, 708, 709, 711, 713, 788, 792, 812, 820
 instability, 729
 limit cycles
 constant-input, 725–728, 806, 807, 812
 granular, 715, 720
 overflow, 715, 717, 718
 zero-input, 719–721, 723, 726, 728
 signal scaling, 697, 700, 702–704, 717, 788, 790, 792, 806, 810, 818
 saturation, 717
 quasi-Newton methods, 388
 Broyden–Fletcher–Goldfarb–Shannon algorithm, 389, 390
 Davidson–Fletcher–Powell algorithm, 388

 ramp function, 8
 random process, 58, 62, 125, 703
 autocorrelation function, 58–60, 62, 125, 411, 413, 426, 490, 589
 autocorrelation matrix, 60, 428, 431, 440, 491, 492
 cross-correlation function, 59
 ensemble, 58
 ergodicity, 60
 joint wide-sense stationarity, 59, 429, 441, 589
 power spectral density, 125–128, 411, 589, 693
 realization, 58
 sample, 58
 strict-sense stationarity, 59
 white noise, 128, 129, 419, 424
 wide-sense cyclostationarity, 490–493
 wide-sense stationarity, 58–62, 125–127, 411, 419, 430, 491–493
 Wiener–Khinchin theorem, 126, 128
 random processes
 autocorrelation function, 62
 random signal, 53, 58, 60, 126
 random variable, 54, 58, 62
 cross-correlation, 57
 cross-correlation, complex, 57
 cross-covariance, 57
 cumulative distribution function, 55
 properties, 55
 energy, 56
 independence, 57
 joint cumulative distribution function, 57

- random variable, (*contd.*)
 joint probability density function, 57, 58, 491
 mean-squared value, 56
 probability density function, 55, 56, 57, 62
 properties, 55
 continuous uniform distribution, 55, 59
 discrete uniform distribution, 55, 56
 Gaussian distribution, 55, 65
 standard deviation, 56
 statistical mean, 56, 60–62
 variance, 56, 60, 62
 reciprocity, 248
 rectangular window, *see* window functions,
 rectangular
 recursive filters, *see* digital filters, IIR
 relative noise variance, 694
 relative power spectral density, 694
 relative power spectrum density, 797
 Remez exchange algorithm, *see* digital filters, FIR,
 Chebyshev method
 residue, 84, 86, 87
 definition, 83
 in MATLAB, *see* MATLAB, command, residuez
 residue theorem, 83, 84, 86, 87, 705, 798
 rounding, *see* quantization errors, arithmetic
 quantization, rounding
 RRS, *see* digital filters, FIR, recursive
 running sum
- sampling frequency, 37, 38, 41, 42, 49, 63, 64,
 145, 602
 change, 455, 456, 465, 493
 in MATLAB, *see* MATLAB, command, resample
 sampling rate, *see* sampling frequency
 sampling theorem, 38, 42, 64, 369
 saturation arithmetic, 717, 718, 732
 scaling function, 605
 Schwartz inequality, 610, 701
 sensitivity, 250, *see* transfer function, sensitivity
 computation, 253, 255, 256
 serial adder, 674
 serial multiplier, 676
 serial subtractor, 675
 short-time Fourier transform, 608, 609, 612
 short-time spectrum, 220
 sign-magnitude representation, *see* digital
 representations, fixed-point, sign-magnitude
 signal flowgraph, 223, 241, 246
 branch, 241
 delayless loop, 243, 828, 830, 832
 interreciprocity, 249, 250
 loop, 247
 matrix representation, 241
 node, 241
- node ordering, 243
 reciprocity, 248
 source node, 241
 topology, 247
 transposition, 249, 250, 255
 signal scaling, 735
 signals
 random, *see* random signal
 continuous-time, 5, 33, 34, 44
 band-limited, 38, 456
 discrete-time, 5, 6, 8, 33, 34
 antisymmetric sequences, 121, 160
 causal, 81
 complex sine, 75, 109
 complex sinusoid, 333
 conjugate antisymmetric sequences, 121, 160
 conjugate symmetric sequences, 121, 160
 cosine, 7, 9
 critically decimated, 504
 decimated, 457, 505
 delayed impulse, 7, 8, 14
 even part, 70
 finite-length sequences, 80
 graphical representation, 7
 imaginary sequences, 97, 159
 impulse, 7
 interpolated, 462
 left-handed, one-sided sequences, 79, 82, 90, 91
 odd part, 70
 periodic, 9, 34, 148
 ramp, 8
 real exponential, 8
 real sequences, 97, 159
 right-handed, one-sided sequences, 79, 82, 90, 92
 step, 7, 8
 symmetric sequences, 121, 160
 two-sided sequences, 79, 82
- signed power-of-two representation, *see* digital
 representations, fixed-point, signed power-of-two
 sine function
 discrete-time, complex, 75, 109
 sine transforms
 even, 204
 single sideband modulation, 780
 single-input, single-output system, 480, 486, 489
 SISO, *see* single-input, single-output system
 spectral estimation, 409
 nonparametric, 411
 averaged periodogram, 413, 414, 443
 Blackman-Tukey, 414, 415, 445, 447
 minimum-variance, 416, 418, 445
 periodogram, 411–414, 416, 426, 443
 windowed periodogram, 413, 414, 445
 parametric, 426, 446
 autocorrelation method, 431, 432, 437, 446, 447
 autoregressive model, 426, 429, 431, 432, 434

- Burg's method, 436, 437, 446, 447
 covariance method, 430, 431, 437, 442
 Itakura–Saito method, 442
 linear prediction, 426, 428, 429, 434, 439
 MUSIC algorithm, 442
 Prony's method, 442
 subspace methods, 442
 SPT, *see* digital representations, fixed-point, signed power-of-two
 SSB, *see* single sideband modulation
 SSS, *see* random process, strict-sense stationarity
 stability
 classification, 369
 with overflow nonlinearity, 729
 stability, BIBO
 classification, 16, 93, 106–109
 definition, 16
 property, 81, 93
 standard deviation, *see* estimation theory, standard deviation
 state-space description
 computation, 245, 253
 definition, 244, 245
 step function, 7, 8
 subsampling, *see* decimation
 system decomposition theorem, 420
 systems
 continuous-time, 6
 sample-and-hold, 44, 46
 discrete-time, 5, 10
 causality, 12, 13, 19, 20, 31, 44
 FIR, 21
 IIR, 21
 linearity, 11–14, 18, 20, 60, 486, 488
 nonrecursive, 21
 recursive, 21, 22
 representation, 17
 shift invariance, 11
 stability, 16, 30
 time invariance, 11–14, 16, 20, 60, 486, 488
- Taylor series, 92, 638
 Tellegen's theorem, 247, 250, 251
 time invariance
 classification, 12, 13, 20
 definition, 11
 time–frequency analysis, 607, 612, 613
 Toeplitz matrix, 431
 Toeplitz-plus-Hankel matrix, 330
 transfer function
 allpass filter, 824
 allpass section, 259, 384
 Butterworth analog filter, 353
 Chebyshev analog filter, 356
 computation, 112
 definition, 105
 delay equalizer, 395
 elliptic analog filter, 358
 FIR filters, 105
 from state-space description, 244, 801
 IIR filters, 105, 232
 phase equalizer, 259, 384
 poles, 81, 84, 105, 106, 114, 245
 rational, 81
 sensitivity, 250, 253, 255, 256, 706, 708, 709, 711, 713
 state space, 254
 zeros, 81, 105, 114, 231, 790
 transmultiplexers, 534
 transposition, 249, 250, 255
 triangular window, *see* window functions, triangular
 truncation, *see* quantization errors, arithmetic quantization, truncation
 two's-complement representation, *see* digital representations, fixed-point, two's-complement
 two-scale difference equations, 660, 661
 uncertainty principle, 610
 unitary transforms, *see* discrete transforms, unitary
 upsampling, *see* interpolation
 variance, *see* estimation theory, variance
 VLSI, *see* implementation, very large scale integration
 Walsh–Hadamard transform
 application, 206
 definition, 206
 fast algorithm, 207
 warping effect, 374
 wave digital filters, 828, 830
 lattice form, 848
 lattice realization, 848, 849, 851, 852
 n-port elements
 parallel adaptor, 839
 series adaptor, 844
 one-port elements
 capacitor, 832
 inductor, 833
 open-circuit, 833
 resistor, 833
 series connection, 833
 short-circuit, 833
 voltage source, 833
 two-port elements
 current generalized immittance converter, 836
 gyrator, 836
 parallel adaptor, 838, 839
 transformer, 836
 voltage generalized immittance converter, 833

- wavelet transform, 207, 599, 604, 612, 638
 9–7 wavelet, 654
 application, 641, 646, 653
 biorthogonality, 617, 620, 622, 623, 632–634,
 639, 654
 computation of, 607
 continuous-time, 612
 Daubechies, 638, 640
 Daubechies 4, 657
 definition, 604
 discrete-time, 614–616, 619
 filter bank relationship, 617, 628, 635
 Haar, 638
 image processing, 641, 643, 646
 in MATLAB, *see* MATLAB, command,
 `wfilters`
 inverse, 605
 mother function, 599, 604, 605, 612, 619
 multiresolution representation, 617, 619, 620,
 622, 635
 of discrete-time signal, 605
 orthogonality, 616, 619, 620, 622, 628, 632,
 634, 639
 regularity, 633–636
 estimation, 635
 examples, 636
 scaling function, 605–607, 620, 633, 635, 636,
 638, 643
 time-frequency analysis, 607, 612
 two-dimensional, 641
 two-scale difference equations, 660, 661
 vanishing moments, 636, 637
 weighted least-squares method, *see* digital filters, FIR,
 weighted least-squares method
 white noise, 128, 129, 413
 Wiener filter, 439, 440
 applications, 441
 computation, 440, 441
 Wiener–Hopf equation, 428–430, 432, 434, 449
 Wiener–Khinchin theorem, 126, 411, 412, 426
 window functions, 608
 Bartlett, 338, 412
 definition, 295
 Blackman, 335, 339
 computation, 299
 definition, 297
 properties, 297
 Dolph–Chebyshev, 339
 definition, 307
 properties, 309
 Gaussian, 612
 generalized Hamming
 definition, 296
 frequency response, 296
 properties, 296
 Hamming, 338, 415, 431
 computation, 299
 definition, 296
 Hann, 339
 computation, 299
 definition, 296
 Kaiser, 339
 computation, 304
 definition, 301
 Fourier transform, 301
 frequency response, 301
 order estimation, 339
 rectangular, 319, 334, 338, 412, 416
 computation, 299
 definition, 294
 frequency response, 295
 triangular, 338
 definition, 295
 windowed periodogram, *see* spectral estimation,
 nonparametric, windowed periodogram
 Winograd Fourier transform, 193
 WLS method, *see* digital filters, FIR, weighted
 least-squares method
 Wold decomposition, 419
 WSCS, *see* random process, wide-sense
 cyclostationarity
 WSS, *see* random process, wide-sense
 cyclostationarity
 Yule–Walker equations, *see* modeling theory,
 Yule–Walker equations
 z transform
 computation, 77, 80, 81
 definition, 76, 208
 inverse, 116, 208
 computation, 84, 86–93
 definition, 83
 Laurent series, 77
 of finite-length sequences, 163
 one-sided, 76
 property
 complex conjugation, 96
 complex differentiation, 95
 initial value theorem, 97
 linearity, 94
 modulation, 95
 of real and imaginary sequences, 97
 time-domain convolution, 98, 105
 time-reversal, 94, 101
 time-shift, 95, 104, 130
 z-domain convolution, 98
 region of convergence, 77, 85, 94–100, 106
 computation, 78, 80, 81
 of finite-length sequences, 80

- of left-handed, one-sided sequences, 79, 82
 - of rational transfer functions, 82
 - of right-handed, one-sided sequences, 79, 82
 - of stable causal systems, 81
 - of two-sided sequences, 79, 82
 - relation with the discrete Fourier transform, 164
- table of, 101
 - two-sided, 76
- zero-padding, *see* discrete Fourier transform,
zero-padding
- zeros, *see* transfer function, zeros
- Zolotarev filters, *see* elliptic filters