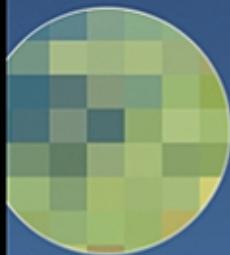


S. ALLEN BROUGHTON • KURT BRYAN

DISCRETE FOURIER ANALYSIS AND WAVELETS

Applications to Signal and Image Processing



DISCRETE FOURIER ANALYSIS AND WAVELETS

This page intentionally left blank

DISCRETE FOURIER ANALYSIS AND WAVELETS

**Applications to Signal and
Image Processing**

**S. ALLEN BROUGHTON
KURT BRYAN**

Rose-Hulman Institute of Technology
Department of Mathematics
Terre Haute, IN



WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2009 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400, fax 978-750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, 201-748-6011, fax 201-748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at 877-762-2974, outside the United States at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic format. For more information about Wiley products, visit our web site at www.wiley.com

Library of Congress Cataloging-in-Publication Data:

Broughton, S. Allen, 1951-

Discrete fourier analysis and wavelets : applications to signal and image processing /

S. Allen Broughton, Kurt Bryan.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-470-29466-6 (cloth)

1. Fourier analysis. 2. Wavelets (Mathematics) 3. Signal processing--Mathematics.

4. Image processing--Mathematics. I. Bryan, Kurt, 1962-. II. Title.

QA403.5.B77 2008

515'.2433--dc22

2008011012

Printed in the United States of America

10 9 8 7 6 5 4 3 2

To our families

This page intentionally left blank

CONTENTS

PREFACE	xi
ACKNOWLEDGMENTS	xv
1 VECTOR SPACES, SIGNALS, AND IMAGES	1
1.1 Overview / 1	
1.2 Some Common Image Processing Problems / 1	
1.3 Signals and Images / 3	
1.4 Vector Space Models for Signals and Images / 9	
1.5 Basic Waveforms—The Analog Case / 17	
1.6 Sampling and Aliasing / 21	
1.7 Basic Waveforms—The Discrete Case / 26	
1.8 Inner Product Spaces and Orthogonality / 29	
1.9 Signal and Image Digitization / 41	
1.10 Infinite-dimensional Inner Product Spaces / 46	
1.11 Matlab Project / 56	
Exercises / 61	
2 THE DISCRETE FOURIER TRANSFORM	71
2.1 Overview / 71	
2.2 The Time Domain and Frequency Domain / 72	
2.3 A Motivational Example / 73	

2.4	The One-dimensional DFT / 78	
2.5	Properties of the DFT / 85	
2.6	The Fast Fourier Transform / 90	
2.7	The Two-dimensional DFT / 93	
2.8	Matlab Project / 97	
	Exercises / 101	
3	THE DISCRETE COSINE TRANSFORM	105
3.1	Motivation for the DCT—Compression / 105	
3.2	Other Compression Issues / 106	
3.3	Initial Examples—Thresholding / 107	
3.4	The Discrete Cosine Transform / 113	
3.5	Properties of the DCT / 117	
3.6	The Two-dimensional DCT / 120	
3.7	Block Transforms / 122	
3.8	JPEG Compression / 124	
3.9	Matlab Project / 132	
	Exercises / 134	
4	CONVOLUTION AND FILTERING	138
4.1	Overview / 138	
4.2	One-dimensional Convolution / 138	
4.3	Convolution Theorem and Filtering / 145	
4.4	2D Convolution—Filtering Images / 150	
4.5	Infinite and Bi-infinite Signal Models / 156	
4.6	Matlab Project / 171	
	Exercises / 174	
5	WINDOWING AND LOCALIZATION	182
5.1	Overview: Nonlocality of the DFT / 182	
5.2	Localization via Windowing / 184	
5.3	Matlab Project / 195	
	Exercises / 197	
6	FILTER BANKS	201
6.1	Overview / 201	
6.2	The Haar Filter Bank / 202	
6.3	The General One-stage Two-channel Filter Bank / 210	

6.4 Multistage Filter Banks / 214	
6.5 Filter Banks for Finite Length Signals / 218	
6.6 The 2D Discrete Wavelet Transform and JPEG 2000 / 231	
6.7 Filter Design / 239	
6.8 Matlab Project / 251	
6.9 Alternate Matlab Project / 255	
Exercises / 258	
7 WAVELETS	267
7.1 Overview / 267	
7.2 The Haar Basis / 269	
7.3 Haar Wavelets versus the Haar Filter Bank / 282	
7.4 Orthogonal Wavelets / 292	
7.5 Biorthogonal Wavelets / 314	
7.6 Matlab Project / 318	
Exercises / 321	
REFERENCES	327
SOLUTIONS	329
INDEX	335

This page intentionally left blank

PREFACE

PHILOSOPHY OF THE TEXT

In this text we provide an introduction to the mathematics that underlies much of modern signal and image processing. In particular, we

- develop the mathematical framework in which signal and image processing takes place, specifically, vector and inner product spaces;
- develop traditional Fourier-based transform techniques, primarily in the discrete case, but also to some extent in the continuous setting;
- provide entry-level material on filtering, convolution, filter banks, and wavelets; and
- make extensive use of computer-based explorations for concept development.

These topics are extremely beautiful and important areas of classical and modern applied mathematics. They also form the foundation for many techniques in science and engineering. However, while they are usually taught (sometimes in bits and pieces) in the engineering and science curricula, they are often overlooked in mathematics courses, or addressed only as brief asides. We hope to change this.

Throughout the text we often use image compression as a concrete and motivational hook for the mathematics, but our goal here is not a technical manual on how to program a JPEG encoder. Rather, we include this topic so that the reader can begin applying the theory to interesting problems relatively quickly. We also touch upon a number of other important applications in addition to compression, for example, progressive transmission of images, image denoising, spectrographic analysis, and

edge detection. We do not discuss the very large topic of image restoration, although the mathematics we develop underlies much of that subject as well. Because audio signals are one-dimensional, they are somewhat simpler to deal with than images, so we often use audio data to illustrate the mathematics of interest before considering images.

This text is suitable for a one-semester course for students who are familiar with calculus and elementary matrix algebra, and possess the “mathematical maturity” typical of a second- or third-year undergraduate. We do not assume any background in image or signal processing. Neither do we assume the reader knows Matlab®, although some experience with computer programming, however modest, is helpful. This text might also serve as a useful supplement for more traditional image processing texts such as *Digital Image Processing Using MATLAB* by Gonzales, Woods, and Eddins and *Digital Image Processing* by Pratt, which are written at a graduate level, have a rather terse mathematical development, and do not make use of tools such as Matlab for concept development or exercises.

Infinite-dimensional vector and inner product spaces form a natural mathematical framework for signal and image processing. Although we do develop a fair amount of general theory in this abstract framework, we focus a bit more on the discrete and finite-dimensional cases. The use of function spaces and the associated analysis is, especially at the beginning, an unnecessary complication for readers learning the basics. Moreover the infinite-dimensional and functional settings do not lend themselves as well to computer-based exercises and experimentation. However, we return to a more general functional framework and some associated analysis in the last chapter, in which we discuss wavelets.

OUTLINE OF THE TEXT

In Chapters 1 through 3 we develop the vector space and matrix framework for analyzing signals and images, as well as the discrete Fourier transform and the discrete cosine transform. We also develop some general theory concerning inner product spaces and orthogonal bases, and consider Fourier series. A principal motivating application throughout these chapters is to develop an understanding of the traditional JPEG compression algorithm. Some easily developed ideas such as noise removal in signals and images are also included. In Chapters 4 and 5 we consider convolution, filtering, and windowing techniques for signals and images. In particular, in Chapter 5, we develop some traditional windowing techniques for “localizing” Fourier-like transforms. In Chapter 6, we develop the theory of filter banks, as a means of understanding the multiscale localized analysis that underlies the JPEG 2000 compression standard. In Chapter 7, we build on Chapter 6 to give a brief account of how filter banks give rise to wavelets.

If time is an issue Chapter 5 can be skipped, since none of the material there is essential for the following chapters. Section 6.7 is also not essential for Chapter 7, but is very useful in helping the reader see the connection between filter banks and wavelets.

Computer Use, Supporting Routines, and Web Page

It is an easy and accurate guess that many concrete problems in signal and image processing are very computationally intensive. As a result both the examples in the text and the exercises make frequent use of Matlab; a computer algebra system like Maple™ or Mathematica® may also be useful. Various Matlab routines used in the text are available on the Web page [<http://www.rose-hulman.edu/mathDFT/>].

Exercises and Projects

At the end of each chapter there is a Matlab project and a number of exercises. The exercises are generally organized by theme. Some of the exercises are simple “by hand” computations, others are fairly large or theoretical, others are Matlab programming or discovery exercises designed to refine intuition about the material or to illustrate a mathematical concept. Many exercises contain hints, and we provide additional hints and answers to selected exercises at the end of the book. The projects are essentially guided explorations that make use of Matlab. Many are suitable for use in a “class laboratory” setting, in which students work alone or in groups.

Most of the exercises and projects make use of standard Matlab commands, but in a few places we make use of commands from various Matlab toolboxes. In Chapter 3, for example, we make use of the “dct” command from Matlab’s Signal Processing Toolbox™. For those readers who don’t have access to this toolbox, we have available on the Web page noted above a substitute “kdct.” It’s not as efficient as Matlab’s dedicated DCT command, but it works. The Matlab project in Chapter 6 makes heavy use of the Wavelet Toolbox™, but again, if the reader doesn’t have this toolbox, we have alternate routines and even an alternate Matlab project.

**S. A. BROUGHTON
K. M. BRYAN**

*Terre Haute, Indiana
December, 2007*

This page intentionally left blank

ACKNOWLEDGMENTS

We would like to acknowledge Rose-Hulman Professors Ed Doering and Roger Lautzenheiser for their suggested improvements to the original set of notes upon which this text is based. We also thank the many Rose-Hulman students who gave us feedback concerning these notes and exercises.

S.A.B., K.M.B

This page intentionally left blank

CHAPTER 1

VECTOR SPACES, SIGNALS, AND IMAGES

1.1 OVERVIEW

In this chapter we introduce the mathematical framework of vector spaces, matrices, and inner products. We motivate the mathematics by using it to model signals and images, both outside the computer (the *analog* signal as it exists in the “real world”) and inside the computer (the *digitized* signal, suitable for computer storage and processing). In either case the signal or image may be viewed as an element of a vector space, so we define and develop some essential concepts concerning these spaces. In particular, to analyze signals and images, we will decompose them into linear combinations of basic sinusoidal or complex exponential waveforms. This is the essence of the discrete Fourier and cosine transforms.

The process of sampling the analog signal and converting it to digital form causes an essential loss of information, called “aliasing” and “quantization error.” We examine these errors rather closely. Analysis of these errors motivates the development of methods to quantify the distortion introduced by an image compression technique, which leads naturally to the concept of the “energy” of a signal and a deeper analysis of inner products and orthogonality.

1.2 SOME COMMON IMAGE PROCESSING PROBLEMS

To open this chapter, we discuss a few common challenges in image processing, to try to give the reader some perspective on the subject and why mathematics is such

an essential tool. In particular, we take a very short look at the following:

- Image compression
- Image restoration and denoising
- Edge and defect detection

We also briefly discuss the “transform” paradigm that forms the basis of so much signal and image processing, and indeed, much of mathematics.

1.2.1 Applications

Compression Digitized images are everywhere. The Internet provides obvious examples, but we also work with digitized images when we take pictures, scan documents into computers, send faxes, photocopy documents, and read books on CD. Digitized images underlie video games, and soon television will go (almost) entirely digital. In each case the memory requirements for storing digitized images are an important issue. For example, in a digital camera we want to pack as many pictures as we can onto the memory card, and we’ve all spent too much time waiting for Web pages to load large images. Minimizing the memory requirements for digitized images is thus important, and this task is what motivates much of the mathematics in this text.

Without going into too much detail, let’s calculate the memory requirement for a typical photograph taken with a digital camera. Assume that we have 24-bit color, so that one byte of memory is required for each of the red, green, and blue components of each pixel. With a 2048×1536 pixel image there will be $2048 \times 1536 \times 3 = 9,431,040$ bytes or 9 megabytes of memory required, if no compression is used. On a camera with a 64-megabyte memory card we can store seven large, gorgeous pictures. This is unacceptably few. We need to do something more sophisticated, to reduce memory requirements by a substantial factor.

However, the compression algorithm we devise cannot sacrifice significant image quality. Even casual users of digital cameras frequently enlarge and print portions of their photographs, so any degradation of the original image will rapidly become apparent. Besides, more than aesthetics may be at stake: medical images (e.g., X rays) may be compressed and stored digitally, and any corruption of the images could have disastrous consequences. The FBI has also digitized and compressed its database of fingerprints, where similar considerations apply; see [7].

At this point the reader might find it motivational to do Exercise 1.1.

Restoration Images can be of poor quality for a variety of reasons: low-quality image capture (e.g., security video cameras), blurring when the picture is taken, physical damage to an actual photo or negative, or noise contamination during the image capture process. Restoration seeks to return the image to its original quality or even “better.” Some of this technology is embedded into image capture devices such as scanners. A very interesting and mathematically sophisticated area of research involves *inpainting*, in which one tries to recover missing portions of an image, perhaps because a film negative was scratched, or a photograph written on.

Edge Detection Sometimes the features of essential interest in an image are the edges, areas of sharp transition that indicate the end of one object and the start of another. Situations such as this may arise in industrial processing, for automatic detection of defects, or in automated vision and robotic manipulation.

1.2.2 Transform-Based Methods

The use of transforms is ubiquitous in mathematics. The general idea is to take a problem posed in one setting, transform to a new domain where the problem is more easily solved, then inverse transform the solution back to the original setting. For example, if you've taken a course in differential equations you may have encountered the Laplace transform, which turns linear differential equations into algebra problems that are more easily solved.

Many imaging processing procedures begin with some type of transform T that is applied to the original image. The transform T takes the image data from its original format in the “image domain” to an altered format in the “frequency domain.” Operations like compression, denoising, or other restoration are sometimes more easily performed in this frequency domain. The modified frequency domain version of the image can then be converted back to the original format in the image domain by applying the inverse of T .

The transform operator T is almost always linear, and for finite-sized signals and images such linear operators are implemented with matrix algebra. The matrix approach thus constitutes a good portion of the mathematical development of the text. Other processes, such as quantization (discussed later), are nonlinear. These are usually the lossy parts of the computation; that is, they cause irreversible but (we hope!) acceptable loss of data.

1.3 SIGNALS AND IMAGES

Before beginning a general discussion of vector spaces, it will be helpful to look at a few specific examples that provide physical realizations of the mathematical objects of interest. We'll begin with one-dimensional signals, then move on to two-dimensional images.

1.3.1 Signals

A signal may be modeled as a real-valued function of a real independent variable t , which is usually time. More specifically, consider a physical process that is dependent on time. Suppose that at each time t within some interval $a \leq t \leq b$ we perform a measurement on some aspect of this process, and this measurement yields a real number that may assume any value in a given range. In this case our measurements are naturally represented by a real-valued function $x(t)$ with domain $a \leq t \leq b$. We will refer to $x(t)$ as an *analog signal*. The function $x(t)$ might represent the intensity of sound at a given location (an audio signal), the current through a wire, the speed of an object, and so on.

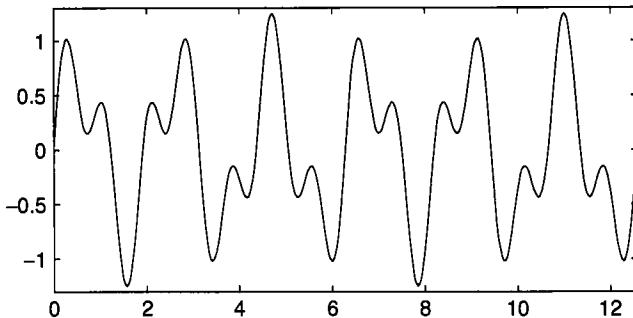


FIGURE 1.1 Analog or continuous model $x(t) = 0.75 \sin(3t) + 0.5 \sin(7t)$.

For example, a signal might be given by the function

$$x(t) = 0.75 \sin(3t) + 0.5 \sin(7t)$$

over the range $0 \leq t \leq 4\pi$. The graph of this function is shown in Figure 1.1. This signal is somewhat unrealistic, however, for it is a linear combination or superposition of a small number of simple sinusoidal functions with no noise. In general, in signal processing we can depend on being vexed by a few persistent annoyances:

- We almost never have an explicit formula for $x(t)$.
- Most signals are very complex.
- Most signals have noise.

Despite the difficulty of writing out an analog description in any specific instance, many physical processes are naturally modeled by analog signals. Analog models also have the advantage of being amenable to analysis using methods from calculus and differential equations. However, most modern signal processing takes place in computers where the computations can be done quickly and flexibly. Unfortunately, analog signals generally cannot be stored in a computer in any meaningful way.

1.3.2 Sampling, Quantization Error, and Noise

To store a signal in a computer we must first *digitize* the signal. The first step in digitization consists of measuring the signal's instantaneous value at specific times over a finite interval of interest. This process is called *sampling*. For the moment let us assume that these measurements can be carried out with “infinite precision.” The process of sampling the signal converts it from an analog form to a finite list of real numbers, and is usually carried out by a piece of hardware known as an *analog-to-digital* (“A-to-D”) converter.

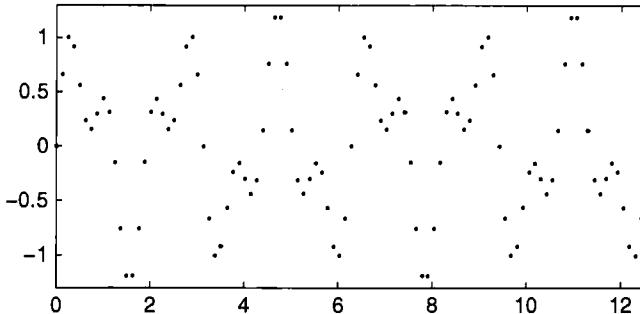


FIGURE 1.2 Discrete or sampled model, $x(t) = 0.75 \sin(3t) + 0.5 \sin(7t)$.

More explicitly, suppose that the signal $x(t)$ is defined on the time interval $a \leq t \leq b$. Choose an integer $N \geq 1$ and define the *sampling interval* $\Delta t = (b - a)/N$. We then measure $x(t)$ at times $t = a, a + \Delta t, a + 2\Delta t, \dots$, to obtain samples

$$x_n = x(a + n\Delta t), \quad n = 0, 1, \dots, N.$$

Define

$$\mathbf{x} = (x_0, x_1, \dots, x_N) \in \mathbb{R}^{N+1}.$$

With the given indexing $x_0 = x(a)$ and $x_N = x(b)$. The vector \mathbf{x} is the sampled version of the signal $x(t)$. The quantity $1/\Delta t$ is the number of samples taken during each time period, so it is called the *sampling rate*.

In Figure 1.2 we have a graphical representation of the sampled signal from Figure 1.1. It should be intuitively clear that sampling causes a loss of information. That is, if we know only the sampled signal, then we have no idea what the underlying analog signal did between the samples. The nature of this information loss can be more carefully quantified, and this gives rise to the concept of *aliasing*, which we examine later.

The sampling of the signal in the independent variable t isn't the only source of error in our A-to-D conversion. In reality, we cannot measure the analog signal's value at any given time with infinite precision, for the computer has only a finite amount of memory. Consider, for example, an analog voltage signal that ranges from 0 to 1 volt. An A-to-D converter might divide up this one volt range into $2^8 = 256$ equally sized intervals, say with the k th interval given by $k\Delta x \leq x < (k + 1)\Delta x$ where $\Delta x = 1/256$ and $0 \leq k \leq 255$. If a measurement of the analog signal at an instant in time falls within the k th interval, then the A-to-D converter might simply record the voltage at this time as $k\Delta x$. This is the *quantization* step, in which a continuously varying quantity is truncated or rounded to the nearest of a finite set of values. An A-to-D converter as above would be said to be "8-bit," because each analog measurement is converted into an 8-bit quantity. The error so introduced is called the *quantization error*.

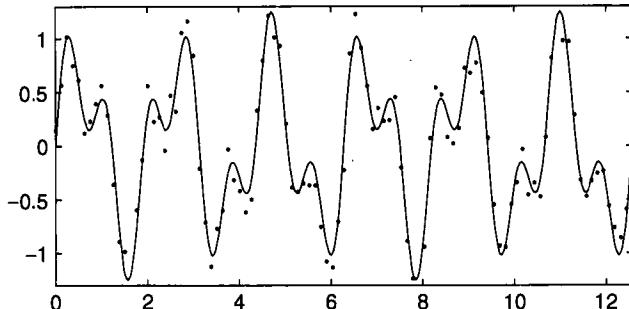


FIGURE 1.3 Analog model and discrete model with noise, $x(t) = 0.75 \sin(3t) + 0.5 \sin(7t)$.

Unfortunately, quantization is a nonlinear process that corrupts the algebraic structure afforded by the vector space model; see Exercise 1.6. In addition quantization introduces irreversible, though usually acceptable, loss of information. This issue is explored further in Section 1.9.

The combination of sampling and quantization allows us to *digitize* a signal or image, and thereby convert it into a form suitable for computer storage and processing.

One last source of error is random noise in the sampled signal. If the noiseless samples are given by x_n as above, the noisy sample values y_n might be modeled as

$$y_n = x_n + \epsilon_n, \quad (1.1)$$

where ϵ_n represents the noise in the n th measurement. The errors ϵ_n are usually assumed to be distributed according to some probability distribution, known or unknown. The noise model in equation (1.1) is additive; that is, the noise is merely added onto the sampled signal. Other models are possible and appropriate in some situations.

In Figure 1.3 we show a discrete signal with noise added. The analog signal and the corrupted discrete signal are graphed together so that the errors introduced by noise may be easily seen.

1.3.3 Grayscale Images

For simplicity we first consider monochrome or *grayscale* images. An analog grayscale image is modeled as a real-valued function $f(x, y)$ defined on a two-dimensional region Ω . Usually Ω is a rectangle, defined in xy coordinates by $a \leq x \leq b$, $c \leq y \leq d$. The value $f(x, y)$ represents the “intensity” of the image at the point (x, y) in Ω . Grayscale images are typically displayed visually so that smaller values of f correspond to darker shades of gray (down to black) and higher values to lighter shades (up to white).

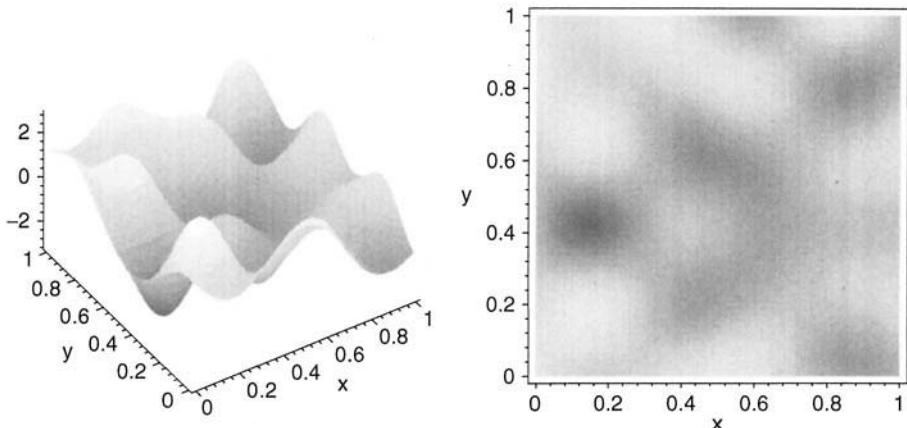


FIGURE 1.4 Grayscale image from two perspectives.

For natural images $f(x, y)$ would never be a simple function. Nonetheless, to illustrate let us consider the image defined by the function

$$\begin{aligned} f(x, y) = & 1.5 \cos(2x) \cos(7y) + 0.75 \cos(5x) \sin(3x) \\ & -1.3 \sin(9x) \cos(15y) + 1.1 \sin(13x) \sin(11y) \end{aligned}$$

on the domain $\Omega = \{(x, y); 0 \leq x \leq 1, 0 \leq y \leq 1\}$. The situation is illustrated in Figure 1.4. The plot on the left is a conventional $z = f(x, y)$ plot with the surface “gray-coded” according to height, where $f = -5$ corresponds to black and $f = 5$ to white. The plot on the right is the same surface but viewed from directly above, looking down the z axis. This is the actual grayscale image encoded by f according to the scheme above.

1.3.4 Sampling Images

As in the one-dimensional case an analog image must be sampled prior to storage or processing in a computer. The simplest model to adopt is the discrete model obtained by sampling the intensity function $f(x, y)$ on a regular grid of points (x_s, y_r) in the plane. For each point (x_s, y_r) the value of $f(x_s, y_r)$ is the “graylevel” or intensity at that location. The values $f(x_s, y_r)$ are collected into a $m \times n$ matrix \mathbf{A} with entries a_{rs} given by

$$a_{rs} = f(x_s, y_r). \quad (1.2)$$

If you’re wondering why it’s $f(x_s, y_r)$ instead of $f(x_r, y_s)$, see Remark 1.1 below.

The sampling points (x_s, y_r) can be chosen in many ways. One approach is as follows: subdivide the rectangle Ω into mn identical subrectangles, with m equal

vertical (y) subdivisions and n equal horizontal (x) subdivisions, of length $\Delta x = (b - a)/n$ and height $\Delta y = (d - c)/m$. We may take the points (x_s, y_r) as the centers of these subrectangles so that

$$(x_s, y_r) = \left(a + \left(s - \frac{1}{2}\right)\Delta x, d - \left(r - \frac{1}{2}\right)\Delta y\right), \quad (1.3)$$

or alternatively as the upper left corners of these rectangles,

$$(x_s, y_r) = (a + s\Delta x, d - r\Delta y), \quad (1.4)$$

where in either case $1 \leq r \leq m$, $1 \leq s \leq n$.

Note that in either case $x_1 < x_2 < \dots < x_n$ and $y_1 > y_2 > \dots > y_m$. Using the centers seems more natural, although the method (1.4) is a bit cleaner mathematically. For images of reasonable size, however, it will make little difference.

Remark 1.1 On a computer screen the pixel coordinates conventionally start in the upper left-hand corner and increase as we move to the right or down, which is precisely how the rows and columns of matrices are indexed. This yields a natural correspondence between the pixels on the screen and the indexes for the matrix \mathbf{A} for either (1.3) or (1.4). However, this is different from the way that coordinates are usually assigned to the plane: with the matrix \mathbf{A} as defined by either (1.3) or (1.4), increasing column index (the index s in a_{rs}) corresponds to the increasing x direction, but increasing row index (the index r in a_{rs}) corresponds to the *decreasing* y direction. Indeed, on those rare occasions when we actually try to identify any (x, y) point with a given pixel or matrix index, we'll take the orientation of the y axis to be reversed, with increasing y as downward.

Remark 1.2 There are other ways to model the sampling of an analog image. For example, we may take a_{rs} as some kind of integral or weighted average of f near the point (x_s, y_r) . These approaches can more accurately model the physical process of sampling an analog image, but the function evaluation model in equation (1.2) has reasonable accuracy and is a simple conceptual model. For almost all of our work in this text we will assume that the sampling has been done and the input image matrix or signal is already in hand.

Remark 1.3 The values of m and n are often decided by the application in mind, or perhaps storage restrictions. It is useful and commonplace to have both m and n to be divisible by some high power of 2.

1.3.5 Color

There are a variety of approaches to modeling color images. One of the simplest is the “RGB” (red, green, blue) model in which a color image is described using three functions $r(x, y)$, $g(x, y)$, and $b(x, y)$, appropriately scaled, that correspond to the intensities of these three additive primary colors at the point (x, y) in the image

domain. For example, if the color components are each scaled in the range 0 to 1, then $r = g = b = 1$ (equal amounts of all colors, full intensity) at a given point in the image would correspond to pure white, while $r = g = b = 0$ is black. The choice $r = 1, g = b = 0$ would be pure red, and so on. See [19] for a general discussion of the theory of color perception and other models of color such as HSI (hue, saturation, intensity) and CMY (cyan, magenta, yellow) models.

For simplicity's sake we are only going to consider grayscale and RGB models, given that computer screens are based on RGB. In fact we will be working almost exclusively with grayscale images in order to keep the discussion simple and focused on the mathematical essentials. An example where the CMY model needs to be considered is in color laser printers that use cyan, magenta, and yellow toner. The printer software automatically makes the translation from RGB to CMY. It is worth noting that the actual JPEG compression standard specifies color images with a slightly different scheme, the luminance-chrominance or "YCbCr" scheme. Images can easily be converted back and forth from this scheme to RGB.

When we consider RGB images, we will assume the sampling has already been done at points (x_s, y_r) as described above for grayscale images. In the sampled image at a given pixel location on the display device the three colors are mixed according to the intensities $r(x_s, y_r)$, $g(x_s, y_r)$, and $b(x_s, y_r)$ to produce the desired color. Thus a sampled m by n pixel image consists of three m by n arrays, one array for each color component.

1.3.6 Quantization and Noise for Images

Just as for one-dimensional signals, quantization error is introduced when an image is digitized. In general, we will structure our grayscale images so that each pixel is assigned an integer value from 0 to 255 (2^8 values) and displayed with 0 as black, 255 as white, and intermediate values as shades of gray. The range is thus quantized with 8-bit precision. Similarly each color component in an RGB image will be assigned value in the 0 to 255 range, so each pixel needs three bytes to determine its color. Some applications require more than 8-bit quantization. For example, medical images are often 12-bit grayscale, offering 4096 shades of gray.

Like one-dimensional signals, images may have noise. For example, let \mathbf{T} be the matrix with entries t_{sr} representing the image on the left in Figure 1.5 and let \mathbf{A} be the matrix with entries a_{sr} representing the noisy image, shown on the right. Analogous to audio signals we can posit an additive noise model

$$a_{sr} = t_{sr} + \epsilon_{sr}, \quad (1.5)$$

where \mathbf{E} has entries ϵ_{sr} and represents the noise. The visual effect is to give the image a kind of "grainy" appearance.

1.4 VECTOR SPACE MODELS FOR SIGNALS AND IMAGES

We now develop a natural mathematical framework for signal and image analysis. At the core of this framework lies the concept of a *vector space*.

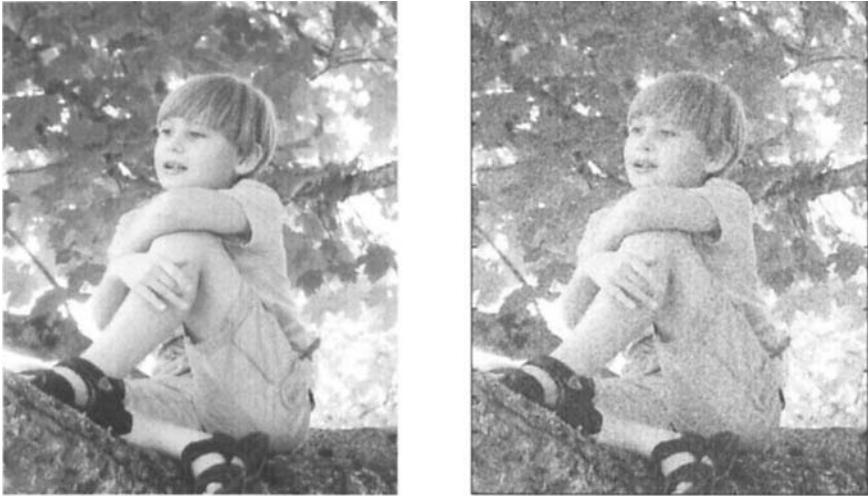


FIGURE 1.5 Image without and with additive noise.

Definition 1.4.1 A vector space over the real numbers \mathbb{R} is a set V with two operations, vector addition and scalar multiplication, with the properties that

1. for all vectors $\mathbf{u}, \mathbf{v} \in V$ the vector sum $\mathbf{u} + \mathbf{v}$ is defined and lies in V (closure under addition);
2. for all $\mathbf{u} \in V$ and scalars $a \in \mathbb{R}$ the scalar multiple $a\mathbf{u}$ is defined and lies in V (closure under scalar multiplication);
3. the “familiar” rules of arithmetic apply, specifically, for all scalars a, b and $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$:
 - a. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$, (addition is commutative),
 - b. $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$ (addition is associative),
 - c. there is a “zero vector” $\mathbf{0}$ such that $\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u}$ (additive identity),
 - d. for each $\mathbf{u} \in V$ there is an additive inverse vector \mathbf{w} such that $\mathbf{u} + \mathbf{w} = \mathbf{0}$, we conventionally write $-\mathbf{u}$ for the additive inverse of \mathbf{u} ,
 - e. $(ab)\mathbf{u} = a(b\mathbf{u})$,
 - f. $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$,
 - g. $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$,
 - h. $1\mathbf{u} = \mathbf{u}$.

If we replace \mathbb{R} above by the field of complex numbers \mathbb{C} , then we obtain the definition of a vector space over the complex numbers.

We'll also make frequent use of subspaces:

Definition 1.4.2 A nonempty subset W of a vector space V is called a “subspace” of V if W is itself closed under addition and scalar multiplication (as defined for V).

Let’s look at a few examples of vector spaces and subspaces, especially those useful in signal and image processing.

1.4.1 Examples—Discrete Spaces

We’ll first consider examples appropriate for sampled signals or images.

■ EXAMPLE 1.1

The vector space \mathbb{R}^N consists of vectors \mathbf{x} of the form

$$\mathbf{x} = (x_1, x_2, \dots, x_N), \quad (1.6)$$

where the x_k are all real numbers. Vector addition and scalar multiplication are defined component by component as

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2, \dots, x_N + y_N), \quad c\mathbf{x} = (cx_1, cx_2, \dots, cx_N),$$

where $\mathbf{y} = (y_1, y_2, \dots, y_N)$ and $c \in \mathbb{R}$. The space \mathbb{R}^N is appropriate when we work with sampled audio or other one-dimensional signals. If we allow the x_k in (1.6) and scalar c to be complex numbers, then we obtain the vector space \mathbb{C}^N . That \mathbb{R}^N or \mathbb{C}^N satisfy the properties of a vector space (with addition and scalar multiplication as defined) follows easily, with zero vector $\mathbf{0} = (0, 0, \dots, 0)$ and additive inverse $(-x_1, -x_2, \dots, -x_n)$ for any vector \mathbf{x} .

As we’ll see, use of the space \mathbb{C}^N can simplify much analysis, even when the signals we work with are real-valued.

Remark 1.4 *Warning:* In later work we will almost always find it convenient to index the components of vectors in \mathbb{R}^N or \mathbb{C}^N starting with index 0, that is, as $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$, rather than the more traditional range 1 to N .

■ EXAMPLE 1.2

The sets $M_{m,n}(\mathbb{R})$ or $M_{m,n}(\mathbb{C})$, $m \times n$ matrices with real or complex entries respectively, form vector spaces. Addition is defined in the usual way for matrices, entry by entry, as is multiplication by scalars. The vector $\mathbf{0}$ is just the matrix with all zero entries, and the additive inverse for a matrix \mathbf{M} with entries m_{jk} is the matrix with entries $-m_{jk}$. Any multiplicative properties of matrices are irrelevant in this context. On closer examination it should be clear that these vector spaces are nothing more than \mathbb{R}^{mn} or \mathbb{C}^{mn} , but spaces where we choose to display the “vectors” as m rows of n components rather than a single row or column with mn entries.

The vector space $M_{m,n}(\mathbb{R})$ is an appropriate model for the discretization of images on a rectangle. As in the one-dimensional case, analysis of images is often facilitated by viewing them as members of space $M_{m,n}(\mathbb{C})$.

■ EXAMPLE 1.3

On occasion it is useful to think of an analog signal $f(t)$ as beginning at some time $t = a$ and continuing “indefinitely.” If we sample such a signal at intervals of Δt starting at time $t = a$ without stopping, we obtain a vector

$$\mathbf{x} = (x_0, x_1, x_2, \dots) \quad (1.7)$$

with real components $x_k = f(a + k\Delta t)$, $k \geq 0$. Given another vector $\mathbf{y} = (y_0, y_1, y_2, \dots)$, we define vector addition and scalar multiplication as

$$c\mathbf{x} = (cx_0, cx_1, cx_2, \dots), \quad \mathbf{x} + \mathbf{y} = (x_0 + y_0, x_1 + y_1, \dots).$$

Let V denote the resulting set with these operations. It’s an easy algebra problem to verify that V is a vector space over the real numbers with zero vector $\mathbf{0} = (0, 0, 0, \dots)$; the additive inverse of \mathbf{x} above is $(-x_0, -x_1, -x_2, \dots)$. And though it may seem painfully obvious, to say that “ $\mathbf{x} = \mathbf{y}$ ” in V means precisely that $x_k = y_k$ for each $k \geq 0$. We will later encounter vector spaces where we have to be quite careful about what is meant by “ $\mathbf{x} = \mathbf{y}$.”

A simple variant of this vector space is the bi-infinite space of vectors

$$\mathbf{x} = (\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots) \quad (1.8)$$

with the analogous vector space structure. A space like this would be appropriate for modeling a physical process with a past and future of indefinite duration.

■ EXAMPLE 1.4

As defined, the set V in the previous example lacks sufficient structure for the kinds of analysis we usually want to do, so we typically impose additional conditions on the components of \mathbf{x} . For example, let us impose the additional condition that for each \mathbf{x} as defined by equation (1.7) there is some number M (which may depend on \mathbf{x}) such that $|x_k| \leq M$ for all $k \geq 0$. In this case the resulting set (with addition and scalar multiplication as defined above for V) is a vector space called $L^\infty(\mathbb{N})$ (here $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of natural numbers), or often just ℓ^∞ . This would be an appropriate space for analyzing the class of sampled signals in which the magnitude of any particular signal remains bounded for all $t \geq 0$.

The verification that $L^\infty(\mathbb{N})$ is a vector space over \mathbb{R} is fairly straightforward. The algebraic properties of item 3 in Definition 1.4.1 are verified exactly as for V in the previous example, where again the zero vector is $(0, 0, 0, \dots)$ and the additive inverse of \mathbf{x} is $(-x_0, -x_1, -x_2, \dots)$. To show closure under vector

addition, consider vectors \mathbf{x} and \mathbf{y} with $|x_k| \leq M_x$ and $|y_k| \leq M_y$ for all $k \geq 0$. From the triangle inequality for real numbers

$$|x_k + y_k| \leq |x_k| + |y_k| \leq M_x + M_y,$$

so the components of $\mathbf{x} + \mathbf{y}$ are bounded in magnitude by $M_x + M_y$. Thus $\mathbf{x} + \mathbf{y} \in L^\infty(\mathbb{N})$, and the set is closed under addition. Similarly for any k the k th component $c x_k$ of $c\mathbf{x}$ is bounded by $|c|M_x$, and the set is closed under scalar multiplication. This makes $L^\infty(\mathbb{N})$ a subspace of the vector space V from the previous example.

If we consider bi-infinite vectors as defined by equation (1.8) with the condition that for each \mathbf{x} there is some number M such that $|x_k| \leq M$ for all $k \in \mathbb{Z}$, then we obtain the vector space $L^\infty(\mathbb{Z})$.

■ EXAMPLE 1.5

We may impose the condition that for each sequence of real numbers \mathbf{x} of the form in (1.7) we have

$$\sum_{k=0}^{\infty} |x_k|^2 < \infty, \quad (1.9)$$

in which case the resulting set is called $L^2(\mathbb{N})$, or often just ℓ^2 . This is even more stringent than the condition for $L^\infty(\mathbb{N})$; verification of this assertion and that $L^2(\mathbb{N})$ is a vector space is left for Exercise 1.11. We may also let the components x_k be complex numbers, and the result is still a vector space.

Conditions like (1.9) that bound the “squared value” of some object are common in applied mathematics and usually correspond to finite energy in an underlying physical process.

A very common variant of $L^2(\mathbb{N})$ is the space $L^2(\mathbb{Z})$, consisting of vectors of the form in equation (1.8) that satisfy

$$\sum_{k=-\infty}^{\infty} |x_k|^2 < \infty.$$

The space $L^2(\mathbb{Z})$ will play an important role throughout Chapters 6 and 7.

Variations on the spaces above are possible, and common. Which vector space we work in depends on our model of the underlying physical process and the analysis we hope to carry out.

1.4.2 Examples—Function Spaces

In the examples above the spaces all consist of vectors that are lists or arrays, finite or infinite, of real or complex numbers. Functions can also be interpreted as elements

of vector spaces, and this is the appropriate setting when dealing with analog signals or images. The mathematics in this case can be more complicated, especially when dealing with issues concerning approximation, limits, and convergence (about which we've said little so far). We'll have limited need to work in this setting, at least until Chapter 7. Here are some relevant examples.

■ EXAMPLE 1.6

Consider the set of all real-valued functions f that are defined and continuous at every point in a closed interval $[a, b]$ of the real line. This means that for any $t_0 \in [a, b]$,

$$\lim_{t \rightarrow t_0} f(t) = f(t_0),$$

where t approaches from the right only in the case that $t_0 = a$ and from the left only in the case that $t_0 = b$. The sum $f + g$ of two functions f and g is the function defined by $(f + g)(t) = f(t) + g(t)$, and the scalar multiple cf is defined via $(cf)(t) = cf(t)$. With these operations this is a vector space over \mathbb{R} , for it is closed under addition since the sum of two continuous functions is continuous. It is also closed under scalar multiplication, since a scalar multiple of a continuous function is continuous. The algebraic properties of item 3 in Definition 1.4.1 are easily verified with the “zero function” as the additive identity and $-f$ as the additive inverse of f . The resulting space is denoted $C[a, b]$.

The closed interval $[a, b]$ can be replaced by the open interval (a, b) to obtain the vector space $C(a, b)$. The spaces $C[a, b]$ and $C(a, b)$ do not coincide, for example, $f(t) = 1/t$ lies in $C(0, 1)$ but not $C[0, 1]$. In this case $1/t$ isn't defined at $t = 0$, and moreover this function can't even be extended to $t = 0$ in a continuous manner.

■ EXAMPLE 1.7

Consider the set of all real-valued functions f that are piecewise continuous on the interval $[a, b]$; that is, f is defined and continuous at all but finitely many points in $[a, b]$. With addition and scalar multiplication as defined in the last example this is a vector space over \mathbb{R} . The requisite algebraic properties are verified in precisely the same manner. To show closure under addition, just note that any point of discontinuity for $f + g$ must be a point of discontinuity for f or g ; hence $f + g$ can have only finitely many points of discontinuity. The discontinuities for cf are precisely those for f .

Both $C(a, b)$ and $C[a, b]$ are subspaces of this vector space (which doesn't have any standard name).

■ EXAMPLE 1.8

Let V denote those functions f in $C(a, b)$ for which

$$\int_a^b f^2(t) dt < \infty. \quad (1.10)$$

A function f that is continuous on (a, b) can have no vertical asymptotes in the open interval, but may be unbounded as t approaches the endpoint $t = a$ or $t = b$. Thus the integral above (and all integrals in this example) should be interpreted as improper integrals, that is,

$$\int_a^b f^2(t) dt = \lim_{p \rightarrow a^+} \int_p^r f^2(t) dt + \lim_{q \rightarrow b^-} \int_r^b f^2(t) dt,$$

where r is any point in (a, b) .

To show that V is closed under scalar multiplication, note that

$$\int_a^b (cf)^2(t) dt = c^2 \int_a^b f^2(t) dt < \infty,$$

since f satisfies the inequality (1.10). To show closure under vector addition, first note that for any real numbers p and q , we have $(p + q)^2 \leq 2p^2 + 2q^2$ (this follows easily from $0 \leq (p - q)^2$). As a consequence, for any two functions f and g in V and any $t \in (a, b)$, we have

$$(f(t) + g(t))^2 \leq 2f^2(t) + 2g^2(t).$$

Integrate both sides above from $t = a$ to $t = b$ (as improper integrals) to obtain

$$\int_a^b (f(t) + g(t))^2 dt \leq 2 \int_a^b f^2(t) dt + 2 \int_a^b g^2(t) dt < \infty,$$

so $f + g$ is in V . The algebraic properties in Definition 1.4.1 follow as before, so that V is a vector space over \mathbb{R} .

The space V as defined above doesn't have any standard name, but it is "almost" the vector space commonly termed $L^2(a, b)$, also called "the space of square integrable functions on (a, b) ." More precisely, the space defined above is the intersection $C(a, b) \cap L^2(a, b)$. Nonetheless, we will generally refer to it as " $L^2(a, b)$," and say more about it in Section 1.10.4. This space will make more appearances in the text, especially in Chapter 7.

Similar to the inequality (1.9), the condition (1.10) comes up fairly often in applied mathematics and usually corresponds to signals of finite energy.

■ EXAMPLE 1.9

Consider the set of functions $f(x, y)$ defined on some rectangular region $\Omega = \{(x, y); a \leq x \leq b, c \leq y \leq d\}$. We make no particular assumptions about the continuity or other nature of the functions. Addition and scalar multiplication are defined in the usual way, as $(f + g)(x, y) = f(x, y) + g(x, y)$ and $(cf)(x, y) = cf(x, y)$. This is a vector space over \mathbb{R} . The proof is in fact the same as in the case of functions of a single variable. This space would be useful for image analysis, with the functions representing graylevel intensities and Ω the image domain.

Of course, we can narrow the class of functions, for example, by considering only those that are continuous on Ω ; this space is denoted $C(\Omega)$. Or we can impose the further restriction that

$$\int_a^b \int_c^d f^2(x, y) dy dx < \infty,$$

which, in analogy to the one-dimensional case, we denote by $L^2(\Omega)$. There are many other important and potentially useful vector spaces of functions.

We could also choose the domain Ω to be infinite, for example, a half-plane or the whole plane. The region is selected to give a good tractable vector space model and to be relevant to the physical situation of interest, though unbounded domains are not generally necessary in image processing.

In addition to the eight basic arithmetic properties listed in Definition 1.4.1, certain other arithmetic properties of vector spaces are worth noting.

Proposition 1.4.1 *If V is a vector space over \mathbb{R} or \mathbb{C} , then*

1. *the vector $\mathbf{0}$ is unique;*
2. *$0\mathbf{u} = \mathbf{0}$ for any vector \mathbf{u} ;*
3. *the additive inverse of any vector \mathbf{u} is unique, and is given by $(-1)\mathbf{u}$.*

These properties look rather obvious and are usually easy to verify in any specific vector space as in Examples 1.1 to 1.9. They also hold in any vector space, and can be shown directly from the eight arithmetic properties for a vector space. The careful proofs can be a bit tricky though! See Exercise 1.12.

We have already started to use the additive vector space structure when we modeled noise in signals and images with equations (1.1) and (1.5). The vector space structure will be indispensable when we discuss the decomposition of signals and images into linear combinations of more “basic” components.

Tables 1.1 and 1.2 give a brief summary of some of the important spaces of interest, as well as when each space might be used. As mentioned, the analog models are used when we consider the actual physical processes that underly signals and images, but for computation we always consider the discrete version.

TABLE 1.1 Discrete Signal Models and Uses

Notation	Vector Space Description
\mathbb{R}^N	$\{\mathbf{x} = (x_1, \dots, x_N) : x_i \in \mathbb{R}\}$, finite sampled signals
\mathbb{C}^N	$\{\mathbf{x} = (x_1, \dots, x_N) : x_i \in \mathbb{C}\}$, analysis of sampled signals
$L^\infty(\mathbb{N})$ or ℓ^∞	$\{\mathbf{x} = (x_0, x_1, \dots) : x_i \in \mathbb{R}, x_i \leq M \text{ for all } i \geq 0\}$ bounded, sampled signals, infinite time
$L^2(\mathbb{N})$ or ℓ^2	$\{\mathbf{x} = (x_0, x_1, \dots) : x_i \in \mathbb{R} \text{ or } x_i \in \mathbb{C}, \sum_k x_k ^2 < \infty\}$ sampled signals, finite energy, infinite time
$L^2(\mathbb{Z})$	$\{\mathbf{x} = (\dots, x_{-1}, x_0, x_1, \dots) : x_i \in \mathbb{R} \text{ or } x_i \in \mathbb{C}, \sum_k x_k ^2 < \infty\}$ sampled signals, finite energy, bi-infinite time
$M_{m,n}(\mathbb{R})$	Real $m \times n$ matrices, sampled rectangular images
$M_{m,n}(\mathbb{C})$	Complex $m \times n$ matrices, analysis of images

1.5 BASIC WAVEFORMS—THE ANALOG CASE

1.5.1 The One-dimensional Waveforms

To analyze signals and images, it can be extremely useful to decompose them into a sum of more elementary pieces or patterns, and then operate on the decomposed version, piece by piece. We will call these simpler pieces the *basic waveforms*. They serve as the essential building blocks for signals and images. In the context of Fourier analysis for analog signals these basic waveforms are simply sines and cosines, or equivalently, complex exponentials. Specifically, the two basic waveforms of interest are $\cos(\omega t)$ and $\sin(\omega t)$, or their complex exponential equivalent $e^{i\omega t}$, where ω acts as a frequency parameter.

The complex exponential basic waveforms will be our preferred approach. Recall Euler's identity,

$$e^{i\theta} = \cos(\theta) + i \sin(\theta). \quad (1.11)$$

TABLE 1.2 Analog Signal Models and Uses

Notation	Vector Space Description
$C(a, b)$ or $C[a, b]$	Continuous functions on (a, b) or $[a, b]$, continuous analog signal
$L^2(a, b)$	f Riemann integrable and $\int_a^b f^2(x) dx < \infty$, analog signal with finite energy
$L^2(\Omega)$ ($\Omega = [a, b] \times [c, d]$)	f Riemann integrable and $\int_a^b \int_c^d f^2(x, y) dy dx < \infty$, analog image

From this we have (with $\theta = \omega t$ and $\theta = -\omega t$)

$$\begin{aligned} e^{i\omega t} &= \cos(\omega t) + i \sin(\omega t), \\ e^{-i\omega t} &= \cos(\omega t) - i \sin(\omega t), \end{aligned} \quad (1.12)$$

which can be solved for $\cos(\omega t)$ and $\sin(\omega t)$ as

$$\begin{aligned} \cos(\omega t) &= \frac{e^{i\omega t} + e^{-i\omega t}}{2}, \\ \sin(\omega t) &= \frac{e^{i\omega t} - e^{-i\omega t}}{2i}. \end{aligned} \quad (1.13)$$

If we can decompose a given signal $x(t)$ into a linear combination of waveforms $\cos(\omega t)$ and $\sin(\omega t)$, then equations (1.13) make it clear that we can also decompose $x(t)$ into a linear combination of appropriate complex exponentials. Similarly equations (1.12) can be used to convert any complex exponential decomposition into sines and cosines. We thus also consider the complex exponential functions $e^{i\omega t}$ as basic waveforms.

Remark 1.5 In the real-valued sine/cosine case we only need to work with $\omega \geq 0$, since $\cos(-\omega t) = \cos(\omega t)$ and $\sin(-\omega t) = -\sin(\omega t)$. Any function that can be constructed as a sum using negative values of ω has an equivalent expression with positive ω .

■ EXAMPLE 1.10

Consider the signal $x(t) = \sin(t) + 3 \sin(-2t) - 2 \cos(-5t)$. From Remark 1.5 we can express $x(t)$ as $x(t) = \sin(t) - 3 \sin(2t) - 2 \cos(5t)$, using only positive values of ω in the expressions $\sin(\omega t)$ and $\cos(\omega t)$. Equations (1.13) also yield

$$x(t) = \frac{1}{2i}e^{it} - \frac{1}{2i}e^{-it} - \frac{3}{2i}e^{2it} + \frac{3}{2i}e^{-2it} - e^{5it} - e^{-5it},$$

a sum of basic complex exponential waveforms. Whether we work in trigonometric functions or complex exponentials matters little from the mathematical perspective. The trigonometric functions, because they're real-valued and familiar, have a natural appeal, but the complex exponentials often yield much cleaner mathematical formulas. As such, we will usually prefer to work with the complex exponential waveforms.

We can visualize $e^{i\omega t}$ by simultaneously graphing the real and imaginary parts as functions of t , as in Figure 1.6, with real parts solid and imaginary parts dashed. Note that

$$\begin{aligned} \cos(\omega t) &= \operatorname{Re}(e^{i\omega t}), \\ \sin(\omega t) &= \operatorname{Im}(e^{i\omega t}). \end{aligned} \quad (1.14)$$

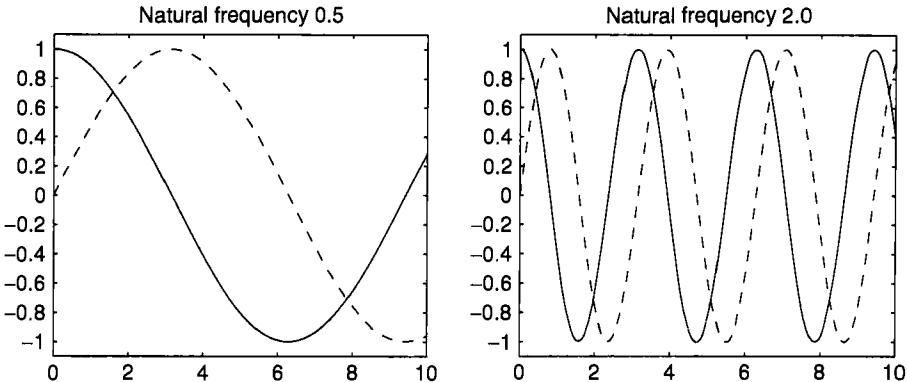


FIGURE 1.6 Real (*solid*) and imaginary (*dashed*) parts of complex exponentials.

Of course the real and imaginary parts, and $e^{i\omega t}$ itself, are periodic and ω controls the frequency of oscillation. The parameter ω is called the *natural frequency* of the waveform.

The period of $e^{i\omega t}$ can be found by considering those values of λ for which $e^{i\omega(t+\lambda)t} = e^{i\omega t}$ for all t , which yields

$$e^{i\omega(t+\lambda)t} = e^{i\omega t} e^{i\omega\lambda t} = e^{i\omega t} (\cos(\omega\lambda) + i \sin(\omega\lambda)), \quad (1.15)$$

so that $e^{i\omega(t+\lambda)t} = e^{i\omega t}$ forces $\cos(\omega\lambda) + i \sin(\omega\lambda) = 1$. The smallest positive value of λ for which this holds satisfies $\lambda|\omega| = 2\pi$. Thus $\lambda = 2\pi/|\omega|$, which is the *period* of $e^{i\omega t}$ (or the *wavelength* if t is a spatial variable).

The quantity $q = \lambda/\lambda = \omega/2\pi$ (so $\omega = 2\pi q$) is the number of oscillations made by the waveform in a unit time interval and is called the *frequency* of the waveform. If t denotes time in seconds, then q has units of *Hertz*, or cycles per second. It is often useful to write the basic waveform $e^{i\omega t}$ as $e^{2\pi iq t}$, to explicitly note the frequency q of oscillation. More precisely, the frequency (in Hertz) of the waveform $e^{2\pi iq t}$ is $|q|$ Hertz, since frequency is by convention nonnegative.

In real-valued terms; we can use $\cos(2\pi qt)$ and $\sin(2\pi qt)$ with $q \geq 0$ in place of $\cos(\omega t)$ and $\sin(\omega t)$ with $\omega \geq 0$.

As we will see later, any “reasonable” (e.g., bounded and piecewise continuous) function $x(t)$ defined on an interval $[-T, T]$ can be written as an infinite sum of basic waveforms $e^{i\omega t}$, as

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{\pi i k t / T} \quad (1.16)$$

for an appropriate choice of the constants c_k . The natural frequency parameter ω assumes the values $\pi k/T$ for $k \in \mathbb{Z}$, or equivalently the frequency q assumes values $k/2T$. An expansion analogous to (1.16) also exists using the sine/cosine waveforms.

1.5.2 2D Basic Waveforms

The 2D basic waveforms are governed by a pair of frequency parameters α and β . Let (x, y) denote coordinates in the plane. The basic waveforms are products of complex exponentials and can be written in either additive form (left side below) or a product form (right side),

$$e^{i(\alpha x + \beta y)} = e^{i\alpha x} e^{i\beta y}. \quad (1.17)$$

As in the one-dimensional case we can convert to a trigonometric form,

$$e^{i(\alpha x + \beta y)} = \cos(\alpha x + \beta y) + i \sin(\alpha x + \beta y),$$

and conversely,

$$\begin{aligned}\cos(\alpha x + \beta y) &= \frac{e^{i(\alpha x + \beta y)} + e^{-i(\alpha x + \beta y)}}{2}, \\ \sin(\alpha x + \beta y) &= \frac{e^{i(\alpha x + \beta y)} - e^{-i(\alpha x + \beta y)}}{2i}.\end{aligned}$$

Thus the family of functions

$$\{\cos(\alpha x + \beta y), \sin(\alpha x + \beta y)\} \quad (1.18)$$

is an alternate set of basic waveforms.

Sometimes a third class of basic waveforms is useful. An application of Euler's formula to both exponentials on the right in equation (1.17) shows that

$$\begin{aligned}e^{i(\alpha x + \beta y)} &= \cos(\alpha x) \cos(\beta y) - \sin(\alpha x) \sin(\beta y) \\ &\quad + i(\cos(\alpha x) \sin(\beta y) + \sin(\alpha x) \cos(\beta y)),\end{aligned}$$

so these complex exponential basic waveforms can be expanded into linear combinations of functions from the family

$$\{\cos(\alpha x) \cos(\beta y), \sin(\alpha x) \sin(\beta y), \cos(\alpha x) \sin(\beta y), \sin(\alpha x) \cos(\beta y)\}. \quad (1.19)$$

Conversely, each of these functions can be written in terms of complex exponentials (see Exercise 1.15). The functions in (1.19) also form a basic set of waveforms.

Just as in the one-dimensional case, for the real-valued basic waveforms (1.18) or (1.19) we can limit our attention to the cases $\alpha, \beta \geq 0$.

We almost always use the complex exponential waveforms in our analysis, however, except when graphing. As in the one-dimensional case these exponential

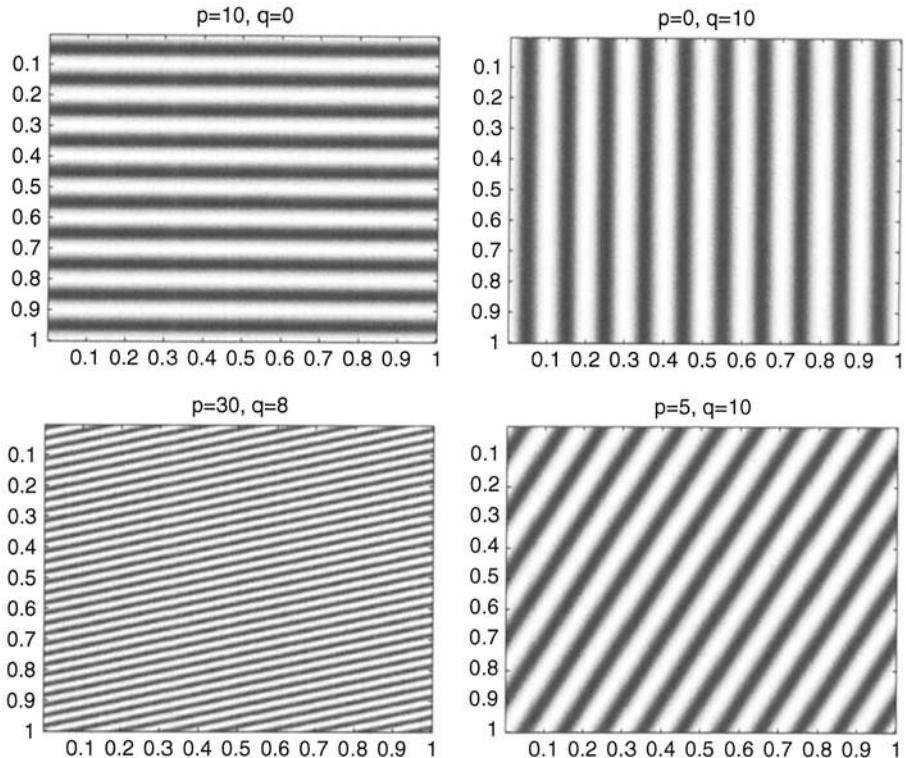


FIGURE 1.7 Grayscale image of $\cos(2\pi(px + qy)) = \operatorname{Re}(e^{2\pi i(px + qy)})$ for various p and q .

waveforms can be written in a frequency format as

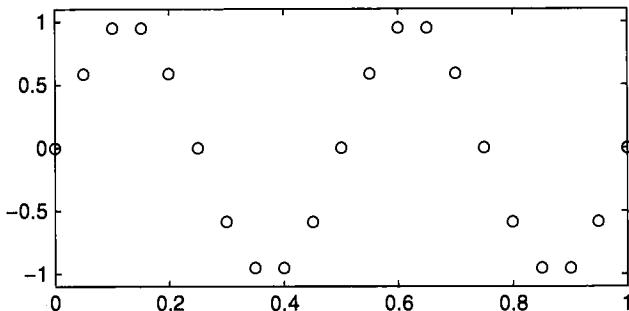
$$e^{i(\alpha x + \beta y)} = e^{2\pi i(px + qy)},$$

where p and q are frequencies in the x and y directions. In the plots in Figure 1.7 we show the real parts of the basic exponential waveforms for several values of p and q , as grayscale images on the unit square $0 \leq x, y \leq 1$, with y downward as per Remark 1.1 on page 8. The waves seem to have a direction and wavelength; see Exercise 1.18.

1.6 SAMPLING AND ALIASING

1.6.1 Introduction

As remarked prior to equation (1.16), an analog signal or function on an interval $[-T, T]$ can be decomposed into a linear combination of basic waveforms $e^{i\omega t}$, or the corresponding sines and cosines. For computational purposes, however, we sample the signal and work with the corresponding discrete quantity, a vector. The

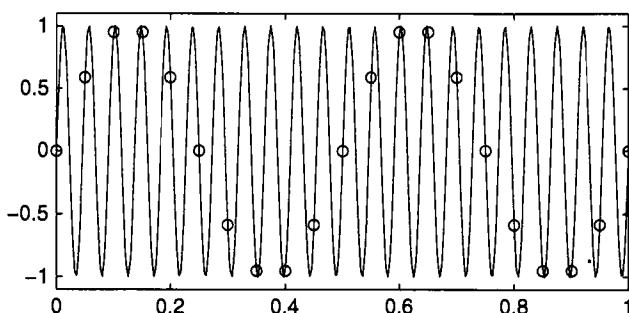
**FIGURE 1.8** Sampled signal, $\Delta T = 0.05$.

analog waveforms $e^{i\omega t}$ must then be replaced with “equivalent” discrete waveforms. What should these waveforms be?

The obvious answer is to use the sampled analog waveforms, but an interesting phenomenon called *aliasing* shows up. It should be intuitively clear that sampling destroys information about the analog signal. It is in the case where the sampling is done on basic waveforms that this loss of information is especially easy to quantify. We will thus take a short detour to discuss aliasing, and then proceed to the discrete model waveforms in the next section.

To illustrate aliasing, consider the sampled signal graphed in Figure 1.8, obtained by sampling the basic waveform $\sin(\omega t)$ for some “unknown” ω on the interval $0 \leq t \leq 1$ at intervals of $\Delta T = 0.05$. The sampled waveform appears to make exactly two full cycles in the time interval $[0, 1]$, corresponding to a frequency of two Hertz and the basic waveform $\sin(4\pi t)$. However, Figure 1.9 shows a plot of the “true” analog signal, superimposed on the sampled signal!

The actual analog waveform is $x(t) = \sin(44\pi t)$, corresponding to a frequency of 22 Hertz ($\omega = 44\pi$). The plot of the sampled signal is quite deceptive and

**FIGURE 1.9** Analog and sampled signal, $\Delta T = 0.05$.

illustrates *aliasing*, in which sampling destroys our ability to distinguish between basic waveforms with certain relative frequencies.

1.6.2 Aliasing for Complex Exponential Waveforms

To quantify this phenomenon, let's first look at the situation for complex waveforms $e^{i\omega t}$. For simplicity we write these in frequency form $e^{2\pi i q t}$ with $q = \omega/2\pi$, so q is in Hertz; note q is not required to be an integer. Suppose that we sample such a waveform N times per second, at times $t = k/N$ for $k = 0, 1, 2, \dots$. The sampled waveform yields values $e^{2\pi i q k / N}$. Under what circumstances will another analog waveform $e^{2\pi i \tilde{q} t}$ at frequency \tilde{q} Hertz yield the same sampled values at times $t = k/N$? Stated quantitatively, this means that

$$e^{2\pi i q k / N} = e^{2\pi i \tilde{q} k / N}$$

for all integers $k \geq 0$. Divide both sides above by $e^{2\pi i q k / N}$ to obtain $1 = e^{2\pi i (\tilde{q} - q) k / N}$, or equivalently,

$$1 = (e^{2\pi i (\tilde{q} - q) / N})^k \quad (1.20)$$

for all $k \geq 0$. Now if a complex number z satisfies $z^k = 1$ for all integers k then $z = 1$ (consider the case $k = 1$). From equation (1.20) we conclude that $e^{2\pi i (\tilde{q} - q) / N} = 1$. Since $e^x = 1$ only when $x = 2\pi i m$ where $m \in \mathbb{Z}$, it follows that $2\pi i (\tilde{q} - q) / N = 2\pi i m$, or $\tilde{q} - q = mN$. Thus the waveforms $e^{2\pi i q t}$ and $e^{2\pi i \tilde{q} t}$ sampled at times $t = k/N$ yield identical values exactly when

$$\tilde{q} - q = mN \quad (1.21)$$

for some integer m .

Equation (1.21) quantifies the phenomenon of aliasing: When sampled with sampling interval $\Delta T = 1/N$ (frequency N Hertz) the two waveforms $e^{2\pi i q t}$ and $e^{2\pi i \tilde{q} t}$ will be *aliased* (yield the same sampled values) whenever \tilde{q} and q differ by any multiple of the sampling rate N . Equivalently, $e^{i\omega t}$ and $e^{i\tilde{\omega} t}$ yield exactly the same sampled values when $\tilde{\omega} - \omega = 2\pi mN$.

Aliasing has two implications, one “physical” and one “mathematical.” The physical implication is that if an analog signal consists of a superposition of basic waveforms $e^{2\pi i q t}$ and is sampled at N samples per second, then for any particular frequency q_0 the waveforms

$$\dots, e^{2\pi i (q_0 - 2N)t}, e^{2\pi i (q_0 - N)t}, e^{2\pi i q_0 t}, e^{2\pi i (q_0 + N)t}, e^{2\pi i (q_0 + 2N)t} \dots$$

are all aliased. Any information concerning their individual characteristics (amplitudes and phases) is lost. The only exception is if we know a priori that the signal

consists only of waveforms in a specific and sufficiently small frequency range. For example, if we know that the signal consists only of waveforms $e^{2\pi i q t}$ with $-N/2 < q \leq N/2$ (i.e., frequencies $|q|$ between 0 and $N/2$), then no aliasing will occur because $q \pm N$, $q \pm 2N$, and so on, do not lie in this range. This might be the case if the signal has been *low-pass filtered* prior to being sampled, to remove (by analog means) all frequencies greater than $N/2$. In this case sampling at frequency N would produce no aliasing.

The mathematical implication of aliasing is this: when analyzing a signal sampled at frequency N , we need only use the sampled waveforms $e^{2\pi i q k/N}$ with $-N/2 < q \leq N/2$. Any discrete basic waveform with frequency outside this range is aliased with, and hence identical to, a basic waveform within this range.

1.6.3 Aliasing for Sines and Cosines

Similar considerations apply when using the sine/cosine waveforms. From equations (1.13) it's easy to see that when sampled at frequency N the functions $\sin(2\pi q t)$ or $\cos(2\pi q t)$ will be aliased with waveforms $\sin(2\pi \tilde{q} t)$ or $\cos(2\pi \tilde{q} t)$ if $\tilde{q} - q = mN$ for any integer m . Indeed, one can see directly that if $\tilde{q} = q + mN$, then

$$\sin(2\pi \tilde{q} k/N) = \sin(2\pi(q + mN)k/N) = \sin(2\pi q k/N + 2\pi km) = \sin(2\pi q k/N),$$

since $\sin(t + 2\pi km) = \sin(t)$ for any t , where k and m are integers. A similar computation holds for the cosine. Thus as in the complex exponential case we may restrict our attention to a frequency interval in q of length N , for example, $-N/2 < q \leq N/2$.

However, in the case of the sine/cosine waveforms the range for q (or ω) can be narrowed a bit further. In light of Remark 1.5 on page 18 we need not consider $q < 0$ if our main interest is the decomposition of a signal into a superposition of sine or cosine waveforms, for $\cos(-2\pi q t) = \cos(2\pi q t)$ and $\sin(-2\pi q t) = -\sin(2\pi q t)$. In the case of the sine/cosine waveforms we need only consider the range $0 \leq q \leq N/2$. This is actually identical to the restriction for the complex exponential case, where the frequency $|q|$ of $e^{2\pi i q t}$ is restricted to $0 \leq |q| \leq N/2$.

1.6.4 The Nyquist Sampling Rate

For both complex exponential waveforms $e^{2\pi i q t}$ and the basic trigonometric waveforms $\sin(2\pi q t)$, $\cos(2\pi q t)$, sampling at N samples per second results in frequencies greater than $N/2$ being aliased with frequencies between 0 and $N/2$. Thus, if an analog signal is known to contain only frequencies of magnitude F and lower, sampling at a frequency $N \geq 2F$ (so $F \leq N/2$) results in no aliasing. This is one form of what is called the *Nyquist sampling rate* or *Nyquist sampling criterion*: to avoid aliasing, we must sample at twice the highest frequency present in an analog signal. This means at least two samples per cycle for the highest frequency. One typically samples at a slightly greater rate to ensure greater fidelity. Thus commercial CD's use a sample rate of 44.1 kHz, which is slightly greater than the generally accepted

maximum audible frequency of 20 kHz. A CD for dogs would require a higher sampling rate!

A closely related result, the Shannon sampling theorem, states that if an analog signal $x(t)$ contains only frequencies in the range 0 to $N/2$ and is sampled at sampling rate N , then $x(t)$ can be perfectly recovered for all t ; see [17, p. 87].

1.6.5 Aliasing in Images

Aliasing also occurs when images are sampled. Consider the simple grayscale image embodied by the function

$$f(x, y) = 256 \sin(2\pi(50x + 70y))$$

on the domain $0 \leq x, y \leq 1$. In Figure 1.10 are images based on sampling f on n by n grids for $n = 60, 100, 300$, and 1000 , and displayed with 0 as black, 255 as white (rounded down). To avoid aliasing, we expect to need a sampling frequency of at least 100 samples per unit distance in the x direction, 140 in the y . The 1000 by 1000 image comes closest to the “true” analog image, while the $n = 60$ and $n = 100$ images completely misrepresent the nature of the underlying analog signal. When $n = 60$, the stripes actually go the wrong way.

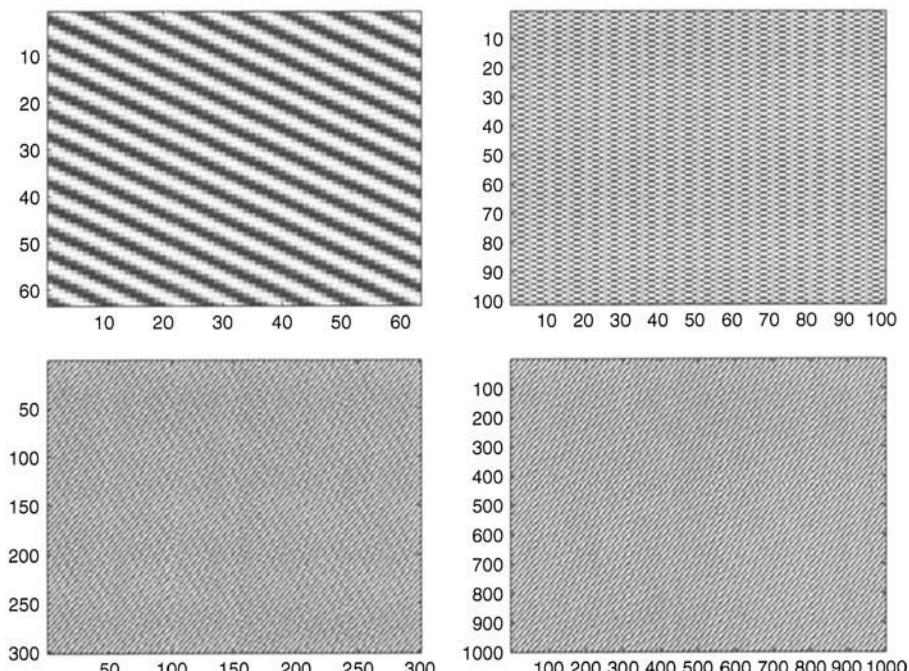


FIGURE 1.10 Aliasing in a 2D region, $n = 60$ (top left), $n = 100$ (top right), $n = 300$ (bottom left), and $n = 1000$ (bottom right).

In general, however, aliasing in images is difficult to convey consistently via the printed page, or even on a computer screen, because the effect is highly dependent on printer and screen resolution. See Section 1.11 at the end of this chapter, in which you can construct your own aliasing examples in Matlab, as well as audio examples.

1.7 BASIC WAVEFORMS—THE DISCRETE CASE

1.7.1 Discrete Basic Waveforms for Finite Signals

Consider a continuous signal $x(t)$ defined on a time interval $[0, T]$, sampled at the N times $t = nT/N$ for $n = 0, 1, 2, \dots, N - 1$; note we don't sample $x(t)$ at $t = T$. This yields discretized signal $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$, where $x_n = nT/N$, a vector in \mathbb{R}^N . In all that follows we will index vectors in \mathbb{R}^N from index 0 to index $N - 1$, as per Remark 1.4 on page 11.

As we show in a later section, the analog signal $x(t)$ can be decomposed into an infinite linear combination of basic analog waveforms, in this case of the form $e^{2\pi i k t/T}$ for $k \in \mathbb{Z}$. As discussed in the previous section, the appropriate basic waveforms are then the discretized versions of the waveforms $e^{2\pi i k t/T}$, obtained by sampling at times $t = nT/N$. This yields a sequence of basic waveform vectors which we denote by $\mathbf{E}_{N,k}$, indexed by k , of the form

$$\mathbf{E}_{N,k} = \begin{bmatrix} e^{2\pi i k 0/N} \\ e^{2\pi i k 1/N} \\ \vdots \\ e^{2\pi i k (N-1)/N} \end{bmatrix}, \quad (1.22)$$

a discrete version of $e^{2\pi i k t/T}$. Note though that the waveform vectors don't depend on T . The m th component $\mathbf{E}_{N,k}(m)$ of $\mathbf{E}_{N,k}$ is given by

$$\mathbf{E}_{N,k}(m) = e^{2\pi i k m / N}. \quad (1.23)$$

For any fixed N we can construct the basic waveform vector $\mathbf{E}_{N,k}$ for any $k \in \mathbb{Z}$, but as shown when we discussed aliasing, $\mathbf{E}_{N,k} = \mathbf{E}_{N,k+mN}$ for any integer m . As a consequence we need only consider the $\mathbf{E}_{N,k}$ for a range in k of length N , say of the form $k_0 + 1 \leq k \leq k_0 + N$ for some k_0 . A “natural” choice is $k_0 = -N/2$ (if N is even) corresponding to $-N/2 < k \leq N/2$ as in the previous aliasing discussion, but the range $0 \leq k \leq N - 1$ is usually more convenient for indexing and matrix algebra. However, no matter which range in k we use to index, we'll always be using the same set of N vectors since $\mathbf{E}_{N,k} = \mathbf{E}_{N,k-N}$.

When there is no potential confusion, we will omit the N index and write simply \mathbf{E}_k , rather than $\mathbf{E}_{N,k}$.

■ EXAMPLE 1.11

As an illustration, here are the vectors $\mathbf{E}_{4,k}$ for $k = -2$ to $k = 3$:

$$\mathbf{E}_{4,-2} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{E}_{4,-1} = \begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix}, \quad \mathbf{E}_{4,0} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\mathbf{E}_{4,1} = \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix}, \quad \mathbf{E}_{4,2} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{E}_{4,3} = \begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix}.$$

Note the aliasing relations $\mathbf{E}_{4,-2} = \mathbf{E}_{4,2}$ and $\mathbf{E}_{4,-1} = \mathbf{E}_{4,3}$. In particular, the sets $\{\mathbf{E}_{4,-1}, \mathbf{E}_{4,0}, \mathbf{E}_{4,1}, \mathbf{E}_{4,2}\}$ and $\{\mathbf{E}_{4,0}, \mathbf{E}_{4,1}, \mathbf{E}_{4,2}, \mathbf{E}_{4,3}\}$ (corresponding to $\mathbf{E}_{N,k}$ on the range $-N/2 < k \leq N/2$ or $0 \leq k \leq N-1$) are identical.

It's also worth noting the relation

$$\overline{\mathbf{E}_{N,k}} = \mathbf{E}_{N,N-k}, \quad (1.24)$$

where the overline denotes complex conjugation. Equation (1.24) is sometimes called “conjugate aliasing.” See Exercise 1.20.

Remark 1.6 There are a lot of periodic functions in the discussion above. For example, the basic waveform $e^{2\pi ikt/T}$ is periodic in t with period T/k . The quantity $\mathbf{E}_{N,k}(m)$ in equation (1.23) is defined for all k and m and periodic in both. As a consequence $\mathbf{E}_{N,k}$ is defined for all k , and periodic with period N . The one entity that is not manifestly periodic is the analog time signal $x(t)$, or its sampled version $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$. At times it will be useful, at least conceptually, to extend either periodically. The sampled signal $\mathbf{x} = (x_0, \dots, x_{N-1})$ can be extended periodically with period N in its index by defining

$$x_m = x_{m \bmod N}$$

for all m outside the range $0 \leq m \leq N-1$. We can also extend the analog signal $x(t)$ periodically to all real t by setting $x(t) = x(t \bmod P)$, where P denotes the period of $x(t)$.

1.7.2 Discrete Basic Waveforms for Images

As with the one-dimensional waveforms, the appropriate discrete waveforms in the two-dimensional case are the sampled basic waveforms. These discrete waveforms are naturally rectangular arrays or matrices.

To be more precise, consider a rectangular domain or image defined by $0 \leq x \leq S$, $0 \leq y \leq R$, but recall Remark 1.1 on page 8; here increasing y is downward. The sampling will take place on an m (in the y direction) by n (x direction) rectangular grid. The basic two-dimensional waveforms were given in equation (1.17). As we will see later, the parameters α and β are most conveniently taken to be of the form $\alpha = 2\pi l/S$ and $\beta = 2\pi k/R$ for integers k and l . Thus the analog basic waveforms to be sampled are the functions $e^{2\pi i(lx/S+ky/R)}$. Each such waveform is sampled at points of the form $x_s = sS/n$, $y_r = rR/m$, with $0 \leq s \leq n - 1$, $0 \leq r \leq m - 1$. The result is an $m \times n$ matrix $\mathcal{E}_{m,n,k,l}$ with row r , column s entry

$$\mathcal{E}_{m,n,k,l}(r, s) = e^{2\pi i(kr/m+ls/n)}. \quad (1.25)$$

Note that $\mathcal{E}_{m,n,k,l}$ does not depend on the image dimensions R or S . The indexes may seem a bit confusing, but recall that m and n are fixed by the discretization size (an m by n pixel image); l and k denote the frequency of the underlying analog waveform in the x and y directions, respectively. The parameters s and r correspond to the x and y coordinates of the sample point. These $m \times n$ matrices $\mathcal{E}_{m,n,k,l}$ constitute the basic waveforms in the discrete setting.

In Exercise 1.17 you are asked to show that $\mathcal{E}_{m,n,k,l}$ can be factored into a product

$$\mathcal{E}_{m,n,k,l} = \mathbf{E}_{m,k} \mathbf{E}_{n,l}^T, \quad (1.26)$$

where the superscript T denotes the matrix transpose operation and where the vectors $\mathbf{E}_{m,k}$ and $\mathbf{E}_{n,l}$ are the discrete basic waveforms in one-dimension, as defined in equation (1.22) (as column vectors). For example,

$$\mathcal{E}_{4,4,1,2} = E_{4,1} E_{4,2}^T = \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ i & -i & i & -i \\ -1 & 1 & -1 & 1 \\ -i & i & -i & i \end{bmatrix}$$

As in the one-dimensional case we will write $\mathcal{E}_{k,l}$ instead of $\mathcal{E}_{m,n,k,l}$ when there is no possibility for confusion.

A variety of aliasing relations for the waveforms $\mathcal{E}_{m,n,k,l}$ follow from equation (1.26) and those for the $E_{m,k}$; see Exercise 1.21. Also the $\mathcal{E}_{m,n,k,l}$ are periodic in k with period m , and periodic in l with period n . If we confine our attention to the ranges $0 \leq k < m$, $0 \leq l < n$, there are exactly mn distinct $\mathcal{E}_{m,n,k,l}$ waveforms. For any index pair (l, k) outside this range the corresponding waveform is identical to one of the mn basic waveforms in this range.

The 2D images of the discrete 2D waveforms look pretty much the same as the analog ones do for low values of k and l . For larger values of k and l the effects of aliasing begin to take over and the waveforms are difficult to accurately graph.

Remark 1.7 As in the one-dimensional case it may occasionally be convenient to extend an image matrix with entries $a_{r,s}$, $0 \leq r \leq m - 1$ and $0 \leq s \leq n - 1$, periodically to the whole plane. We can do this as

$$a_{r,s} = a_{r \bmod m, s \bmod n}.$$

1.8 INNER PRODUCT SPACES AND ORTHOGONALITY

1.8.1 Inner Products and Norms

Vector spaces provide a convenient framework for analyzing signals and images. However, it is helpful to have a bit more mathematical structure to carry out the analysis, specifically some ideas from geometry. Most of the vector spaces we'll be concerned with can be endowed with geometric notions such as "length" and "angle." Of special importance is the idea of "orthogonality." All of these notions can be quantified by adopting an *inner product* on the vector space of interest.

Inner Products The inner product is just a generalization of the familiar dot product from basic multivariable calculus. In the definition below a "function on $V \times V$ " means a function whose domain consists of ordered pairs of vectors from V .

Definition 1.8.1 Let V be a vector space over \mathbb{C} (resp., \mathbb{R}). An inner product (or scalar product) on V is a function from $V \times V$ to \mathbb{C} (resp., \mathbb{R}). We use (\mathbf{v}, \mathbf{w}) to denote the inner product of vectors \mathbf{v} and \mathbf{w} and require that for all vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and scalars a, b in \mathbb{C} (resp., \mathbb{R}).

1. $(\mathbf{v}, \mathbf{w}) = \overline{(\mathbf{w}, \mathbf{v})}$, (conjugate symmetry)
2. $(a\mathbf{u} + b\mathbf{v}, \mathbf{w}) = a(\mathbf{u}, \mathbf{w}) + b(\mathbf{v}, \mathbf{w})$, (linearity in the first argument)
3. $(\mathbf{v}, \mathbf{v}) \geq 0$, and $(\mathbf{v}, \mathbf{v}) = 0$ if and only if $\mathbf{v} = \mathbf{0}$

In the case where V is a vector space over \mathbb{R} , condition 1 is simply $(\mathbf{v}, \mathbf{w}) = (\mathbf{w}, \mathbf{v})$. If V is over \mathbb{C} , then condition 1 also immediately implies that $(\mathbf{v}, \mathbf{v}) = \overline{(\mathbf{v}, \mathbf{v})}$ so that (\mathbf{v}, \mathbf{v}) is always real-valued, and hence condition 3 makes sense. The linearity with respect to the first variable in property 2 easily extends to any finite linear combination.

One additional fact worth noting is that the inner product is conjugate-linear in the second argument, that is,

$$\begin{aligned} (\mathbf{w}, a\mathbf{u} + b\mathbf{v}) &= \overline{(a\mathbf{u} + b\mathbf{v}, \mathbf{w})}, && \text{by property 1,} \\ &= \bar{a} \overline{(\mathbf{u}, \mathbf{w})} + \bar{b} \overline{(\mathbf{v}, \mathbf{w})}, && \text{by property 2,} \\ &= \bar{a} (\mathbf{w}, \mathbf{u}) + \bar{b} (\mathbf{w}, \mathbf{v}), && \text{by property 1.} \end{aligned} \tag{1.27}$$

A vector space equipped with an inner product is called an *inner product space*.

Norms Another useful geometric notion on a vector space V is that of a *norm*, a way of quantifying the size or length of vectors in V . This also allows us to quantify the distance between elements of V .

Definition 1.8.2 A norm on a vector space V (over \mathbb{R} or \mathbb{C}) is a function $\|\mathbf{v}\|$ from V to \mathbb{R} with the properties that

1. $\|\mathbf{v}\| \geq 0$, and $\|\mathbf{v}\| = 0$ if and only if $\mathbf{v} = \mathbf{0}$
2. $\|a\mathbf{v}\| = |a|\|\mathbf{v}\|$
3. $\|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|$ (the triangle inequality)

for all vectors $\mathbf{v}, \mathbf{w} \in V$, and scalars a .

A vector space equipped with a norm is called (not surprisingly) a *normed vector space*, or sometimes a *normed linear space*.

An inner product (\mathbf{v}, \mathbf{w}) on a vector space V always induces a corresponding norm via the relation

$$\|\mathbf{v}\| = \sqrt{(\mathbf{v}, \mathbf{v})}. \quad (1.28)$$

(See Exercise 1.27.) Thus every inner product space is a normed vector space, but the converse is not true; see Example 1.16 and Exercise 1.28. We'll make frequent use of equation (1.28) in the form $\|\mathbf{v}\|^2 = (\mathbf{v}, \mathbf{v})$.

Remark 1.8 It's also useful to define the distance between two vectors \mathbf{v} and \mathbf{w} in a normed vector space as $\|\mathbf{v} - \mathbf{w}\|$. For example, in \mathbb{R}^n ,

$$\|\mathbf{v} - \mathbf{w}\| = ((v_1 - w_1)^2 + \cdots + (v_n - w_n)^2)^{1/2},$$

which is the usual distance formula.

Generally, the function $\|\mathbf{v} - \mathbf{w}\|$ on $V \times V$ defines a *metric* on V , a way to measure the distance between the elements of the space, and turns V into a *metric space*. The study of metrics and metric spaces is a large area of mathematics, but in this text we won't need this much generality.

Remark 1.9 If the normed vector space V in question has any kind of physical interpretation, then the quantity $\|\mathbf{v}\|^2$ frequently turns out to be proportional to some natural measure of "energy." The expression for the energy of most physical systems is quadratic in nature with respect to the variables that characterize the state of the system. For example, the kinetic energy of a particle with mass m and speed v is $\frac{1}{2}mv^2$, quadratic in v . The energy dissipated by a resistor is V^2/R , quadratic in V , where R is the resistance and V the potential drop across the resistor. The concept of energy is also important in signal and image analysis, and the quantification of the energy in these settings is quadratic in nature. We'll say more on this later.

1.8.2 Examples

Here are some specific, useful inner product spaces, and the corresponding norms.

■ EXAMPLE 1.12

\mathbb{R}^n (real Euclidian space): The most common inner product on \mathbb{R}^n is the dot product, defined by

$$(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

for vectors $\mathbf{x} = (x_1, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$ in \mathbb{R}^n . Properties 1 to 3 for inner products are easily verified.

The corresponding norm from equation (1.28) is

$$\|\mathbf{x}\| = (x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2},$$

the usual Euclidean norm.

■ EXAMPLE 1.13

\mathbb{C}^n (complex Euclidean space): On \mathbb{C}^n the usual inner product is

$$(\mathbf{x}, \mathbf{y}) = x_1 \overline{y_1} + x_2 \overline{y_2} + \cdots + x_n \overline{y_n}$$

for vectors $\mathbf{x} = (x_1, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$ in \mathbb{C}^n . Conjugation of the second vector's components is important! The corresponding norm from equation (1.28) is

$$\|\mathbf{x}\| = (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{1/2},$$

where we have made use of the fact that $z\bar{z} = |z|^2$ for any complex number z .

■ EXAMPLE 1.14

$M_{m,n}(\mathbb{C})$ ($m \times n$ matrices with complex entries): On $M_{m,n}(\mathbb{C})$ an inner product is given by

$$(\mathbf{A}, \mathbf{B}) = \sum_{j=1}^m \sum_{k=1}^n a_{j,k} \overline{b_{j,k}}$$

for matrices \mathbf{A} and \mathbf{B} with entries $a_{j,k}$ and $b_{j,k}$, respectively. The corresponding norm from equation (1.28) is

$$\|\mathbf{A}\| = \left(\sum_{j=1}^m \sum_{k=1}^n |a_{j,k}|^2 \right)^{1/2},$$

called the *Frobenius norm*. As in Example 1.2, as an inner product space $M_{m,n}(\mathbb{C})$ is really “identical” to \mathbb{C}^{mn} .

■ EXAMPLE 1.15

$C[a, b]$ (continuous functions on $[a, b]$): Consider the vector space $C[a, b]$, and suppose that the functions can assume complex values. An inner product on this space is given by

$$(f, g) = \int_a^b f(t)\overline{g(t)} dt.$$

It's not hard to see that the integral is well-defined, for f and g are continuous functions on a closed interval, hence bounded, so that the product fg is continuous and bounded. The integral therefore converges. Of course, if the functions are real-valued, the conjugation is unnecessary.

Properties 1 and 2 for the inner product follow easily from properties of the Riemann integral. Only property 3 for inner products needs comment. First,

$$(f, f) = \int_a^b |f(t)|^2 dt \geq 0, \quad (1.29)$$

since the integrand is nonnegative and the integral of a nonnegative function is nonnegative. However, the second assertion in property 3 needs some thought—if the integral in (1.29) actually equals zero, must f be the zero function?

We can prove that this is so by contradiction: suppose that f is not identically zero, say $f(t_0) \neq 0$ for some $t_0 \in [a, b]$. Then $|f(t_0)|^2 > 0$. Moreover, since f is continuous, so is $|f(t)|^2$. We can thus find some small interval $(t_0 - \delta, t_0 + \delta)$ with $\delta > 0$ on which $|f(t)|^2 \geq |f(t_0)|^2/2$. Then

$$(f, f) = \int_a^{t_0-\delta} |f(t)|^2 dt + \int_{t_0-\delta}^{t_0+\delta} |f(t)|^2 dt + \int_{t_0+\delta}^b |f(t)|^2 dt.$$

Since $|f(t)|^2 \geq |f(t_0)|^2/2$ for $t \in (t_0 - \delta, t_0 + \delta)$ the middle integral on the right above is positive and greater than $\delta|f(t_0)|^2$ (the area of a 2δ width by $|f(t_0)|^2/2$ tall rectangle under the graph of $|f(t)|^2$). The other two integrals are at least nonnegative. We conclude that if $f \in C[a, b]$ is not the zero function, then $(f, f) > 0$. Equivalently $(f, f) = 0$ only if $f \equiv 0$.

The corresponding norm for this inner product is

$$\|f\| = \left(\int_a^b |f(t)|^2 dt \right)^{1/2}.$$

In light of the discussion above $\|f\| = 0$ if and only if $f \equiv 0$.

■ EXAMPLE 1.16

Another commonly used norm on the space $C[a, b]$ is the *supremum* norm, defined by

$$\|f\|_{\infty} = \sup_{x \in [a, b]} |f(x)|.$$

Recall that the supremum of a set $A \subset \mathbb{R}$ is the smallest real number M such $a \leq M$ for every $a \in A$, meaning M is the “least upper bound” for the elements of A . If f is continuous, then we can replace “sup” in the definition of $\|f\|_{\infty}$ with “max,” since a continuous function on a closed bounded interval $[a, b]$ must attain its supremum.

The supremum norm does not come from any inner product in equation (1.28); see Exercise 1.28.

■ EXAMPLE 1.17

$C(\Omega)$ (the set of continuous complex-valued functions on a closed rectangle $\Omega = \{(x, y); a \leq x \leq b, c \leq y \leq d\}$): An inner product on this space is given by

$$(f, g) = \int_a^b \int_c^d f(x, y) \overline{g(x, y)} dy dx.$$

As in the one-dimensional case, the integral is well-defined since f and g must be bounded; hence the product fg is continuous and bounded. The integral therefore converges. An argument similar to that of the previous example shows that property 3 for inner products holds.

The corresponding norm is

$$\|f\| = \left(\int_a^b \int_c^d |f(x, y)|^2 dy dx \right)^{1/2}.$$

This space can be considerably enlarged, to include many discontinuous functions that satisfy $\|f\| < \infty$.

When we work in a function space like $C[a, b]$ we’ll sometimes use the notation $\|f\|_2$ (rather than just $\|f\|$) to indicate the Euclidean norm that stems from the inner product, and so avoid confusion with the supremum norm (or other norms).

1.8.3 Orthogonality

Recall from elementary vector calculus that the dot product (\mathbf{v}, \mathbf{w}) of two vectors \mathbf{v} and \mathbf{w} in \mathbb{R}^2 or \mathbb{R}^3 satisfies the relation

$$(\mathbf{v}, \mathbf{w}) = \|\mathbf{v}\| \|\mathbf{w}\| \cos(\theta), \quad (1.30)$$

where $\|\mathbf{v}\|$ is the length of \mathbf{v} , $\|\mathbf{w}\|$ the length of \mathbf{w} , and θ is the angle between \mathbf{v} and \mathbf{w} . In particular, it's easy to see that $(\mathbf{v}, \mathbf{w}) = 0$ exactly when $\theta = \pi/2$ radians, so \mathbf{v} and \mathbf{w} are orthogonal to each other. The notion of orthogonality can be an incredibly powerful tool. This motivates the following general definition:

Definition 1.8.3 Two vectors \mathbf{v} and \mathbf{w} in an inner product space V are “orthogonal” if $(\mathbf{v}, \mathbf{w}) = 0$.

The notion of orthogonality depends on not only the vectors but also the inner product we are using (there may be more than one!) We will say that a subset S (finite or infinite) of vectors in an inner product space V is *pairwise orthogonal*, or more commonly just *orthogonal*, if $(\mathbf{v}, \mathbf{w}) = 0$ for any pair of distinct ($\mathbf{v} \neq \mathbf{w}$) vectors $\mathbf{v}, \mathbf{w} \in S$.

■ EXAMPLE 1.18

Let $S = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ denote the standard basis vectors in \mathbb{R}^n (\mathbf{e}_k has a “1” in the k th position, zeros elsewhere), with the usual Euclidean inner product (and indexing from 1 to n). The set S is orthogonal since $(\mathbf{e}_j, \mathbf{e}_k) = 0$ when $j \neq k$.

■ EXAMPLE 1.19

Let S denote the set of functions $e^{\pi i kt/T}$, $k \in \mathbb{Z}$, in the vector space $C[-T, T]$, with the inner product as defined in Example 1.15. The set S is orthogonal, for if $k \neq m$, then

$$\begin{aligned} (e^{\pi i kt/T}, e^{\pi imt/T}) &= \int_{-T}^T e^{\pi i kt/T} \overline{e^{\pi imt/T}} dt \\ &= \int_{-T}^T e^{\pi i kt/T} e^{-\pi imt/T} dt \\ &= \int_{-T}^T e^{\pi i(k-m)t/T} dt \\ &= \frac{T(e^{\pi i(k-m)} - e^{-\pi i(k-m)})}{\pi i(k-m)} \\ &= 0, \end{aligned}$$

since $e^{\pi in} - e^{-\pi in} = 0$ for any integer n .

■ EXAMPLE 1.20

Let S denote the set of basic discrete waveforms $\mathbf{E}_{N,k} \in \mathbb{C}^N$, $-N/2 < k \leq N/2$, as defined in equation (1.22). With the inner product on \mathbb{C}^N as defined in

Example 1.13, but on the index range 0 to $N - 1$ instead of 1 to N , the set S is orthogonal. The proof is based on the surprisingly versatile algebraic identity

$$1 + z + z^2 + \cdots + z^{N-1} = \frac{1 - z^N}{1 - z} \quad \text{if } z \neq 1. \quad (1.31)$$

If $k \neq l$, then

$$\begin{aligned} (\mathbf{E}_k, \mathbf{E}_l) &= \sum_{r=0}^{N-1} e^{2\pi i kr/N} \overline{e^{2\pi i lr/N}} \\ &= \sum_{r=0}^{N-1} e^{2\pi i kr/N} e^{-2\pi i lr/N} \\ &= \sum_{r=0}^{N-1} e^{2\pi i(k-l)r/N} \\ &= \sum_{r=0}^{N-1} (e^{2\pi i(k-l)/N})^r. \end{aligned} \quad (1.32)$$

Let $z = e^{2\pi i(k-l)/N}$ in (1.31) and equation (1.32) becomes

$$\begin{aligned} (\mathbf{E}_k, \mathbf{E}_l) &= \frac{1 - (e^{2\pi i(k-l)/N})^N}{1 - e^{2\pi i(k-l)/N}} \\ &= \frac{1 - e^{2\pi i(k-l)}}{1 - e^{2\pi i(k-l)/N}} \\ &= 0, \end{aligned}$$

since $e^{2\pi i(k-l)} = 1$. Moreover the denominator above cannot equal zero for $-N/2 < k, l \leq N/2$ if $k \neq l$. Note the similarity of this computation to the computation in Example 1.19.

Remark 1.10 A very similar computation to that of Example 1.20 shows that the waveforms or matrices $\mathcal{E}_{m,n,k,l}$ in $M_{m,n}(\mathbb{C})$ (with the inner product from Example 1.14 but indexing from 0) are also orthogonal.

1.8.4 The Cauchy–Schwarz Inequality

The following inequality will be extremely useful. It has wide-ranging application and is one of the most famous inequalities in mathematics.

Theorem 1.8.1 (Cauchy–Schwarz) *For any vectors \mathbf{v} and \mathbf{w} in an inner product space V over \mathbb{C} or \mathbb{R} ,*

$$|(\mathbf{v}, \mathbf{w})| \leq \|\mathbf{v}\| \|\mathbf{w}\|,$$

where $\|\cdot\|$ is the norm induced by the inner product via equation (1.28).

Proof Note that $0 \leq (\mathbf{v} - c\mathbf{w}, \mathbf{v} - c\mathbf{w})$ for any scalar c , from property 3 for inner products. If we expand this out by using the properties of the inner product (including equation (1.27)), we find that

$$\begin{aligned} 0 &\leq (\mathbf{v} - c\mathbf{w}, \mathbf{v} - c\mathbf{w}) \\ &= (\mathbf{v}, \mathbf{v}) - c(\mathbf{w}, \mathbf{v}) - \bar{c}(\mathbf{v}, \mathbf{w}) + (c\mathbf{w}, c\mathbf{w}) \\ &= \|\mathbf{v}\|^2 - c(\mathbf{w}, \mathbf{v}) - \bar{c}(\mathbf{v}, \mathbf{w}) + |c|^2 \|\mathbf{w}\|^2. \end{aligned}$$

Let us suppose that $\mathbf{w} \neq \mathbf{0}$, for otherwise, Cauchy–Schwarz is obvious. Choose $c = (\mathbf{v}, \mathbf{w})/(\mathbf{w}, \mathbf{w})$ so that $\bar{c} = (\mathbf{w}, \mathbf{v})/(\mathbf{w}, \mathbf{w})$; note that (\mathbf{w}, \mathbf{w}) is real and positive. Then $c(\mathbf{w}, \mathbf{v}) = \bar{c}(\mathbf{v}, \mathbf{w}) = |(\mathbf{v}, \mathbf{w})|^2/\|\mathbf{w}\|^2$ and

$$0 \leq \|\mathbf{v}\|^2 - 2 \frac{|(\mathbf{v}, \mathbf{w})|^2}{\|\mathbf{w}\|^2} + \frac{|(\mathbf{v}, \mathbf{w})|^2}{\|\mathbf{w}\|^2} = \|\mathbf{v}\|^2 - \frac{|(\mathbf{v}, \mathbf{w})|^2}{\|\mathbf{w}\|^2}$$

from which the Cauchy–Schwarz inequality follows. ■

1.8.5 Bases and Orthogonal Decomposition

Bases Recall that the set of vectors $S = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ in \mathbb{R}^n is called the *standard basis*. The reason is that any vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ in \mathbb{R}^n can be written as a linear combination

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \cdots + x_n \mathbf{e}_n$$

of elements from S , in one and only one way. The \mathbf{e}_k thus form a convenient set of building blocks for \mathbb{R}^n . The set is “minimal” in the sense that any vector \mathbf{x} can be constructed from the \mathbf{e}_k in only one way.

The following concepts may be familiar from elementary linear algebra in \mathbb{R}^N , but they are useful in any vector space.

Definition 1.8.4 *A set S (finite or infinite) in a vector space V over \mathbb{C} (resp., \mathbb{R}) is said to “span V ” if every vector $\mathbf{v} \in V$ can be constructed as a finite linear combination of elements of S ,*

$$\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_n \mathbf{v}_n,$$

for suitable scalars α_k in \mathbb{C} (resp., \mathbb{R}) and vectors $\mathbf{v}_k \in S$.

Definition 1.8.5 A set S (finite or infinite) in a vector space V over \mathbb{C} (resp., \mathbb{R}) is said to be “linearly independent” if, for any finite set of vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ in S , the only solution to

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n = \mathbf{0}$$

is $\alpha_k = 0$ for all $1 \leq k \leq n$.

A set S that spans V is thus sufficient to build any vector in V by superposition. Linear independence ensures that no vector can be built in more than one way. A set S that both spans V and is linearly independent is especially useful, for each vector in V can be built from elements of S in a unique way.

Definition 1.8.6 A linearly independent set S that spans a vector space V is called a basis for V .

Be careful: a basis S for V may have infinitely many elements, but according to the definition above we must be able to construct any *specific* vector in V using only a *finite* linear combination of vectors in S . The word “basis” has a variety of meanings in mathematics, and the more accurate term for the type of basis defined above, in which only finite combinations are allowed, is a “Hamel basis.” If infinite linear combinations of basis vectors are allowed (as in Section 1.10), then issues concerning limits and convergence arise. In either case, however, we’ll continue to use the term “basis,” and no confusion should arise.

It’s worth noting that no linearly independent set and hence no basis can contain the zero vector.

The standard basis in \mathbb{R}^n or \mathbb{C}^n , of course, provides an example of a basis. Here are a couple slightly more interesting examples.

■ EXAMPLE 1.21

Consider the space $M_{m,n}(\mathbb{C})$ of $m \times n$ complex matrices, and define mn distinct elements $\mathbf{A}_{p,q} \in M_{m,n}(\mathbb{C})$ as follows: let the row p , column q entry of $\mathbf{A}_{p,q}$ equal 1, and set all other entries of $\mathbf{A}_{p,q}$ equal to zero (quite analogous to the standard basis of \mathbb{R}^n or \mathbb{C}^n). The set $S = \{\mathbf{A}_{p,q}; 1 \leq p \leq m, 1 \leq q \leq n\}$ forms a basis for $M_{m,n}(\mathbb{C})$.

■ EXAMPLE 1.22

Let P denote the vector space consisting of all polynomials in the variable x ,

$$p(x) = a_0 + a_1x + \cdots + a_nx^n$$

with real coefficients a_k and no condition on the degree n . You should convince yourself that this is indeed a vector space over \mathbb{R} , with the obvious operations. And

note that a polynomial has a highest degree term; we're not allowing expressions like $1 + x + x^2 + \dots$, that is, power series. One basis for P is given by the infinite set

$$S = \{1, x, x^2, x^3, \dots\}$$

It's not hard to see that any polynomial can be expressed as a finite linear combination of elements of S , and in only one way.

A vector space typically has many different bases. In fact each vector space that will interest us in this text has infinitely many different bases. Which basis we use depends on what we're trying to do.

If a vector space V has a finite basis $S = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, then V is said to be *finite-dimensional*. It is a fact from elementary linear algebra that any other basis for V must also contain exactly n vectors. In this case V is called an n -dimensional vector space. In light of the remarks above we can see that \mathbb{R}^n really is an n -dimensional vector space over \mathbb{R} (surprise!), while \mathbb{C}^n is n -dimensional over \mathbb{C} . Based on Example 1.21 the spaces $M_{m,n}(\mathbb{R})$ and $M_{m,n}(\mathbb{C})$ are both mn -dimensional vector spaces over \mathbb{R} or \mathbb{C} , respectively. The space P in Example 1.22 is infinite-dimensional.

Orthogonal and Orthonormal Bases A lot of vector algebra becomes ridiculously easy when the vectors involved are orthogonal. In particular, finding an orthogonal set of basis vectors for a vector space V can greatly aid analysis and facilitate certain computations.

One very useful observation is the following theorem.

Theorem 1.8.2 *If a set $S \subset V$ of non-zero vectors is orthogonal then S is linearly independent.*

Proof Consider the equation

$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_n \mathbf{v}_n = \mathbf{0},$$

where the \mathbf{v}_k are elements of S . Form the inner product of both sides above with any one of the vectors \mathbf{v}_m , $1 \leq m \leq n$, and use the linearity of the inner product in the first variable to obtain

$$\sum_{k=1}^n \alpha_k (\mathbf{v}_k, \mathbf{v}_m) = (\mathbf{0}, \mathbf{v}_m).$$

The right side above is, of course, 0. Since S is orthogonal, $(\mathbf{v}_k, \mathbf{v}_m) = 0$ unless $k = m$, so the equation above degenerates to $\alpha_m (\mathbf{v}_m, \mathbf{v}_m) = 0$. Because $(\mathbf{v}_m, \mathbf{v}_m) > 0$ (each \mathbf{v}_m is nonzero by hypothesis), we obtain $\alpha_m = 0$ for $1 \leq m \leq n$. ■

In the remainder of this section we assume that V is a finite-dimensional vector space over either \mathbb{R} or \mathbb{C} . Of special interest are bases for V that are orthogonal so that $(\mathbf{v}_k, \mathbf{v}_m) = 0$ for any two distinct basis vectors. In this case it's easy to explicitly write any $\mathbf{v} \in V$ as a linear combination of basis vectors.

Theorem 1.8.3 *Let $S = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ be an orthogonal basis for a vector space V . Then any $\mathbf{v} \in V$ can be expressed as*

$$\mathbf{v} = \sum_{k=1}^n \alpha_k \mathbf{v}_k, \quad (1.33)$$

where $\alpha_k = (\mathbf{v}, \mathbf{v}_k)/(\mathbf{v}_k, \mathbf{v}_k)$.

Proof This proof is very similar to that of Theorem 1.8.2. First, since S is a basis, there is some set of α_k that work in equation (1.33). Form the inner product of both sides of equation (1.33) with any \mathbf{v}_m , $1 \leq m \leq n$, and use the linearity of the inner product in the first variable to obtain

$$(\mathbf{v}, \mathbf{v}_m) = \sum_{k=1}^n \alpha_k (\mathbf{v}_k, \mathbf{v}_m).$$

Since S is orthogonal $(\mathbf{v}_k, \mathbf{v}_m) = 0$ unless $k = m$, in which case the equation above becomes $(\mathbf{v}, \mathbf{v}_m) = \alpha_m (\mathbf{v}_m, \mathbf{v}_m)$. Thus $\alpha_m = (\mathbf{v}, \mathbf{v}_m)/(\mathbf{v}_m, \mathbf{v}_m)$ is uniquely determined. The denominator $(\mathbf{v}_m, \mathbf{v}_m)$ cannot be zero, since \mathbf{v}_m cannot be the zero vector (because \mathbf{v}_m is part of a linearly independent set). ■

Definition 1.8.7 *An orthogonal set S is orthonormal if $\|\mathbf{v}\| = 1$ for each $\mathbf{v} \in S$.*

In the case where a basis S for V forms an orthonormal set, the expansion in Theorem 1.8.3 becomes a bit simpler since $(\mathbf{v}_k, \mathbf{v}_k) = \|\mathbf{v}_k\|^2 = 1$, so we can take $\alpha_k = (\mathbf{v}, \mathbf{v}_k)$ in equation (1.33).

Remark 1.11 Any orthogonal basis S can be replaced by a “re-scaled” basis that is orthonormal. Specifically, if S is an orthogonal basis for a vector space V , let S' denote the set obtained by replacing each vector $\mathbf{x} \in S$ by the re-scaled vector $\mathbf{x}' = \mathbf{x}/\|\mathbf{x}\|$ of length 1. If a vector $\mathbf{v} \in V$ can be expanded according to Theorem 1.8.3, then \mathbf{v} can also be written as a superposition of elements of S' , as

$$\mathbf{v} = \sum_{k=1}^n (\alpha_k \|\mathbf{v}_k\|) \mathbf{v}'_k,$$

where $\mathbf{v}'_k = \mathbf{v}_k/\|\mathbf{v}_k\|$ has norm one.

■ EXAMPLE 1.23

The standard basis for \mathbb{C}^N certainly has its place, but for many types of analysis the basic waveforms $\mathbf{E}_{N,k}$ are often more useful. In fact they also form an orthogonal basis for \mathbb{C}^N . We've already shown this to be true when in Example 1.20 we showed that the N vectors $\mathbf{E}_{N,k}$ for $0 \leq k \leq N - 1$ (or $-N/2 < k \leq N/2$) are mutually orthogonal. By Theorem 1.8.2, the vectors are necessarily linearly independent, and since a set of N linearly independent vectors in \mathbb{C}^N must span \mathbb{C}^N , the $\mathbf{E}_{N,k}$ form a basis for \mathbb{C}^N .

From equation 1.33 we then obtain a simple decomposition formula

$$\begin{aligned}\mathbf{x} &= \sum_{k=0}^{N-1} \frac{(\mathbf{x}, \mathbf{E}_{N,k})}{(\mathbf{E}_{N,k}, \mathbf{E}_{N,k})} \mathbf{E}_{N,k} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} (\mathbf{x}, \mathbf{E}_{N,k}) \mathbf{E}_{N,k}\end{aligned}\quad (1.34)$$

for any vector $\mathbf{x} \in \mathbb{C}^N$, where we've made use of $(\mathbf{E}_{N,k}, \mathbf{E}_{N,k}) = N$ for each k (see Exercise 1.30). Equation (1.34) will be of paramount importance later; indeed most of Chapter 3 is devoted to the study of equation (1.34)!

■ EXAMPLE 1.24

An entirely analogous argument shows that the matrices $\mathcal{E}_{m,n,k,l}$ form a basis for $M_{m,n}(\mathbb{C})$, and for any matrix $\mathbf{A} \in M_{m,n}(\mathbb{C})$

$$\begin{aligned}\mathbf{A} &= \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} \frac{(\mathbf{A}, \mathcal{E}_{m,n,k,l})}{(\mathcal{E}_{m,n,k,l}, \mathcal{E}_{m,n,k,l})} \mathcal{E}_{m,n,k,l} \\ &= \frac{1}{mn} \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} (\mathbf{A}, \mathcal{E}_{m,n,k,l}) \mathcal{E}_{m,n,k,l},\end{aligned}\quad (1.35)$$

where we've made use of $(\mathcal{E}_{m,n,k,l}, \mathcal{E}_{m,n,k,l}) = mn$ (see Exercise 1.31).

Parseval's Identity Suppose that S is an orthonormal basis for an n -dimensional vector space V . Let $\mathbf{v} \in V$ be expanded according to equation (1.33). Then

$$\begin{aligned}\|\mathbf{v}\|^2 &= (\mathbf{v}, \mathbf{v}) \\ &= \left(\sum_j \alpha_j \mathbf{v}_j, \sum_k \alpha_k \mathbf{v}_k \right)\end{aligned}$$

$$\begin{aligned}
 &= \sum_{j,k=1}^n \alpha_j \overline{\alpha_k} (\mathbf{v}_j, \mathbf{v}_k) \\
 &= \sum_{k=1}^n |\alpha_k|^2,
 \end{aligned} \tag{1.36}$$

where we have used the properties of the inner product (including equation (1.27)), $\alpha_k \overline{\alpha_k} = |\alpha_k|^2$, and the fact that S is orthonormal so $(\mathbf{v}_k, \mathbf{v}_k) = 1$. Equation (1.36) is called *Parseval's identity*.

As noted in Remark 1.9 on page 30, $\|\mathbf{v}\|^2$ is often interpreted as the energy of the discretized signal. If the basis set S is orthonormal, then each vector \mathbf{v}_k represents a basis signal that is scaled to have energy equal to 1, and it's easy to see that the signal $\alpha_k \mathbf{v}_k$ has energy $|\alpha_k|^2$. In this case Parseval's identity asserts that the “energy of the sum is the sum of the energies.”

1.9 SIGNAL AND IMAGE DIGITIZATION

As discussed earlier in the chapter, general analog signals and images cannot be meaningfully stored in a computer but must be converted to digital form. As noted in Section 1.3.2, this introduces a quantization error. It's tempting to minimize this error by storing the underlying real numbers as high-precision floating point values, but this would be expensive in terms of storage. In Matlab a grayscale image stored as double precision floating point numbers requires eight bytes per pixel, compared to one byte per pixel for the eight-bit quantization scheme discussed Section 1.3.6. Furthermore, if very fast processing is required, it's usually better to use integer arithmetic chips than floating point hardware. Thus we must balance quantization error with the storage and computational costs associated with more accurate digitization. In order to better understand this issue, we now take a closer look at quantization and a more general scheme than that presented in Section 1.3.2.

1.9.1 Quantization and Dequantization

Let's start with a simple but representative example.

■ EXAMPLE 1.25

Consider an analog signal $x(t)$ that can assume “any” real value at any particular time t . The signal would, of course, be sampled to produce a string of real numbers x_k for k in some range, say $0 \leq k \leq n$. This still doesn't suffice for computer storage though, since we can't store even a single real number x_k to infinite precision. What we'll do is this: divide the real line into “bins,” say the disjoint intervals $(-\infty, -5]$, $(-5, 3]$, $(3, 7]$, and $(7, \infty)$. Note that these are chosen arbitrarily here, solely for the sake of example. We thus have four *quantization*

intervals. Every real number falls into exactly one of these intervals. We'll refer to the interval $(-\infty, -5]$ as “interval 0,” $(-5, 3]$ as “interval 1,” $(3, 7]$ as “interval 2,” and $(7, \infty)$ as “interval 3.” In this manner any real number z can be associated with an integer in the range 0 to 3, according to the quantization interval in which z lies. This defines a *quantization map* q from \mathbb{R} to the set $\{0, 1, 2, 3\}$. Rather than storing z we store (with some obvious loss of information) $q(z)$. Indeed, since $q(z)$ can assume only four distinct values, it can be stored with just two bits. We can store the entire discretized signal $x(t)$ with just $2(n + 1)$ bits, for example, as “00” for a sample x_k in interval 0, “01” for interval 1, “10” for interval 2, “11” for interval 3.

To reconstruct an approximation to any given sample x_k , we proceed as follows: for each quantization interval we choose a representative value z_k for quantization interval k . For example, we can take $z_0 = -10$, $z_1 = -1$, $z_2 = 5$, and $z_3 = 10$. Here z_1 and z_2 are chosen as the midpoints of the corresponding interval, z_0 and z_3 as “representative” of their intervals. If a sample x_k falls in quantization interval 0 (i.e., was stored as the bit sequence “00”), we reconstruct approximately as $\tilde{x}_k = z_0$. A similar computation is performed for the other intervals. This yields a *dequantization map* \tilde{q} from the set $\{0, 1, 2, 3\}$ back to \mathbb{R} .

As a specific example, consider the sampled signal $\mathbf{x} \in \mathbb{R}^5$ with components $x_0 = -1.2$, $x_1 = 2.3$, $x_2 = 4.4$, $x_3 = 8.8$, and $x_4 = -2.8$. The quantization map yields $q(\mathbf{x}) = (1, 1, 2, 3, 1)$ when q is applied component-by-component to \mathbf{x} . The reconstructed version of \mathbf{x} is $\tilde{q}(q(\mathbf{x})) = (-1, -1, 5, 10, -1)$.

The General Quantization Scheme The quantization scheme above generalizes as follows. Let r be the number of quantization levels ($r = 4$ in the previous example). Choose $r - 1$ distinct quantization “jump points” $\{y_1, \dots, y_{r-1}\}$, real numbers that satisfy $-\infty < y_1 < y_2 < \dots < y_{r-1} < \infty$ (we had $y_1 = -5$, $y_2 = 3$, $y_3 = 7$ above). Let $y_0 = -\infty$ and $y_r = \infty$. We call the interval $(y_k, y_{k+1}]$ the *kth quantization interval*, where $0 \leq k \leq r - 1$ (we should really use an open interval (y_{r-1}, ∞) for the last interval). The leftmost and rightmost intervals are unbounded. Each real number belongs to exactly one of the r quantization intervals.

The *quantization map* $q : \mathbb{R} \rightarrow \{0, 1, \dots, r - 1\}$ assigns an integer quantization level $q(x)$ to each $x \in \mathbb{R}$ as follows: $q(x)$ is the index k such that $y_k < x \leq y_{k+1}$, meaning, x belongs to the k th quantization interval. The function q can be written more explicitly if we define the Heaviside function

$$H(x) = \begin{cases} 0, & x \leq 0, \\ 1, & x > 0, \end{cases}$$

in which case

$$q(x) = \sum_{k=1}^{r-1} H(x - y_k).$$

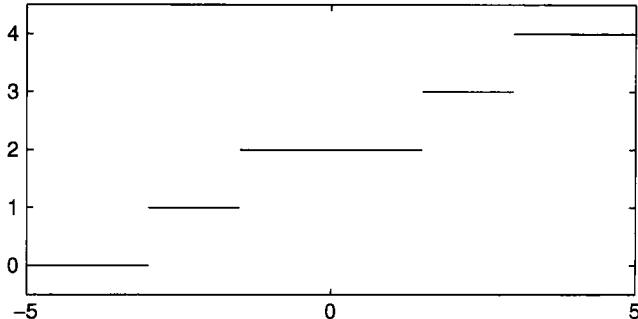


FIGURE 1.11 Quantization function $H(x + 3) + H(x + 1.5) + H(x - 1.5) + H(x - 3)$.

A sample graph of a quantization function is given in Figure 1.11, in which we have chosen $y_1 = -3$, $y_2 = -1.5$, $y_3 = 1.5$, $y_4 = 3$, and $y_0 = -\infty$, $y_5 = \infty$. Notice that the quantization interval around zero is bigger than the rest. This is not uncommon in practice.

The quantization map is used as follows: Let \mathbf{x} denote a sampled signal, so each component x_j of \mathbf{x} is a real number and has not yet been quantized. The quantized version of \mathbf{x} is just $q(\mathbf{x})$, in which q is applied component-by-component to \mathbf{x} . Each x_j is thus assigned to one of the r quantization intervals. If $r = 2^b$ for some $b > 0$, then each quantized x_j can be stored using b bits, and we refer to this as “ b -bit quantization.” A similar procedure would be applied to images.

Dequantization Once quantized, the vector \mathbf{x} cannot be exactly recovered because q is not invertible. If we need to approximately reconstruct \mathbf{x} after quantization we do the following: pick real numbers z_0, z_1, \dots, z_{r-1} such that z_k lies in the k th quantization interval, that is $y_{k-1} < z_{k-1} \leq y_k$. Ideally the value of z_k should be a good approximation to the average value of the entries of \mathbf{x} that fall in the k th quantization interval. A simple choice is to take z_k as the midpoint of the k th quantization interval, as in the example above. The set $\{z_k : 0 \leq k \leq r - 1\}$ is called the *codebook* and the z_k are called the *codewords*. Define the *dequantization map* $\tilde{q} : \{0, \dots, r - 1\} \rightarrow \mathbb{R}$ that takes k to z_k . Then define the approximate reconstruction of \mathbf{x} as the vector $\tilde{\mathbf{x}}$ with components \tilde{x}_j where

$$\tilde{x}_j = \tilde{q}(q(x_j)) = z_{q(x_j)}.$$

If $y_{k-1} < x_j \leq y_k$, then x_j is mapped to the k th quantization interval (i.e., $q(x_j) = k$) and $\tilde{x}_j = \tilde{q}(q(x_j)) = z_k$ where $y_{k-1} < z_k \leq y_k$. In other words, x_j and the quantized/dequantized quantity \tilde{x}_j both lie in the interval $y_{k-1} < x_j \leq y_k$, so at worst the discrepancy is

$$|x_j - \tilde{x}_j| \leq y_k - y_{k-1}. \quad (1.37)$$

If the y_k are finely spaced, the error won't be too large. But, of course, more y_k means more bits are needed for storage. The same procedure can be applied to images/matrices.

Measuring Error Ideally we'd like to choose our quantization/dequantization functions to minimize the distortion or error introduced by this process. This requires a way to quantify the distortion. One simple measure of the distortion is $\|\mathbf{x} - \tilde{\mathbf{x}}\|^2$, the squared distance between \mathbf{x} and $\tilde{\mathbf{x}}$ as vectors in \mathbb{R}^n (or \mathbb{C}^n). Actually it is slightly more useful to quantify distortion in relative terms, as a fraction of the original signal energy, so we use $(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)/\|\mathbf{x}\|^2$. By adjusting the values of the y_k 's and the z_k 's, we can, in principle, minimize distortion for any specific signal \mathbf{x} or image. In the case of image compression, the quantization levels $\{z_k\}$ can be stored with the compressed file. For other applications, we may want the quantization levels to be fixed ahead of time, say in an audio application. In this case, it's sensible to minimize the distortion over an entire class of signals or images. We would also like the quantization and dequantization computations to be simple. The following example gives a scheme that works fairly well.

■ EXAMPLE 1.26

In this example we'll quantize an image, but the same principles apply to a one-dimensional signal. Let us assume that the intensity values of a class of grayscale images of interest satisfy $m \leq a(x, y) \leq M$ on some rectangle Ω , where $a(x, y)$ is the analog image intensity. Let \mathbf{A} denote the matrix with components a_{jk} obtained by sampling (but not quantizing) the analog image $a(x, y)$. Select the y_k so that they split up the interval $[m, M]$ into r subintervals of equal length, and let z_k be the mid-point of each interval. If we define $h = (M - m)/r$, then we obtain the following formulas for the y_k 's, the z_k 's, q , and \tilde{q} :

$$y_k = m + kh, \quad k = 1, \dots, r - 1, \quad y_0 = -\infty, \quad y_r = \infty,$$

$$z_k = m + \left(k + \frac{1}{2} \right) h, \quad k = 0, \dots, r - 1,$$

$$q(x) = \text{ceil} \left(r \frac{x - m}{M - m} \right) - 1 \quad \text{for } x > m, \quad q(m) = 0,$$

$$\tilde{q}(k) = m + \left(\frac{k + 1}{2} \right) h.$$

The ceiling function "ceil" from \mathbb{R} to \mathbb{Z} is defined by taking $\text{ceil}(x)$ as the smallest integer greater than or equal to x .

To illustrate, the image at the top left in Figure 1.12 has a sample matrix \mathbf{A} (stored as double precision floating point) with limits $0 \leq a_{ij} \leq 255$. The quantization method above at 5 bits per pixel (32 quantization intervals) or greater gives no measurable distortion. In Figure 1.12 we illustrate quantization at each of $b = 4, 2, 1$ bits per pixel (bpp) in order to see the distortion. The measure of

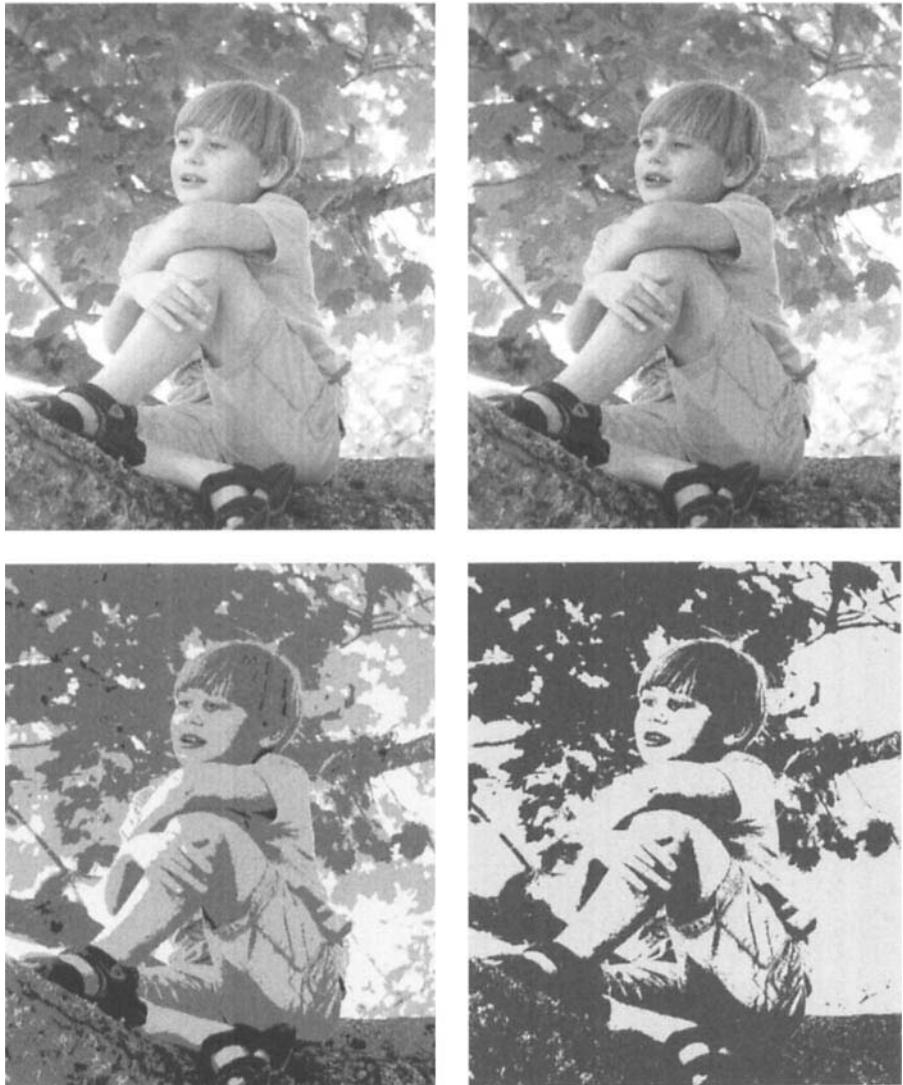


FIGURE 1.12 Original image (*top left*) and quantization at 4 bits (*top right*), 2 bits (*bottom left*) and 1 bit (*bottom right*).

the distortion mD is reported as a percentage of the total image energy,

$$mD = 100 \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|^2}{\|\mathbf{A}\|^2},$$

where $\|\cdot\|$ denotes the Frobenius norm of Example 1.14. The resulting errors are 0.2, 3.6, and 15.2 percent for the 4, 2, and 1 bit quantizations, respectively.

Example 1.26 illustrates uniform quantization with midpoint codewords. It also yields an improvement over the error estimate 1.37, namely

$$|a_{ij} - \tilde{a}_{ij}| \leq \frac{h}{2}. \quad (1.38)$$

1.9.2 Quantifying Signal and Image Distortion More Generally

Suppose that we have signal that we want to compress or denoise to produce a processed approximation. The quantization discussion above provides a concrete example, but other similar situations will arise later. In general, what is a reasonable way to quantify the accuracy of the approximation?

Many approaches are possible, but we'll do essentially as we did for the image in Example 1.26. For a discretized image (or signal) \mathbf{A} approximated by $\tilde{\mathbf{A}}$, write

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E},$$

where $\mathbf{E} = \tilde{\mathbf{A}} - \mathbf{A}$ is the error introduced by using the approximation $\tilde{\mathbf{A}}$ for \mathbf{A} . We could also consider \mathbf{E} as some kind of random noise. Our measure of distortion (or noise level, if appropriate) is

$$mD = \frac{\|\tilde{\mathbf{A}} - \mathbf{A}\|^2}{\|\mathbf{A}\|^2} = \frac{\|\mathbf{E}\|^2}{\|\mathbf{A}\|^2}, \quad (1.39)$$

the relative error as is typically done in any physical application. We will often report this error as a percentage as we did with the images above, by multiplying by 100.

1.10 INFINITE-DIMENSIONAL INNER PRODUCT SPACES

Analog signals and images are naturally modeled by functions of one or more real variables, and as such the proper setting for the analysis of these objects is a function space such as $C[a, b]$. However, vector spaces of functions are infinite-dimensional, and some of the techniques developed for finite-dimensional vector spaces need a bit of adjustment. In this section we give an outline of the mathematics necessary to carry out orthogonal expansions in these function spaces, especially orthogonal expansions with regard to complex exponentials. The ideas in this section play a huge role in applied mathematics. They also provide a nice parallel to the discrete ideas.

Example: An Infinite-dimensional Space Let's focus on the vector space $C[a, b]$ for the moment. This space is not finite-dimensional. This can be shown by demonstrating the existence of m linearly independent functions in $C[a, b]$ for any integer $m > 0$. To do this, let $h = (b - a)/m$ and set $x_k = a + (k - 1)h$ for $0 \leq k \leq m$; the points x_k partition $[a, b]$ into m equal subintervals, each of length h .

(with $x_0 = a$, $x_m = b$). Let $I_k = [x_{k-1}, x_k]$ for $1 \leq k \leq m$, and define m functions

$$\phi_k(x) = \begin{cases} 0, & \text{if } x \text{ is not in } I_k, \\ \frac{2}{h}(x - x_{k-1}), & x_{k-1} \leq x \leq (x_{k-1} + x_k)/2, \\ \frac{2}{h}(x_k - x), & (x_{k-1} + x_k)/2 < x \leq x_k, \end{cases}$$

for $1 \leq k \leq m$. Each function ϕ_k is continuous and piecewise linear, identically zero outside I_k , with $\phi_k = 1$ at the midpoint of I_k (such a function is sometimes called a “tent” function—draw a picture). It’s easy to show that the functions ϕ_k are linearly independent, for if

$$\sum_{k=1}^m c_k \phi_k(x) = 0$$

for $a \leq x \leq b$, then evaluating the left side above at the midpoint of I_j immediately yields $c_j = 0$. The ϕ_k thus form a linearly independent set. If $C[a, b]$ were finite-dimensional, say of dimension n , then we would not be able to find a set of $m > n$ linearly independent functions. Thus $C[a, b]$ is not finite-dimensional.

A similar argument can be used to show that any of the vector spaces of functions from Section 1.4.2 are infinite-dimensional.

1.10.1 Orthogonal Bases in Inner Product Spaces

It can be shown that any vector space, even an infinite-dimensional space, has a basis in the sense of Definition 1.8.6 (a Hamel basis), in which only finite combinations of the basis vectors are allowed. However, such bases are usually difficult to exhibit explicitly and of little use for computation. As such, we’re going to expand our notion of basis to allow infinite combinations of the basis vectors. Of course, the word “infinite” always means limits are involved, and if so, we need some measure of distance or length, since limits involve some quantity “getting close” to another.

In light of this criterion it’s helpful to restrict our attention to vector spaces in which we have some notion of distance, such as a normed vector space. But orthogonality, especially in the infinite-dimensional case, is such a valuable asset that we’re going to restrict our attention to inner product spaces. The norm will be that associated with the inner product via equation (1.28).

Let V be an inner product space, over either \mathbb{R} or \mathbb{C} . We seek a set S of vectors in V ,

$$S = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots\}, \tag{1.40}$$

that can act as a basis for V in some reasonable sense. Note that we are assuming that the elements of S can be listed, meaning put into a one-to-one correspondence with the positive integers. Such a set is said to be *countable* (many infinite sets are not!).

Remark 1.12 In what follows it's not important that the elements of the set S be indexed as 1, 2, 3, Indeed the elements can be indexed from any subset of \mathbb{Z} , \mathbb{Z} itself, or even $\mathbb{Z} \times \mathbb{Z}$. The main point is that we must be able to sum over the elements of S using traditional summation notation, \sum . We index the elements starting from 1 in the discussion that follows solely to fix notation.

In the finite-dimensional case the basis vectors must be linearly independent, and this was an automatic consequence of orthogonality. In the infinite-dimensional case we'll cut straight to the chase: in this text we will only consider as prospective bases those sets that are orthogonal (though nonorthogonal bases can be constructed).

In the finite-dimensional case, a basis must also span the vector space. In the present case, we are allowing infinite linear combinations of basis vectors, and want to be able to write

$$\mathbf{v} = \sum_{k=1}^{\infty} \alpha_k \mathbf{v}_k \quad (1.41)$$

for any $\mathbf{v} \in V$ by choosing the coefficients α_k suitably. But infinite linear combinations have no meaning in a general vector space. How should equation (1.41) be interpreted?

Recall that in elementary calculus the precise definition of an infinite sum

$$\sum_{k=1}^{\infty} a_k = A$$

is that $\lim_{n \rightarrow \infty} (\sum_{k=1}^n a_k) = A$ (the sequence of partial sums converges to A). This is equivalent to

$$\lim_{n \rightarrow \infty} \left| \sum_{k=1}^n a_k - A \right| = 0.$$

This motivates our interpretation of equation (1.41) and the definition of what it means for the set S in (1.40) to span V . However, in an infinite-dimensional inner product space the term “span” is replaced by “complete.”

Definition 1.10.1 An (orthogonal) set S as in (1.40) is “complete” if for each $\mathbf{v} \in V$ there are scalars α_k , $k \geq 1$, such that

$$\lim_{n \rightarrow \infty} \left\| \sum_{k=1}^n \alpha_k \mathbf{v}_k - \mathbf{v} \right\| = 0. \quad (1.42)$$

The limit in (1.42) is just an ordinary limit for a sequence of real numbers. The norm on the inner product space takes the place of absolute value in \mathbb{R} .

We can now define

Definition 1.10.2 A set S as in (1.40) is called an “orthogonal basis” for V if S is complete and orthogonal. If S is orthonormal, then S is called an “orthonormal basis.”

The existence of an orthogonal basis as in Definition 1.10.2 is not assured but depends on the particular inner product space. However, all of the function spaces of interest from Section 1.8.2 have such bases. In the case of spaces consisting of functions of a single real variable (e.g., $C[a, b]$), we can write out a basis explicitly, and also for function spaces defined on a rectangle in the plane. We’ll do this shortly.

1.10.2 The Cauchy–Schwarz Inequality and Orthogonal Expansions

For now let’s assume that an orthogonal basis $S = \{\mathbf{v}_1, \mathbf{v}_2, \dots\}$ exists. How can we compute the α_k in the expansion (1.41)? Are they uniquely determined? It’s tempting to mimic the procedure used in the finite-dimensional case: take the inner product of both sides of (1.41) with a specific basis vector \mathbf{v}_m to obtain $(\mathbf{v}, \mathbf{v}_m) = (\sum_k \alpha_k \mathbf{v}_k, \mathbf{v}_m)$ then use linearity of the inner product in the first argument to obtain $(\mathbf{v}, \mathbf{v}_m) = \sum_k \alpha_k (\mathbf{v}_k, \mathbf{v}_m) = \alpha_m (\mathbf{v}_m, \mathbf{v}_m)$. This immediately yields $\alpha_m = (\mathbf{v}, \mathbf{v}_m) / (\mathbf{v}_m, \mathbf{v}_m)$, just as in the finite-dimensional case. This reasoning is a bit suspect, though, because it requires us to invoke linearity for the inner product with respect to an infinite sum. Unfortunately, the definition of the inner product makes no statements concerning infinite sums. We need to be a bit more careful (though the answer for α_m is correct!).

To demonstrate the validity of the conclusion above more carefully we’ll use the Cauchy–Schwarz inequality in Theorem 1.8.1. Specifically, suppose that $S = \{\mathbf{v}_1, \mathbf{v}_2, \dots\}$ is an orthogonal basis so that for any $\mathbf{v} \in V$ there is some choice of scalars α_k for which equation (1.42) holds. For some fixed m consider the inner product $(\mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k, \mathbf{v}_m)$. If we expand this inner product and suppose $n \geq m$ while taking absolute values throughout, we find that

$$\begin{aligned} \left| \left(\mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k, \mathbf{v}_m \right) \right| &= \left| (\mathbf{v}, \mathbf{v}_m) - \sum_{k=1}^n \alpha_k (\mathbf{v}_k, \mathbf{v}_m) \right| \\ &= |(\mathbf{v}, \mathbf{v}_m) - \alpha_m (\mathbf{v}_m, \mathbf{v}_m)|. \end{aligned} \quad (1.43)$$

Note that all sums above are finite. On the other hand, the Cauchy–Schwarz inequality yields

$$\left| \left(\mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k, \mathbf{v}_m \right) \right| \leq \| \mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k \| \| \mathbf{v}_m \| . \quad (1.44)$$

Combine (1.43) and (1.44) to find that for $n \geq m$

$$|(\mathbf{v}, \mathbf{v}_m) - \alpha_m (\mathbf{v}_m, \mathbf{v}_m)| \leq \left\| \mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k \right\| \| \mathbf{v}_m \| . \quad (1.45)$$

The completeness assumption forces $\|\mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k\| \rightarrow 0$ as $n \rightarrow \infty$ on the right side of (1.45) (and $\|\mathbf{v}_m\|$ remains fixed, since m is fixed). The left side of (1.45) must also approach zero, but the left side doesn't depend on n . We must conclude that $(\mathbf{v}, \mathbf{v}_m) - \alpha_m (\mathbf{v}_m, \mathbf{v}_m) = 0$, so $\alpha_m = (\mathbf{v}, \mathbf{v}_m)/(\mathbf{v}_m, \mathbf{v}_m)$. Of course, if S is orthonormal, this becomes just $\alpha_m = (\mathbf{v}, \mathbf{v}_m)$.

Note that we assumed an expansion as in (1.42) exists. What we've shown above is that IF such an expansion exists (i.e., if S is complete), THEN the α_k are uniquely determined and given by the formula derived. Let's state this as a theorem.

Theorem 1.10.1 *If S is an orthogonal basis for an inner product space V , then for any $\mathbf{v} \in V$, equation (1.42) holds where $\alpha_k = (\mathbf{v}, \mathbf{v}_k)/(\mathbf{v}_k, \mathbf{v}_k)$.*

It's conventional to write the expansion of \mathbf{v} in the shorthand form of (1.41), with the understanding that the precise meaning is the limit in (1.42).

Interestingly, Parseval's identity still holds. Note that if S is an orthonormal basis and $\alpha_k = (\mathbf{v}, \mathbf{v}_k)$, then

$$\begin{aligned} \left\| \mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k \right\|^2 &= \left(\mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k, \mathbf{v} - \sum_{k=1}^n \alpha_k \mathbf{v}_k \right) \\ &= (\mathbf{v}, \mathbf{v}) - \sum_{k=1}^n \alpha_k (\mathbf{v}_k, \mathbf{v}) - \sum_{k=1}^n \overline{\alpha_k} (\mathbf{v}, \mathbf{v}_k) + \sum_{k=1}^n |\alpha_k|^2 \\ &= \|\mathbf{v}\|^2 - \sum_{k=1}^n |\alpha_k|^2. \end{aligned}$$

As $n \rightarrow \infty$ the left side approaches zero; hence so does the right side, and we obtain

$$\sum_{k=1}^{\infty} |\alpha_k|^2 = \|\mathbf{v}\|^2, \quad (1.46)$$

which is Parseval's identity in the infinite-dimensional case.

There are a variety of equivalent characterizations or definitions of what it means for a set S to be complete (e.g., S is complete if Parseval's identity holds for all \mathbf{v}), and some may make it easier or harder to verify that any given set S is complete. Proving that a given set of vectors in a specific inner product space is complete always involves some analysis ("analysis" in the sense of limits, inequalities, and estimates). We won't go into these issues in much detail in this text. We will simply exhibit, without proof, some orthogonal bases for common spaces of interest.

1.10.3 The Basic Waveforms and Fourier Series

For the moment let's focus on the space $C[-T, T]$ of continuous complex-valued functions on the closed interval $[-T, T]$, where $T > 0$. This, of course, includes the real-valued functions on the interval. The vector space $C[-T, T]$ becomes an inner product space if we define the inner product as in Example 1.15.

Complex Exponential Fourier Series Let S denote the set of basic analog waveforms $\phi_k(t) = e^{ik\pi t/T}$ for $k \in \mathbb{Z}$ introduced in Example 1.19. In that example it was shown that this set is orthogonal. It can be shown with a bit of nontrivial analysis that this set is complete in $C[-T, T]$ or $L^2(-T, T)$ (see, e.g., [21] or [10], sec. II.4.4). As a consequence S is a basis.

Note that the basis ϕ_k here is indexed in k , which ranges over all of \mathbb{Z} , but according to Remark 1.12 on page 48 this makes no essential difference.

Consider a typical function $f(t)$ in $C[-T, T]$. From Theorem 1.10.1 we have the expansion

$$f(t) = \sum_{k=-\infty}^{\infty} \alpha_k e^{ik\pi t/T}, \quad (1.47)$$

where

$$\alpha_k = \frac{(f, e^{ik\pi t/T})}{(e^{ik\pi t/T}, e^{ik\pi t/T})} = \frac{1}{2T} \int_{-T}^T f(t) e^{-ik\pi t/T} dt, \quad (1.48)$$

since $\int_{-T}^T e^{ik\pi t/T} e^{-ik\pi t/T} dt = 2T$. The right side of equation (1.47) is called the *Fourier series* for f , and the α_k of equation (1.48) are called the *Fourier coefficients*.

■ EXAMPLE 1.27

Let $T = 1$, and consider the function $f(t) = t^2$ in $C[-1, 1]$. The Fourier coefficients from equation (1.48) are given by

$$\begin{aligned} \alpha_k &= \frac{1}{2} \int_{-1}^1 t^2 e^{-ik\pi t} dt \\ &= \frac{2(-1)^k}{\pi^2 k^2} \end{aligned}$$

for $k \neq 0$ (integrate by parts twice), while $\alpha_0 = \frac{1}{3}$. We can write out the Fourier series for this function as

$$f(t) = \frac{1}{3} + \frac{2}{\pi^2} \sum_{k=-\infty, k \neq 0}^{\infty} \frac{(-1)^k e^{ik\pi t}}{k^2}.$$

It should be emphasized that the series on the right “equals” $f(t)$ only in the sense defined by (1.42). As such, it is instructive to plot $f(t)$ and the right side above summed from $k = -n$ to n for a few values of n , as in Figure 1.13. In the case where $n = 0$ the approximation is just $f(t) \approx a_0 = (\int_{-T}^T f(t) dt)/2T$, the average value of f over the interval. In general, there is no guarantee that the Fourier series for any specific value $t = t_0$ converges to the value $f(t_0)$, unless one knows something more about f . However, the Fourier series will converge in the sense of equation (1.42) for any function $f \in L^2(-T, T)$.

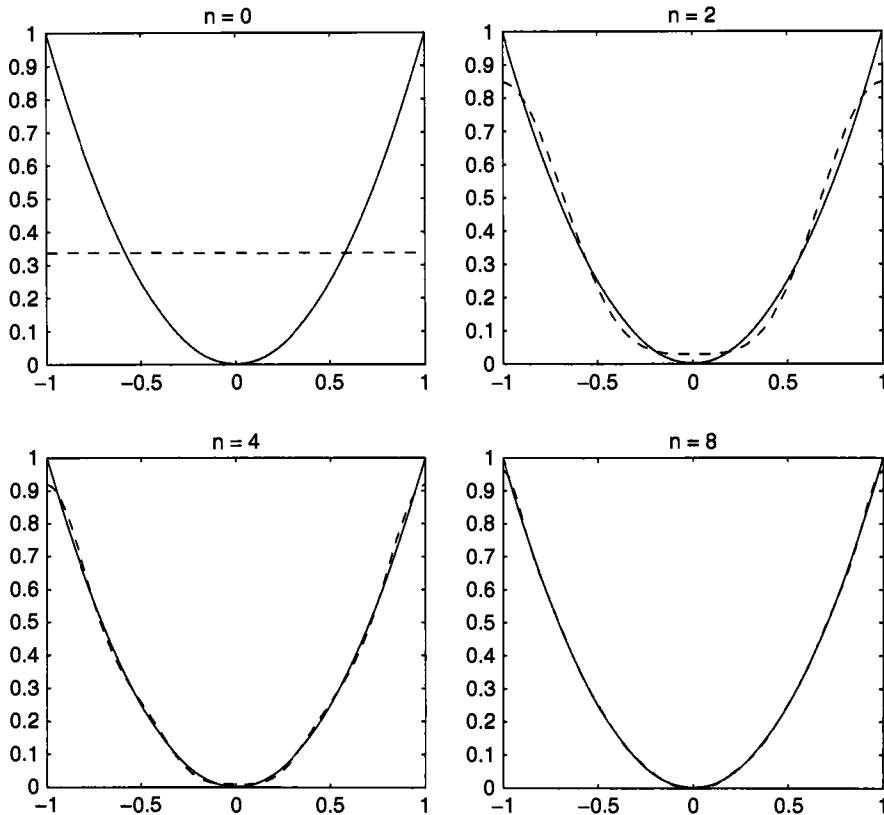


FIGURE 1.13 Function $f(t) = t^2$ (solid) and Fourier series approximations (dashed), $n = 0, 2, 4, 8$.

It's also worth noting that the Fourier series itself continues periodically outside the interval $[-T, T]$ with period $2T$, since each of the basic waveforms are periodic with period $2T$. This occurs even though f does not need to be defined outside this interval. This case is shown in Figure 1.14.

Sines and Cosines Fourier expansions can be also written using sines and cosines. Specifically, we can write the Fourier coefficients of a function $f(t)$ on $[-T, T]$ as

$$\begin{aligned}
 \alpha_k &= \frac{1}{2T} \int_{-T}^T f(t) e^{-ik\pi t/T} dt \\
 &= \frac{1}{2T} \int_{-T}^T f(t) \left(\cos\left(\frac{k\pi t}{T}\right) - i \sin\left(\frac{k\pi t}{T}\right) \right) dt \\
 &= a_k - i b_k
 \end{aligned} \tag{1.49}$$

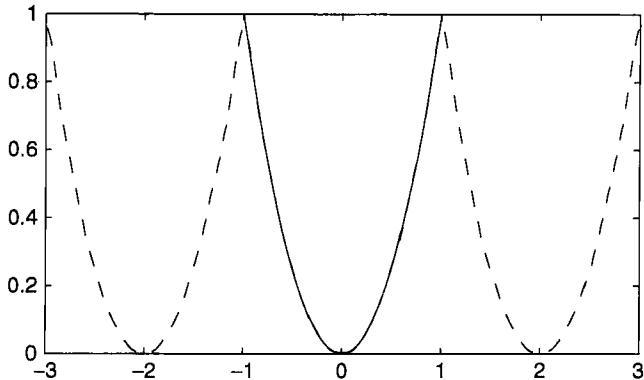


FIGURE 1.14 Function $f(t) = t^2$, $-1 \leq t \leq 1$, and eight-term Fourier series approximation extended to $[-3, 3]$.

where

$$a_k = \frac{1}{2T} \int_{-T}^T f(t) \cos\left(\frac{k\pi t}{T}\right) dt, \quad b_k = \frac{1}{2T} \int_{-T}^T f(t) \sin\left(\frac{k\pi t}{T}\right) dt. \quad (1.50)$$

Observe that $b_0 = 0$, while $a_0 = \alpha_0$ and $\alpha_{-k} = a_k + i b_k$. Let us rewrite the complex exponential Fourier series as

$$\begin{aligned} f(t) &= \sum_{k=-\infty}^{\infty} \alpha_k e^{ik\pi t/T} \\ &= \alpha_0 + \sum_{k=1}^{\infty} (\alpha_k e^{ik\pi t/T} + \alpha_{-k} e^{-ik\pi t/T}) \\ &= \alpha_0 + \sum_{k=1}^{\infty} ((a_k - i b_k) e^{ik\pi t/T} + (a_k + i b_k) e^{-ik\pi t/T}) \\ &= \alpha_0 + \sum_{k=1}^{\infty} (a_k (e^{ik\pi t/T} + e^{-ik\pi t/T}) - i b_k (e^{ik\pi t/T} - e^{-ik\pi t/T})) \\ &= \alpha_0 + 2 \sum_{k=1}^{\infty} (a_k \cos(k\pi t/T) + b_k \sin(k\pi t/T)) \end{aligned} \quad (1.51)$$

where we've made use of equation (1.13). If f is real-valued, then the Fourier series with respect to the sine/cosine basis can be computed from (1.50) and (1.51) with no reference to complex numbers.

Fourier expansions can be performed on any interval $[a, b]$, by translating and scaling the functions $e^{i\pi kt/T}$ appropriately. The linear mapping $\phi : t \rightarrow T(2t - (a + b))/(b - a)$ takes the interval $[a, b]$ to $[-T, T]$. As a consequence we can check that the functions

$$e^{i\pi k\phi(t)/T} = e^{ik\pi(2t-(a+b))/(b-a)}$$

are orthogonal on $[a, b]$ with respect to the $L^2(a, b)$ inner product, and so form a basis.

Fourier Series on Rectangles The set of functions $\phi_{k,m}(x, y)$, $k \in \mathbb{Z}$, and $m \in \mathbb{Z}$ defined by

$$\phi_{k,m}(x, y) = e^{2\pi i(kx/a+my/b)}$$

forms an orthogonal basis for the set $C(\Omega)$ where $\Omega = \{(x, y); 0 \leq x \leq a, 0 \leq y \leq b\}$. As in the one-dimensional case, we can also write out an equivalent basis of sines and cosines.

Orthogonal bases can be proved to exist for more general two-dimensional regions, or regions in higher dimensions, but they cannot usually be written out explicitly.

1.10.4 Hilbert Spaces and $L^2(a, b)$

The inner product space of continuous, square-integrable functions on an interval (a, b) was defined in Example 1.15. This space is satisfactory for some purposes, but in many cases it's helpful to enlarge the space to contain discontinuous functions. The inclusion of discontinuous functions brings with it certain technical difficulties that, in full generality and rigor, require some sophisticated analysis to resolve. This analysis is beyond the scope of this text. Nonetheless, we give below a brief sketch of the difficulties, the resolution, and a useful fact concerning orthogonal bases.

Expanding the Space of Functions In our Fourier series examples above we worked in inner product spaces that consist of continuous functions. However, the definition of the inner product

$$(f, g) = \int_a^b f(t)\overline{g(t)} dt$$

requires much less of the functions involved. Indeed we might want to write out Fourier expansions for functions that have discontinuities. Such expansions can be a very useful, so it makes sense to enlarge this inner product space to include more than just continuous functions.

The Cauchy–Schwarz inequality indicates the direction we should move. The inner product (f, g) of two functions is guaranteed to be finite if the L^2 norms $\|f\|$

and $\|g\|$ are both finite. This was exactly the condition imposed in Example 1.8. We will thus enlarge our inner product space to include all functions f that satisfy

$$\int_a^b |f(t)|^2 dt < \infty. \quad (1.52)$$

This guarantees that the inner product (f, g) is defined for all pairs $f, g \in V$. However, equation (1.52) presumes that the function $|f(t)|^2$ can be meaningfully integrated using the Riemann integral. This may seem like a technicality, but while continuous functions are automatically Riemann integrable, arbitrary functions are not. We thus require Riemann integrability of f (which ensures that $|f|^2$ is Riemann integrable). Our notion of $L^2(a, b)$ is thus enlarged to include many types of discontinuous functions.

Complications This enlargement of the space creates a problem of its own though. Consider a function f that is zero at all but one point in $[a, b]$. Such a function, as well as $|f(t)|^2$, is Riemann integrable, and indeed the integral of $|f(t)|^2$ will be zero. In short, $\|f\| = 0$ and also $(f, f) = 0$, even though f is not identically zero. This violates property 1 for the norm and property 3 for the inner product. More generally, functions f and g that differ from each other at only finitely many (perhaps even more) points satisfy $\|f - g\| = 0$, even though $f \neq g$. Enlarging our inner product space has destroyed the inner product structure and notion of distance! This wasn't a problem when the functions were assumed to be continuous; recall Example 1.15.

The fix for this problem requires us to redefine what we mean by " $f = g$ " in $L^2(a, b)$; recall the remark at the end of Example 1.3. Functions f_1 and f_2 that differ at sufficiently few points are considered identical under an equivalence relation " \sim ," where " $f_1 \sim f_2$ " means $\int |f_1 - f_2|^2 dt = 0$. The elements of the space of $L^2(a, b)$ thus consist of equivalence classes of functions that differ at sufficiently few points (just as the rational numbers consist of "fractions," but with a_1/b_1 and a_2/b_2 identified under an equivalence relation, namely, $a_1/b_1 \sim a_2/b_2$ meaning $a_1b_2 = a_2b_1$; thus $\frac{1}{2}$, $\frac{2}{4}$, and $\frac{3}{6}$ are all the same rational number). As a consequence functions in $L^2(a, b)$ do not have well-defined point values. When we carry out computations in $L^2(a, b)$, it is almost always via integration or inner products.

These changes also require some technical modifications to our notion of integration. It turns out Riemann integration is not sufficient for the task, especially when we need to deal with sequences or series of functions in $L^2(a, b)$. The usual approach requires replacing the Riemann integral with the Lebesgue integral, a slightly more versatile approach to integration. The modifications also have the happy consequence of "completing" the inner product space by guaranteeing that Cauchy sequences in the space converge to a limit within the space, which greatly simplifies the analysis. A complete inner product space is known as a *Hilbert space*. The Hilbert space obtained in this situation is known as $L^2(a, b)$. These spaces play an important role in applied mathematics and physics.

Despite all of these complications, we will use the notation $L^2(a, b)$ for the set of all Riemann integrable functions that satisfy inequality (1.52). Indeed, if we limit

our attention to the subset of piecewise continuous functions, then real no difficulties arise. For the details of the full definition of $L^2(a, b)$ the interested reader can refer to [20] or any advanced analysis text.

A Converse to Parseval Parseval's identity (1.46) shows that if $\phi_k, k \geq 0$, is an orthonormal basis for $L^2(a, b)$ and $f \in L^2(a, b)$, then f can be expanded as $f = \sum_k c_k \phi_k$ and $\sum_k |c_k|^2 = \|f\|^2 < \infty$. That is, every function in $L^2(a, b)$ generates a sequence $\mathbf{c} = (c_0, c_1, \dots)$ in $L^2(\mathbb{N})$. In our new and improved space $L^2(a, b)$ it turns out that the converse is true: if we choose any sequence $\mathbf{c} = (c_0, c_1, \dots)$ with $\sum_k c_k^2 < \infty$, then the sum

$$\sum_{k=0}^{\infty} c_k \phi_k \quad (1.53)$$

converges to some function $f \in L^2(a, b)$, in the sense that

$$\lim_{n \rightarrow \infty} \left\| f - \sum_{k=0}^n c_k \phi_k \right\| = 0.$$

In short, the functions in $L^2(a, b)$ and sequences in $L^2(\mathbb{N})$ can be matched up one to one (the correspondence depends on the basis ϕ_k). This is worth stating as a theorem.

Theorem 1.10.2 *Let $\phi_k, k \geq 0$, be an orthonormal basis for $L^2(a, b)$. There is an invertible linear mapping $\Phi : L^2(a, b) \rightarrow L^2(\mathbb{N})$ defined by*

$$\Phi(f) = \mathbf{c},$$

where $\mathbf{c} = (c_0, c_1, \dots)$ with $c_k = (f, \phi_k)$. Moreover $\|f\|_{L^2(a,b)} = \|\mathbf{c}\|_{L^2(\mathbb{N})}$ (the mapping is an “isometry,” that is, length-preserving.)

1.11 MATLAB PROJECT

This section is designed as a series of Matlab explorations that allow the reader to play with some of the ideas in the text. It also introduces a few Matlab commands and techniques we'll find useful later. Matlab commands that should be executed are in a bold `typewriter` font and usually displayed prominently.

1. Start Matlab. Most of the computation in the text is based on indexing vectors beginning with index 0, but Matlab indexes vectors from 1. This won't usually cause any headaches, but keep it in mind.
2. Consider sampling the function $f(t) = \sin(2\pi(440)t)$ on the interval $0 \leq t < 1$, at 8192 points (sampling interval $\Delta T = 1/8192$) to obtain samples

$f_k = f(k\Delta T) = \sin(2\pi(440)k/8192)$ for $0 \leq k \leq 8191$. The samples can be arranged in a vector \mathbf{f} . You can do this in Matlab with

```
f = sin(2 * pi * 440 / 8192 * (0:8191));
```

Don't forget the semicolon or Matlab will print out \mathbf{f} !

The sample vector \mathbf{f} is stored in double precision floating point, about 15 significant figures. However, we'll consider \mathbf{f} as not yet quantized. That is, the individual components f_k of \mathbf{f} can be thought of as real numbers that vary continuously, since 15 digits is pretty close to continuous for our purposes.

- What is the frequency of the sinewave $\sin(2\pi(440)t)$, in Hertz?
- Plot the sampled signal with the command `plot(f)`. It probably doesn't look too good, as it goes up and down 440 times in the plot range. You can plot a smaller range, say the first 100 samples, with `plot(f(1:100))`.
- At the sampling rate 8192 Hertz, what is the Nyquist frequency? Is the frequency of $f(t)$ above or below the Nyquist frequency?
- Type `sound(f)` to play the sound out of the computer speaker. By default, Matlab plays all sound files at 8192 samples per second, and assumes the sampled audio signal is in the range -1 to 1 . Our signal satisfies these conditions.
- As an example of aliasing, consider a second signal $g(t) = \sin(2\pi(440 + 8192)t)$. Repeat parts (a) through (d) with sampled signal

```
g = sin(2 * pi * (440+8192) / 8192 * (0:8191));
```

The analog signal $g(t)$ oscillates much faster than $f(t)$, and we could expect it to yield a higher pitch. However, when sampled at frequency 8192 Hertz, $f(t)$ and $g(t)$ are aliased and yield precisely the same sampled vectors \mathbf{f} and \mathbf{g} . They should sound the same too.

- To illustrate the effect of quantization error, let us construct a 2-bit (4 quantization levels) version of the audio signal $f(t)$ as in the scheme of Example 1.26. With that notation we have minimum value $m = -1$ and maximum value $M = 1$ for our signal, with $r = 4$. The command

```
qf = ceil(2 * (f+1)) - 1;
```

produces the quantized signal $q(\mathbf{f})$. Sample values of $f(t)$ in the ranges $(-1, -0.5]$, $(-0.5, 0]$, $(0, 0.5]$, and $(0.5, 1]$ are mapped to the integers $0, 1, 2, 3$, respectively.

To approximately reconstruct the quantized signal, we apply the dequantization formula to construct $\tilde{\mathbf{f}}$ as

```
ftilde = -1 + 0.5 * (qf + 0.5);
```

This maps the integers $0, 1, 2$, and 3 to values $-0.75, -0.25, 0.25$, and 0.75 , respectively (the codewords in this scheme).

- g. Plot the first hundred values of \tilde{f} with `plot(ftilde(1:100))`; Play the quantized signal with `sound(ftilde)`; It should sound harsh compared to f .
- h. Compute the distortion (as a percentage) of the quantized/dequantized signal using equation (1.39). In Matlab this is implemented as

```
100 * norm(f-ftilde)^2/norm(f)^2
```

The `norm` command computes the standard Euclidean norm of a vector.

- i. Repeat parts (f) through (h) using 3-,4-,5-,6-,7-, and 8-bit quantization. For example, 5-bit quantization is accomplished with `qf=ceil(16*(f+1))-1`, dequantization with `ftilde=-1+(qf+0.5)/16`. Here $16 = 2^{5-1}$. Make sure to play the sound in each case. Make up a table showing the number of bits in the quantization scheme, the corresponding distortion, and your subjective rating of the sound quality.

At what point can your ear no longer distinguish the original audio signal from the quantized version?

- 3. Type `load('splat')`. This loads in an audio signal sampled at 8192 samples per second. By default the signal is loaded into a vector "y" in Matlab, while the sampling rate is loaded into a variable "Fs". Execute the Matlab command `whos` to verify this (or look at the workspace window that shows the currently defined variables).
 - a. Play the sampled signal with `sound(y)`.
 - b. Plot the sampled signal. Based on the size of y and sample rate 8192 Hertz, what is T , the length (in seconds) of the sampled sound?
 - c. The audio signal was sampled at frequency 8192. We can mimic a lower sampling rate $8192/m$ by taking every m th entry of the original vector y ; this is called *downsampling*. In particular, let's try $m = 2$. Execute `y2 = y(1:2:10001)`; The downsampled vector $y2$ is the same sound, but sampled at frequency $8192/2 = 4096$ Hertz.
- Play the downsampled $y2$ with `sound(y2, 4096)`. The second argument to the `sound` command indicates the sound should be played back at the corresponding rate (in Hertz).
- d. Comment: why does the whistling sound in the original audio signal fall steadily in pitch, while the downsampled version seems to rise and fall?
- 4. You can clear from Matlab memory all variables defined to this point with the command `clear`. Do so now.
 - a. Find an image (JPEG will do) and store it in Matlab's current working directory. You can load the image into Matlab with

```
z = imread('myimage.jpg');
```

(change the name to whatever your image is called!) If the image is color, the "`imread`" command automatically converts the $m \times n$ pixel image (in

whatever conventional format it exists) into three $m \times n$ arrays, one for each of red, blue, and green as in Section 1.3.5. Each array consists of unsigned eight-bit integers, that is, integers in the range 0 to 255. Thus the variable z is now an $m \times n \times 3$ array. If the image is grayscale you'll get only one array.

- b. The command

```
image(z);
```

displays the image in color, under the convention that z consists of unsigned integers and the colors are scaled in the 0 to 255 range. If we pass an $m \times n$ by 3 array of floating point numbers to the `image` command, it is assumed that each color array is scaled from 0.0 to 1.0. Consult the help page for the “`image`” command for more information.

- c. A simple way to construct an artificial grayscale image is by picking off one of the color components and using it as a grayscale intensity, for example, $zg = \text{double}(z(:,:,1))$; . However, a slightly more natural result is obtained by taking a weighted average of the color components, as

```
zg = 0.2989 * double(z(:,:,1))
+ 0.5870 * double(z(:,:,2)) + 0.1140 * double(z(:,:,3));
```

The `double` command indicates that the array should be converted from unsigned integers to double precision floating point numbers, for example, 13 becomes 13.0. It's not strictly necessary unless we want to do floating point arithmetic on zg (which we do). The weighting coefficients above stem from the NTSC (television) color scheme; see [14] for more information. Now we set up an appropriate grayscale color map with the commands

```
L = 255;
colormap([(0:L)/L; (0:L)/L; (0:L)/L]');
```

Type `help colormap` for more information on this command. Very briefly, the array that is passed to the `colormap` command should have three columns, any number of rows. The three entries in the k th row should be scaled in the 0.0 to 1.0 range, and indicate the intensity of red, green, and blue that should be displayed for a pixel that is assigned integer value k . In our `colormap` command above, the k th row consists of elements $((k - 1)/255, (k - 1)/255, (k - 1)/255)$, which correspond to a shade of gray (equal amounts of red, blue, and green), with $k = 0$ as black and $k = 255$ as white.

Now display the grayscale image with

```
image(zg);
```

It's not necessary to execute the `colormap` command prior to displaying every image—once will do, unless you close the display window, in which case you must re-initialize the color map.

- d. The image is currently quantized at eight bit precision (each pixel's graylevel specified by one of $0, 1, 2, \dots, 255$). We can mimic a cruder quantization level, say six-bit, with

```
qz = 4 * floor(zg/4);
```

followed by `image(qz)`. This command has the effect of rounding each entry of `zg` to the next lowest multiple of 4, so each pixel is now encoded as one of the 64 numbers $0, 4, 8, \dots, 252$. Can you tell the difference in the image?

Compute the percent distortion introduced with

```
100 * norm(zg-qz,'fro') .^ 2 / norm(zg,'fro') .^ 2
```

The “`fro`” argument indicates that the Frobenius norm of Example 1.14 should be used for the matrices, that is, take the square root of the sum of the squares of the matrix entries.

Repeat the above computations for other b -bit quantizations with $b = 1, 2, 3, 4, 5$. Display the image in case, and compute the distortion. At what point does the quantization become objectionable?

- e. We can add noise to the image with

```
zn = zg + 50 * (rand(size(zg))-0.5);
```

which should, of course, be followed by `image(zn)`. The `size` command returns the dimensions of the matrix `zg`, and the `rand` command generates a matrix of that size consisting of uniformly distributed random numbers (double precision floats) on the interval 0 to 1. Thus $50 * (\text{rand}(\text{size}(\text{zg})) - 0.5)$ yields an array of random numbers in the range -25 to 25 that is added to `zg`. Any values in `zn` that are out of range (less than 0 or greater than 255) are “clipped” to 0 or 255, respectively.

5. This exercise illustrates aliasing for images. First, clear all variables with `clear`, and then execute the commands `L = 255`; and

```
colormap([(0:L)/L;(0:L)/L;(0:L)/L]');
```

as in the previous exercise.

Let's sample and display an analog image, say

$$f(x, y) = 128(1 + \sin(2\pi(20)x)\sin(2\pi(30)y)),$$

on the square $0 \leq x, y \leq 1$; we've chosen f to span the range 0 to 256. We will sample on an m by n grid for various values of m and n . This can be accomplished with

```
m = 50; X = [0:m-1]/m;
n = 50; Y = [0:n-1]/n;
f = 128 * (1 + sin(2*pi*30*Y)' * sin(2*pi*20*X));
```

The first portion $\sin(2\pi \cdot 30 \cdot Y)'$ are the y values of the grid points (as a column vector) and $\sin(2\pi \cdot 20 \cdot X)$ are the x values (as a row vector). Plot the image with `image(f)`:

Try various values of m and n from 10 to 500. Large values of m and n should produce a more “faithful” image, while small values should produce obvious visual artifacts; in particular, try $m = 20, n = 30$. The effect is highly dependent on screen resolution.

EXERCISES

JPEG Compression

- 1.1** Find at least five different color JPEG images on a computer (or with a Web browser); they'll have a “.jpg” suffix on the file name. Try to find a variety of images, for example, one that is “mostly dark” (astronomical images are a good bet). For each image determine the following:

- Its pixel by pixel size—how many pixels wide, how many tall. This can usually be determined by right mouse clicking on the image and selecting “Properties.”
- The memory storage requirements for the image if it was stored “naively” as in the example in the text above.
- The actual memory storage requirement (the file size).
- The ratio of the actual and “naive” storage requirements.

Summarize your work in a table. Note the range of compression ratios obtained. Can you detect any correlation between the nature or quality of the images and the compression ratios achieved?

Complex Numbers

- 1.2** Use Euler's identity (1.11) to prove that if x is real, then

- $e^{-ix} = \overline{e^{ix}}$,
- $e^{2\pi ix} = 1$ if and only if x is an integer.

- 1.3 The complex number z is an N th root of unity if and only if $z^N = 1$. Draw the eighth roots of unity on the unit circle.
- 1.4 Prove that the entries of $\mathbf{E}_{N,k}$ are N th roots of unity (see Exercise 1.3).
- 1.5 Suppose that

$$\begin{aligned}x(t) &= a \cos(\omega t) + b \sin(\omega t) \\&= ce^{i\omega t} + de^{-i\omega t}\end{aligned}$$

for all real t . Show that

$$\begin{aligned}a &= c + d, & b &= ic - id, \\c &= \frac{a - ib}{2}, & d &= \frac{a + ib}{2}.\end{aligned}$$

Sampling and Quantization

- 1.6 Let $x(t) = 1.3t$ and $y(t) = \sin(\frac{\pi}{2}t)$ be analog signals on the interval $0 \leq t \leq 1$.
- Sample $x(t)$ at times $t = 0, 0.25, 0.5, 0.75$ to produce sampled vector $\mathbf{x} = (x(0), x(0.25), x(0.5), x(0.75)) \in \mathbb{R}^4$. Sample $y(t)$ at the same times to produce vector $\mathbf{y} \in \mathbb{R}^4$. Verify that the sampled version (same times) of the analog signal $x(t) + y(t)$ is just $\mathbf{x} + \mathbf{y}$ (this should be painfully clear).
 - Let q denote a function that takes any real number r and rounds it to the nearest integer, a simple form of quantization. Use q to quantize \mathbf{x} from part (a) component by component, to produce a quantized vector $q(\mathbf{x}) = (q(x_0), q(x_1), q(x_2), q(x_3))$. Do the same for \mathbf{y} and $\mathbf{x} + \mathbf{y}$. Show that $q(\mathbf{x}) + q(\mathbf{y}) \neq q(\mathbf{x} + \mathbf{y})$, and also that $q(2\mathbf{x}) \neq 2q(\mathbf{x})$. Quantization is a nonlinear operation!

Vector Spaces

- 1.7 Is the set of all quadratic polynomials in x with real-valued coefficients (with polynomial addition and scalar multiplication defined in the usual way) a vector space over \mathbb{R} ? Why or why not? (Consider something like $a_0 + a_1x + 0x^2$ a quadratic polynomial.)
- 1.8 Is the set of all continuous real-valued functions $f(x)$ defined on $[0, 1]$ that satisfy

$$\int_0^1 f(x) dx = 3$$

a vector space over \mathbb{R} ? Assume function addition and scalar multiplication are defined as usual.

- 1.9** Clearly, \mathbb{R}^n is a subset of \mathbb{C}^n . Is \mathbb{R}^n a subspace of \mathbb{C}^n (where \mathbb{C}^n is considered a vector space over \mathbb{C})?
- 1.10** Verify that the set in Example 1.3 with given operations is a vector space.
- 1.11** Verify that the set $L^2(\mathbb{N})$ in Example 1.5 with given operations is a vector space. *Hint:* This closely parallels Example 1.8, with summation in place of integrals.
Explain why $L^2(\mathbb{N})$ is a subspace of $L^\infty(\mathbb{N})$.
- 1.12** The point of this exercise is to prove the assertions in Proposition 1.4.1 for an abstract vector space.
- Show that the $\mathbf{0}$ vector is unique. To do this, suppose that there are two vectors, $\mathbf{0}_1$ and $\mathbf{0}_2$, both of which play the role of the zero vector. Show that $\mathbf{0}_1 = \mathbf{0}_2$. *Hint:* Consider $\mathbf{0}_1 + \mathbf{0}_2$.
 - Below is a proof that $0\mathbf{u} = \mathbf{0}$ in any vector space. In this proof $-\mathbf{u}$ denotes the additive inverse for \mathbf{u} , so $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$. What property or properties of the eight listed in Definition 1.4.1 justifies each step?

$$\begin{aligned}(1+0)\mathbf{u} &= 1\mathbf{u} + 0\mathbf{u} \\ 1\mathbf{u} &= \mathbf{u} + 0\mathbf{u} \\ \mathbf{u} &= \mathbf{u} + 0\mathbf{u}, \\ \mathbf{u} + (-\mathbf{u}) &= (\mathbf{u} + (-\mathbf{u})) + 0\mathbf{u} \\ \mathbf{0} &= \mathbf{0} + 0\mathbf{u} \\ \mathbf{0} &= 0\mathbf{u}.\end{aligned}$$

- c. Show that if $\mathbf{u} + \mathbf{v} = \mathbf{0}$, then $\mathbf{v} = (-1)\mathbf{u}$ (this shows that the additive inverse of \mathbf{u} is $(-1)\mathbf{u}$).

Basic Analog and Discrete Waveforms

- 1.13** Write out the basic waveforms $\mathbf{E}_{2,k}$ for $k = -2, -1, 0, 1, 2$, and verify that the resulting vectors are periodic with period 2 with respect to the index k .
Repeat for $\mathbf{E}_{3,k}$ (same k range). Verify periodicity with period 3.
- 1.14** Let $a = re^{i\theta}$ be a complex number (where $r > 0$ and θ are real).
- Show that the function $f(t) = ae^{i\omega t}$ satisfies $|f(t)| = r$ for all t .
 - Show that $f(t) = ae^{i\omega t}$ is shifted a fraction θ/ω cycles or periods to the left, compared to $|r|e^{i\omega t}$.
- 1.15** Show that each of the four types of waveforms in (1.19) can be expressed as a linear combination of waveforms $e^{\pm i\alpha x \pm i\beta y}$ of the form (1.17).
- 1.16** Let \mathbf{C}_k be the vector obtained by sampling the function $\cos(2\pi kt)$ at the points $t = 0, 1/N, 2/N, \dots, (N-1)/N$, and let \mathbf{S}_k be similarly defined with respect

to the sine function. Prove the following vector analogs of the equations (1.12), (1.13), and (1.14) relating the exponential and trigonometric wave forms.

$$\begin{aligned}\mathbf{E}_k &= \mathbf{C}_k + i\mathbf{S}_k, \quad \overline{\mathbf{E}_k} = \mathbf{C}_k - i\mathbf{S}_k, \\ \mathbf{C}_k &= \frac{1}{2}(\mathbf{E}_k + \overline{\mathbf{E}_k}), \quad \mathbf{S}_k = \frac{1}{2i}(\mathbf{E}_k - \overline{\mathbf{E}_k}), \\ \mathbf{C}_k &= \text{Re}(\mathbf{E}_k), \quad \mathbf{S}_k = \text{Im}(\mathbf{E}_k),\end{aligned}$$

where $\mathbf{E}_k = \mathbf{E}_{N,k}$ is as defined in equation (1.22).

- 1.17** Show that we can factor the basic two-dimensional waveform $\mathcal{E}_{m,n,k,l}$ as

$$\mathcal{E}_{m,n,k,l} = \mathbf{E}_{m,k}\mathbf{E}_{n,l}^T$$

(recall superscript T denotes the matrix/vector transpose operation), where the vectors $\mathbf{E}_{m,k}$ and $\mathbf{E}_{n,l}$ are the discrete basic waveforms in one-dimension as defined in equation (1.22), as column vectors.

- 1.18** Consider an exponential waveform

$$f(x, y) = e^{2\pi i(px+qy)}$$

as was discussed in Section 1.5.2 (p and q need not be integers). Figure 1.7 in that section indicates that this waveform has a natural “direction” and “wavelength.” The goal of this problem is to understand the sense in which this is true, and how these quantities depend on p and q .

Define $\mathbf{v} = (p, q)$, so \mathbf{v} is a two-dimensional vector. Consider a line L through an arbitrary point (x_0, y_0) in the direction of a unit vector $\mathbf{u} = (u_1, u_2)$ (so $\|\mathbf{u}\| = 1$). The line L can be parameterized with respect to arc length as

$$x(t) = x_0 + tu_1, \quad y(t) = y_0 + tu_2.$$

- a. Show that the function $g(t) = f(x(t), y(t))$ with $x(t)$, $y(t)$ as above (i.e., f evaluated along the line L) is given by

$$g(t) = Ae^{2\pi i\|\mathbf{v}\|\cos(\theta)t},$$

where A is some complex number that doesn’t depend on t and θ is the angle between \mathbf{v} and \mathbf{u} . Hint: Use equation (1.30).

- b. Show that if L is orthogonal to \mathbf{v} , then the function g (and so f) remains constant.
c. Find the frequency (oscillations per unit distance moved) of g as a function of t , in terms of p , q , and θ .
d. Find the value of θ that maximizes the frequency at which $g(t)$ oscillates. This θ dictates the direction one should move, relative to \mathbf{v} so that f

oscillates as rapidly as possible. How does this value of θ compare to the θ value in question (b)? What is this maximal frequency of oscillation, in terms of p and q ?

- e. Find the “peak-to-peak” distance or wavelength of the waveform $f(x, y)$, in terms of p and q .

Aliasing and Basic Waveforms

- 1.19 Write out the vectors $\mathbf{E}_{6,0}, \mathbf{E}_{6,1}, \dots, \mathbf{E}_{6,5}$ as in Section 1.7.1. Determine all aliasing relations or redundancies (including conjugate aliasing) you can from the chart. (Remember to index the vector components from 0 to 5.)
- 1.20 For a pure 1D wave form of N samples prove the aliasing relation

$$\mathbf{E}_{N-k} = \overline{\mathbf{E}_k}.$$

- 1.21 Find all the aliasing relations you can (including conjugate aliasing) for $\mathcal{E}_{m,n,k,l}$. This can be done directly, or you might use equation (1.26) and the aliasing relations for the $\mathbf{E}_{N,k}$.

Inner Products and Norms

- 1.22 Let $S = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, where $\mathbf{v}_1 = (1, 1, 0), \mathbf{v}_2 = (-1, 1, 1)$, and $\mathbf{v}_3 = (1, -1, 2)$ are vectors in \mathbb{R}^3 .
 - a. Verify that S is orthogonal with respect to the usual inner product. This shows that S must be a basis for \mathbb{R}^3 .
 - b. Write the vector $\mathbf{w} = (3, 4, 5)$ as a linear combination of the basis vectors in S . Verify that the linear combination you obtain actually reproduces \mathbf{w} !
 - c. Rescale the vectors in S as per Remark 1.11 on page 39 to produce an equivalent set S' of orthonormal vectors.
 - d. Write the vector $\mathbf{w} = (3, 4, 5)$ as a linear combination of the basis vectors in S' .
 - e. Use the results of part (d) to check that Parseval’s identity holds.
- 1.23 Let $S = \{\mathbf{E}_{4,0}, \mathbf{E}_{4,1}, \mathbf{E}_{4,2}, \mathbf{E}_{4,3}\}$ (these vectors are written out explicitly just prior to equation (1.24)). The set S is orthogonal and a basis for \mathbb{R}^4 .
 - a. Use Theorem 1.8.3 to write the vector $\mathbf{v} = (1, 5, -2, 3)$ as a linear combination of the basis vectors in S .
 - b. Rescale the vectors in S as per Remark 1.11 on page 39 to produce an equivalent set S' of orthonormal vectors.
 - c. Write the vector $\mathbf{v} = (1, 5, -2, 3)$ as a linear combination of the basis vectors in S' .
 - d. Use the results of part (c) to check that Parseval’s identity holds.

- 1.24** There are infinitely many other inner products on \mathbb{R}^n besides the standard dot product, and they can be quite useful too.

Let $\mathbf{d} = (d_1, d_2, \dots, d_n) \in \mathbb{R}^n$. Suppose that $d_k > 0$ for $1 \leq k \leq n$.

- a. Let $\mathbf{v} = (v_1, v_2, \dots, v_n)$ and $\mathbf{w} = (w_1, w_2, \dots, w_n)$ be vectors in \mathbb{R}^n . Show that the function

$$(\mathbf{v}, \mathbf{w})_d = \sum_{k=1}^n d_k v_k w_k$$

defines an inner product on \mathbb{R}^n . Write out the corresponding norm.

- b. Let $\mathbf{d} = (1, 5)$ in \mathbb{R}^2 , and let $S = \{\mathbf{v}_1, \mathbf{v}_2\}$ with $\mathbf{v}_1 = (2, 1)$, $\mathbf{v}_2 = (5, -2)$. Show that S is orthogonal with respect to the $(\cdot, \cdot)_d$ inner product.
c. Find the length of each vector in S with respect to the norm induced by this inner product.
d. Write the vector $\mathbf{w} = (-2, 5)$ as a linear combination of the basis vectors in S . Verify that the linear combination you obtain actually reproduces \mathbf{w} !

- 1.25** Let \mathbf{v} and \mathbf{w} be elements of a normed vector space. Prove the reverse triangle inequality,

$$\|\mathbf{v}\| - \|\mathbf{w}\| \leq \|\mathbf{v} - \mathbf{w}\|. \quad (1.54)$$

Hint: Start with $\mathbf{v} = (\mathbf{v} - \mathbf{w}) + \mathbf{w}$, take the norm of both sides, and use the usual triangle inequality.

- 1.26** Let $d(t)$ be a real-valued, positive, continuous function that is bounded away from 0 on an interval $[a, b]$; that is, $d(t) \geq \delta > 0$ for some δ and all $t \in [a, b]$. Verify that

$$(f, g)_d = \int_a^b d(t)f(t)\overline{g(t)} dt$$

defines an inner product on $C[a, b]$ (with complex-valued functions). Write out the corresponding norm.

- 1.27** Suppose that V is an inner product space with inner product (\mathbf{v}, \mathbf{w}) . Show that if we define

$$\|\mathbf{v}\| = \sqrt{(\mathbf{v}, \mathbf{v})},$$

then $\|\mathbf{v}\|$ satisfies the properties of a norm. *Hints:* All the properties are straightforward, except the triangle inequality. To show this, note that

$$\|\mathbf{v} + \mathbf{w}\|^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 + (\mathbf{v}, \mathbf{w}) + (\mathbf{w}, \mathbf{v}).$$

Apply the Cauchy-Schwarz inequality to both inner products on the right side above, and note that for any $z \in \mathbb{C}$ we have $|\operatorname{Re}(z)| \leq |z|$.

- 1.28** Suppose that V is an inner product space, with a norm $\|\mathbf{v}\| = \sqrt{(\mathbf{v}, \mathbf{v})}$ that comes from the inner product.

- a. Show that this norm must satisfy the *parallelogram* identity

$$2\|\mathbf{u}\|^2 + 2\|\mathbf{v}\|^2 = \|\mathbf{u} + \mathbf{v}\|^2 + \|\mathbf{u} - \mathbf{v}\|^2.$$

- b. Let $f(x) = x$ and $g(x) = x(1-x)$ be elements of the normed vector space $C[0, 1]$ with the supremum norm. Compute each of $\|f\|_\infty$, $\|g\|_\infty$, $\|f + g\|_\infty$, and $\|f - g\|_\infty$, and verify that the parallelogram identity from part (a) does not hold. Hence the supremum norm cannot come from an inner product.

- 1.29** Suppose that S is an orthogonal but not orthonormal basis for \mathbb{R}^n consisting of vectors \mathbf{v}_k , $1 \leq k \leq n$. Show that Parseval's identity becomes

$$\|\mathbf{v}\|^2 = \sum_{k=1}^n |\alpha_k|^2 \|\mathbf{v}_k\|^2.$$

where $\mathbf{v} = \sum_{k=1}^n \alpha_k \mathbf{v}_k$. Hint: Just chase through the derivation of equation (1.36).

- 1.30** For the basic waveforms defined by equation (1.22) show that

$$(\mathbf{E}_{N,k}, \mathbf{E}_{N,k}) = N$$

with the inner product defined in Example 1.13.

- 1.31** For 2D wave forms which are $m \times n$ matrices defined by equation (1.25) prove that

$$(\mathcal{E}_{k,l}, \mathcal{E}_{p,q}) = 0$$

when $k \neq p$ or $l \neq q$, and

$$(\mathcal{E}_{k,l}, \mathcal{E}_{k,l}) = mn$$

with the inner product on $M_{m,n}(\mathbb{C})$ defined in Example 1.14. It may be helpful to look at Example 1.20.

- 1.32** Let \mathbf{v}_k , $1 \leq k \leq n$, be any set of vectors in \mathbb{C}^n , considered as column vectors. Define the $n \times n$ matrix

$$\mathbf{A} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n].$$

Let $\mathbf{A}^* = (\overline{\mathbf{A}})^T$ (conjugate and transpose entries). What is the relationship between the entries $b_{m,k}$ of the $n \times n$ matrix $\mathbf{B} = (\mathbf{A}^*)(\mathbf{A})$ and the inner products $(\mathbf{v}_k, \mathbf{v}_m)$ (inner product defined as in Example 1.13)?

- 1.33** Use the result of the last exercise to show that if S is an orthonormal set of vectors $\mathbf{v}_k \in \mathbb{C}^n$, $1 \leq k \leq n$, and \mathbf{A} is the matrix from the previous problem, then $(\mathbf{A}^*)(\mathbf{A}) = \mathbf{I}$, where \mathbf{I} denotes the $n \times n$ identity matrix.

Infinite-Dimensional Inner Product Spaces and Fourier Series

- 1.34** Let ϕ_k , $1 \leq k < \infty$ be an orthonormal set in an inner product space V (no assumption that ϕ_k is complete). Let $f \in V$ and define $c_k = (f, \phi_k)$. Prove Bessel's inequality,

$$\sum_{k=1}^{\infty} |c_k|^2 \leq \|f\|^2. \quad (1.55)$$

Hint: Start with

$$0 \leq \left(f - \sum_{k=1}^n c_k \phi_k, f - \sum_{k=1}^n c_k \phi_k \right)$$

(first explain why this inequality is true).

- 1.35** Suppose that V is a normed vector space and \mathbf{v}_n a sequence of elements in V that converge to $\mathbf{v} \in V$, that is,

$$\lim_{n \rightarrow \infty} \|\mathbf{v}_n - \mathbf{v}\| = 0.$$

Show that

$$\lim_{n \rightarrow \infty} \|\mathbf{v}_n\| = \|\mathbf{v}\|.$$

Hint: Use equation (1.54) from Exercise 1.25.

Show that the converse of this statement is false (provide a counterexample).

- 1.36** We can endow the vector space $L^2(\mathbb{N})$ in Example 1.5 (see also Exercise 1.11) with an inner product

$$(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{\infty} x_k y_k,$$

where $\mathbf{x} = (x_0, x_1, x_2, \dots)$ and $\mathbf{y} = (y_0, y_1, y_2, \dots)$ and all components are real-valued.

- a. Verify that this really does define an inner product. In particular, you should first show that the inner product of any two elements is actually defined; that is, the infinite sum converges. (*Hint:* Use $|x_k y_k| \leq (x_k^2 + y_k^2)/2$ as in Example 1.8.)

What is the corresponding norm on $L^2(\mathbb{N})$?

- b. Let \mathbf{e}_k denote that element of $L^2(\mathbb{N})$ that has $x_k = 1$ and all other $x_m = 0$. Show that $S = \cup_{k=0}^{\infty} \mathbf{e}_k$ is an orthonormal set.
- c. For an arbitrary $\mathbf{x} \in L^2(\mathbb{N})$, show that

$$\mathbf{x} = \sum_{k=0}^{\infty} \alpha_k \mathbf{e}_k$$

for a suitable choice of the α_k . *Hint:* This is very straightforward; just use Theorem 1.10.1.

- 1.37** Let V be an infinite-dimensional inner product space with orthonormal basis $\phi_k, k \geq 1$. Suppose that \mathbf{x} and \mathbf{y} are elements of V and

$$\mathbf{x} = \sum_{k=1}^{\infty} a_k \phi_k \quad \mathbf{y} = \sum_{k=1}^{\infty} b_k \phi_k.$$

Of course, $a_k = (\mathbf{x}, \phi_k)$ and $b_k = (\mathbf{y}, \phi_k)$.

- a. Define partial sums

$$\mathbf{x}_N = \sum_{k=1}^N a_k \phi_k \quad \mathbf{y}_N = \sum_{k=1}^N b_k \phi_k.$$

Show that

$$(\mathbf{x}_N, \mathbf{y}_N) = \sum_{k=1}^N a_k b_k.$$

- b. Show that

$$(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{\infty} a_k b_k.$$

Hint: Note that for any N ,

$$(\mathbf{x}, \mathbf{y}) - (\mathbf{x}_N, \mathbf{y}_N) = (\mathbf{x}, \mathbf{y} - \mathbf{y}_N) + (\mathbf{x} - \mathbf{x}_N, \mathbf{y}_N),$$

and of course, $\mathbf{x}_N \rightarrow \mathbf{x}$ and $\mathbf{y}_N \rightarrow \mathbf{y}$. Use the Cauchy–Schwarz inequality to show that the right side above goes to zero; then invoke part (a). The result of Exercise 1.35 may also be helpful.

- 1.38** Show that if \mathbf{v} and \mathbf{w} are nonzero vectors, then equality is obtained in the Cauchy–Schwarz inequality (i.e., $|(\mathbf{v}, \mathbf{w})| = \|\mathbf{v}\| \|\mathbf{w}\|$) if and only if $\mathbf{v} = c\mathbf{w}$ for some scalar c . (One direction is easy, the other is somewhat challenging.)

1.39 Let $\phi_k(t) = e^{i\pi kt}/\sqrt{2}$ for $k \in \mathbb{Z}$.

- a. Verify that the ϕ_k form an orthonormal set in $C[-1, 1]$ (complex-valued functions) with the inner product defined in Example 1.15.
- b. Find the Fourier coefficients α_k of the function $f(t) = t$ with respect to the ϕ_k explicitly in terms of k by using equation (1.48). Hint: It's a pretty easy integration by parts, and α_0 doesn't fit the pattern.
- c. Use the result of part (b) to prove the amusing result that

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}.$$

1.40 Let $f(t)$ be a real-valued continuous function defined on an interval $[-T, T]$.

- a. Show that if $f(t)$ is an even function ($f(-t) = f(t)$), then the Fourier series in (1.51) contains only cosine terms (and the constant term).
- b. Show that if $f(t)$ is an odd function ($f(-t) = -f(t)$), then the Fourier series in (1.51) contains only sine terms.

1.41 Let $\phi_k(t) = \cos(k\pi t)$ for $k \geq 0$.

- a. Verify that the ϕ_k are orthogonal on $[0, 1]$ with respect to the usual inner product (note $\phi_0(t) = 1$).
- b. Show that this set of functions is complete as follows (where we'll make use of the given fact that the functions $e^{i\pi kt}$ are complete in $C[-1, 1]$): Let $f(t)$ be a function in $C[0, 1]$. We can extend f to an even function $\tilde{f}(t)$ in $C[-1, 1]$ as

$$\tilde{f}(t) = \begin{cases} f(t), & t \geq 0, \\ f(-t), & t < 0. \end{cases}$$

Now use the result of Exercise 1.40 to show that \tilde{f} has a Fourier expansion in appropriate cosines on $[-1, 1]$. Why does this show that $\cos(k\pi t)$ is complete on $[0, 1]$?

CHAPTER 2

THE DISCRETE FOURIER TRANSFORM

2.1 OVERVIEW

In this chapter we introduce the *discrete Fourier transform* (DFT) and *inverse discrete Fourier transform* (IDFT) for analyzing sampled signals and images. We've already developed the mathematics that underlies these tools: the formula $\alpha_k = (\mathbf{v}, \mathbf{E}_k)/N$ behind equation (1.34) in the last chapter is, up to a minor modification, the DFT, while equation (1.34) is itself the IDFT.

But rather than jumping straight to the definition of the DFT and IDFT, it may be useful to first discuss some basic facts about the notions of *time domain* and *frequency domain*. We'll also pause to a look at a very simple computational example that can help build some important intuition about how the DFT and frequency domain are used. As usual, we'll consider the one-dimensional case of signals first and then move to two dimensions.

Before proceeding futher, we should remark that the definition of these transforms varies slightly from text to text. However, the definitions usually only differ by a scaling or “normalizing” constant. We selected our normalization to be algebraically convenient and consistent with Matlab. For a more comprehensive account of the DFT, see [8].

2.2 THE TIME DOMAIN AND FREQUENCY DOMAIN

In the previous chapter we presented a few different mathematical models associated to one-dimensional signals. In the analog case, the signal is a function $x(t)$ of a real variable t , and we usually view t as time. In the discrete case, the signal is the vector \mathbf{x} obtained by sampling the underlying function at regular times $t = a + k\Delta T$ for some range of the integer k . The indexing integer k thus acts as a kind of time variable. For this reason the original signal, either $x(t)$ or the sampled version $x_k = x(a + k\Delta T)$, is said to exist in the *time domain*. In the two-dimensional image case the phrase “time domain” doesn’t make sense, but we’ll still use it. The original analog image $f(x, y)$ or sampled version might also be said to exist in the “image domain.”

Every object that exists in the time domain has a representation in something called the *frequency domain*. We’ll make this more precise shortly. The DFT and IDFT allow us to move easily back and forth between the discrete time and frequency domains. If you’ve encountered the Laplace transform before, then you’ve already seen the general idea: every suitable function $f(t)$ has a counterpart $F(s)$ in the “Laplace s -domain” (a kind of frequency domain), and the Laplace and inverse Laplace transforms move us back and forth.

The motivation for moving to the frequency domain in the first place is that many operations are easier, both computationally and conceptually, in the frequency domain. We transform the time domain signal to the frequency domain, perform the operation of interest there, and then transform the altered signal back to the time domain.

In many cases it turns out that removing noise from a time domain signal can be accomplished, at least approximately, by removing the signal’s “high frequency components” in the frequency domain. Basically this is done according to the diagram in Figure 2.1. The precise mathematical structure of the frequency domain depends on the nature of the time domain signals we’re trying to analyze and the transform used. In this chapter we use the DFT for the analysis step to compute the frequency domain representation of the signal. In the smoothing step we eliminate or reduce the high frequencies, which is a simple algebraic operation. Finally, in the synthesis step we use the IDFT to re-synthesize a less noisy version of the time domain signal from its altered frequency domain representation. This three-step process is typical of a lot of signal and image processing.

A specific example is presented in the next section. We’ll illustrate the transformation of a signal to the frequency domain via decomposition into basic waveforms, as well as how this frequency domain information can be presented graphically, altered to remove noise, and finally re-synthesized into a denoised version of the original signal.

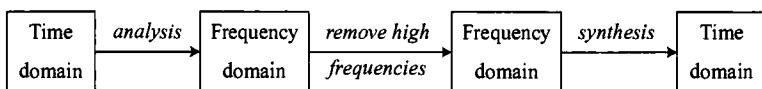


FIGURE 2.1 Processing a signal in the frequency domain.

2.3 A MOTIVATIONAL EXAMPLE

2.3.1 A Simple Signal

Let's begin by considering the analog signal

$$x(t) = 2.0 \cos(2\pi \cdot 5t) + 0.8 \sin(2\pi \cdot 12t) + 0.3 \cos(2\pi \cdot 47t) \quad (2.1)$$

on the time interval $[0, 1]$. As no other frequencies are present, $x(t)$ is built from a superposition of trigonometric waveforms with frequencies of 5, 12, and 47 Hertz. A plot of $x(t)$ is shown in Figure 2.2.

Of course, in practice, only a sampled version of $x(t)$ is available (with quantization error too, but let's assume that's negligible for now). Suppose that we sample with period $\Delta T = 1/128$ and obtain sample vector

$$\mathbf{x} = \left(x(0), x\left(\frac{1}{128}\right), x\left(\frac{2}{128}\right), \dots, x\left(\frac{127}{128}\right) \right).$$

The goal for the moment is simple: Use the discretized signal \mathbf{x} to determine, or at least approximate, the frequencies that make up the original analog signal $x(t)$. If sampling occurs at 128 Hertz, then no aliasing will occur provided that we are willing to assume or assure that the analog signal contains no frequencies above 64 Hertz.

The first 10 samples of $x(t)$ (the components of \mathbf{x}) are, to 4 significant figures,

$$2.300, 2.183, 2.474, 2.507, 1.383, 0.984, -0.023, -1.230, -1.289, -1.958.$$

Of course, there are 118 more samples. Trying to ferret out the frequencies that make up $x(t)$ from these data looks like a formidable task, but we've already solved the problem in Chapter 1! Let us compute the expansion of \mathbf{x} as a linear combination of the basic discrete waveforms $\mathbf{E}_{128,k}$ by using equation (1.34).

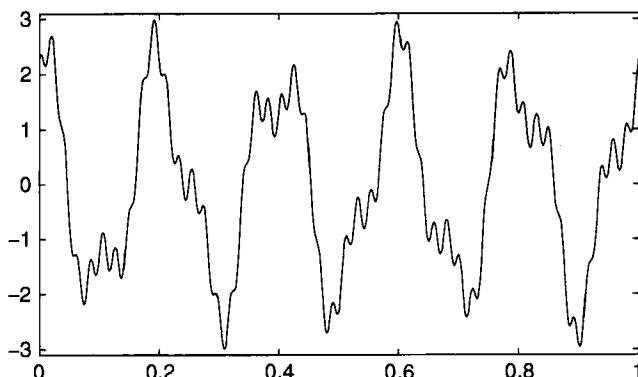


FIGURE 2.2 Time domain analog signal.

2.3.2 Decomposition Into Basic Waveforms

First recall that because of the aliasing relation $\mathbf{E}_{128,k} = \mathbf{E}_{128,k+128m}$ for any integers k and m , we may as well restrict our attention to a range in the index k of length 128. For the moment we'll use the range $-63 \leq k \leq 64$, though later it will be more convenient to use $0 \leq k \leq 127$. According to equation (1.34),

$$\mathbf{x} = \sum_{k=-63}^{64} c_k \mathbf{E}_k, \quad (2.2)$$

where we write just \mathbf{E}_k instead of $\mathbf{E}_{128,k}$. The c_k for $-63 \leq k \leq 64$ are given by

$$c_k = \frac{(\mathbf{x}, \mathbf{E}_k)}{(\mathbf{E}_k, \mathbf{E}_k)} = \frac{1}{128} \sum_{m=0}^{127} x_m e^{-2\pi i km/128},$$

where we've used equation (1.23) for the m th component of \mathbf{E}_k and the fact that $(\mathbf{E}_k, \mathbf{E}_k) = 128$. Remember that the components of \mathbf{E}_k must be conjugated because of the nature of the inner product on \mathbb{C}^N . We are also indexing the components of vectors in the time domain \mathbb{R}^{128} from 0 to 127.

The c_k turn out to be

$$c_{-47} = 0.15, \quad c_{-12} = 0.4i, \quad c_{-5} = 1.0, \quad c_5 = 1.0, \quad c_{12} = -0.4i, \quad c_{47} = 0.15$$

and all other c_k are zero (to round-off error). The sampled signal \mathbf{x} can thus be written as the superposition

$$\mathbf{x} = 0.15 \cdot (\mathbf{E}_{47} + \mathbf{E}_{-47}) - 0.4i \cdot (\mathbf{E}_{12} - \mathbf{E}_{-12}) + 1.0 \cdot (\mathbf{E}_5 + \mathbf{E}_{-5}).$$

This is precisely the data that would be obtained by sampling the analog signal

$$g(t) = 0.15 (e^{2\pi i(47)t} - e^{2\pi i(-47)t}) - 0.4i (e^{2\pi i(12)t} + e^{2\pi i(-12)t}) \\ + (e^{2\pi i(5)t} - e^{2\pi i(-5)t})$$

at times $t = 0, 1/128, 2/128, \dots, 127/128$. From equations (1.13) we have $g(t) = 0.3 \cos(2\pi \cdot 47t) + 0.8 \sin(2\pi \cdot 12t) + 2.0 \cos(2\pi \cdot 5t)$, which is precisely the original analog signal $x(t)$!

2.3.3 Energy at Each Frequency

It's worth examining this computation from the point of view of Remark 1.9 on page 30 in which the quantity $\|\mathbf{x}\|^2$ is interpreted as the energy of a sampled signal \mathbf{x} .

With \mathbf{x} expanded according to equation (2.2),

$$\begin{aligned}
 \|\mathbf{x}\|^2 &= (\mathbf{x}, \mathbf{x}) \\
 &= \left(\sum_{k=-63}^{64} c_k \mathbf{E}_k, \sum_{m=-63}^{64} c_m \mathbf{E}_m \right) \\
 &= \sum_{k=-63}^{64} \sum_{m=-63}^{64} c_k \overline{c_m} (\mathbf{E}_k, \mathbf{E}_m) \\
 &= \sum_{k=-63}^{64} |c_k|^2 \|\mathbf{E}_k\|^2. \tag{2.3}
 \end{aligned}$$

Note that we use different indexes of summation in the first and second argument to the inner product in line 2, to avoid confusion. Of course, we also use the fact that $(\mathbf{E}_k, \mathbf{E}_m) = 0$ for $k \neq m$.

The vector $c_k \mathbf{E}_k$ is that multiple of the basic waveform \mathbf{E}_k needed to synthesize the sampled signal \mathbf{x} . The quantity $|c_k|^2 \|\mathbf{E}_k\|^2 = \|c_k \mathbf{E}_k\|^2$ is the energy of this constituent waveform. Equation (2.3) can thus be viewed as a decomposition of the total energy in the signal into the sum of the energies of the constituent orthogonal waveforms. In particular, for the example signal of interest we have (since $\|\mathbf{E}_k\|^2 = (\mathbf{E}_k, \mathbf{E}_k) = 128$ for all k)

$$\begin{aligned}
 \|\mathbf{x}\|^2 &= 128|c_{-5}|^2 + 128|c_5|^2 + 128|c_{-12}|^2 + 128|c_{12}|^2 \\
 &\quad + 128|c_{-47}|^2 + 128|c_{47}|^2 \\
 &= 128 + 128 + 20.48 + 20.48 + 2.88 + 2.88 \\
 &= 302.72. \tag{2.4}
 \end{aligned}$$

Now recall that the basic waveform \mathbf{E}_k is the sampled version of $e^{2\pi i k t/T}$ and moreover that the frequency of $e^{2\pi i k t/T}$ (and \mathbf{E}_k) is $|k|/T$ Hertz. In this case $T = 1$, so the frequency is merely $|k|$. Equation (2.4) states that the total energy contribution of the 5 Hertz frequency components (index $k = \pm 5$) to \mathbf{x} is $128|c_{-5}|^2 + 128|c_5|^2 = 256$. Of course, 128 units of this comes from the $k = -5$ waveform and 128 from the $k = 5$ waveform, but the quantities can be lumped together because we are interested in the energy at a given frequency. Similarly the contribution of the 12 Hertz frequency components is $128|c_{-12}|^2 + 128|c_{12}|^2 = 40.96$ and the 47 Hertz components contribute energy $128|c_{-47}|^2 + 128|c_{47}|^2 = 5.76$. On a percentage basis the 5 Hertz frequency component constitutes about 84.6% of the total signal energy, the 12 Hertz contributes 13.5%, and the 47 Hertz components constitute the remaining 1.9%.

2.3.4 Graphing the Results

One informative way to present frequency information is graphically as in Figure 2.3, where we plot the quantity $128|c_k|^2$ (the energy contributed by the corresponding

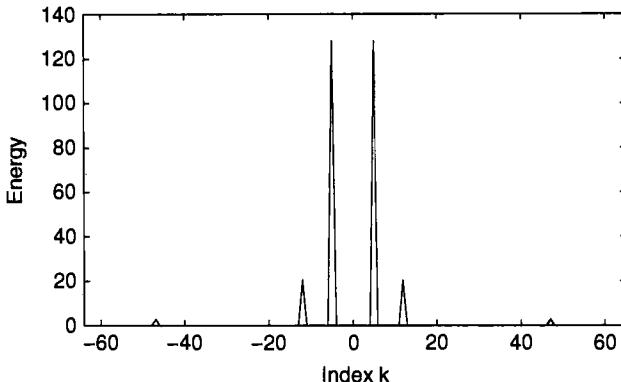


FIGURE 2.3 Spectrum of example signal.

waveform) versus the index k for $-63 \leq k \leq 64$. The plot really consists of a discrete set of points $(k, 128|c_k|^2)$ for $-63 \leq k \leq 64$, but “connecting the dots” makes the plot more easily readable. Clearly, the plot is symmetric about $k = 0$, since in this example $|c_{-k}| = |c_k|$. As we’ll see, this is always the case when the signal \mathbf{x} is real-valued, but not if the signal is complex-valued.

There is another, slightly different way to present such data. Recall that $\mathbf{E}_{N,k}$ is defined for any $k \in \mathbb{Z}$. As a consequence the coefficient $c_k = (\mathbf{x}, \mathbf{E}_{N,k}) / (\mathbf{E}_{N,k}, \mathbf{E}_{N,k})$ is also defined for any k . However, from the aliasing relation $\mathbf{E}_{N,k} = \mathbf{E}_{N,k+mN}$ it follows that $c_k = c_{k+mN}$; that is, the c_k are periodic with period N . In analyzing the frequency information embodied by the c_k , we can thus confine our attention to any range in k of length N . In Figure 2.3 we chose $-63 \leq k \leq 64$, but it will often be convenient to use the range $0 \leq k \leq N - 1$ in the general case, or $0 \leq k \leq 127$ in this example.

If we adopt this convention for the present example, then the values of c_5 , c_{12} , and c_{47} remain the same, and we obtain $c_{81} = c_{-47} = 0.15$, $c_{116} = c_{-12} = 0.4i$, and $c_{123} = c_{-5} = 1.0$. The resulting plot of the points $(k, 128|c_k|^2)$ for $0 \leq k \leq 127$ is shown in Figure 2.4. The previous symmetry at $k = 0$ is now symmetry about $k = 64$. In effect, the entire range $-63 \leq k \leq -1$ has been picked up and moved rigidly to the right 128 units, to the range $65 \leq k \leq 127$.

Finally, if our primary interest is the total energy contributed by the frequency $|k|$ waveforms, then we can lump together these energies and just plot the points $(|k|, 128(|c_k|^2 + |c_{-k}|^2))$ for $k = 1$ to 63 . However, $k = 0$ and $k = 64$ are special. In the case $k = 0$ we should just plot $(0, 128|c_0|^2)$, for using $128(|c_k|^2 + |c_{-k}|^2)$ in this case double counts the energy. Similarly for $k = 64$ we just plot $(64, 128|c_{64}|^2)$. Such a plot is shown in Figure 2.5.

The frequency information contained in the c_k over whatever k range we elect to work over is called the *spectrum* of the signal. How this spectral information is scaled or presented graphically varies widely.

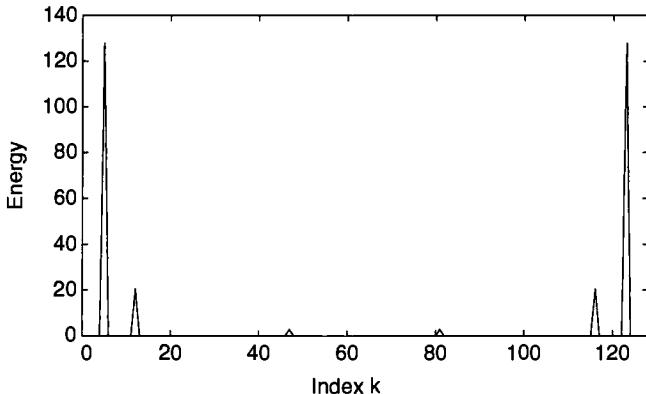


FIGURE 2.4 Spectrum of example signal.

2.3.5 Removing Noise

Signals often contain noise, and one common task in signal and image processing is to “clean up” a signal or image by removing the noise. In the time domain, random noise can contribute to a jagged appearance of an otherwise smooth signal. In the frequency domain, random noise manifests itself across the entire spectrum, including high frequencies where an otherwise smooth signal should have little energy. Thus one very simple approach to noise reduction is this: we decide (based on physical insight, preferably) on a “cutoff” frequency. Any spectral energy above this cutoff is considered to have come from noise, while anything below is signal. Of course, the delineation is rarely clean-cut, but it’s sufficient for illustrative purposes at the moment.

In the present example let’s take 40 Hertz as our cutoff. We compute the c_k as above; in doing so, we’ve moved to the frequency domain. Based on our cutoff frequency we decide that the c_{-47} and c_{47} coefficients are artifacts due to noise

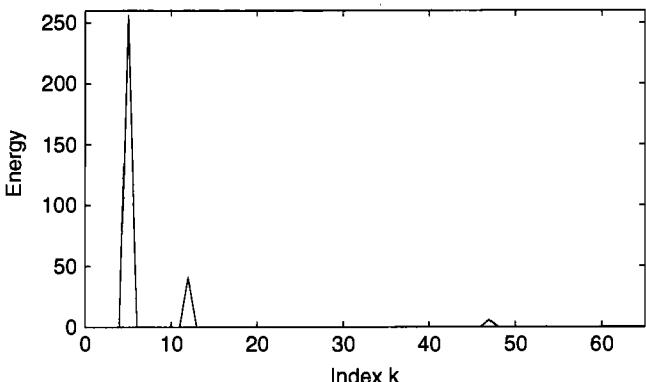


FIGURE 2.5 Spectrum of example signal, frequency energies lumped.

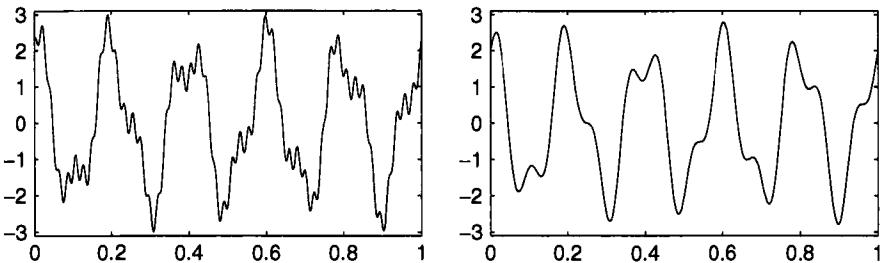


FIGURE 2.6 Noisy and denoised signal.

polluting the signal, and hence we zero them out, a very simple algebraic operation. The other coefficients may also be polluted by noise, but (at least with this approach) there's nothing we can do about it. We now reconstruct the denoised signal $\tilde{\mathbf{x}}$ embodied by the remaining coefficients using equation (2.2), which moves us back to the time domain. A plot of the resulting denoised signal $\tilde{\mathbf{x}}$ is shown in Figure 2.6, along with a plot of the original noisy signal.

In the next section we'll look at the case for a general sampled signal, and in particular, how the coefficients c_k allow us to break a signal up into constituent frequencies and analyze the energy contributed by each.

2.4 THE ONE-DIMENSIONAL DFT

2.4.1 Definition of the DFT

In all that follows we assume that vectors in \mathbb{R}^N or \mathbb{C}^N are indexed from 0 to $N - 1$, for example, $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$. Whenever we use matrix methods, all vectors will be written in a column format. Whether a vector \mathbf{x} is actually obtained by sampling an analog signal is irrelevant for now. Unless otherwise noted, we also use the range $0 \leq k \leq N - 1$ for the basic vectors $\mathbf{E}_{N,k}$.

In the last section we saw that the numbers

$$c_k = \frac{(\mathbf{x}, \mathbf{E}_{N,k})}{(\mathbf{E}_{N,k}, \mathbf{E}_{N,k})} \quad (2.5)$$

in the orthogonal decomposition of \mathbf{x} carry a lot of information about a signal. We now define a transform $\mathbf{x} \rightarrow \mathbf{X}$ on signals, called the *discrete Fourier transform* (DFT), that collects all this information together into one vector.

Definition 2.4.1 Let $\mathbf{x} \in \mathbb{C}^N$ be a vector $(x_0, x_1, \dots, x_{N-1})$. The discrete Fourier transform (DFT) of \mathbf{x} is the vector $\mathbf{X} \in \mathbb{C}^N$ with components

$$X_k = (\mathbf{x}, \mathbf{E}_{N,k}) = \sum_{m=0}^{N-1} x_m e^{-2\pi i km/N} \quad (2.6)$$

for $0 \leq k \leq N - 1$.

Comparison of equations (2.5) and (2.6) shows that $X_k = Nc_k$, since the $\mathbf{E}_{N,k}$ satisfy $(\mathbf{E}_{N,k}, \mathbf{E}_{N,k}) = N$. Traditionally the c_k are themselves called the *Fourier coefficients* of \mathbf{x} , but some texts may use this terminology for X_k . When in doubt, we refer to the appropriate equation!

Remark 2.1 The coefficient $c_0 = X_0/N$ measures the contribution of the constant basic waveform $\mathbf{E}_0 = (1, 1, \dots, 1)$ to \mathbf{x} . In fact

$$c_0 = \frac{X_0}{N} = \frac{1}{N} \sum_{m=0}^{N-1} \mathbf{x}_m$$

is the average value of \mathbf{x} . The coefficient c_0 is usually referred to as the *dc coefficient* because it measures the strength of the constant or “direct current” component of the signal, in an electrical setting.

Remark 2.2 Equation (2.6) actually defines the DFT coefficients X_k for any index k , and the resulting X_k are periodic with period N in the index. We’ll thus sometimes refer to the X_k on other ranges of the index, for example, $-N/2 < k \leq N/2$ when N is even. Actually, even if N is odd, the range $-N/2 < k \leq N/2$ works because k is required to be an integer. Knowledge of X_k on such a range allows us to compute the DFT coefficients for all k .

The original signal \mathbf{x} can be reconstructed from the c_k by using equation (1.33) in Theorem 1.8.3. Since $c_k = X_k/N$, this means we can reconstruct \mathbf{x} from \mathbf{X} .

Definition 2.4.2 Let $\mathbf{X} \in \mathbb{C}^N$ be a vector $(X_0, X_1, \dots, X_{N-1})$. The inverse discrete Fourier transform (IDFT) of \mathbf{X} is the vector $\mathbf{x} = \frac{1}{N} \sum_{m=0}^{N-1} X_m \mathbf{E}_{N,m} \in \mathbb{C}^N$ with components

$$x_k = \frac{(\mathbf{X}, \mathbf{E}_{N,k})}{N} = \frac{1}{N} \sum_{m=0}^{N-1} X_m e^{2\pi i km/N}. \quad (2.7)$$

Note the $1/N$ term in front of the sum in (2.7) used to compensate for the fact that $X_k = Nc_k$. The inverse DFT shows how to synthesize \mathbf{x} from the $\mathbf{E}_{N,k}$, which are the DFT basic waveforms. As mentioned in the first section of this chapter, various definitions of the DFT/IDFT pair exist. Some put a $1/\sqrt{N}$ factor in front of both sums in the DFT and IDFT. (The computer algebra system maple scales transforms like this). In any case, the DFT and IDFT will “undo” each other.

We will occasionally write $\mathbf{X} = DFT(\mathbf{x})$ and $\mathbf{x} = IDFT(\mathbf{X})$ to indicate the application of the DFT or IDFT to a given vector.

We can now be a little more precise about the time and frequency domains for discretized one-dimensional signals. A signal vector $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ in \mathbb{C}^N exists in the time domain. We can write out \mathbf{x} with respect to the standard basis \mathbf{e}_k ,

$0 \leq k \leq N - 1$, as

$$\mathbf{x} = \sum_{k=0}^{N-1} x_k \mathbf{e}_k.$$

The standard basis is most natural for working with signals in the time domain. The vector $\mathbf{X} = DFT(\mathbf{x}) \in \mathbb{C}^N$ is the frequency domain version of the time domain signal \mathbf{x} and shows how to represent \mathbf{x} as a linear combination of the basic waveforms \mathbf{E}_k ,

$$\mathbf{x} = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathbf{E}_k.$$

Vectors in the time and frequency domain are paired one-to-one, via the transform relations $\mathbf{X} = DFT(\mathbf{x})$ and $\mathbf{x} = IDFT(\mathbf{X})$.

2.4.2 Sample Signal and DFT Pairs

In the next section we'll develop some of the mathematical properties of the DFT, but let's pause now to look at a few examples.

An Aliasing Example Consider the analog signal $x(t)$ defined by equation (2.1) in Section 2.3, but this time let's sample $x(t)$ at only 64 Hertz on the interval $0 \leq t \leq 1$. The Nyquist frequency drops to 32 Hertz, below the frequency of the 47 Hertz component of $x(t)$. The sampled signal is the vector $\mathbf{x} \in \mathbb{R}^{64}$ with components $x_k = x(k/64)$, $0 \leq k \leq 63$. In Figure 2.7 we plot $64(|c_{-k}|^2 + |c_k|^2)$ versus k , or equivalently, $(|X_{-k}|^2 + |X_k|^2)/64$ versus k . As in the previous case k corresponds to frequency in Hertz. The 47 Hertz energy now masquerades as energy at $64 - 47 = 17$

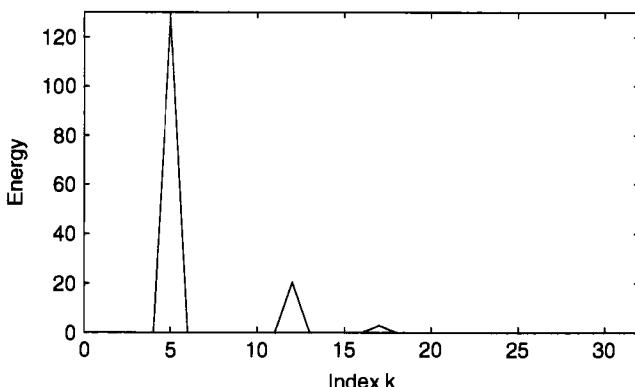


FIGURE 2.7 Spectrum of example signal, with aliasing.

Hertz, since

$$\begin{aligned}\cos\left(\frac{2\pi \cdot 47k}{64}\right) &= \cos\left(\frac{2\pi(-47)k}{64}\right) \\ &= \cos\left(\frac{2\pi(-47+64)k}{64}\right) \\ &= \cos\left(\frac{2\pi \cdot 17k}{64}\right)\end{aligned}$$

for all k . The signal $x(t) = 2.0 \cos(2\pi \cdot 5t) + 0.8 \sin(2\pi \cdot 12t) + 0.3 \cos(2\pi \cdot 47t)$ is thus identical to $\tilde{x}(t) = 2.0 \cos(2\pi \cdot 5t) + 0.8 \sin(2\pi \cdot 12t) + 0.3 \cos(2\pi \cdot 17t)$ when sampled at 64 Hertz.

Square Pulses The previous examples have consisted of a superposition of relatively few basic waveforms. Let's examine some signals with energy at many frequencies.

Consider a discrete square waveform $\mathbf{x} \in \mathbb{R}^N$ with components $x_k = 1$ for $0 \leq k \leq R - 1$ and $x_k = 0$ for $R \leq k \leq N - 1$, where $0 < R < N$. In Exercise 2.11 you are asked to show that the DFT of this signal is given by

$$X_k = \frac{1 - e^{-2\pi i k R / N}}{1 - e^{-2\pi i k / N}}$$

for $k > 0$, while $X_0 = R$. The magnitude of X_k for $k > 0$ is

$$\begin{aligned}|X_k| &= \frac{|1 - e^{-2\pi i k R / N}|}{|1 - e^{-2\pi i k / N}|} \\ &= \frac{((1 - \cos(2\pi k R / N))^2 + \sin^2(2\pi k R / N))^{1/2}}{((1 - \cos(2\pi k / N))^2 + \sin^2(2\pi k / N))^{1/2}} \\ &= \sqrt{\frac{1 - \cos(2\pi k R / N)}{1 - \cos(2\pi k / N)}},\end{aligned}$$

where we make use of $|z_1 z_2| = |z_1| |z_2|$ and $\sin^2(\omega) + \cos^2(\omega) = 1$.

In Figures 2.8 through 2.10 we plot \mathbf{x} and the corresponding points $(k, |X_k|^2 / N)$ for $N = 256$ and several values of R . In each case we use the range $-N/2 < k \leq N/2$ for the DFT, rather than $0 \leq k \leq N - 1$ (recall Remark 2.2 on page 79); this allows us to show more clearly the relation between \mathbf{x} and the DFT \mathbf{X} . In Figure 2.8 we use $R = 50$, in Figure 2.9 we take $R = 10$, and in Figure 2.10 we take $R = 2$.

As the square wave pulse becomes shorter in the time domain, the DFT becomes broader in the frequency domain. This is generally true. Indeed in the extreme case

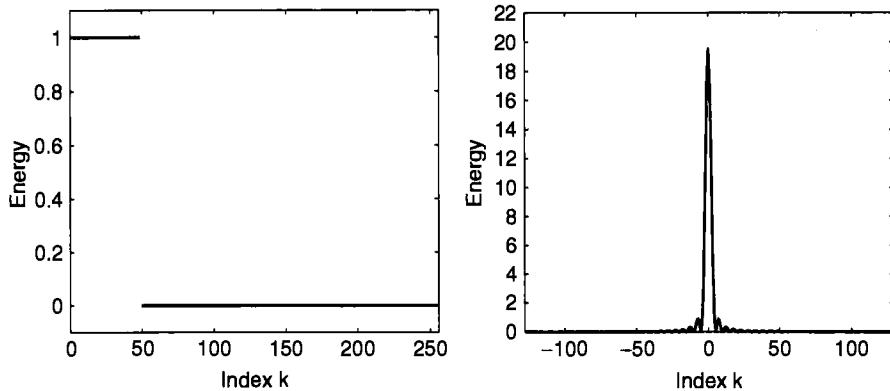


FIGURE 2.8 Time domain square wave and spectrum, $R = 50$.

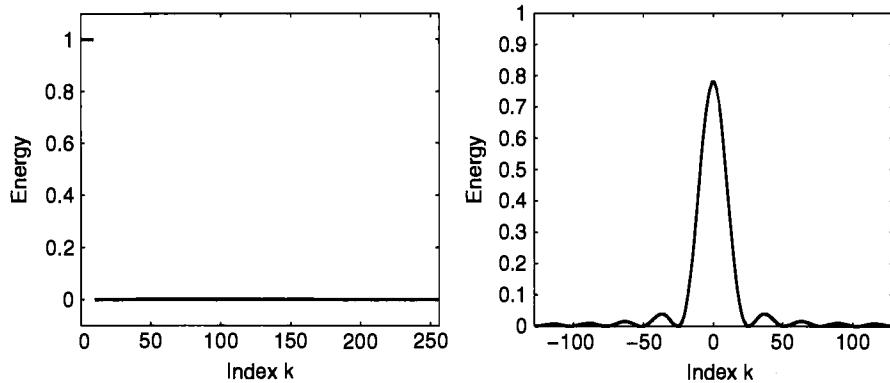


FIGURE 2.9 Time domain square wave and spectrum, $R = 10$.

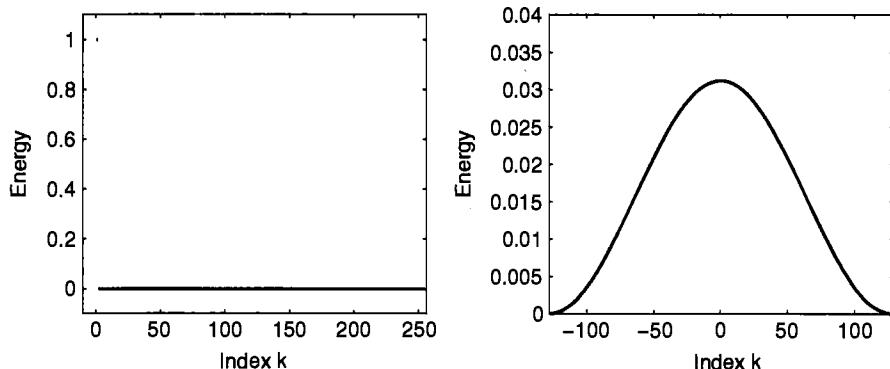


FIGURE 2.10 Time domain square wave and spectrum, $R = 2$.

that $R = 1$ (so $x_0 = 1$ and $x_k = 0$ for $k \geq 1$) we find that the DFT coefficients satisfy $|X_k| = 1$ for all k , so the spectral plot is flat; see Exercise 2.4.

Noise Since we will occasionally have to deal with random noise, it will be helpful to say a bit about what noise looks like, in both the time and frequency domain. However, it is important to point out that almost all of the modeling and mathematics in this text is concerned with deterministic (nonrandom) signals and images. The study of signals that are predominantly random in nature is itself a large area of applied mathematics and statistics, and usually goes under the name “time series analysis.” Although this subject has many similarities and tools in common with the mathematics we’ve presented so far, it is based on somewhat different assumptions about the nature of the signals.

Nonetheless, here’s a simple example of what noise often looks like. Consider a discrete signal $\mathbf{x} \in \mathbb{R}^N$ in which each of the samples x_k is a random variable. More precisely, let us assume that each x_k is a normally distributed random variable with zero mean and variance σ^2 , and that the x_k are independent of each other. We could use any other distribution, but the normality assumption is a common one. (If you haven’t encountered the normal distribution, then the previous statements won’t make sense. In that case, just think of x_k as a “random number” in the range $(-3\sigma, 3\sigma)$.) Equation (2.6) shows that the X_k will themselves be random variables. To illustrate, consider Figure 2.11 below. On the left is the time domain signal of the form described above with $N = 256$ and variance $\sigma^2 = 1$. On the right is plotted the quantity $|X_k|^2/N$ versus k on the range $0 \leq k \leq 255$; note the symmetry about $k = 128$.

Not surprisingly, the spectrum itself has energy at all frequencies. It turns out that this energy is, “on average,” distributed evenly across the frequencies. This kind of random signal is called *white noise*. It can be shown (see Exercise 2.12, if you’ve had a little statistics) that both $\text{Re}(X_k)$ and $\text{Im}(X_k)$ are themselves independent normal random variables, with zero mean and a variance that can be computed explicitly, and that $|X_k|^2/N$ is exponentially distributed.

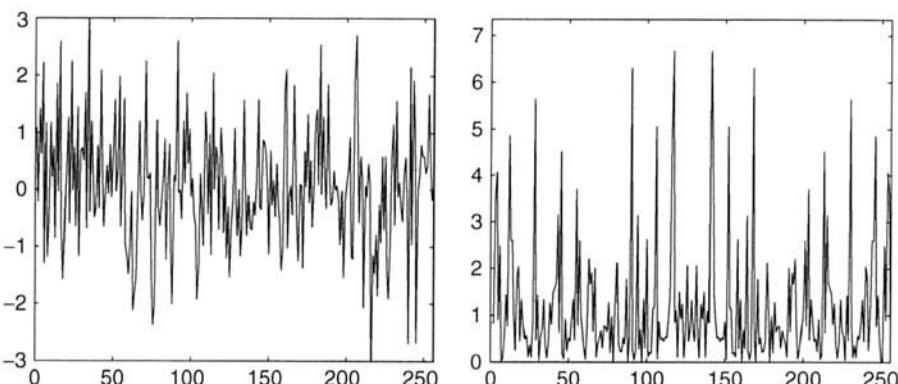


FIGURE 2.11 Random signal and its DFT magnitude.

2.4.3 Suggestions on Plotting DFTs

In graphing DFTs, we often do one or more of the following:

- Plot only the magnitude $|X_k|$, the squared magnitude $|X_k|^2$, or some rescaled versions thereof. In particular, we may not be concerned with the phase or argument of the complex number X_k .
- Make a change of variable by plotting $\log(|X_k|)$, $\log(1 + |X_k|)$, or something similar. This is very helpful when the $|X_k|$ vary widely in magnitude, which is often the case with images.
- Plot the spectral data $(k, |X_k|)$ on whatever k range best illustrates the point we want to make. For purely computational and analytical purposes we usually use $0 \leq k \leq N - 1$, but using a range in k that is symmetric about $k = 0$ is sometimes more informative when graphing. In fact, if all we care about is the energy at any given frequency, can we plot quantities $(|X_{-k}|^2 + |X_k|^2)/N$, $1 \leq k \leq N/2 - 1$, along with $|X_0|^2/N$ and $|X_{N/2}|^2/N$ as in Figure 2.5.

Remark 2.3 Up to this point the horizontal axis on our spectral plots is always indexed by the integer k . It's worth remembering that if \mathbf{x} comes from a sampled analog signal, then each k corresponds to a particular analog frequency, in Hertz. Specifically, let the sampling rate be M samples per second over a time interval of length T seconds. For simplicity assume MT is an integer, so the sample vector has length $N = MT$. The corresponding basic waveforms are then $\mathbf{E}_{N,k}$ from equation (1.22) in Chapter 1. Recall that we may restrict our attention to integers in the range $-N/2 < k \leq N/2$ because of aliasing. The vector $\mathbf{E}_{N,k}$ is obtained by sampling the analog waveform $e^{2\pi i k t/T}$, which has frequency $|k|/T$ Hertz (recall that $e^{2\pi q t}$ makes q cycles per second). As a consequence the discrete waveform $\mathbf{E}_{N,k}$ and hence the index k correspond to a frequency of $|k|/T$ Hertz. The range $0 \leq |k| \leq N/2$ corresponds to a frequency range 0 to $N/2T = M/2$ Hertz, the Nyquist frequency.

2.4.4 An Audio Example

Let us finish this section by applying the DFT to a real audio signal. The train whistle signal graphed on the left in Figure 2.12 comes as a part of the Matlab installation. It is sampled at $M = 8192$ Hertz and is a vector of length $N = 12880$ samples. The time span of the sampled signal is thus $T = N/M \approx 1.5723$ seconds. On the right in Figure 2.12 we graph the lumped frequency energies $(|X_k|^2 + |X_{N-k}|^2)/N$. In light of Remark 2.3 on page 84, any given horizontal coordinate k in the spectral plot below corresponds to frequency $|k|/T \approx 0.636|k|$ Hertz. The right end of the plot corresponds to the Nyquist frequency 4096 Hertz. The DFT makes it clear that essentially three frequencies or closely grouped sets of frequencies dominate the signal.

See the Matlab project later in this chapter, where you can play around with this signal yourself.

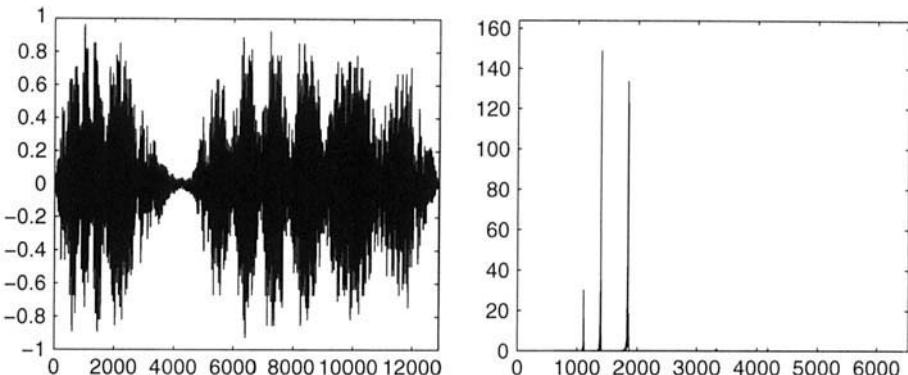


FIGURE 2.12 Train whistle signal and DFT magnitude.

2.5 PROPERTIES OF THE DFT

In this section we demonstrate some important algebraic properties of the discrete Fourier transform.

2.5.1 Matrix Formulation and Linearity

The DFT as a Matrix The application of the DFT to a vector $\mathbf{x} \in \mathbb{C}^N$ can be implemented as a matrix multiplication $\mathbf{x} \rightarrow \mathbf{F}_N \mathbf{x}$ for an appropriate $N \times N$ matrix \mathbf{F}_N . Whenever we use matrix methods to analyze the DFT, it's important to write all vectors in a column format.

Let's start with a look at the specific case $N = 4$, and then proceed to the general case. Let \mathbf{x} be a vector of samples

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

From the definition of the DFT we compute the components X_k of \mathbf{X} as

$$X_k = (\mathbf{x}, \mathbf{E}_{4,k}) = \mathbf{E}_{4,k}^* \mathbf{x},$$

where $\mathbf{E}_{4,k}^*$ is the row vector or 1×4 matrix obtained as the conjugate transpose of $\mathbf{E}_{4,k}$. Of course, $\mathbf{E}_{4,k}^* \mathbf{x}$ is the matrix product of the 1×4 matrix $\mathbf{E}_{4,k}^*$ with the 4×1 matrix (column vector) \mathbf{x} , and this product is just the scalar X_k . For $N = 4$ this

amounts to the equations:

$$X_0 = [\bar{1} \quad \bar{1} \quad \bar{1} \quad \bar{1}] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

$$X_1 = [\bar{1} \quad \bar{i} \quad \bar{-1} \quad \bar{-i}] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

$$X_2 = [\bar{1} \quad \bar{-1} \quad \bar{1} \quad \bar{-1}] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

$$X_3 = [\bar{1} \quad \bar{-i} \quad \bar{-1} \quad \bar{i}] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

In more compact notation (and after we perform the complex conjugation) this is

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad (2.8)$$

or $\mathbf{X} = \mathbf{F}_4 \mathbf{x}$.

To analyze the general N -dimensional case, first recall from linear algebra that if \mathbf{A} is an $N \times N$ matrix and \mathbf{x} an N -dimensional column vector, then the product \mathbf{Ax} is that N -dimensional column vector \mathbf{v} with components

$$v_m = \sum_{k=0}^{N-1} A_{mk} x_k,$$

where we index from 0 to $N - 1$ as usual. Compare this to the definition (2.6) of the DFT—they are identical, with v_m above replaced by X_m and where \mathbf{A} denotes the matrix with row m , column k entry $e^{-2\pi i m k / N}$. We have shown the following theorem:

Theorem 2.5.1 Let $\mathbf{x} \in \mathbb{C}^N$ and $\mathbf{X} = DFT(\mathbf{x})$. Then

$$\mathbf{X} = \mathbf{F}_N \mathbf{x},$$

where \mathbf{F}_N is the $N \times N$ matrix (rows and columns indexed 0 to $N - 1$) with row m , column k entry $e^{-2\pi imk/N}$.

If we let $z = e^{-2\pi i/N}$ (so $e^{2\pi imk/N} = z^{mk}$), then \mathbf{F}_N can be written in the form

$$\mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & z & z^2 & z^3 & \cdots & z^{N-1} \\ 1 & z^2 & z^{2 \cdot 2} & z^{2 \cdot 3} & \cdots & z^{2 \cdot (N-1)} \\ 1 & z^3 & z^{3 \cdot 2} & z^{3 \cdot 3} & \cdots & z^{3 \cdot (N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z^{N-1} & z^{(N-1) \cdot 2} & z^{(N-1) \cdot 3} & \cdots & z^{(N-1) \cdot (N-1)} \end{bmatrix}. \quad (2.9)$$

The matrix \mathbf{F}_N has a lot of structure. This structure can even be exploited to develop an algorithm called the “fast Fourier transform” that provides a very efficient method for computing DFT’s without actually doing the full matrix multiplication. We’ll look at this in Section 2.6.

The Inverse DFT as a Matrix The IDFT defined by equation (2.7) also has a matrix formulation. Indeed essentially identical reasoning to that used for the DFT (with equation (2.7) in place of (2.6)) shows that $\mathbf{x} = \tilde{\mathbf{F}}_N \mathbf{X}$ where $\tilde{\mathbf{F}}_N$ is the $N \times N$ matrix with row m , column k entry $e^{2\pi imk/N}/N$. This matrix is symmetric, like \mathbf{F}_N .

In fact $\tilde{\mathbf{F}}_N$ is easily obtained from \mathbf{F}_N . If we conjugate \mathbf{F}_N entry by entry, we obtain the matrix $N\tilde{\mathbf{F}}_N$ so that $\overline{\mathbf{F}_N} = N\tilde{\mathbf{F}}_N$ or

$$\tilde{\mathbf{F}}_N = \frac{1}{N} \overline{\mathbf{F}_N}.$$

However, it will later be more useful to replace the entry-by-entry conjugation of \mathbf{F}_N with the Hermitian transpose \mathbf{F}_N^* of \mathbf{F}_N , a conjugation followed by transposition. Because \mathbf{F}_N is symmetric, this transposition may seem pointless, but the Hermitian transpose operation is more standard in linear algebra and has some nice properties. In this case we find that

$$\tilde{\mathbf{F}}_N = \frac{1}{N} \mathbf{F}_N^*.$$

We can summarize with the following theorem:

Theorem 2.5.2 Let $\mathbf{x} \in \mathbb{C}^N$ and $\mathbf{X} = DFT(\mathbf{x})$. Let \mathbf{F}_N denote the N -point DFT matrix. Then

$$\begin{aligned}\mathbf{X} &= \mathbf{F}_N \mathbf{x}, \\ \mathbf{x} &= \frac{1}{N} \mathbf{F}_N^* \mathbf{X}.\end{aligned}$$

Of course, this, also shows that $\mathbf{F}_N^{-1} = \frac{1}{N} \mathbf{F}_N^*$.

We haven't stated the following explicitly, but Theorem 2.5.2 (or equations (2.6)–(2.7)) yields a subsequent important fact.

Theorem 2.5.3 The discrete Fourier transform $DFT : \mathbb{C}^N \rightarrow \mathbb{C}^N$ is a linear mapping,

$$DFT(a\mathbf{x} + b\mathbf{y}) = a \cdot DFT(\mathbf{x}) + b \cdot DFT(\mathbf{y})$$

for $\mathbf{x}, \mathbf{y} \in \mathbb{C}^N$ and $a, b \in \mathbb{C}$. The inverse DFT is also linear.

2.5.2 Symmetries for Real Signals

In all the examples we've looked at so far the DFT has possessed obvious symmetries. The symmetries stemmed from the fact that the underlying signals were real-valued. This is quantified in the following proposition, in which we implicitly use the fact that X_k is defined for all k (recall Remark 2.2 on page 79).

Proposition 2.5.1 A vector $\mathbf{x} \in \mathbb{C}^N$ has all components x_r real if and only if $\mathbf{X} = DFT(\mathbf{x})$ has components that satisfy

$$X_{-k} = \overline{X}_k,$$

or equivalently,

$$\operatorname{Re}(X_{-k}) = \operatorname{Re}(X_k) \quad \text{and} \quad \operatorname{Im}(X_{-k}) = -\operatorname{Im}(X_k)$$

for all $k \in \mathbb{Z}$.

Proof Suppose that \mathbf{x} has all components x_r real so that $\bar{x}_r = x_r$. Then

$$\begin{aligned}X_{-k} &= \sum_{r=0}^{N-1} x_r e^{-2\pi i(-k)r/N} \\ &= \sum_{r=0}^{N-1} x_r e^{2\pi ikr/N}\end{aligned}$$

$$\begin{aligned}
 &= \sum_{r=0}^{N-1} \overline{x_r e^{-2\pi i kr/N}} \\
 &= \overline{\sum_{r=0}^{N-1} x_r e^{-2\pi i kr/N}} \\
 &= \overline{X_k}.
 \end{aligned}$$

Here we've used $\overline{x_r} = x_r$, as well as elementary properties of complex conjugation. This shows that $X_{-k} = \overline{X_k}$.

To show the converse, suppose that $X_{-k} = \overline{X_k}$ for all k , so $X_k = \overline{X_{-k}}$. We have from the IDFT that

$$\begin{aligned}
 x_r &= \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{2\pi i kr/N} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} \overline{X_{-k}} e^{2\pi i kr/N} \\
 &= \frac{1}{N} \overline{\sum_{k=0}^{N-1} X_{-k} e^{-2\pi i kr/N}} \\
 &= \frac{1}{N} \overline{\sum_{n=0}^{N-1} X_n e^{2\pi i nr/N}} \quad (\text{substitute } n = -k) \\
 &= \overline{x_r}.
 \end{aligned}$$

Since $x_r = \overline{x_r}$, we conclude that each x_r is real.

The symmetry and antisymmetry of the real and imaginary parts are a simple consequence of the definition of conjugation. ■

Proposition 2.5.1 is really a special case of a more general symmetry for the DFT; see Exercise 2.8.

Remark 2.4 From Proposition 2.5.1, if \mathbf{x} has real components, then $X_{-k} = \overline{X_k}$. Also, since X_k is periodic with period N in k (whether or not \mathbf{x} is real-valued), we have $X_{-k} = X_{N-k}$. We conclude that

$$X_{N-k} = \overline{X_k}. \tag{2.10}$$

In particular, if $k = N/2 + j$ in equation (2.10), then

$$X_{N/2-j} = \overline{X_{N/2+j}}.$$

Thus, if \mathbf{x} is real, X_k is conjugate-symmetric about $N/2$. (If N is odd, this still makes sense if we let $j = m + 1/2$ for $m \in \mathbb{Z}$.) We also conclude that $|X_{N/2-j}| = |X_{N/2+j}|$, which is why for real-valued signals we need only plot $|X_k|$ in the range $0 \leq k \leq N/2$.

2.6 THE FAST FOURIER TRANSFORM

The fast Fourier transform (FFT) is an algorithm, or really a class of algorithms, for computing the DFT and IDFT efficiently. The FFT is one of the most influential algorithms of the twentieth century. The “modern” version of the algorithm was published by Cooley and Tukey [6] in 1965, but some of the ideas appeared earlier. Given the very regular structure of the matrix \mathbf{F}_N , it’s not entirely surprising that shortcuts can be found for computing the matrix-vector product $\mathbf{F}_N \mathbf{x}$. All modern software uses the FFT algorithm for computing discrete Fourier transforms. The details are usually transparent to the user. Thus knowledge of the FFT algorithm is not generally essential for using or understanding DFTs.

There are many of points of view on the FFT. A computer scientist would classify the FFT as a classic “divide and conquer” algorithm. A mathematician might view the FFT as a natural consequence of the structure of certain finite groups. Other practitioners might simply view it as a clever technique for organizing the sums involved in equation (2.6), or factoring the matrix \mathbf{F}_N into simpler pieces. Our intention is merely to give a very brief introduction to the idea behind one form of the algorithm. The material in this section is not essential for anything that follows. For a more comprehensive treatment of the fast Fourier transform, see [4].

2.6.1 DFT Operation Count

Let’s begin by looking at the computational cost of evaluating the sums in equation (2.6), or equivalently, computing the matrix-vector product of Theorem 2.5.1. Let $\mathbf{x} \in \mathbb{C}^N$, and let \mathbf{X} denote the DFT of \mathbf{x} . How many floating point operations does it take to compute \mathbf{X} from \mathbf{x} using equation (2.6)? The computation of any given X_k requires N complex multiplications (the product $x_m e^{-2\pi i km/N}$ for $0 \leq m \leq N - 1$) followed by $N - 1$ additions, a total of $2N - 1$ operations. Since we have to compute X_k for $0 \leq k \leq N - 1$, we must perform $N(2N - 1) = 2N^2 - N$ operations for this straightforward approach, where an “operation” means a complex addition or multiplication. Of course, this ignores the cost of computing the exponentials. However, if we let $z = e^{-2\pi i/N}$, then all exponentials are of the form z^m for m in the range $0 \leq m \leq N - 1$, and these can be pre-computed and tabled. We’ll thus ignore this computational cost, which, at any rate, is only on the order of N operations and insignificant with respect to N^2 .

All in all, the straightforward approach to the DFT requires $2N^2 - N$ operations, or roughly just $2N^2$ if N is large. If the signal \mathbf{x} is real, then there are some tricks that can be performed to lower the operation count, but the computational cost is still proportional to N^2 .

2.6.2 The FFT

The fast Fourier transform cuts the work for computing an N -sample DFT from $2N^2$ floating point operations down to $CN \log(N)$ operations for some constant C , a significant savings if N is large or many transforms have to be computed quickly. We'll consider one of the most common varieties of the many FFT algorithms, and restrict our attention to the special but fairly typical case where $N = 2^n$ for some positive integer n .

Our first goal is to split the task of computing an N -point DFT into the computation of two transforms of size $N/2$. For the k index appearing in equation (2.6), we can write $k = (N/2)k_1 + k_0$, where $k_1 = 0$ or $k_1 = 1$ and $0 \leq k_0 < N/2$. Of course, as k_0 and k_1 range over these limits, k ranges from 0 to $N - 1$. So we write the DFT in equation (2.6) as

$$\begin{aligned} X_{(N/2)k_1+k_0} &= \sum_{m=0}^{N-1} x_m e^{-2\pi i ((N/2)k_1+k_0)m/N} \\ &= \sum_{m=0}^{N-1} x_m e^{-\pi i k_1 m} e^{-2\pi i k_0 m/N} \\ &= \sum_{m=0}^{N-1} x_m (-1)^{k_1 m} e^{-2\pi i k_0 m/N}, \end{aligned} \quad (2.11)$$

where we've split the exponential and used $e^{-\pi i k_1 m} = (-1)^{k_1 m}$. Next we split the sum over m in equation (2.11) into even and odd indexes to obtain

$$\begin{aligned} X_{(N/2)k_1+k_0} &= \sum_{m=0}^{N/2-1} x_{2m} (-1)^{2k_1 m} e^{-2\pi i k_0 (2m)/N} \quad (\text{even indexes}) \\ &\quad + \sum_{m=0}^{N/2-1} x_{2m+1} (-1)^{k_1 (2m+1)} e^{-2\pi i k_0 (2m+1)/N} \quad (\text{odd indexes}) \end{aligned}$$

A bit of simplification yields

$$\begin{aligned} X_{(N/2)k_1+k_0} &= \sum_{m=0}^{N/2-1} x_{2m} e^{-2\pi i k_0 m/(N/2)} \\ &\quad + \sum_{m=0}^{N/2-1} x_{2m+1} (-1)^{k_1} e^{-2\pi i k_0 /N} e^{-2\pi i k_0 m/(N/2)} \end{aligned}$$

$$\begin{aligned}
&= \sum_{m=0}^{N/2-1} x_{2m} e^{-2\pi i k_0 m / (N/2)} \\
&+ (-1)^{k_1} e^{-2\pi i k_0 / N} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-2\pi i k_0 m / (N/2)}. \quad (2.12)
\end{aligned}$$

Let's use $F_1(k_0)$ and $F_2(k_0)$ to denote the first and second sums on the right in equation (2.12), that is,

$$F_1(k_0) = \sum_{m=0}^{N/2-1} x_{2m} e^{-2\pi i k_0 m / (N/2)} \text{ and } F_2(k_0) = \sum_{m=0}^{N/2-1} x_{2m+1} e^{-2\pi i k_0 m / (N/2)}.$$

Note that F_1 (as k_0 ranges from 0 to $N/2 - 1$) is exactly the DFT of the $N/2$ -component vector $(x_0, x_2, x_4, \dots, x_{N-2})$. Likewise F_2 is exactly the DFT of the $N/2$ -component vector $(x_1, x_3, x_5, \dots, x_{N-1})$.

We can now write equation (2.12) as

$$X_{(N/2)k_1+k_0} = F_1(k_0) + (-1)^{k_1} e^{-2\pi i k_0 / N} F_2(k_0). \quad (2.13)$$

Remember, $0 \leq k_0 < N/2$ and $k_1 = 0$ or $k_1 = 1$. In the case where $k_1 = 0$, equation (2.13) yields

$$X_{k_0} = F_1(k_0) + e^{-(2\pi i / N)k_0} F_2(k_0), \quad (2.14)$$

while $k_1 = 1$ in (2.13) gives

$$X_{N/2+k_0} = F_1(k_0) - e^{-(2\pi i / N)k_0} F_2(k_0). \quad (2.15)$$

In short, if we compute the DFT of the even and odd indexed entries of \mathbf{x} , equations (2.14) and (2.15) tell us how to combine this information to get the full DFT of \mathbf{x} ; equation (2.14) yields the first $N/2$ components of \mathbf{X} and (2.15) yields the last $N/2$. An N -point transform is thus reduced to the computation and appropriate mixing of two $N/2$ -point transforms.

2.6.3 The Operation Count

What's the operation count for the full N -point transform with this approach? Let $W_{N/2}$ denote the number of operations necessary to compute each transform F_1 and F_2 , both of size $N/2$. The computation of F_1 and F_2 together requires a total of $2W_{N/2}$ operations. Also equations (2.14) and (2.15) require a total of N multiplications and N additions to weave together the full N -point transform. Thus, if W_N denotes the

number of operations for the full N -point transform, we have

$$W_N = 2N + 2W_{N/2}.$$

But we can now apply the same approach recursively to compute each $N/2$ -point transform, to conclude that $W_{N/2} = N + 2W_{N/4}$. Similarly $W_{N/4} = N/2 + 2W_{N/8}$, and so on. We conclude that

$$W_N = 2N + 2\left(\frac{N}{2} + 2\left(\frac{N}{4} + 2\left(\frac{N}{8} + \cdots + 2W_1\right)\right)\right) = 2(N + N + \cdots + N),$$

where there are a total of $\log_2(N)$ terms in the sum. The net result is an algorithm that takes about $2N \log_2(N)$ complex operations.

The FFT is usually implemented without explicit recursion; see [4] for various practical implementations. The FFT is not limited to N that are powers of 2. In general, an efficient FFT algorithm can be derived most easily when N is “highly composite,” that is, factors completely into small integers.

2.7 THE TWO-DIMENSIONAL DFT

Let $\mathbf{A} \in M_{m,n}(\mathbb{C})$; we might consider \mathbf{A} as a sampled image. The general framework developed in Chapter 1 makes it easy to define the two-dimensional DFT of \mathbf{A} . We simply let the basic two-dimensional waveforms $\mathcal{E}_{m,n,k,l}$ take over for the one-dimensional waveforms $\mathbf{E}_{n,k}$. In what follows we write $\mathcal{E}_{k,l}$ instead of $\mathcal{E}_{m,n,k,l}$ for the two-dimensional waveforms, since m and n remain fixed in any given computation.

Recall that the $\mathcal{E}_{k,l}$ for $0 \leq k \leq m - 1, 0 \leq l \leq n - 1$ form an orthogonal basis for $M_{m,n}(\mathbb{C})$ with respect to the usual inner product. This allowed us to develop the expansion of equation (1.35). A slight variation on this yields the two-dimensional DFT.

Definition 2.7.1 Let $\mathbf{A} \in M_{m,n}(\mathbb{C})$ have components a_{rs} , $0 \leq r \leq m - 1, 0 \leq s \leq n - 1$. The two-dimensional discrete Fourier transform of \mathbf{A} is the matrix $\hat{\mathbf{A}} \in M_{m,n}(\mathbb{C})$ with components

$$\hat{a}_{k,l} = (\mathbf{A}, \mathcal{E}_{k,l}) = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} e^{-2\pi i (kr/m + ls/n)}, \quad (2.16)$$

where $0 \leq k \leq m - 1, 0 \leq l \leq n - 1$.

The definition of $\hat{a}_{k,l}$ above differs from the coefficients $(\mathbf{A}, \mathcal{E}_{k,l})/mn$ of equation (1.35) only in that $\hat{a}_{k,l}$ has no $1/mn$ factor in front. In general, we'll use a “hat” symbol over a matrix \mathbf{A} to denote the two-dimensional DFT of \mathbf{A} .

In light of equation (1.35) and the fact that the $\mathcal{E}_{k,l}$ form an orthogonal basis, we can reconstruct \mathbf{A} using the two-dimensional inverse DFT defined as follows:

Definition 2.7.2 Let $\widehat{\mathbf{A}} \in M_{m,n}(\mathbb{C})$ be a matrix with components $\widehat{a}_{k,l}$. The two-dimensional inverse discrete Fourier transform of $\widehat{\mathbf{A}}$ is the matrix

$$\mathbf{A} = \frac{1}{mn} \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} \widehat{a}_{k,l} \mathcal{E}_{k,l}$$

in $M_{m,n}(\mathbb{C})$ with components

$$a_{r,s} = \frac{1}{mn} \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} \widehat{a}_{k,l} e^{2\pi i(kr/m+ls/n)}. \quad (2.17)$$

The $1/mn$ factor is placed in the IDFT definition. Like the one-dimensional DFT/IDFT, definitions may vary slightly from text to text.

Remark 2.5 It's clear from equation (2.16) that the component $\widehat{a}_{k,l}$ is defined for any integers k and l . Moreover, since $e^{-2\pi i(kr/m+ls/n)}$ is periodic in k with period m and in l with period n , the same is true for $\widehat{a}_{k,l}$, so $\widehat{a}_{k,l} = \widehat{a}_{k+pm,l+qn}$ for any integers p and q . As a result we may, as in the case of the one-dimensional DFT, consider $\widehat{a}_{k,l}$ on alternate ranges of k and l . One common variation is to use the range $-m/2 < k \leq m/2$, $-n/2 < l \leq n/2$.

Recall from Chapter 1 that each basic waveform $\mathcal{E}_{k,l}$ has a certain “frequency” and “direction” (recall the figures in Section 1.5.2). The two-dimensional DFT of an image indicates how much of each basic waveform is necessary to synthesize the image as a superposition of basic waveforms. The “low-frequency” Fourier coefficients (small k and l) stem from slowly varying features and patterns in the image, such as the blue sky in an outdoor picture. Rapidly varying features such as a striped shirt or other fine textures contribute to Fourier coefficients with larger indexes.

The two-dimensional DFT also has a matrix formulation.

Proposition 2.7.1 The two-dimensional DFT in equation (2.16) may be computed as

$$\widehat{\mathbf{A}} = \mathbf{F}_m \mathbf{A} \mathbf{F}_n = \mathbf{F}_m \mathbf{A} \mathbf{F}_n^T,$$

where \mathbf{F}_m and \mathbf{F}_n are the DFT matrices of Theorem 2.5.1.

Proof Consider the matrix product $\mathbf{F}_m \mathbf{A}$; note \mathbf{F}_m is $m \times m$, while \mathbf{A} is $m \times n$. The product is an $m \times n$ matrix with row k , column s entry $(\mathbf{F}_m \mathbf{A})_{k,s}$ given by

$$(\mathbf{F}_m \mathbf{A})_{k,s} = \sum_{r=0}^{m-1} a_{r,s} e^{-2\pi i kr/m}. \quad (2.18)$$

This follows from the definition of matrix multiplication, where note that $e^{-2\pi i kr/m}$ is the row k , column r entry of \mathbf{F}_m .

Now consider the product $(\mathbf{F}_m \mathbf{A}) \mathbf{F}_n$. The matrix $(\mathbf{F}_m \mathbf{A})$ is $m \times n$, while \mathbf{F}_n is $n \times n$. The product $(\mathbf{F}_m \mathbf{A}) \mathbf{F}_n$ is thus $m \times n$. The row s , column l entry of \mathbf{F}_n is $e^{-2\pi i ls/n}$. If we make use of equation (2.18), we find that the row k , column l entry of the product $(\mathbf{F}_m \mathbf{A}) \mathbf{F}_n$ is given by

$$\begin{aligned} (\mathbf{F}_m \mathbf{A} \mathbf{F}_n)_{k,l} &= \sum_{s=0}^{n-1} (\mathbf{F}_m \mathbf{A})_{k,s} e^{-2\pi i ls/n} \\ &= \sum_{s=0}^{n-1} \sum_{r=0}^{m-1} a_{r,s} e^{-2\pi i kr/m} e^{-2\pi i ls/n} \\ &= \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} e^{-2\pi i (kr/m + ls/n)}. \end{aligned}$$

This is precisely the row k , column l entry of the $m \times n$ matrix $\widehat{\mathbf{A}} = DFT(\mathbf{A})$, so $\widehat{\mathbf{A}} = \mathbf{F}_m \mathbf{A} \mathbf{F}_n$. The formula $\widehat{\mathbf{A}} = \mathbf{F}_m \mathbf{A} \mathbf{F}_n^T$ follows from the fact that $\mathbf{F}_k^T = \mathbf{F}_k$. ■

Proposition 2.7.1 also has a nice algorithmic interpretation. First note these facts:

- If \mathbf{B} is an $m \times n$ matrix, then the operation $\mathbf{B} \rightarrow \mathbf{F}_m \mathbf{B}$ performs an m -point one-dimensional DFT on each column of \mathbf{B} .
- The operation $\mathbf{B} \rightarrow \mathbf{B} \mathbf{F}_n^T$ performs an n -point one-dimensional DFT on each row of \mathbf{B} , and leaves each transformed row as a row vector. This is easy to see by noting that \mathbf{B}^T swaps the rows and columns of \mathbf{B} so that $\mathbf{F}_n \mathbf{B}^T$ performs an n -point transform on each column of \mathbf{B}^T (i.e., rows of \mathbf{B}). If we now transpose back, the transformed columns are swapped back into rows, and the result is $(\mathbf{F}_n \mathbf{B}^T)^T = \mathbf{B} \mathbf{F}_n^T$.

According to the remarks above then, the operation

$$\mathbf{A} \rightarrow \mathbf{F}_m \mathbf{A}$$

performs a one-dimensional m -point DFT on each column of the matrix \mathbf{A} . The operation

$$(\mathbf{F}_m \mathbf{A}) \rightarrow (\mathbf{F}_m \mathbf{A}) \mathbf{F}_n^T$$

then transforms the result row by row. From Proposition 2.7.1 the result is precisely the two-dimensional DFT of \mathbf{A} . Thus the two-dimensional DFT can be computed by transforming \mathbf{A} column by column, and then transforming the result row by row. We

can also perform a two-dimensional DFT by first transforming the rows and then the columns. This corresponds to associating the product $\mathbf{F}_m \mathbf{A} \mathbf{F}_n^T$ as $\mathbf{F}_m (\mathbf{A} \mathbf{F}_n^T)$.

Like the one-dimensional DFT, the two-dimensional DFT is also linear and the DFT of a real-valued matrix possesses many symmetries; see Exercise 2.19. There are also two-dimensional versions of the FFT. Indeed, since the two-dimensional DFT can be computed as a sequence of one-dimensional DFTs, the one-dimensional FFT has an easy extension to two dimensions (though a tailor-made two-dimensional FFT algorithm may be slightly more efficient).

2.7.1 Interpretation and Examples of the 2D DFT

In Figure 2.13 we show a plotted image (336×448 pixels) and several renditions of the magnitude of the DFT for this image. We have taken the logarithm of the absolute value of the Fourier transform in the upper right and lower left pictures. The reason is that there are a few relatively large coefficients and many that are quite small. For natural images the dominant frequencies are almost always low (small k and l), since most of the energy is concentrated in the broad features of the image. In plotting $\log(1 + |\hat{a}_{k,l}|)$, we are able to see patterns over several orders of magnitude. We take $\log(1 + |\hat{a}_{k,l}|)$ rather than $\log(|\hat{a}_{k,l}|)$ to avoid taking the log of 0.

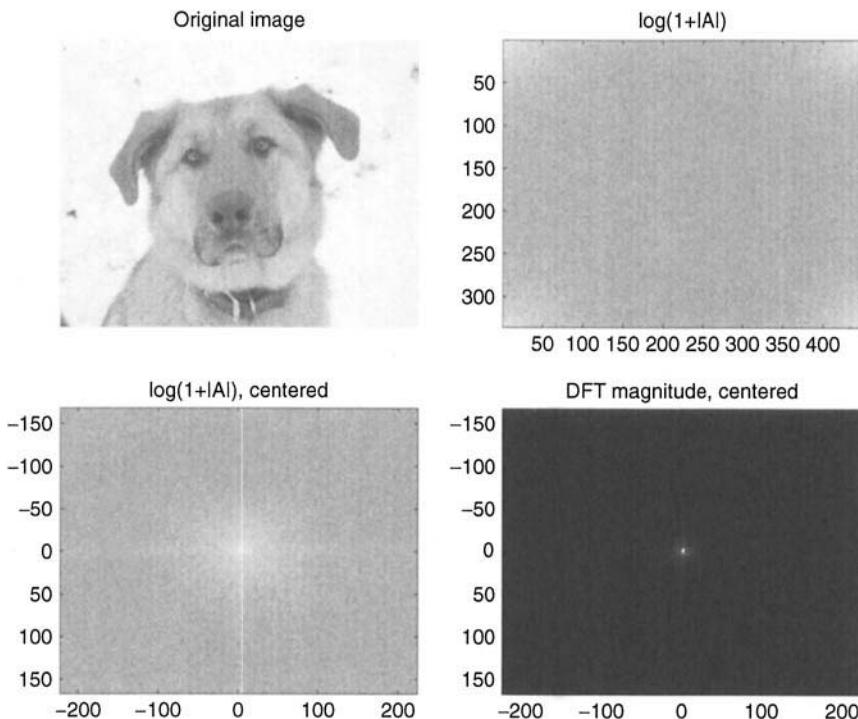


FIGURE 2.13 Image and various presentations of the DFT magnitude.

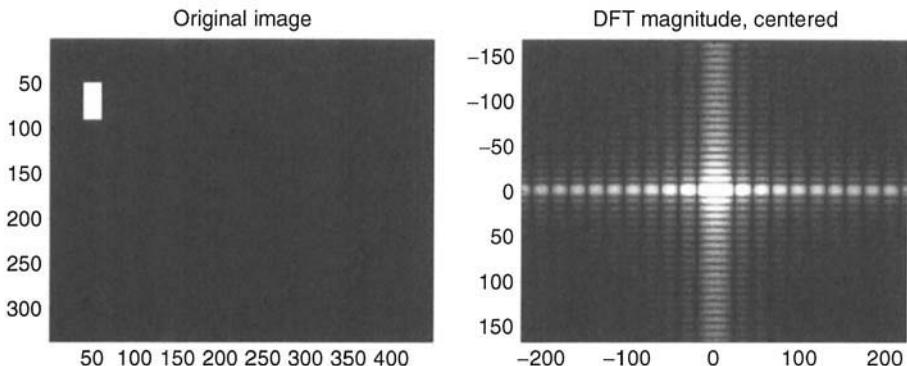


FIGURE 2.14 Artificial image and DFT magnitude $\log(1 + |\hat{a}_{r,s}|)$

Note that in the bottom two figures we have “centered” the DFT by using the range $-m/2 \leq k \leq m/2$, $-n/2 \leq l \leq n/2$. The centered transform puts low frequencies with lots of energy in the middle and higher frequencies with lower energy at the boundary. This is especially clear from the lower right figure, in which we do not take the logarithm. All the energy is obviously concentrated in the very lowest frequencies near the center of the plot, and the remaining frequencies are practically absent.

Our last example in Figure 2.14 is an artificially generated image that illustrates how the DFT reacts to discontinuities. The image is of a white rectangle on a black background. It is a product of rectangular pulses, namely $A_{r,s} = x_r y_s$ where x and y are two square-wave pulse functions similar to the square wave example in Section 2.4.2. It can be shown (see Exercise 2.14) that the coefficients of $\widehat{\mathbf{A}} = DFT(\mathbf{A})$ are the products $\hat{a}_{k,l} = X_k Y_l$ where X and Y are the one-dimensional DFT’s of x and y . Since each of X and Y looks like the DFTs in Figures 2.8 through 2.10, we see can why $\widehat{\mathbf{A}}$ looks like it does. In general, discontinuities or other abrupt features generate a lot of energy at high frequencies.

2.8 MATLAB PROJECT

This section contains Matlab explorations that make use of the DFT for analyzing signals and images.

2.8.1 Audio Explorations

1. Start Matlab. Load in the “train” signal with the command `load ('train');`. Recall that the audio signal is loaded into a variable “y” and the sampling rate into “Fs.” The sampling rate is 8192 Hertz, and the signal contains 12,880 samples. If we consider this signal as sampled on an interval $[0, T]$, then $T = 12880/8192 \approx 1.5723$ seconds.
2. Compute the DFT of the signal with `Y=fft (y);`. Display the magnitude of the Fourier transform with `plot (abs (Y))`. The DFT should have length 12,880 and be symmetric about the center.

Since Matlab indexes from 1, the DFT coefficient Y_k as defined by equation (2.6) is actually $Y(k+1)$ in Matlab. Also, according to Remark 2.3 on page 84, Y_k corresponds to frequency $k/1.5723$, and so $Y(k)$ corresponds to frequency $(k - 1)/1.5723$ Hertz.

3. You can plot only the first half of the DFT with `plot (abs (Y(1:6441)))`. Do so. Use the data cursor button on the plot window to pick out the frequency and amplitude of the three (obviously) largest frequencies in the train signal. Compute the actual value of each frequency in Hertz.
4. Let f_1 , f_2 , and f_3 denote these largest frequencies, in Hertz, and let A_1 , A_2 , A_3 denote the corresponding amplitudes from the plot above. Define these variables in Matlab. Synthesize a new signal using only these frequencies, sampled at 8192 Hertz on the interval $[0, 1.5]$, with

```
t = [0:1/8192:1.5];
ysynth = (A1 * sin(2 * pi * f1 * t) + A2 * sin(2 * pi * f2 * t)
+ A3 * sin(2 * pi * f3 * t)) / (A1+A2+A3);
```

The division by $(A_1 + A_2 + A_3)$ guarantees that the synthesized signal `ysynth` lies in the range $[-1, 1]$, which is the range Matlab uses for audio signals.

5. Play the original train sound with `sound(y)`, and the synthesized version of only three frequencies with `sound(ysynth)`. Note that our computations do not take into account the phase information at these frequencies, merely the amplitude. Does the artificially generated signal capture the tone of the original?
6. Here is a simple approach to compressing an audio or other one-dimensional signal. The idea is to transform the audio signal to the frequency domain with the DFT. We then eliminate the insignificant frequencies by “thresholding,” that is, zeroing out any Fourier coefficients below a given threshold. This becomes the compressed version of the signal. To recover an approximation to the signal, we use the IDFT to take the thresholded transform back to the time domain.

For the train audio signal we can threshold as follows: First, we compute the maximum value of $|Y_k|$, with

```
M = max (abs (Y))
```

Then choose a threshold parameter `thresh` between 0 and 1. Let's start with

```
thresh = 0.1
```

Finally, we zero out all frequencies in `Y` that fall below a value `thresh*M` in magnitude. This can be done with

```
Ythresh = (abs(Y)>thresh*M).*Y;
```

which installs the thresholded transform into “Ythresh.” You should plot the thresholded transform with `plot (abs (Ythresh))`, just to make sure it worked. You can also see what fraction of the Fourier coefficients survived the cut with

```
sum (abs (Ythresh) >0) /12,880.
```

We’ll call this the “compression ratio.”

To recover an approximation to the original signal inverse transform with

```
ythresh = real(ifft(Ythresh));
```

and play the “compressed” audio with `sound(ythresh)`. The “real” command above truncates any vestigial imaginary round-off error in the `ifft` command.

You can compute the distortion of the compressed signal with

```
100 * norm(y-ythresh)^2/norm(y)^2
```

Repeat the computations above for threshold values 0.001, 0.01, 0.1, and 0.5. In each case compute the compression ratio, the distortion, and of course, play the audio signal and rate its quality.

2.8.2 Images

1. Close any open graphics windows and clear all Matlab variables with `clear`. Then load an image into Matlab with

```
z=imread('myimage.jpg');
```

where “myimage.jpg” is any image you want (loaded from the current working directory). Construct a grayscale image with

```
zg = 0.2989 * double(z(:,:,1)) + 0.5870 * double(z(:,:,2))  
+ 0.1140 * double(z(:,:,3));
```

and the colormap

```
L = 255;  
colormap([(0:L)/L; (0:L)/L; (0:L)/L]');
```

View the image with `image (zg)`.

2. Compute the two-dimensional DFT of `zg` with

```
Z = fft2(zg);
```

You can view the DFT itself as an image as follows: First, as discussed in the text, we'll look at $\log(1 + |Z|)$ in order to better see components that vary over many orders of magnitude. Thus let

$$Zlog = \log(1+abs(Z));$$

In order to properly scale the image, it's helpful to compute the maximum value of Zlog with

$$M = max(max(Zlog)).$$

Finally, we can view the transform (using the grayscale colormap set up previously) with

$$image(255 * Zlog/M)$$

You should be able to see certain symmetries in the two-dimensional DFT, but otherwise, the data are difficult to interpret. The original image certainly isn't evident.

3. As for audio signals, we can try a simple image compression technique based on thresholding the DFT (still stored in the variable Z).

To begin we compute the maximum value of $|Z_{k,l}|$, with

$$M = max(max(abs(Z)))$$

We also choose a threshold parameter thresh between 0 and 1. Let's start with

$$thresh = 0.0001$$

We zero out all frequencies in Z that fall below a value thresh*M in magnitude. This is done with

$$Zthresh = (abs(Z) > thresh * M) .* Z;$$

which installs the thresholded transform into "Zthresh." You can see what fraction of the Fourier coefficients survived the cut with

$$sum(sum(abs(Zthresh) > 0)) / m/n$$

where m and n are the pixel counts in each direction. As in the audio case we'll call this the "compression ratio."

To uncompress and view the image, inverse transform with

$$zthresh = real(ifft2(Zthresh));$$

and view with `image(zthresh)`. As in the audio case the `real` command above truncates any vestigial imaginary round-off error in the `ifft2` command.

You can compute the distortion of the compressed signal with

```
100 * norm(zg-zthresh, 'fro')^2/norm(zg, 'fro')^2
```

Repeat the computation above for threshold values 0.001, 0.01, and 0.05, or even larger values. In each case compute the compression ratio, the distortion, display the image, and rate its quality. Construct a plot of the distortion (vertical axis) versus compression ratio. It may be helpful to use a logarithmic scale on one or both axes.

EXERCISES

DFT Computation

- 2.1 Write out the matrix \mathbf{F}_2 that governs the 2-point DFT. Verify explicitly that $\frac{1}{2}\mathbf{F}_2^* = \mathbf{F}_2^{-1}$.
- 2.2 Use the matrix \mathbf{F}_4 to compute the DFT of the vector $\mathbf{x} = (1, 2, 0, -1)$.
- 2.3 Use the matrix \mathbf{F}_4^* to compute the inverse DFT of the vector $\mathbf{X} = (3, 1 + i, 1, 1 - i)$.
- 2.4 Let $\mathbf{e}_k \in \mathbb{C}^N$ denote standard basis vector with a “1” in the k th position, zeroes elsewhere (components indexed $0 \leq k \leq N - 1$). Find the DFT of \mathbf{e}_k , and also the magnitude of each DFT coefficient.
- 2.5 Let N be even. Compute the DFT of the vector $\mathbf{x} \in \mathbb{C}^N$ with components $x_k = (-1)^k$, $0 \leq k \leq N - 1$. Hint: $-1 = e^{i\pi}$.
- 2.6 Let $\mathbf{e}_{j,k}$ denote an $m \times n$ matrix that has a “1” in the row j , column k position and zeroes elsewhere. Find the two-dimensional DFT of $\mathbf{e}_{j,k}$, and also the magnitude of each DFT coefficient.
- 2.7 In this exercise we show that $\mathbf{F}_N^4 = N^2\mathbf{I}$, where \mathbf{F}_N is the $N \times N$ DFT matrix and \mathbf{I} is the $N \times N$ identity matrix. Thus the DFT operation is, up to a scalar multiple, periodic with period 4.
All matrices below should have columns and rows indexed from 0 to $N - 1$.
 - a. Use the definition of matrix multiplication to show that row j , column m entry of \mathbf{F}_N^2 is given by

$$(\mathbf{F}_N)_{j,m}^2 = 0$$

if $j + m \neq 0$ and $j + m \neq N$. Hint: use $1 + z + z^2 + \dots + z^{N-1} = (1 - z^N)/(1 - z)$.

Show that if $j + m = 0$ or $j + m = N$, then $(\mathbf{F}_N)_{j,m}^2 = N$.

You may find it helpful to explicitly write out \mathbf{F}_4^2 , in order to see the pattern.

- b. Use part (a) to show that $\mathbf{F}_N^4 = N^2 \mathbf{I}$. It may be helpful to note that \mathbf{F}_N^2 is symmetric, and each column or row of \mathbf{F}_N^2 has only one nonzero entry.

2.8 Let \mathbf{x} and \mathbf{y} be vectors in \mathbb{C}^N .

- a. Show that $y_r = \overline{x_r}$ for $0 \leq r \leq N - 1$ if and only if the DFT coefficients satisfy $Y_m = \overline{X_{-m}}$.
- b. Deduce Proposition 2.5.1 as a special case of part (a).

dc Coefficients

2.9 Let $\mathbf{b} = \mathbf{a} - \text{mean}(\mathbf{a})$ denote the grayscale image \mathbf{b} in which the average value \mathbf{a} is subtracted from each entry of \mathbf{a} . Show that $B_{0,0} = 0$ (where $\mathbf{B} = DFT(\mathbf{b})$) but argue that \mathbf{b} and \mathbf{a} encode the same image pattern.

Energy and Inner Products

2.10 Show that if $\mathbf{x} \in \mathbb{C}^N$ and $\mathbf{X} = DFT(\mathbf{x})$, then

$$N \|\mathbf{x}\|^2 = \|\mathbf{X}\|^2$$

where we use the usual norm on \mathbb{C}^N . *Hint:* It's a simple variation on Parseval's identity.

Pulses

2.11 Consider a discrete square waveform $\mathbf{x} \in \mathbb{R}^N$ with components $x_k = 1$ for $0 \leq k \leq R - 1$ and $x_k = 0$ for $R \leq k \leq N - 1$, where $0 < R < N$.

- a. Show that the DFT of this pulse is given by

$$X_m = \frac{1 - e^{-2\pi i m R/N}}{1 - e^{-2\pi i m/N}}$$

for $m > 0$, while $X_0 = R$. *Hint:* X_0 is easy. For X_m with $m > 0$ write out the DFT sum and use the geometric summation formula $1 + z + z^2 + \cdots + z^n = (1 - z^{n+1})/(1 - z)$.

- b. Plot the phase of X_m as a function of m for the case $N = 16$. What do you notice? How does the result depend on R ?

Noise

2.12 Warning! This exercise requires a little background in statistics or probability, specifically, that if x_0, x_1, \dots, x_{N-1} are *independent* normal random variables (real-valued) with mean 0 and variance σ^2 , then $x = \sum_k a_k x_k$ is also a normal random variable, with mean 0 and variance $\sigma^2 \sum_k a_k^2$ (where the a_k are fixed real numbers). Also, by these assumptions, the random variables $x = \sum_k a_k x_k$ and $y = \sum_k b_k x_k$ will be independent if $\sum_k a_k b_k = 0$.

Consider a “noise signal” $\mathbf{x} \in \mathbb{R}^N$ as in Section 2.4.2, in which the x_k form a collection of independent normal random variables, each with mean 0 and variance σ^2 . Let \mathbf{X} denote the DFT of \mathbf{x} .

- a. Show that both $\operatorname{Re}(\mathbf{X}_m)$ and $\operatorname{Im}(\mathbf{X}_m)$ are normal random variables for $0 \leq m \leq N - 1$, and that both have mean zero and variance $N\sigma^2/2$ for $m > 0$, while the mean and variance of X_0 are 0 and $N\sigma^2$, respectively. Hint: From $\cos^2(z) = \frac{1}{2} + \frac{1}{2}\cos(2z)$ we have

$$\sum_{k=0}^{N-1} \cos^2\left(\frac{2\pi km}{N}\right) = \frac{N}{2} + \frac{1}{2} \sum_{k=0}^{N-1} \cos\left(\frac{4\pi km}{N}\right).$$

Show the sum on the right above is zero. A similar identity $\sin^2(z) = \frac{1}{2} - \frac{1}{2}\cos(2z)$ holds for the sine.

- b. Show that $\operatorname{Re}(\mathbf{X}_m)$ and $\operatorname{Im}(\mathbf{X}_m)$ are independent.
c. Find the probability distribution function for the random variable $(\operatorname{Re}(\mathbf{X}_m)^2 + \operatorname{Im}(\mathbf{X}_m)^2)/N$ for $m > 0$, and show that the expected value of $(\operatorname{Re}(\mathbf{X}_m)^2 + \operatorname{Im}(\mathbf{X}_m)^2)/N$ (the energy for waveform \mathbf{E}_m) is the same for all $m > 0$. Hence “on average” this signal has a flat spectrum. That’s why such signals are called *white noise*—all frequencies are present in an equal amount.

2D DFTs

- 2.13** Compute (by hand) the two-dimensional DFT of the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}.$$

Then inverse transform the result.

- 2.14** Let \mathbf{x} and \mathbf{y} be two column vectors of length m and n respectively, with one-dimensional DFT’s \mathbf{X} and \mathbf{Y} . Let \mathbf{z} be the product matrix defined by

$$z_{r,s} = x_r y_s$$

for $0 \leq r \leq m - 1$, $0 \leq s \leq n - 1$, and \mathbf{Z} the two-dimensional DFT of \mathbf{z} .

- a. Show that \mathbf{z} is an $m \times n$ matrix satisfying $\mathbf{z} = \mathbf{x}\mathbf{y}^T$ (where \mathbf{y}^T is the transpose of \mathbf{y}).
b. Show that

$$Z_{k,l} = X_k Y_l$$

or equivalently $\mathbf{Z} = \mathbf{XY}^T$, where $Z_{k,l}$ denotes the row k column l entry of \mathbf{Z} .

- 2.15** Let \mathbf{z} denote an $m \times n$ matrix with $z_{j,k} = 1$ for $0 \leq j \leq R - 1, 0 \leq k \leq S - 1$, and $z_{j,k} = 0$ otherwise. Find the two-dimensional DFT of \mathbf{z} as explicitly as possible. *Hint:* Make use of Exercises 2.11 and 2.14.

Translation Effects

- 2.16** Let $\mathbf{x} \in \mathbb{C}^N$ have DFT \mathbf{X} . Let $\mathbf{y} \in \mathbb{C}^N$ be that vector obtained by circularly shifting \mathbf{x} by m indexes as

$$y_k = x_{(k+m) \bmod N}.$$

Show that the DFT of \mathbf{y} has components $Y_r = e^{2\pi i rm/N} X_r$, and that $|X_r| = |Y_r|$ for all r .

- 2.17** Let \mathbf{A} denote an $m \times n$ matrix with entries $a_{r,s}$ and DFT $\widehat{\mathbf{A}}$ with components $\hat{a}_{k,l}$. Let \mathbf{B} be the matrix obtained by circularly shifting \mathbf{x} by p indexes and q indexes, as

$$b_{r,s} = a_{(r+p) \bmod m, (s+q) \bmod n}.$$

Show that the two-dimensional DFT of \mathbf{B} has components

$$\hat{b}_{k,l} = e^{2\pi i (pk/m + ql/n)} \hat{a}_{k,l}$$

and that $|\hat{a}_{k,l}| = |\hat{b}_{k,l}|$ for all k and l .

- 2.18** This exercise involves Matlab:

- a. Load an image \mathbf{A} into Matlab and let \mathbf{B} be the shifted image as in Exercise 2.17. Shift by a reasonable amount, say 30 to 40 pixels in each direction. Print out both images.
- b. Compute and show both of $|\widehat{\mathbf{A}}|$ and $|\widehat{\mathbf{B}}|$. Use the centered view, logarithmically scaled. Are the results consistent with Exercise 2.17?

Symmetry

- 2.19** In Section 2.5.2 certain symmetry properties of the DFT were discussed, under the assumption that the time domain signal is real-valued. Show that if an $m \times n$ matrix \mathbf{A} is real-valued then certain relationships (equality or “conjugate equality”) exists among the 2D Fourier coefficients $\hat{a}_{k,l}, \hat{a}_{m-k,l}, \hat{a}_{m,n-l}$, and $\hat{a}_{m-k,n-l}$. Take note of Remark 2.5 on page 94.

Compression

- 2.20** This exercise involves Matlab. Try the audio compression technique of Section 2.8.1, part 6, on the “splat” signal in Matlab. Make a table showing the compression ratio obtained using various thresholds, the distortion introduced, and your subjective rating of the quality of the sound. It’s probably more difficult to get compression comparable to that attainable with the “train” signal, while maintaining quality. Why?

CHAPTER 3

THE DISCRETE COSINE TRANSFORM

3.1 MOTIVATION FOR THE DCT—COMPRESSION

The goal of any compression algorithm is to make the data that represent the underlying signal or image as small as possible without undue loss of information. Other considerations include simplicity, speed of computation, and flexibility.

In this chapter we present the discrete cosine transform and introduce the notion of localizing frequency analysis by breaking a signal or image into smaller pieces. One motivation for this is to give the reader an understanding of the mathematical basis of classical JPEG compression. In order to keep the discussion focused on the essential mathematical principles, we'll make a few simplifying assumptions. First, we'll assume that the signal or image has already been digitized, and so there are no image capture or sampling issues involved. We'll also assume that the quantization error when the signal is captured is negligible so that the vector space models of Chapter 1 continue to apply. The quantization that occurs when the image is first digitized is in the time domain. Quantization will also be an issue at a later point, but in the frequency domain. We'll work only with grayscale images.

Before proceeding it's helpful to define what we mean by "lossless" and "lossy" compression.

Definition 3.1.1 *A compression algorithm or a specific step in a compression algorithm is called "lossless" if it is reversible so that the input can be perfectly reconstructed from the output. A process that is not lossless is called "lossy."*

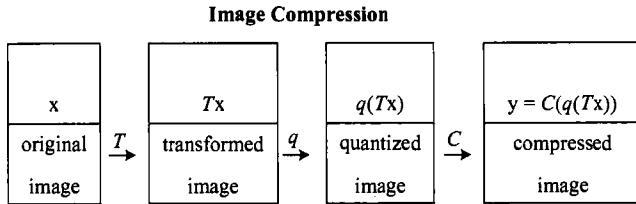
**FIGURE 3.1** Overview of compression.

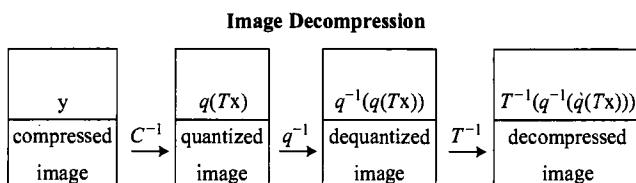
Figure 3.1 illustrates the overall flow of a compression algorithm. The first step in the algorithms we'll consider consists of applying a linear transform T like the DFT to the object of interest, a vector or matrix \mathbf{x} . Aside from round-off error the transform is invertible, so this step is essentially lossless. The next step is quantization of the frequency domain version of the signal using the techniques from Section 1.9; this is represented by the function q . Some care is required in the design of the transform and quantization scheme in order to achieve good compression. Ideally the quantized frequency domain representation of the original time domain object should have a large proportion of zeros, which will make the frequency domain data easy to compress. The quantization step is lossy.

The last step in the compression process is the application of some standard data compression technique, represented by C , not necessarily specific to audio or image compression. This often involves Huffman coding or simple run-length encoding. We will not discuss the details of this step in the process, since it really is quite separate from the mathematics we've developed so far. See Chapter 8 of [14] for more information on this topic.

Decompressing the compressed image requires reversing the above steps and is illustrated in Figure 3.2. In most of this chapter the transform T will be the discrete cosine transform (DCT), described below. In later chapters the transform T will become a *filter bank* or *wavelet* transform.

3.2 OTHER COMPRESSION ISSUES

We always face a compromise between good compression and fidelity to the original signal or image. There are several ways to quantify the efficiency of a compression

**FIGURE 3.2** Overview of decompression.

method. The most obvious is to simply compute the ratio of the original file size to the compressed file size. In the case of the full JPEG algorithm, however, this would involve the final compression step (the Huffman or run-length compression), which we don't want to get into here. A slightly cruder alternative is to look at the number of nonzero elements in the image after it has been transformed to the frequency domain and quantized. This is generally the approach we take.

In addition to measuring the compression achieved we also need to measure image quality. Since image (or audio) quality is very subjective, it is impossible to find a perfect quantitative measure. However, the measure of distortion that we used in Chapter 1 is usually reasonable, since throwing away a lot of the energy in a signal or image generally means that the result has poor fidelity.

3.3 INITIAL EXAMPLES—THRESHOLDING

Before proceeding, it will be very helpful to look at a simple compression scheme that makes use of the mathematics we already have at hand, in particular, the DFT. It will also serve to illustrate certain shortcomings in using the DFT for image compression. The examples below are closely related to the audio and image exercises from Section 2.8 in the previous chapter.

Consider a sampled signal \mathbf{x} with components $x_k = f(k\Delta T)$, $0 \leq k \leq N - 1$ for some function f defined on $[0, 1]$, where $\Delta T = 1/N$. The first simple compression scheme will be as follows: Choose a threshold parameter $0 \leq c \leq 1$. Then

1. compute $\mathbf{X} = DFT(\mathbf{x})$. Let $M = \max_{0 \leq k \leq N-1}(|X_k|)$;
2. define $\tilde{\mathbf{X}} \in \mathbb{C}^N$ with components

$$\tilde{X}_k = \begin{cases} X_k, & \text{if } |X_k| \geq cM, \\ 0, & \text{if } |X_k| < cM. \end{cases}$$

The vector $\tilde{\mathbf{X}} \in \mathbb{C}^N$ will become the compressed version of the signal. Of course, if most of the components $\tilde{X}_k = 0$, then $\tilde{\mathbf{X}}$ should be easy to compress. To transmit the image, we could just send the nonzero \tilde{X}_k . Decompression is done by inverse transforming, as $\tilde{\mathbf{x}} = IDFT(\tilde{\mathbf{X}})$. The vector $\tilde{\mathbf{x}}$ should approximate \mathbf{x} .

In practice, the vector \mathbf{X} would not simply be thresholded but rather subjected to a more sophisticated quantization scheme like that in Section 1.9. Ideally this scheme should zero out many X_k , like thresholding, and allow the remaining X_k to be stored economically. Nonetheless, we'll proceed with a simple thresholding approach for now and apply more sophisticated methods later in this chapter.

We can crudely measure the efficiency of our compression scheme by computing the fraction of the X_k that exceeds the threshold. We'll consider the efficiency to be

a function of the parameter c as

$$P(c) = \frac{\#\{k : |\tilde{X}_k| > 0\}}{N} = \frac{\#\{k : |X_k| > cM\}}{N},$$

where $\#\{A\}$ denotes the number of elements in the set A . Of course, $P(c) = 0$ indicates perfect compression, whereas $P(c) = 1$ indicates no compression at all. We'll measure the distortion of the compressed image as in Chapter 1, as

$$D(c) = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{\|\mathbf{x}\|^2}$$

or on a percentage basis as $100D(c)$. If $D(c) = 0$ then the reconstruction is perfect.

Ideally both $P(c)$ and $D(c)$ should be close to zero, but that can't happen simultaneously. Let's look at a few specific examples and see how the numbers work out.

3.3.1 Compression Example 1: A Smooth Function

Let $f(t) = (t - t^2)^2$ in the scheme above, with $N = 256$. This function is plotted on the left in Figure 3.3. On the right we plot the DFT \mathbf{X} , as pairs $(k, |X_k|^2/N)$ on the range $-127 \leq k \leq 128$. The plot makes it clear that most of the energy lies in a few low frequencies. Almost any threshold parameter $c > 0$ should leave these energetic frequencies unchanged while zeroing out the remainder. We thus expect good compression and little distortion. Table 3.1 shows various values of c and the corresponding values of $P(c)$ and $D(c)$. In this case it's clear that we can obtain good compression and low distortion. Indeed we can compress the signal 10-fold and still obtain distortion less than 10^{-8} . At this level of compression the original signal and reconstructed compressed signal are visually indistinguishable.

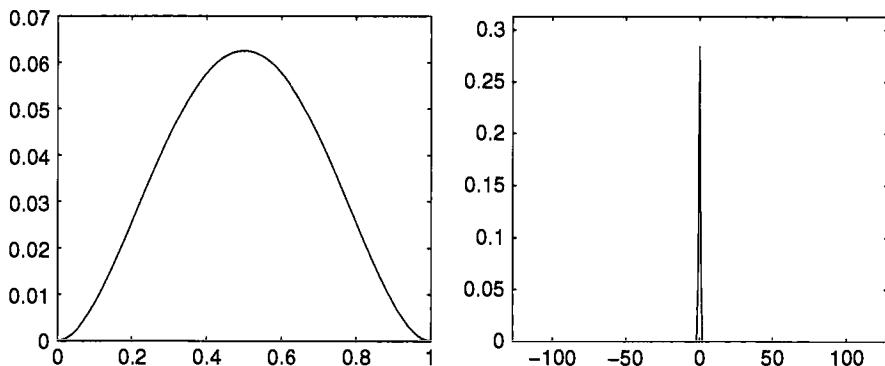


FIGURE 3.3 $f(t) = (t - t^2)^2$ and DFT.

TABLE 3.1 Compression Data for $f(t) = (t - t^2)^2$

c	0.5	0.1	0.01	0.001	10^{-4}	10^{-8}
$P(c)$	0.004	0.012	0.02	0.035	0.066	0.652
$D(c)$	0.300	1.2×10^{-3}	5.2×10^{-5}	1.0×10^{-6}	1.3×10^{-8}	< 10^{-15}

3.3.2 Compression Example 2: A Discontinuity

This time let

$$f(t) = \begin{cases} 1, & t \leq 0.2, \\ 0, & t > 0.2, \end{cases} \quad (3.1)$$

on $[0, 1]$, a discontinuous step function. We again take $N = 256$ and sample as in the previous example. On the left in Figure 3.4, we plot the DFT \mathbf{X} , as pairs $(k, |X_k|^2/N)$ on the range $-127 \leq k \leq 128$, while on the right, we zoom in to the range $-9 \leq k \leq 9$. In this case most of the energy still lies in the lower frequencies, but compared to the previous case the energy has spread out a bit. Table 3.2 shows various values of c and the corresponding values of $P(c)$ and $D(c)$. Compression here is a bit tougher than for the first example. In particular, a 10-fold compression ($P(c) = 0.1$) with this approach brings the distortion up to 2.9×10^{-2} . In Figure 3.5 we show the original signal and compressed/decompressed signal at this 10-fold compression level.

The essential difficulty in this example is the discontinuity: it takes a lot of different basic waveforms to synthesize a jump.

3.3.3 Compression Example 3

In this case let $f(t) = t$ on the interval $[0, 1]$ with $N = 256$, sampled as above. The function f is continuous on the closed interval. If we apply the same procedure as

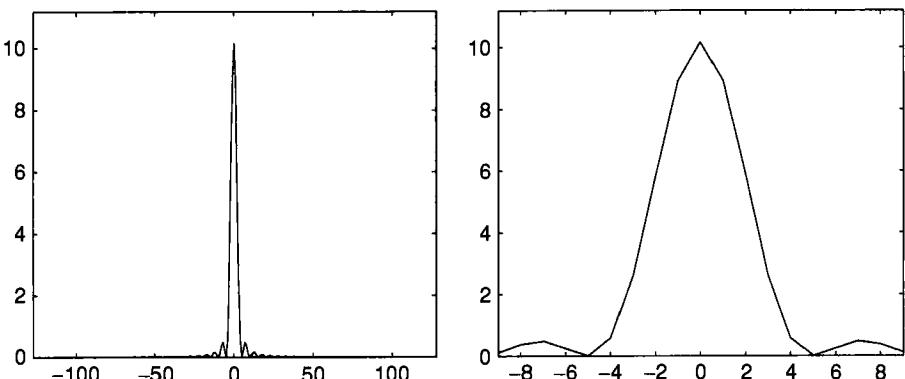


FIGURE 3.4 Two views of the DFT for step function $f(t)$.

TABLE 3.2 Compression Data for Discontinuous $f(t)$ of Equation (3.1)

c	0.5	0.1	0.03	0.02	0.01	0.001
$P(c)$	0.027	0.082	0.262	0.543	0.793	0.988
$D(c)$	0.124	0.039	0.011	0.0035	0.364×10^{-4}	1.23×10^{-7}

above to this function we obtain the data in Table 3.3. Compression for this signal also seems more difficult than for the first example. In particular, to achieve a 10-fold compression with this approach, we have to accept distortion 1.12×10^{-2} . In Figure 3.6 we show the original signal and compressed-decompressed signal at the 10-fold compression level.

Why is this nicely continuous function so hard to compress? The problem is that we need a slightly refined notion of continuity when we use the DFT. The difficulty in this case is that $f(0) \neq f(1)$. To see why this causes trouble, recall that the DFT/IDFT pair allows us to synthesize the vector \mathbf{x} as a superposition of periodic waveforms $\mathbf{E}_{N,m}$, as $\mathbf{x} = \frac{1}{N} \sum_{m=0}^{N-1} X_m \mathbf{E}_{N,m}$. What this synthesis really provides though is a “bi-infinite” vector $\mathbf{w} = (\dots, w_{-2}, w_{-1}, w_0, w_1, \dots)$ given by

$$w_k = \frac{1}{N} \sum_{m=0}^{N-1} X_m e^{2\pi i km/N}$$

with $x_k = w_k$ for $0 \leq k \leq N - 1$. But w_k is defined for all $k \in \mathbb{Z}$ and periodic in k with period N . If $f(0) \neq f(1)$, then the DFT/IDFT must synthesize the jump in \mathbf{w}

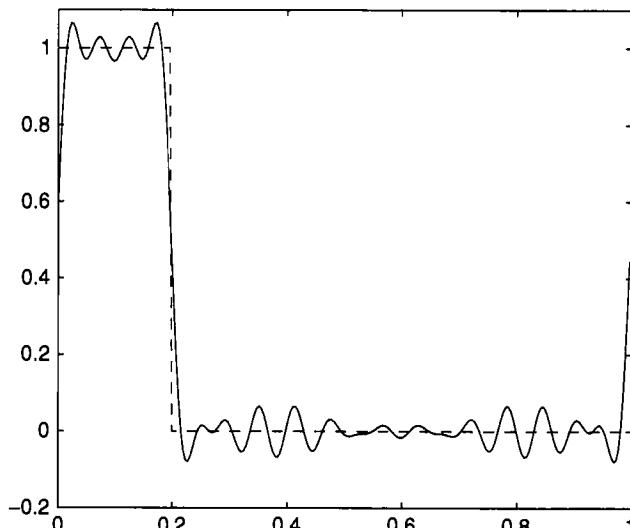
**FIGURE 3.5** Original (dashed) and decompressed (solid) signal, 10-fold compression.

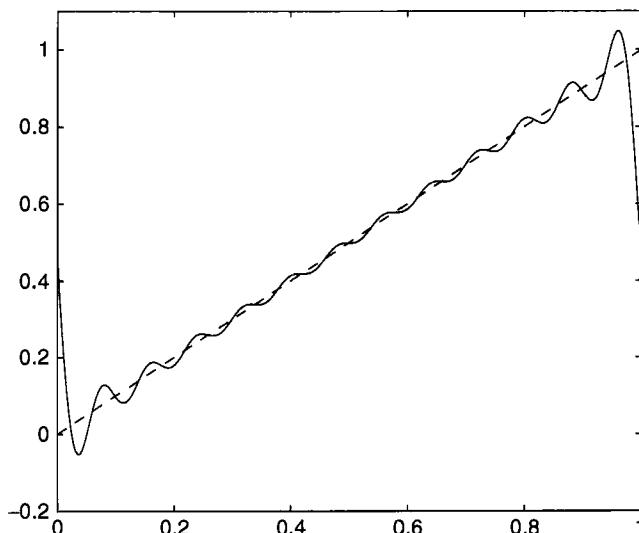
TABLE 3.3 Compression Data for $f(t) = t$

c	0.5	0.1	0.03	0.02	0.01	0.005
$P(c)$	0.004	0.027	0.082	0.129	0.254	0.574
$D(c)$	0.252	0.043	1.44×10^{-2}	9.2×10^{-3}	4.49×10^{-3}	1.5×10^{-3}

that occurs whenever the index k is a multiple of N , just as if the jump occurred in the middle of the interval. This requires the superposition of many frequencies. The situation is illustrated in Figure 3.7 in which we plot \mathbf{w} , the periodic extension of the partial Fourier synthesis of \mathbf{x} from Figure 3.6, superimposed over the periodic extension of the original signal.

Another way to look at the difficulty in compressing this signal is via Exercise 2.16, in which we consider a signal \mathbf{y} obtained by shifting a signal \mathbf{x} circularly to the left m units, as $y_k = x_{(k+m) \bmod N}$. If \mathbf{x} stems from sampling a function $f(t)$ with $f(0) \neq f(1)$, then \mathbf{y} can be obtained by sampling the function $g(t) = f((t+m/N) \bmod 1)$. If $1 \leq m \leq N-1$, then g has a discontinuity at $1-m/N \in (0, 1)$. We would thus expect the transform \mathbf{Y} to have many large coefficients. But according to Exercise 2.16, $|X_k| = |Y_k|$ for $0 \leq k \leq N-1$, and so $\mathbf{X} = DFT(\mathbf{x})$ must also have many large coefficients. In the eyes of the DFT any discrepancy in the endpoint values is like having a discontinuity in the middle of the domain!

Unfortunately, this “edge effect” also appears in images when the two-dimensional DFT is used. If points on directly opposite sides of the image have different values,

**FIGURE 3.6** Original (dashed) and decompressed (solid) signal, 10-fold compression.

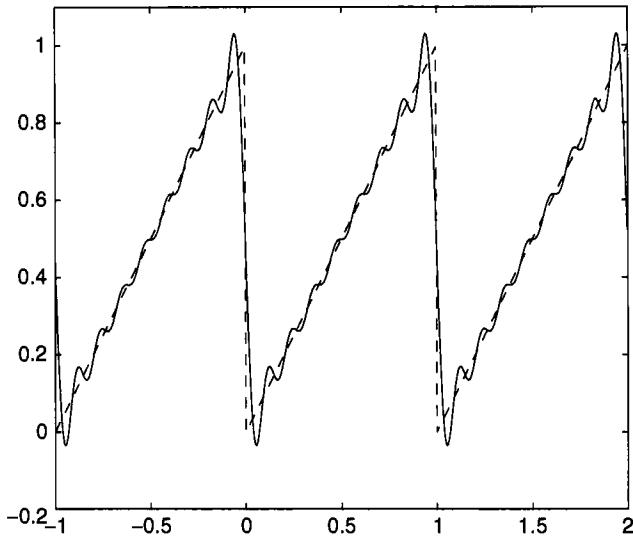


FIGURE 3.7 Partial Fourier synthesis (*solid*) and periodic extension w (*dashed*) of signal x from Figure 3.6.

then the DFT perceives this as a discontinuity, and many frequencies are needed to accurately synthesize the image. The same argument given in the paragraph above shows this, with the result of Exercise 2.17 in place of Exercise 2.16. Consequently thresholding the transform won't zero out many frequencies unless the threshold cutoff is rather high, in which case the reconstruction will suffer in quality.

3.3.4 Observations

The examples above and those from the Matlab project of Section 2.8 should be somewhat encouraging, but it's clear that the current approach has some deficiencies. The DFT behaves badly in the presence of discontinuities, which "pollute" the entire transform. One way to overcome this is to break the signal into shorter blocks and then compress each block individually. Since any given discontinuity appears in only one block, its effect is confined to that block's transform. If the blocks are relatively short, then most blocks will contain no discontinuities, and probably few frequencies, and so compress more easily.

Similar considerations apply to images. In light of this, our overall compression strategy will be to break signals and images up into smaller blocks, and then compress each individually. Indeed the traditional JPEG standard specifies that the image be broken up into 8 by 8 pixel blocks. This partly alleviates one difficulty with this approach to compression.

Finally, we have not really taken advantage of quantization in the frequency domain. When we threshold as above small frequencies are "quantized" to zero, but the others remain full double precision floating point.

We'll address these issues in the next few sections.

3.4 THE DISCRETE COSINE TRANSFORM

3.4.1 DFT Compression Drawbacks

The blocking strategy partially removes the DFT's difficulty with discontinuities, but as the example in the last section showed, even the edge discontinuities will be a problem. Moreover, breaking the signal up into blocks will make this even worse, since every block boundary is now a potential discontinuity! This can be overcome by replacing the DFT with the closely related discrete cosine transform (DCT).

3.4.2 The Discrete Cosine Transform

Let's begin with the one-dimensional case. The overall approach to fixing the DFT's difficulty with edge discontinuities is to extend the signal to twice its original length by reflection about one end of the signal, compute the DFT of the double-length signal, then "restrict back" to the appropriate length.

Here are the specifics. Let $\mathbf{x} \in \mathbb{C}^N$.

Symmetric Reflection Compression difficulties arise when x_0 differs substantially from x_{N-1} . Let us thus define an extension $\tilde{\mathbf{x}} \in \mathbb{C}^{2N}$ of \mathbf{x} as

$$\tilde{x}_k = \begin{cases} x_k, & 0 \leq k \leq N - 1, \\ x_{2N-k-1}, & N \leq k \leq 2N - 1. \end{cases} \quad (3.2)$$

We then have $\tilde{\mathbf{x}} = (x_0, x_1, \dots, x_{N-2}, x_{N-1}, x_{N-1}, x_{N-2}, \dots, x_1, x_0)$, so $\tilde{\mathbf{x}}$ is just \mathbf{x} reflected about the right endpoint and $\tilde{x}_0 = \tilde{x}_{2N-1} = x_0$. When we take the $2N$ -point DFT of $\tilde{\mathbf{x}}$ we won't encounter the kind of edge effects discussed above. Note that we duplicate x_{N-1} in our extension, called the *half-point symmetric extension*. There are other ways to extend \mathbf{x} ; we'll say more about this later.

DFT of the Extension The DFT $\tilde{\mathbf{X}}$ of the vector $\tilde{\mathbf{x}} \in \mathbb{C}^{2N}$ has components

$$\tilde{X}_k = \sum_{m=0}^{2N-1} \tilde{x}_m e^{-2\pi i km/(2N)} = \sum_{m=0}^{2N-1} \tilde{x}_m e^{-\pi i km/N}. \quad (3.3)$$

Note that we use a $2N$ -point DFT. We can split the sum on the right in equation (3.3) and obtain

$$\tilde{X}_k = \sum_{m=0}^{N-1} (\tilde{x}_m e^{-\pi i km/N} + \tilde{x}_{2N-m-1} e^{-\pi i k(2N-m-1)/N}), \quad (3.4)$$

since as the index of summation m in (3.4) assumes values $m = 0, \dots, N - 1$, the quantity $2N - m - 1$ in the second part of the summand assumes values $2N - 1, \dots, N$ in that order. Thus the sums in (3.3) and (3.4) are in fact identical.

From equation (3.2) we have $\tilde{x}_{2N-m-1} = \tilde{x}_m = x_m$ for $0 \leq m \leq N - 1$. Use this in (3.4), along with

$$e^{-\pi ik(2N-m-1)/N} = e^{i\pi k(m+1)/N} e^{2\pi ik} = e^{i\pi k(m+1)/N}$$

to obtain

$$\begin{aligned}\tilde{X}_k &= \sum_{m=0}^{N-1} (x_m e^{-\pi i km/N} + x_m e^{i\pi k(m+1)/N}) \\ &= e^{\pi ik/2N} \sum_{m=0}^{N-1} (x_m e^{-\pi ik(m+1/2)/N} + x_m e^{\pi ik(m+1/2)/N}) \\ &= 2e^{\pi ik/2N} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi k(m+1/2)}{N}\right).\end{aligned}\quad (3.5)$$

This defines the DFT coefficients \tilde{X}_k on the range $0 \leq k \leq 2N - 1$. As usual however, the formula makes sense for any $k \in \mathbb{Z}$ and the resulting extension is periodic in k with period $2N$.

DCT/IDCT Derivation Equation (3.5) is “almost” what we’ll call the discrete cosine transform. Let’s define

$$c_k = 2 \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi k(m+1/2)}{N}\right)\quad (3.6)$$

so that $\tilde{X}_k = e^{\pi ik/2N} c_k$. Here are a few easy-to-check facts about c_k :

- $c_{-k} = c_k$ and $c_{k+2N} = -c_k$.
- If \mathbf{x} is real-valued, then c_k is also real.

In particular, note that from the first bulleted item above $c_{2N-r} = c_{-2N+r} = -c_{-2N+r+2N} = -c_r$, so

$$c_{2N-r} = -c_r.\quad (3.7)$$

Taking $r = N$ shows that $c_N = 0$. If we consider the range $1 \leq r \leq N - 1$ (so $2N - r$ ranges from $2N - 1$ down to $N + 1$), we see that knowledge of c_k on the range $0 \leq k \leq N - 1$ allows us to compute c_k for $N \leq k \leq 2N - 1$. All this stems from the fact that the underlying signal $\tilde{\mathbf{x}}$ possesses a symmetry obtained from reflection.

We can consider equation (3.6) as a transform that maps $\mathbf{x} \in \mathbb{C}^N$ to a vector $\mathbf{c} \in \mathbb{C}^N$, where $\mathbf{c} = (c_0, c_1, \dots, c_{N-1})$. The transform must be invertible, for as remarked above, knowledge of c_k for $0 \leq k \leq N - 1$ allows us to determine c_k for all $0 \leq k \leq 2N - 1$.

$2N - 1$. But we know that $\tilde{X}_k = e^{\pi i k / 2N} c_k$ and so we can recover $\tilde{\mathbf{x}} = IDFT(\mathbf{X})$ and hence \mathbf{x} .

Let's write out this inverse transform explicitly. Begin with the inverse DFT ($2N$ points!) of \mathbf{X}

$$\tilde{x}_k = \frac{1}{2N} \sum_{r=0}^{2N-1} \tilde{X}_r e^{2\pi i kr/(2N)} = \frac{1}{2N} \sum_{r=0}^{2N-1} \tilde{X}_r e^{\pi i kr/N}.$$

Now fill in $\tilde{X}_r = e^{\pi i r / 2N} c_r$ and split the sum as

$$\begin{aligned} \tilde{x}_k &= \frac{1}{2N} \left(c_0 + \sum_{r=1}^{2N-1} c_r e^{\pi i (k+1/2)r/N} \right) \\ &= \frac{1}{2N} \left(c_0 + \sum_{r=1}^{N-1} c_r e^{\pi i (k+1/2)r/N} + \sum_{r=N+1}^{2N-1} c_r e^{\pi i (k+1/2)r/N} \right) \\ &= \frac{1}{2N} \left(c_0 + \sum_{r=1}^{N-1} c_r e^{\pi i (k+1/2)r/N} + \sum_{r=1}^{N-1} c_{2N-r} e^{\pi i (k+1/2)(2N-r)/N} \right) \\ &= \frac{1}{2N} \left(c_0 + \sum_{r=1}^{N-1} (c_r e^{\pi i (k+1/2)r/N} - c_{2N-r} e^{-\pi i (k+1/2)r/N}) \right), \end{aligned} \quad (3.8)$$

where we've explicitly broken out the $r = 0$ term and ignored $r = N$, since $c_N = 0$. In the third line we made the substitution $r \rightarrow 2N - r$, and altered the range of summation accordingly. In the transition to (3.8) we also substituted

$$e^{\pi i (k+1/2)(2N-r)/N} = e^{2\pi i k} e^{i\pi} e^{-\pi i (k+1/2)r/N} = -e^{-\pi i (k+1/2)r/N}.$$

Finally, use equation (3.7) to write equation (3.8) in the form

$$\begin{aligned} \tilde{x}_k &= \frac{1}{2N} \left(c_0 + \sum_{r=1}^{N-1} (c_r e^{\pi i (k+1/2)r/N} + c_r e^{-\pi i (k+1/2)r/N}) \right) \\ &= \frac{c_0}{2N} + \frac{1}{N} \sum_{r=1}^{N-1} c_r \cos \left(\frac{\pi(k+1/2)r}{N} \right) \end{aligned} \quad (3.9)$$

We can use equation (3.9) to recover \tilde{x}_k for $0 \leq k \leq 2N - 1$. Then, of course, we can recover $x_k = \tilde{x}_k$ for $0 \leq k \leq N - 1$.

Definition of the DCT and IDCT Equations (3.6) through (3.9) form a natural transform/inverse transform pair and are, up to a minor change of scale, the discrete cosine transform. Let us modify the definition of the c_k slightly (we will of course

compensate in equation (3.9)), and replace the c_k by C_k with

$$C_0 = \sqrt{\frac{1}{N}} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi 0(m+1/2)}{N}\right) = \sqrt{\frac{1}{N}} \sum_{m=0}^{N-1} x_m, \quad (3.10)$$

$$C_k = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x_m \cos\left(\frac{\pi k(m+1/2)}{N}\right), \quad 1 \leq k \leq N-1. \quad (3.11)$$

Definition 3.4.1 Let $\mathbf{x} \in \mathbb{C}^N$. The discrete cosine transform (type II) of \mathbf{x} is the vector $\mathbf{C} \in \mathbb{C}^N$ with components C_k defined by equations (3.10) and (3.11).

As with the DFT, the DCT is a linear transformation. The DCT maps \mathbb{C}^N to \mathbb{C}^N , but if $\mathbf{x} \in \mathbb{R}^N$, then $\mathbf{C} \in \mathbb{R}^N$, unlike the DFT. The “type II” in the definition stems from the fact that there are many variations on the DCT; this version is the most commonly used.

If we compensate appropriately in equation (3.9), we obtain the inverse discrete cosine transform (IDCT).

Definition 3.4.2 Let $\mathbf{C} \in \mathbb{C}^N$. The inverse discrete cosine transform of \mathbf{C} is the vector $\mathbf{x} \in \mathbb{C}^N$ defined by

$$x_m = \frac{1}{\sqrt{N}} C_0 + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} C_k \cos\left(\frac{\pi k(m+1/2)}{N}\right) \quad (3.12)$$

for $0 \leq m \leq N-1$.

3.4.3 Matrix Formulation of the DCT

As a linear mapping the DCT, like the DFT, can be represented by a matrix. From equations (3.10) and (3.11) it's easy to see that we can compute the DCT as $\mathbf{C} = \mathcal{C}_N \mathbf{x}$, where

$$\mathcal{C}_N = \begin{bmatrix} \frac{1}{\sqrt{N}} & \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N} \frac{1}{2}\right) & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N} \frac{3}{2}\right) & \cdots & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N} \frac{2N-1}{2}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N} \frac{1}{2}\right) & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N} \frac{3}{2}\right) & \cdots & \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N} \frac{2N-1}{2}\right) \end{bmatrix}. \quad (3.13)$$

The first row consists entirely of entries $1/\sqrt{N}$. For $k \geq 1$ the row k , column m entry is given by $\sqrt{2/N} \cos([\pi k(2m + 1)]/2N)$.

A careful examination of the inverse DCT in equation (3.12) shows that $\mathbf{x} = \mathcal{C}_N^T \mathbf{C}$ so that $\mathcal{C}_N^{-1} = \mathcal{C}_N^T$. The matrix \mathcal{C}_N is thus orthogonal, that is,

$$\mathcal{C}_N^T \mathcal{C}_N = \mathcal{C}_N \mathcal{C}_N^T = \mathbf{I}_N. \quad (3.14)$$

This is the motivation for the $\sqrt{1/N}$ and $\sqrt{2/N}$ scaling factors in the DFT/IDFT definition—to make the relevant matrices orthogonal.

3.5 PROPERTIES OF THE DCT

3.5.1 Basic Waveforms for the DCT

The DFT allows us to write an arbitrary vector as a superposition of basic waveforms $\mathbf{E}_{N,k}$. The DCT does the same, even though we didn't develop it in this manner. Specifically, consider the matrix form of the IDCT. Basic matrix algebra shows that we can write $\mathbf{x} = \mathcal{C}_N^T \mathbf{C}$ in the form

$$\mathbf{x} = \sum_{k=0}^{N-1} C_k \mathcal{C}_{N,k}, \quad (3.15)$$

where $\mathcal{C}_{N,k}$ denotes the k th column of \mathcal{C}_N^T or the k th the row of \mathcal{C}_N (indexing rows and columns as usual from 0 to $N - 1$). The vectors $\mathcal{C}_{N,k}$ play the role of the basic waveforms $\mathbf{E}_{N,k}$ for the DCT. Equation (3.14) implies that the vectors $\mathcal{C}_{N,k}$ are orthonormal in \mathbb{R}^N . The $\mathcal{C}_{N,k}$ are also orthonormal in \mathbb{C}^N because the conjugation in the standard inner product makes no difference.

As with the DFT we can compute the coefficients C_k by taking the inner product of both sides of (3.15) with the vector $\mathcal{C}_{N,k}$ and using the orthonormality of the $\mathcal{C}_{N,k}$ to find that

$$C_k = (\mathbf{x}, \mathcal{C}_{N,k}). \quad (3.16)$$

Of course, equation (3.16) is merely a restatement of the definition of $\mathbf{C} = \mathcal{C}_N \mathbf{x}$, the matrix form of the DCT.

3.5.2 The Frequency Domain for the DCT

Just as the waveforms $\mathbf{E}_{N,k}$ can be obtained by sampling the function $e^{2\pi i kt/T}$ at times $t = 0, T/N, 2T/N, \dots, (N - 1)T/N$, we can obtain $\mathcal{C}_{N,k}$ by sampling the function

$$\phi_k(t) = \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k t}{T}\right)$$

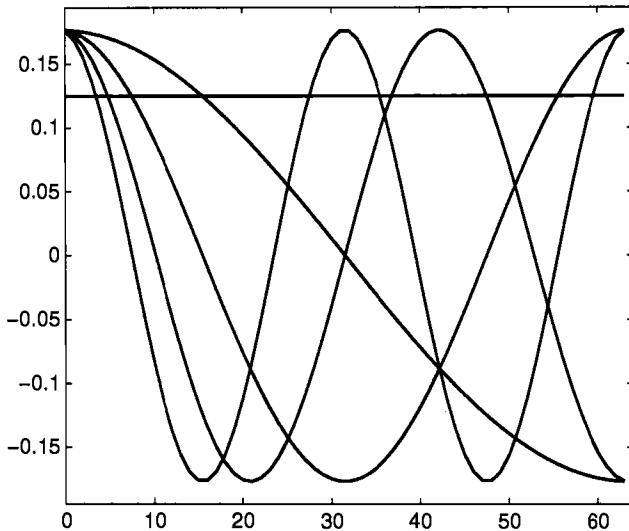


FIGURE 3.8 DCT basis functions $C_{64,k}$, $0 \leq k \leq 4$.

at times $t = T/2N, 3T/2N, 5T/2N, \dots, [(2N - 1)T]/2N$. These times are the midpoints of each interval of the form $[mT/N, ((m + 1)T)/N]$, $0 \leq m \leq N - 1$ within the interval $[0, T]$.

For $0 \leq k \leq N - 1$ the function $\cos(\pi kt/T)$ ranges from frequency 0 Hertz to $(N - 1)/2T$ Hertz. This is effectively the same frequency range, dc to the Nyquist frequency, as in the complex exponential case or sine/cosine case; recall Remark 2.3 on page 84.

Figure 3.8 shows a few of the waveforms $C_{N,k}$ when $N = 64$ and $k = 0, 1, 2, 3, 4$.

If it does not cause confusion, we will write C_k for $C_{N,k}$. Note that $C_k \neq \operatorname{Re}(E_k)$. Observe that the cosine basis function C_k crosses the horizontal axis k times in the interval $0 \leq k \leq N - 1$, in accordance with the frequency $k/2$ function $\cos(k\pi t/T)$ plotted on the range $0 \leq t \leq T$.

3.5.3 DCT and Compression Examples

In order to gain a better understanding of the DCT, let's go back to the thresholding compression examples from Section 3.3.

Before proceeding we should point out that as the FFT provides a fast algorithm for computing the DFT, fast algorithms also exist for computing the DCT. Indeed, since we derived the DCT from an appropriate DFT computation, we can use the FFT to obtain an order $N \log(N)$ algorithm for the DCT. Specifically tailored algorithms for the DCT are bit more efficient, though.

Let's begin by reconsidering the first example of Section 3.3, but with the DCT taking the place of the DFT. In Figure 3.9 we plot the magnitude of the DCT for this signal, on the range $0 \leq k \leq 20$; the remaining coefficients C_k are all effectively

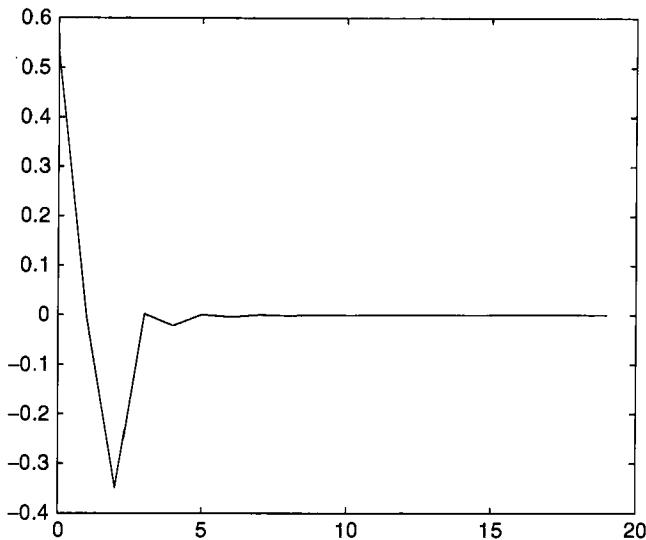


FIGURE 3.9 DCT of sampled function $f(t) = (t - t^2)^2$.

zero. In Table 3.4 we show the compression and distortion for a number of threshold parameters. As in the DFT case, compression is efficient. We can get a 10-fold compression with distortion less than 10^{-8} .

For the second example involving a discontinuous f from Section 3.3, the data are shown in Table 3.5. These are pretty similar to the DFT data. In particular, at 10-fold compression we obtain distortion 1.54×10^{-2} . This is expected because the discontinuity is in the interior of the interval. The reconstruction of the signal at this compression level looks quite similar to the DFT case.

For the third example from Section 3.3 the data are shown in Table 3.6. This is obviously a dramatic improvement over the DFT, which is not surprising since

TABLE 3.4 Compression Data for $f(t) = (t - t^2)^2$

c	0.5	0.1	0.01	0.001	10^{-4}	10^{-8}
$P(c)$	0.008	0.008	0.012	0.035	0.070	0.805
$D(c)$	0.0013	0.0013	1.0×10^{-6}	8.1×10^{-7}	3.5×10^{-8}	$< 10^{-15}$

TABLE 3.5 Compression Data for Discontinuous f of Equation (3.1)

c	0.5	0.1	0.03	0.02	0.01	0.001
$P(c)$	0.016	0.043	0.137	0.250	0.559	0.973
$D(c)$	0.124	0.039	0.012	0.0062	0.0015	5.8×10^{-7}

TABLE 3.6 Compression Data for $f(t) = t$

c	0.5	0.1	0.03	0.02	0.01	0.005
$P(c)$	0.008	0.008	0.012	0.016	0.020	0.023
$D(c)$	0.0036	0.0036	5.7×10^{-4}	2.0×10^{-4}	8.1×10^{-5}	3.6×10^{-5}

the DCT was designed to address this situation. In particular, with $c = 0.0002$ we obtain 10-fold compression but with distortion only 2.3×10^{-7} . The compressed and original signals are visually indistinguishable.

One way to summarize the data in this example is by plotting $\log(D(c))$ versus $P(c)$ for the data in Table 3.3. This is shown in the panel on the right in Figure 3.10, while the corresponding compression/distortion data for the DFT from Table 3.3 is shown in the panel on the left. In each case the positive direction along the horizontal axis indicates poorer compression, while downward indicates lower distortion. This helps us visualize the compromise between low distortion and good compression. Note the rather different horizontal scales on each plot; the DCT achieves dramatically better compression on this signal while introducing much less distortion.

3.6 THE TWO-DIMENSIONAL DCT

We can derive a two-dimensional version of the DCT in the same way we derived the one-dimensional DCT. An $m \times n$ image matrix \mathbf{A} is extended to a $2m \times 2n$ matrix by appropriately reflecting about its bottom and right edge, and then reflecting one of these to fill in the fourth corner of the enlarged image. An example is shown in Figure 3.11 with the image of the dog from the previous chapter. DFT difficulties

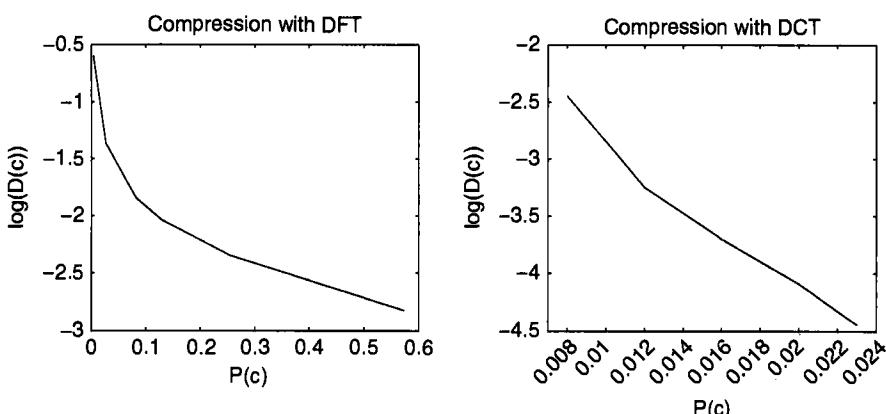


FIGURE 3.10 Data $\log(D(c))$ versus $P(c)$ for compression of $f(t) = t$ using DFT (left) and DCT (right).

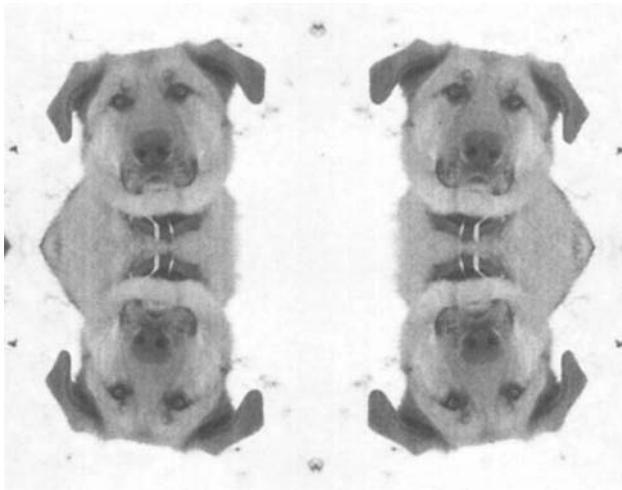


FIGURE 3.11 Even reflections of dog image.

with mismatched edges are gone, since opposite edges now match. We can apply the two-dimensional DFT to the extended image and then restrict back to an appropriate set of $m \times n$ Fourier components, as in the one-dimensional case.

However, we aren't going to work out the details of this approach, for it turns out to be equivalent to the following much simpler scheme. Specifically, we'll define the two-dimensional DCT with inspiration from Proposition 2.7.1.

Definition 3.6.1 *The two-dimensional DCT of an $m \times n$ matrix \mathbf{A} is the $m \times n$ matrix*

$$\widehat{\mathbf{A}} = \mathcal{C}_m \mathbf{A} \mathcal{C}_n^T. \quad (3.17)$$

Exactly the same argument given for the DFT shows that this definition is equivalent to performing an m -point DCT on \mathbf{A} column by column and then an n -point DCT on the result, row by row. Alternatively, we can transform rows first, then columns.

An explicit formula for the two-dimensional DCT easily falls out of equation (3.17) and is given by

$$\hat{a}_{k,l} = u_k v_l \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} \cos\left(\frac{\pi}{m} k \left(r + \frac{1}{2}\right)\right) \cos\left(\frac{\pi}{n} l \left(s + \frac{1}{2}\right)\right), \quad (3.18)$$

where

$$\begin{aligned} u_0 &= \sqrt{\frac{1}{m}}, \quad u_k = \sqrt{\frac{2}{m}}, \quad k > 0, \\ v_0 &= \sqrt{\frac{1}{n}}, \quad v_l = \sqrt{\frac{2}{n}}, \quad l > 0. \end{aligned} \quad (3.19)$$

The basic waveforms are the $m \times n$ matrices $\mathcal{C}_{m,n,k,l}$ (or $\mathcal{C}_{k,l}$ when m, n are fixed) where $0 \leq k \leq m - 1, 0 \leq l \leq n - 1$. The row r , column s entry of $\mathcal{C}_{k,l}$ is given by

$$\mathcal{C}_{k,l}(r, s) = u_k v_l \cos\left(\frac{\pi}{m}k\left(r + \frac{1}{2}\right)\right) \cos\left(\frac{\pi}{n}l\left(s + \frac{1}{2}\right)\right).$$

The two-dimensional inverse DCT is easily derived from equations (3.14) and (3.17), and is given by

$$\mathbf{A} = \mathcal{C}_m^T \widehat{\mathbf{A}} \mathcal{C}_n. \quad (3.20)$$

See Exercise 3.12.

3.7 BLOCK TRANSFORMS

As was discussed in Section 3.3, it can aid compression if we cut up the signal or image into smaller blocks and transform each piece. For a variety of reasons the block size for traditional JPEG compression has been standardized to 8×8 pixel blocks, but in theory any block size could be used. To get some feeling for the process, let's apply these ideas to the image of the dog from previous chapters.

In Figure 3.12 we show both the DCT (on the left) and “block DCT” (on the right) magnitude of the 336×448 pixel dog image, logarithmically scaled. The standard DCT on the left is computed with equation (3.18), with $0 \leq k \leq 335, 0 \leq l \leq 447$. It doesn't really have any discernible information, at least to the naked eye.

For the block DCT the image is divided up into disjoint 8×8 pixel submatrices or blocks. Each 8×8 block of the original grayscale image is then replaced by the two-dimensional DCT of that block, itself an 8×8 (real-valued) matrix. The row k , column l element of each such submatrix corresponds to a frequency vector pair (k, l)

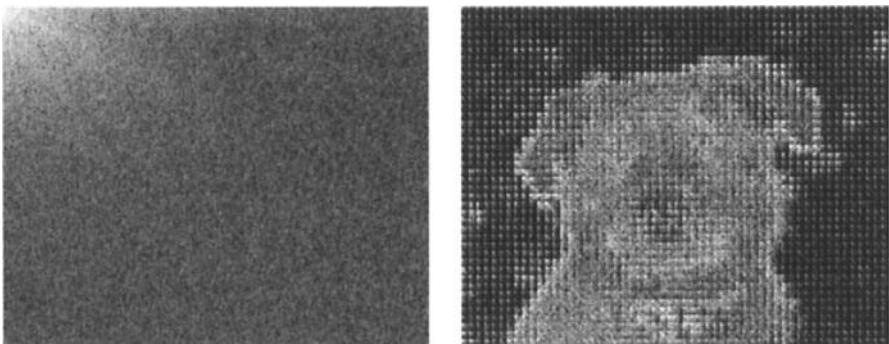


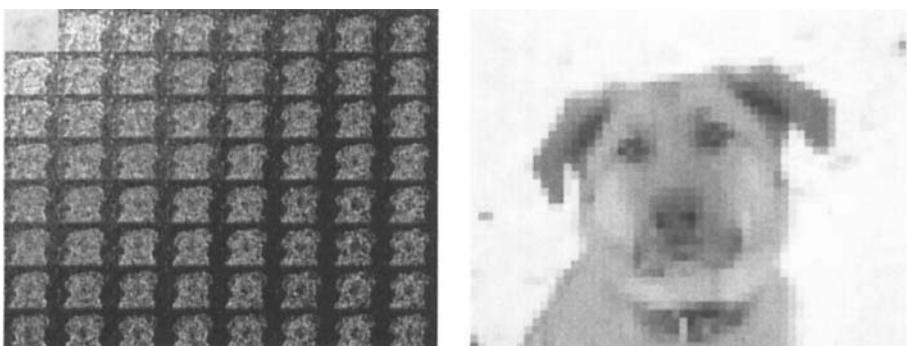
FIGURE 3.12 DCT and block DCT for dog image.

TABLE 3.7 Frequency Vectors for the Block DCT

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	(0, 7)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)	(1, 7)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)	(2, 7)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)	(3, 7)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)	(4, 7)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)	(5, 7)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)	(6, 7)
(7, 0)	(7, 1)	(7, 2)	(7, 3)	(7, 4)	(7, 5)	(7, 6)	(7, 7)

in equation (3.18), where (k, l) is one of the pairs from Table 3.7. It's interesting that we can now perceive the original image in some fashion in this block DCT (think about why!)

The block transform can have more visual impact if we regroup the block DCT coefficients by frequency vector rather than block location. We show this "frequency grouping" presentation on the left in Figure 3.13. Each of the 64 distinct subblocks encodes the information from one of the (k, l) frequency pairs. For example, in each 8×8 subblock on the right in Figure 3.12, the $(0, 0)$ or dc coefficient measures the energy of the constant function in the DCT expansion of the corresponding block. When the dc coefficients are all grouped together in the upper left block of size $(336/8) \times (448/8) = 42 \times 56$ in the image on the left in Figure 3.13, we get a "thumbnail" of the original image. The other 63 blocks are obtained by gathering together coefficients of like frequency vectors from the other blocks. The image on the right in Figure 3.13 is a close-up of the dc coefficient 42×56 subblock from the image on the left.

**FIGURE 3.13** Regrouped block DCT (*left*) and dc components only (*right*).

Remark 3.1 For an 8×8 DCT calculation it's actually faster in Matlab to use matrix multiplication than to use the two-dimensional DCT command “`dct2`”. The `dct2` command is much faster than matrix multiplication when the images are somewhat larger. The matrix multiplication computation of the DCT for an 8×8 matrix \mathbf{A} is done using equation (3.17), as $\mathbf{A} \rightarrow \mathcal{C}_8 \mathbf{A} \mathcal{C}_8^T$.

3.8 JPEG COMPRESSION

3.8.1 Overall Outline

We are now ready to discuss the fundamentals of how traditional JPEG compression works. There are many options for JPEG compression and many parameters that may be specified. Our goal is not a “bit-by-bit” account of the algorithm but rather to illustrate the role of the mathematics we've been studying, specifically the DCT. Here is the basic procedure:

1. *Separate color*, if applicable: A color image is decomposed into its color components, usually using the YCbCr color scheme. This color representation scheme, like RGB, dictates the color of each pixel with three numbers. The entire image can thus be represented by three arrays of appropriate size. The details here need not concern us, since we will be working with monochrome images. See Chapter 6 of [14] for more information on how color images are represented and handled.
2. *Transform*: Perform a block DCT on the image using 8×8 pixel blocks. If the image pixel count in either dimension is not a multiple of 8 the image is padded in some way to a higher multiple of 8. Again, the details are unimportant at the moment.
3. *Quantize*: Each 8×8 pixel block has an 8×8 DCT consisting of real numbers. Each of the 64 components or frequencies in this DCT is quantized in the manner described in Section 1.9. This is the main lossy step in the compression.
4. *Compress*: The image is compressed by using run-length encoding on each block, and then Huffman coding the result. The dc coefficient is often treated separately. This is the step where actual file compression occurs.

Decoding reverses the steps:

- 4'. *Decompress*: Recover the quantized DCT blocks.
- 3'. *Dequantize*: Again, this is done using a scheme as in Section 1.9.
- 2'. *Inverse transform*: Apply the inverse DCT to each block.
- 1'. *Mix colors*: If applicable, and display block.

3.8.2 DCT and Quantization Details

Since we are considering only monochrome images, we need not worry about step 1 above, but let's take a closer look at steps 2 and 3. This is where all the mathematics

we've developed comes into play. Suppose that \mathbf{B} is an 8×8 pixel block from the original image with each pixel encoded by a number $-127 \leq b_{k,m} \leq 128$ (if the grayscale image was originally in the 0 to 255 range, we just subtract 127 from each pixel). We apply the DCT to \mathbf{B} to obtain $\tilde{\mathbf{B}} = DCT(\mathbf{B})$. If the original 8×8 pixel matrix \mathbf{B} is in the range $-127 \leq b_{k,m} \leq 128$, then the DCT $\tilde{\mathbf{B}}$ has all entries in the range $(-2048, 2048]$ (see Exercise 3.13), although the entries will generally be much closer to zero.

The next step is quantization. We quantize $\tilde{\mathbf{B}}$ component by component as

$$q(\hat{b}_{k,m}) = \text{round}\left(\frac{\hat{b}_{k,m}}{e_{k,m}}\right), \quad (3.21)$$

where the “round” function means round to the nearest integer. Here $e_{k,m}$ is a scale factor that depends on the particular frequency (k, m) being quantized. The 8×8 matrix \mathbf{e} with components $e_{k,m}$ is called the *quantization matrix*. A typical quantization matrix is (see [26])

$$\mathbf{e} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \quad (3.22)$$

The quantized matrix $q(\tilde{\mathbf{B}})$ will consist mostly of small integers and, we hope, many zeros. As a result $q(\tilde{\mathbf{B}})$ should be easy to compress.

The next step is the actual compression. Here a combination of run-length encoding followed by Huffman coding is employed. In run-length encoding the length of sequences of zeros in $q(\tilde{\mathbf{B}})$ are recorded in a list, along with the nonzero elements that interrupt the sequences. The placement of the zeros may be exploited by using the so-called zigzag encoding. See [26, p. 406] for further details.

The reconstruction consists of decoding the compressed data stream to obtain each quantized DCT $q(\tilde{\mathbf{B}})$. For any given 8×8 DCT we then dequantize each entry as

$$\tilde{b}_{k,m} = q(\hat{b}_{k,m})e_{k,m}, \quad 0 \leq k, m \leq 7, \quad (3.23)$$

where $e_{k,m}$ is the corresponding element of the quantization matrix. The inverse DCT is then applied to the 8×8 array $\tilde{\mathbf{B}}$ to obtain the relevant 8×8 portion of the reconstructed image. The resulting inverse transform consists of floating point numbers, but it would be rounded or quantized to integers.

■ EXAMPLE 3.1

Here is an example that shows how a single 8×8 block might be handled. Consider the matrix \mathbf{B} of grayscale intensities below:

$$\mathbf{B} = \begin{bmatrix} 47 & 32 & 75 & 148 & 192 & 215 & 216 & 207 \\ 36 & 82 & 161 & 196 & 205 & 207 & 190 & 140 \\ 86 & 154 & 200 & 203 & 213 & 181 & 143 & 82 \\ 154 & 202 & 209 & 203 & 159 & 145 & 147 & 127 \\ 184 & 207 & 199 & 147 & 134 & 127 & 137 & 138 \\ 205 & 203 & 125 & 72 & 123 & 129 & 150 & 115 \\ 209 & 167 & 126 & 107 & 111 & 94 & 105 & 107 \\ 191 & 129 & 126 & 136 & 106 & 54 & 99 & 165 \end{bmatrix}.$$

This image is scaled from 0 to 255. Viewed as a 8-bit grayscale image, the matrix \mathbf{B} is shown on the left in Figure 3.14. If we subtract 127 from each entry and perform a DCT, we obtain (rounded to one figure after the decimal)

$$\hat{\mathbf{B}} = \begin{bmatrix} 148.9 & 9.5 & -43.3 & -4.1 & -11.1 & 2.9 & -16.0 & 17.5 \\ 66.0 & -216.4 & -190.8 & 4.8 & -38.8 & 13.8 & 6.9 & -10.2 \\ -88.3 & -148.2 & 11.6 & 22.6 & 71.9 & 22.5 & 12.7 & 0.4 \\ -19.7 & -55.2 & 90.5 & 60.8 & -16.6 & -12.5 & -2.1 & 3.8 \\ 16.1 & -22.8 & 42.3 & -44.1 & 40.6 & 12.8 & 8.3 & -11.3 \\ -5.3 & 15.7 & 11.3 & 23.3 & 5.9 & 0.0 & -6.3 & -9.1 \\ -9.6 & -9.3 & 11.9 & 24.7 & -1.8 & -14.7 & -9.1 & 8.0 \\ -6.5 & 16.6 & -17.7 & -2.3 & 4.1 & -4.7 & 10.2 & 4.6 \end{bmatrix}$$

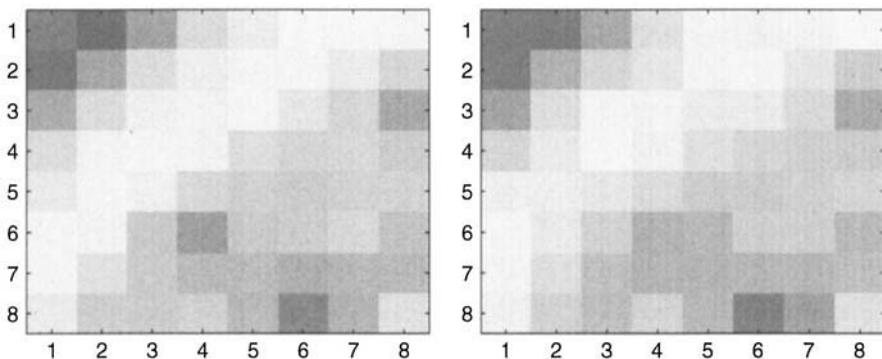


FIGURE 3.14 Grayscale view of \mathbf{B} and compressed/decompressed $\hat{\mathbf{B}}$.

Quantization of $\widehat{\mathbf{B}}$ with equation (3.21) and the quantization matrix in equation (3.22) yields

$$q(\widehat{\mathbf{B}}) = \begin{bmatrix} 9 & 1 & -4 & 0 & 0 & 0 & 0 & 0 \\ 6 & -18 & -14 & 0 & -1 & 0 & 0 & 0 \\ -6 & -11 & 1 & 1 & 2 & 0 & 0 & 0 \\ -1 & -3 & 4 & 2 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The matrix above would become part of the compressed image. As desired $q(\widehat{\mathbf{B}})$ contains a large fraction of zeros and many other integer entries that are close to zero.

Let's look at what happens when we decompress the image. From equation (3.23) the dequantized version of the DCT for this block is given by

$$\widetilde{\mathbf{B}} = \begin{bmatrix} 144 & 11 & -40 & 0 & 0 & 0 & 0 & 0 \\ 72 & -216 & -196 & 0 & -26 & 0 & 0 & 0 \\ -84 & -143 & 16 & 24 & 90 & 0 & 0 & 0 \\ -14 & -51 & 88 & 58 & 0 & 0 & 0 & 0 \\ 18 & -22 & 37 & -56 & 68 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Compare this to $\widehat{\mathbf{B}}$ above. If we apply the inverse DCT to $\widetilde{\mathbf{B}}$ above, add 128 to all entries, and round to the nearest integer, we obtain

$$\begin{bmatrix} 43 & 39 & 80 & 159 & 206 & 205 & 206 & 223 \\ 44 & 94 & 145 & 176 & 206 & 219 & 182 & 127 \\ 77 & 156 & 210 & 203 & 195 & 195 & 151 & 84 \\ 143 & 190 & 221 & 202 & 165 & 144 & 135 & 128 \\ 199 & 198 & 181 & 150 & 128 & 128 & 137 & 144 \\ 217 & 188 & 135 & 95 & 105 & 137 & 135 & 108 \\ 208 & 161 & 114 & 100 & 103 & 105 & 109 & 116 \\ 200 & 133 & 113 & 139 & 110 & 49 & 79 & 171 \end{bmatrix}$$

The matrix above is shown as a grayscale image on the right in Figure 3.14. This distortion is about 6.8%. Below we show for comparison the difference of that

matrix minus the original matrix \mathbf{B} :

$$\begin{bmatrix} -4 & 7 & 5 & 11 & 14 & -10 & -10 & 16 \\ 8 & 12 & -16 & -20 & 1 & 12 & -8 & -13 \\ -9 & 2 & 10 & 0 & -18 & 14 & 8 & 2 \\ -11 & -12 & 12 & -1 & 6 & -1 & -12 & 1 \\ 15 & -9 & -18 & 3 & -6 & 1 & 0 & 6 \\ 12 & -15 & 10 & 23 & -18 & 8 & -15 & -7 \\ -1 & -6 & -12 & -7 & -8 & 11 & 4 & 9 \\ 9 & 4 & -13 & 3 & 4 & -5 & -20 & 6 \end{bmatrix}.$$

At an additional level of sophistication the entire quantization matrix \mathbf{e} could be multiplied by a number r greater or less than 1 depending on the characteristics of the block. The choice $r > 1$ then reduces the number of bits required to store each entry by producing a “coarser” quantization. This, of course, decreases the fidelity of the reconstruction. Taking $r < 1$ has the opposite effect. The choice of r could vary on a block-by-block basis.

Figure 3.15 shows the original image and reconstructions for $r = 0.1$, 0.5 , and 2 , along with the original. The distortions are 2.4% , 5.0% , and 9% , respectively.

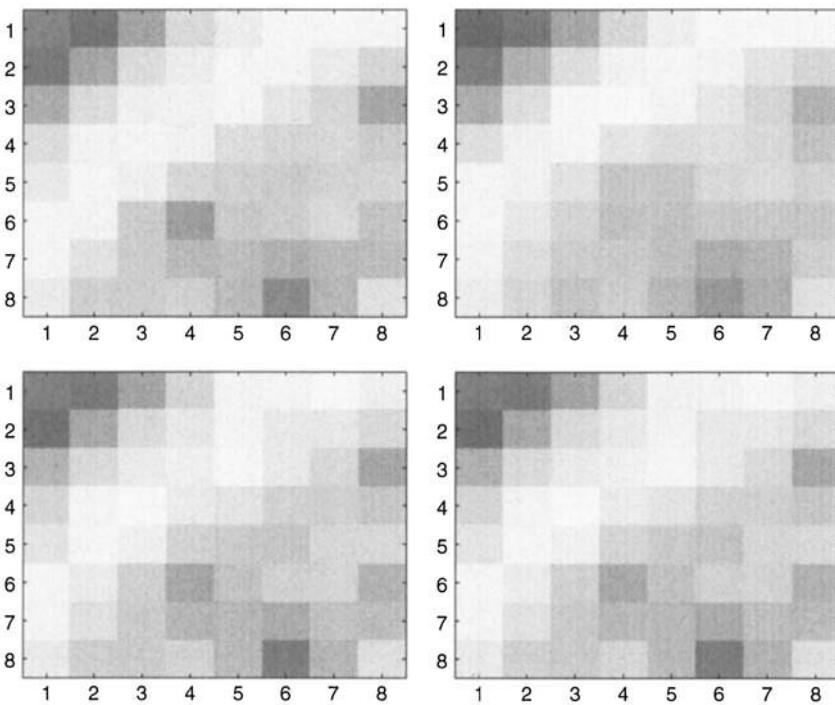


FIGURE 3.15 Original image (top left) and reconstructions with $r = 0.1$ (top right), $r = 0.5$ (bottom left), and $r = 2.0$ (bottom right).

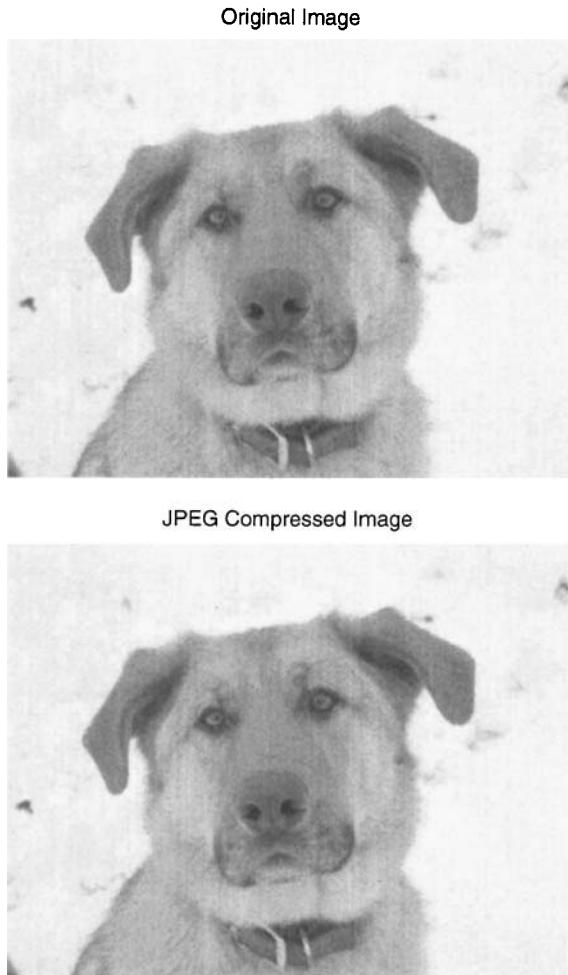


FIGURE 3.16 Original dog image (*top*) and JPEG compressed dog (*bottom*).

3.8.3 The JPEG Dog

In Figure 3.16 on the top we see the dog processed with the quantization matrix \mathbf{e} of equation (3.22) and then reconstructed; the original dog image is on the bottom. The resulting quantized block DCT is 90% zeros, which should compress well. The distortion is 2.4%. Some blocking artifacts are evident.

3.8.4 Sequential versus Progressive Encoding

After the block DCT is computed and quantized, the data can be organized for compression in a couple different ways. One way is *sequentially*, block by block, so that as the image is decoded and displayed each 8×8 block is displayed with full

resolution, typically starting from the upper left and sweeping left to right and down until the bottom right corner of the image is displayed.

An alternative is progressive encoding/decoding, in which the lowest frequency DCT coefficients (starting with the dc coefficients) for each 8×8 block are transmitted, followed by the higher frequency components. As the information arrives and is decoded and displayed, the visual effect is that of a blurry image that gets progressively sharper and sharper.

To be more specific, let $\widehat{\mathbf{A}}$ be the full $m \times n$ block DCT of the $m \times n$ image \mathbf{A} . Let $\widehat{\mathbf{A}}_{k,l}$ be the $m \times n$ matrix obtained by zeroing out all DCT coefficients in $\widehat{\mathbf{A}}$ except the one in each 8×8 block corresponding to the frequency vector (k, l) given in Table 3.7. For example, $\widehat{\mathbf{A}}_{0,0}$ would be obtained by zeroing all elements in each 8×8 block EXCEPT the dc coefficient.

Define $\widehat{\mathbf{A}}_s$ as

$$\widehat{\mathbf{A}}_s = \sum_{k+l=s} \widehat{\mathbf{A}}_{k,l},$$

for example, $\widehat{\mathbf{A}}_0 = \widehat{\mathbf{A}}_{0,0}$, $\widehat{\mathbf{A}}_1 = \widehat{\mathbf{A}}_{1,0} + \widehat{\mathbf{A}}_{0,1}$, and $\widehat{\mathbf{A}}_2 = \widehat{\mathbf{A}}_{2,0} + \widehat{\mathbf{A}}_{1,1} + \widehat{\mathbf{A}}_{0,2}$. If $k + l$ is small, then $\widehat{\mathbf{A}}_{k,l}$ carries low frequency information about the image, which corresponds to low resolution features in the image. If $k + l$ is large, then $\widehat{\mathbf{A}}_{k,l}$ carries higher frequency information, which corresponds to detailed features in the image. As a result the $m \times n$ matrix $\widehat{\mathbf{A}}_s$ encodes higher and higher frequency information (finer and finer detail) about the image as s increases. For the 8×8 block size the highest frequency block in the DCT is $k = 7, l = 7$, so there's no point in using $s > 14$.

Let us also define

$$\mathbf{A}_s = \text{iblkdct}(\widehat{\mathbf{A}}_s)$$

where “iblkdct” denotes the inverse of the block DCT transform. Each matrix \mathbf{A}_s lives back in the image domain and contains finer and finer detail information about the original image as s increases. The quantity

$$\sum_{s=0}^p \mathbf{A}_s \tag{3.24}$$

amalgamates the information in the \mathbf{A}_s and provides a sharper and sharper version of the original image as p increases. If $p = 14$, the sum in (3.24) reconstructs the full image \mathbf{A} , since

$$\widehat{\mathbf{A}} = \sum_{k=0}^7 \sum_{l=0}^7 \widehat{\mathbf{A}}_{k,l} = \sum_{s=0}^{14} \sum_{k+l=s} \widehat{\mathbf{A}}_{k,l} = \sum_{s=0}^{14} \widehat{\mathbf{A}}_s$$

so applying the linear operator “iblkdct” to both sides above yields

$$\mathbf{A} = \sum_{s=0}^{14} \mathbf{A}_s.$$

If the data are suitably arranged (transmit the dc coefficient for all blocks first, followed by the $(0, 1)$ frequency for each block, then the $(1, 1)$, the $(2, 0)$, $(1, 1)$, $(0, 2)$ frequencies, etc.), we can implement progressive transmission as follows:

1. Compute each $\hat{\mathbf{A}}_s$ and \mathbf{A}_s as the data become available.
2. For each $0 \leq p \leq 14$, once all \mathbf{A}_s for $0 \leq s \leq p$ have been constructed, display the sum in equation (3.24) while waiting for further data to arrive.

■ EXAMPLE 3.2

The original image \mathbf{A} is shown at the top left in Figure 3.17. The image embodied by \mathbf{A}_0 is that obtained by reconstructing only from the dc coefficients, and was shown on the right in Figure 3.13. The image at the top right in Figure 3.17 is \mathbf{A}_1 in the scheme's notation given above. The images at the bottom left and right are \mathbf{A}_2 and \mathbf{A}_3 , respectively.

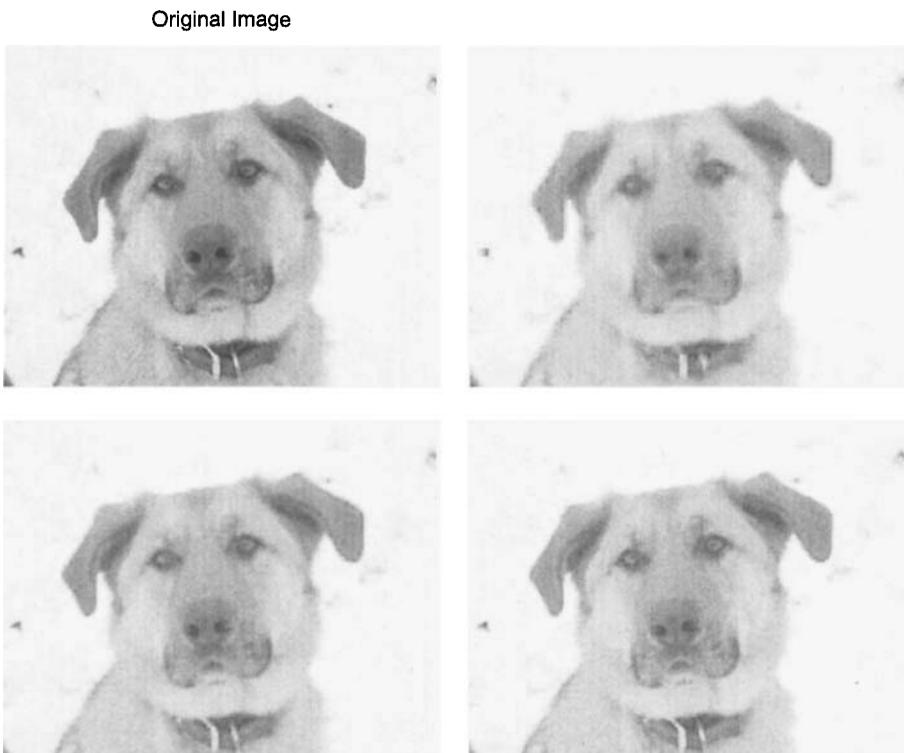


FIGURE 3.17 Original image (*top left*), \mathbf{A}_1 (*top right*), \mathbf{A}_2 (*bottom left*), and \mathbf{A}_3 (*bottom right*).

3.9 MATLAB PROJECT

In this section we make use of Matlab to explore the DCT and its use in compression. If you don't have the Matlab Signal Processing Toolbox (which contains the `dct` and `dct2` commands), alternate versions `kdct` and `kdct2` are available at the Web site [28].

1. Start Matlab. Load in the "train" signal with the command `load('train')`; Recall that the audio signal is loaded into a variable "y" and the sampling rate into "Fs." The sampling rate is 8192 Hertz.
 - a. Compute the DFT of y with `Y = fft(y);`, then display the magnitude of the DFT up to the Nyquist frequency with the command `plot(abs(Y(1:6441)))`.
 - b. Compute the DCT of y with `Yc = dct(y);`, then display the magnitude of the DCT on a separate plot (open a new plotting window with `figure(2)`) with `plot(abs(Yc))`. Compare to the DFT plot.
2. Here's a simple compression scheme for audio that's in the spirit of JPEG. The supplied command `audiocompress` accepts as an input argument a column vector `y` of double-precision floating point numbers between -1 and 1. The command also accepts a positive integer `m` and a scaling parameter `r`. A DCT is applied to consecutive non-overlapping sections of `y` of size `m`. Each frequency Y_k in the `m`-point DCT is then quantized to an integer value as `round(Y_k/r)`. If $r > 1$, the quantization is coarser and presumably more "compressible"; $r < 1$ has the opposite effect.

Try using the command with the "splat" signal (`load('splat')`). Start with block size $m = 50$ and $r = 0.01$, as

```
[ycomp, P] = audiocompress(y, 50, 0.01);
```

The output variable `ycomp` is the compressed/decompressed audio signal, and `P` is the fraction of frequencies in the block DCT that are not zeroed out by the quantization. In the case above this should be about one-third.

Try varying the block size m by choosing a few other values in the range $m = 10$ to $m = 5000$. In each case adjust r so that the fraction of nonzero quantized coefficients is about one-third, listen to the compressed/decompressed signal, and compute the distortion. Make a table showing the value of m , the distortion, and your subjective rating of the sound quality. Is there an optimal choice for m ?

3. Alter the `audiocompress` command to use the DFT instead of the DCT (change the `dct` to `fft`, and `idct` to `ifft`; you'll also need to enclose the `ifft` command inside `real`, to eliminate any imaginary round-off errors). Repeat the last problem for each block size and compare the performance of the DFT to the DCT for this application. Is the DFT consistently better or worse than the DCT? Why?

How high can the compression go before you can detect it by listening to the sound?

4. The supplied command `jpegdemo` accepts as an input argument an array A of integers or double-precision floats, assumed to encode a grayscale image in the range 0 to 255. It also accepts a scaling parameter r . The image array is block transformed and quantized as described in Section 3.8. The output of the command is the “compressed/decompressed” version of the image as an array of double-precision floating numbers in the interval [0, 255], and the fraction of nonzero DCT coefficients remaining. In reality we’d return the output image as unsigned integers, but with the floating point numbers we can do arithmetic.

Load in an image file, preferably a losslessly encoded image—for example, something in a TIFF format (but JPEG will do) with the command (command `y = imread('myimage.tif');`). Use this to simulate a grayscale image; for example, if the image is loaded into `z`, then use

```
zg = 0.2989*double(z(:,:,1))+0.5870*double(z(:,:,2))
+0.1140*double(z(:,:,3));
```

Subject the image to simulated JPEG compression with

```
[z2,P] = jpegdemo(zg, 1.0);
```

Take note of the compression factor P . And of course, display the image and compute the distortion as in equation (1.39). Recall that the command `colormap([(0:n)/n; (0:n)/n; (0:n)/n]');` sets up a grayscale color map on the range 0 to n .

Vary the scaling parameter r over a wide range (e.g., 0.1 to 100). Make a table showing the value of r , the compression ratio, distortion, and visual appearance of the image.

5. Does repeatedly subjecting an image to JPEG compression continue to degrade the image? Execute `[z2,P] = jpegdemo(zg, 1.0);` and then run the command

```
[z2,P] = jpegdemo(z2, 1.0);
```

10 times or more, to repeatedly subject the image to compression and decompression. Compute the distortion when done, and compare to the distortion after only one iteration. Can you see any degradation in the image? Where in this process would any degradation (after the first application of `jpegdemo`) occur?

6. The supplied command `jpegprogressive` is identical to the previously introduced command `jpegdemo`, but accepts a third integer argument “ p ” as in equation (3.24). As described in Section 3.8.1, with the choice $p = 0$ the

image will be reconstructed from the DC coefficients only, while $p = 14$ will yield the full resolution reconstruction.

Use the your image from the last problem and execute

```
[z2,P] = jpegprogressive(zg, 1.0, p);
```

for $p = 0, 1, 2, 5, 10, 14$. Why does the $p = 0$ image look the way it does?

EXERCISES

Computation

- 3.1 Write out the matrix \mathcal{C}_3 , and use it to compute the DCT of the vector $\mathbf{x} = (1, 2, -1)$.
- 3.2 Use the matrix \mathcal{C}_3^T to compute the inverse DCT of the vector $\mathbf{X} = (3, 0, 1)$.
- 3.3 Let \mathbf{e}_m , $0 \leq m \leq N - 1$, denote the m th standard basis vector in \mathbb{R}^N . Write out the DCT of \mathbf{e}_m explicitly.
- 3.4 Let $f(t) = t^2$ on the interval $0 \leq t \leq 1$. Let \mathbf{x} be the vector in \mathbb{R}^{1024} obtained by sampling $f(t)$ at $t = k/1024$ for $0 \leq k \leq 1023$. Compute the DCT of \mathbf{x} in Matlab. Threshold the DCT \mathbf{X} at levels suitable to obtain compression to (roughly) 0.1%, 0.5%, 1.0%, and 5.0% of the original size of \mathbf{X} (i.e., eliminate 99.9%, 99.5%, 99.0%, and 95.0% of the DCT coefficients). For each case compute the distortion in the reconstructed signal.

Repeat with the DFT in place of the DCT. Which transform yields better results, and why?

- 3.5 In Matlab the command `dct(A)` computes the DCT of the matrix A , column by column (as opposed to `dct2`, which is the full two-dimensional DCT). Explain why the k th row of the matrix obtained from `dct(eye(N))` is the vector $\mathcal{C}_{N,k}$.
- 3.6 Use Matlab to plot the cosine basis vectors or waveforms $\mathcal{C}_{N,k}$ for $N = 100$ and several values of k . Verify that the graph of $\mathcal{C}_{N,k}$ crosses the horizontal axis k times. Exercise 3.5 may be useful.
- 3.7 Let

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 5 & 7 & 9 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 5 & 3 & 1 & 4 \\ 5 & 3 & 3 & 0 & 0 \end{bmatrix}.$$

Use Matlab to compute the two-dimensional DCT of \mathbf{A} (command `dct2`). Quantize/dequantize the DCT \mathbf{A} by applying the function $x \rightarrow d \text{ round}(x/d)$

to $\widehat{\mathbf{A}}$ component by component, using $d = 1.0$ (thus rounding each $\hat{a}_{k,l}$ to the nearest integer), and then inverse transform. Compute the percentage error $100\|\mathbf{A} - \widehat{\mathbf{A}}\|^2/\|\mathbf{A}\|^2$, where $\widehat{\mathbf{A}}$ is the inverse transformed quantized matrix. Here $\|\mathbf{A}\|$ is the “Frobenius” norm of Example 1.14. In Matlab execute `norm(A, 'fro')`.

Repeat with $d = 0.01, 0.1, 10.0$, and $d = 100.0$, and report the percentage error in each case.

DCT Properties

- 3.8** Show that the two-dimensional DCT of the $N \times N$ identity matrix is again the $N \times N$ identity matrix.
- 3.9** Compute (by hand) the two-dimensional DCT of the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}.$$

Then inverse transform the result.

- 3.10** Show that the DCT is an orthogonal transform, that is,

$$\|\text{DCT}(\mathbf{x})\|^2 = \|\mathbf{x}\|^2$$

using the usual Euclidian norm for vectors in \mathbb{C}^N . This is the DCT version of Parseval’s identity. *Hint:* For any vector $\mathbf{v} \in \mathbb{C}^N$, we have $\|\mathbf{v}\|^2 = \mathbf{v}^*\mathbf{v}$, where \mathbf{v}^* is the row vector obtained as the conjugate transpose of \mathbf{v} .

- 3.11** Let \mathbf{x} be a vector in \mathbb{R}^N with DCT \mathbf{X} . Show that

$$X_0 = \sqrt{N} \text{ ave}(\mathbf{x}),$$

where $\text{ave}(\mathbf{x})$ denotes the mean value of the components of \mathbf{x} .

Show also that if $\mathbf{A} \in M_{m,n}(\mathbb{R})$ has two-dimensional DCT $\widehat{\mathbf{A}}$ then

$$\hat{a}_{0,0} = \sqrt{mn} \text{ ave}(\mathbf{A}).$$

- 3.12** Prove equation (3.20). Use this to find the corresponding explicit inversion formula for the two-dimensional DCT (analogous to the explicit formula (3.18) for the two-dimensional DCT).
- 3.13** Suppose that $\mathbf{x} \in \mathbb{R}^N$ has components that satisfy $|x_k| \leq M$. Show that if $\mathbf{X} = \text{DCT}(\mathbf{x})$, then $|X_k| \leq M\sqrt{2N}$. Derive an analogous bound for the two-dimensional DCT.

- 3.14** Suppose that $\mathbf{x} \in \mathbb{R}^N$ has DCT \mathbf{X} . Let $\tilde{\mathbf{x}}$ denote the vector with components $\tilde{x}_k = x_{N-k-1}$, that is,

$$\tilde{\mathbf{x}} = (x_{N-1}, x_{N-2}, \dots, x_1, x_0).$$

- a. Show that the DCT $\tilde{\mathbf{X}}$ of $\tilde{\mathbf{x}}$ has components $\tilde{X}_k = (-1)^k X_k$.
 - b. Suppose that “ R ” denotes the reversal operation that takes \mathbf{x} to $\tilde{\mathbf{x}}$ and “ C ” denotes compression via the DCT by thresholding. Explain why the result of part (a) shows that $R(C(\mathbf{x})) = C(R(\mathbf{x}))$. That is, our approach to compression is invariant with respect to “time-reversal.” *Hint:* Show that $R(C(\mathbf{x}))$ and $C(R(\mathbf{x}))$ have the same DCT (and since the DCT is invertible, must be the same vector).
- 3.15** Use Matlab to compute C_3^k for $k = 1, 2, 3, \dots$. Does it appear that C_3^k is a multiple of \mathbf{I} for any choice of k ? (Compare to Exercise 2.7 in Chapter 2.)
Is there any evidence that C_N^k is a multiple of \mathbf{I} for some $N > 2$ and some k ?
- 3.16** This is a simple variation on Exercise 2.14, in which the DCT replaces the DFT.

Let \mathbf{x} and \mathbf{y} be two column vectors of length m and n , respectively, with one-dimensional DCT’s \mathbf{X} and \mathbf{Y} . Let \mathbf{z} be the product image defined by

$$z_{r,s} = x_r y_s$$

for $0 \leq r \leq m-1$, $0 \leq s \leq n-1$, and \mathbf{Z} the two-dimensional DCT of \mathbf{z} .

- a. Show that \mathbf{z} is an $m \times n$ matrix satisfying $\mathbf{z} = \mathbf{x}\mathbf{y}^T$.
- b. Show that

$$Z_{k,l} = X_k Y_l$$

or equivalently $\mathbf{Z} = \mathbf{XY}^T$.

- 3.17** This is a 2D version of Exercise 3.14. Let \mathbf{A} denote be an $m \times n$ image matrix and let \mathbf{B} denote the “90 degree clockwise rotation” of \mathbf{A} . (This isn’t quite the same as taking the transpose!) Note that \mathbf{B} is $n \times m$, and in fact

$$b_{r,s} = a_{m-s-1,r}$$

with all rows and columns indexed from 0. Make sure you believe this, for example, $b_{0,0} = a_{m-1,0}$, $b_{0,m-1} = a_{0,0}$, $b_{n-1,m-1} = a_{0,n-1}$, and $b_{n-1,0} = a_{m-1,n-1}$.

- a. Find the coefficients $\hat{b}_{r,s}$ of the two-dimensional DCT $\widehat{\mathbf{B}}$ in terms of the coefficients $\hat{a}_{r,s}$ of the DCT of \mathbf{A} .

- b.** Suppose that “ R ” denotes the clockwise rotation operation that takes \mathbf{A} to \mathbf{B} (i.e., $\mathbf{B} = R(\mathbf{A})$) and “ C ” denotes compression via the DCT and thresholding. Explain why the result of part (b) shows that $C(R(\mathbf{x})) = R(C(\mathbf{x}))$ (rotating and compressing yields the same result as compressing and then rotating, or equivalently, $R^{-1}(C(R(\mathbf{x}))) = C(\mathbf{x})$). Our approach to compression is thus invariant with respect to rotation.

Hint: Work in the frequency domain. Show that $R(C(\mathbf{A}))$ and $C(R(\mathbf{A}))$ have the same 2D DCT (and since the DCT is invertible, must be the same vector).

JPEG

- 3.18** Suppose that \mathbf{A} is a grayscale image in which each pixel consists of an integer in the range 0 to 255. We compress and then decompress \mathbf{A} , using the JPEG scheme outlined in the text, to produce image $\tilde{\mathbf{A}}$, consisting of integers in the range 0 to 255. Ignore any floating point round-off error; that is, assume all computations involving the block DCT are done “exactly.” At which point(s) in this process is information lost?

- 3.19** The quantization used in JPEG in the frequency domain is of the form

$$q(x) = \text{round}\left(\frac{x}{d}\right)$$

for some real number d (see equation (3.21)). What are the quantization intervals in this scheme, and how do they depend on d ? (There are in principle infinitely many quantization intervals here, so this quantization isn’t perfectly in accordance with Section 1.9.)

If we dequantize as $\tilde{q}(q(x)) = dq(x)$ (equation (3.23)), what are the corresponding codewords?

- 3.20** Is there any choice for the quantization matrix \mathbf{e} for which the quantization process of equations (3.21) through (3.23) will yield lossless compression? Why or why not?

CHAPTER 4

CONVOLUTION AND FILTERING

4.1 OVERVIEW

The term *filtering* refers to the systematic alteration of the frequency content of a signal or image. In particular, we sometimes want to “filter out” certain frequencies. The operation is often linear and usually (but not always) performed on the time domain version of the signal.

One important tool for filtering in the time domain is *convolution*. The form taken by convolution depends on the vector space in which the signals reside. We’ll begin by examining convolution for finite-length one-dimensional signals, then look at convolution in two-dimensions (images), and finally we’ll examine convolution for infinite and bi-infinite signals. We also introduce the *z-transform*, a powerful tool in signal and image processing.

4.2 ONE-DIMENSIONAL CONVOLUTION

4.2.1 Example: Low-pass Filtering and Noise Removal

Recall the example of Section 2.3, in which the function $x(t) = 2.0 \cos(2\pi \cdot 5t) + 0.8 \sin(2\pi \cdot 12t) + 0.3 \cos(2\pi \cdot 47t)$ was sampled on the time interval $[0, 1]$ at times $t = k/128$ for integers $0 \leq k \leq 127$. The goal in that example was the removal of the 47 Hertz component in the sampled signal, perhaps because it represented noise. In that case the noise was deterministic, not random, but that doesn’t matter for

the moment. We removed the high-frequency portion of the signal with frequency domain methods, by performing a DFT (though we hadn't actually defined the DFT at that point), zeroing out the 47 Hertz component, and then re-synthesizing the signal without the noise by using an inverse DFT.

Let's consider how such a goal might be achieved, at least approximately, with strictly time domain methods. Let $\mathbf{x} \in \mathbb{R}^N$ be obtained by sampling some function $x(t)$ on time interval $[0, 1]$, at times $t = k/N$ for $0 \leq k \leq N - 1$. As usual, we index vector components from 0 to $N - 1$. For simplicity we assume that $x(t)$ is periodic in time with period 1, and that $x(t)$ consists of a superposition of low frequencies that represent the "real" signal and high frequencies that represent unwanted noise, random or deterministic. The frequency content of the analog signal $x(t)$ is assumed to lie in the range 0 to $N/2$ Hertz, so aliasing is not an issue.

A simple approach to partially removing the high frequencies from the signal while leaving low frequencies is to use a *moving average*, by replacing each sample x_k with the average value of itself and nearby samples. For example, let us take $\mathbf{w} \in \mathbb{R}^N$ as the vector defined by

$$w_k = \frac{1}{2}x_{k-1} + \frac{1}{2}x_k \quad (4.1)$$

for $0 \leq k \leq N - 1$. Equation (4.1) presents a problem in the case where $k = 0$, since x_{-1} is undefined. We will thus interpret all indexes "modulo N ," so, for example, $x_{-1} = x_{N-1}$. This is in keeping with the assumption that the function $x(t)$ is periodic with period 1, so $x_{-1} = x(-1/N) = x((N-1)/N) = x_{N-1}$. We can thus consider x_k as defined for all k via $x_k = x_{k \bmod N}$, if necessary.

The vector \mathbf{w} defined by equation (4.1) will be a smoothed version of \mathbf{x} , with much less high-frequency content. The is best seen by considering this two-point averaging scheme's effect in the case that $x(t)$ consists of a basic waveform, complex exponential or trigonometric. For example, if $x(t) = \sin(2\pi qt)$, then

$$\begin{aligned} w_k &= \frac{1}{2}x_{k-1} + \frac{1}{2}x_k \\ &= \frac{1}{2} \sin\left(\frac{2\pi q(k-1)}{N}\right) + \frac{1}{2} \sin\left(\frac{2\pi qk}{N}\right) \\ &= \left(\frac{1 + \cos(2\pi q/N)}{2}\right) \sin\left(\frac{2\pi qk}{N}\right) - \frac{1}{2} \sin\left(\frac{2\pi q}{N}\right) \cos\left(\frac{2\pi qk}{N}\right) \\ &= A \sin\left(\frac{2\pi qk}{N}\right) - B \cos\left(\frac{2\pi qk}{N}\right), \end{aligned}$$

where $A = (1 + \cos(2\pi q/N))/2$, $B = \frac{1}{2} \sin(2\pi q/N)$, and we make use of $\sin(a - b) = \sin(a)\cos(b) - \cos(a)\sin(b)$ with $a = 2\pi qk/N$, $b = 2\pi q/N$. If q is close to zero (more accurately, if q/N is close to zero), then $A \approx 1$ and $B \approx 0$. As a consequence $w_k \approx \sin(2\pi qk/N) = x_k$. In short, a low frequency waveform passes through the two-point averaging process largely unchanged. On the other hand, if $q \approx N/2$

(the Nyquist frequency), then $A \approx 0$ and $B \approx 0$, so $w_k \approx 0$. The highest frequencies we can represent will be nearly zeroed out by this process.

■ EXAMPLE 4.1

In Figure 4.1 we show on the left the original signal vector \mathbf{x} from Section 2.3 and on the right the vector \mathbf{w} produced by application of equation (4.1), both plotted as functions on the interval $[0, 1]$. In Figure 4.2 we plot the DFT magnitude for both the original and smoothed signals, on the frequency range 0 to 64 Hertz. The magnitude of the 5 Hertz spike has changed from 128 to 126.1, while the 12 Hertz spike has changed from 51.2 to 49.0. The 47 Hertz portion has diminished in magnitude from 19.2 to 7.8, considerably more (on a percentage basis) than the lower two frequencies.

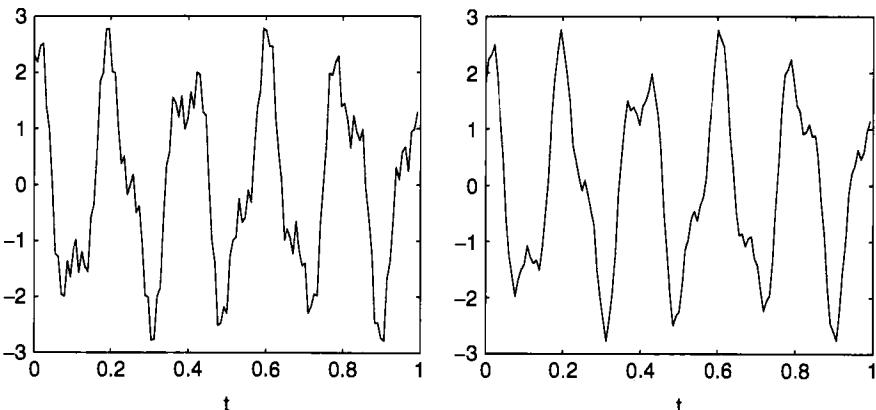


FIGURE 4.1 Original and smoothed signal.

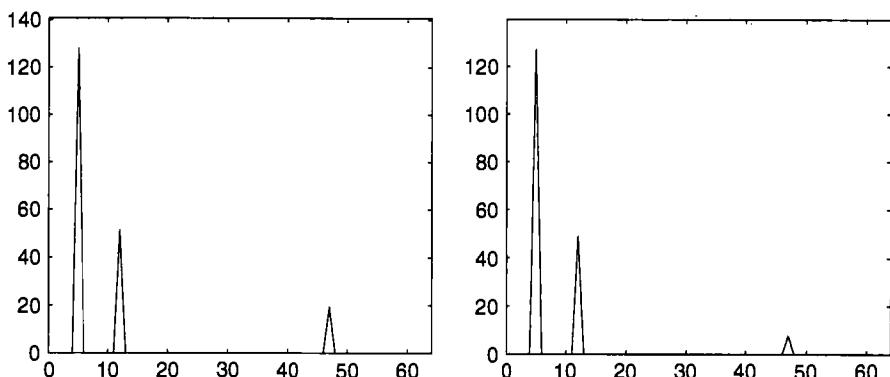


FIGURE 4.2 Original and smoothed signal DFT.

Example 4.1 illustrates *low-pass filtering*, in which one attempts to remove high frequencies from a signal while leaving lower frequencies unchanged. The two-point moving average procedure is obviously imperfect in this application, for the 5 and 12 Hertz frequencies were slightly altered and the 47 Hertz component was not completely removed. We can improve the performance of this low-pass filter, and indeed design other types of filters, by using more sophisticated averages as discussed below.

It's worth noting that the two-point moving average filter above has a simple matrix interpretation. In the case $N = 8$, for example, we have $\mathbf{w} = \mathbf{M}\mathbf{x}$ for an appropriate matrix \mathbf{M} :

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_0 \\ x_2 + x_1 \\ x_3 + x_2 \\ x_4 + x_3 \\ x_5 + x_4 \\ x_6 + x_5 \\ x_7 + x_6 \end{bmatrix}.$$

A two-point average for larger sample vectors has an analogous structure. In particular, it's clear that this process is linear.

4.2.2 Convolution

The low-pass filtering operation above is a special case of *convolution*, an operation that plays an important role in signal and image processing, and indeed many areas of mathematics. In what follows we will assume that all vectors in \mathbb{C}^N are indexed from 0 to $N - 1$. Moreover, when convenient, we will assume that the vectors have been extended periodically with period N in the relevant index, via $x_k = x_{k \bmod N}$.

Remark 4.1 If we extend a vector \mathbf{x} periodically to all index values k via $x_k = x_{k \bmod N}$, then for any value of m we have

$$\sum_{k=m}^{m+N-1} x_k = \sum_{k=0}^{N-1} x_{k+m} = \sum_{k=0}^{N-1} x_k.$$

In short, the sum of any N consecutive components of \mathbf{x} is the same, a fact we'll use over and over again; see Exercise 4.4.

Convolution Definition Let's recast the filtering operation above in a more general format. We begin with a definition.

Definition 4.2.1 Let \mathbf{x} and \mathbf{y} be vectors in \mathbb{C}^N . The circular convolution of \mathbf{x} and \mathbf{y} is the vector $\mathbf{w} \in \mathbb{C}^N$ with components

$$w_r = \sum_{k=0}^{N-1} x_k y_{(r-k) \bmod N} \quad (4.2)$$

for $0 \leq r \leq N - 1$. The circular convolution is denoted $\mathbf{w} = \mathbf{x} * \mathbf{y}$.

For now we'll drop the "circular" prefix and refer to this operation as just "convolution." Of course, since we assume that all vectors are extended periodically, we can ignore the modulo N arithmetic on indexes, but we put it explicitly in equation (4.2) for the sake of clarity.

Convolution looks a bit confusing, but here's a simple way to visualize it. We compute the quantity w_0 by taking the vector \mathbf{x} and the vector \mathbf{y} indexed in reverse order starting at $k = 0$, and lining them up:

$$\begin{array}{cccccc} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ y_0 & y_{N-1} & y_{N-2} & \cdots & y_1 \end{array}$$

Then w_0 is simply the dot product of these two rows, $w_0 = x_0 y_0 + x_1 y_{N-1} + \cdots + x_{N-1} y_1$. To compute w_1 , take the y_1 at the end of the second row and move it to the front, pushing other components to the right, as

$$\begin{array}{cccccc} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ y_1 & y_0 & y_{N-1} & \cdots & y_2 \end{array}$$

The dot product of these two row vectors is w_1 . To compute w_2 , move y_2 to the front, push all other components to the right, and compute the dot product. Do this a total of N times, at which point \mathbf{y} will be back in its original position. The overall computation can be summarized as

	x_0	x_1	x_2	\cdots	x_{N-1}
w_0	y_0	y_{N-1}	y_{N-2}	\cdots	y_1
w_1	y_1	y_0	y_{N-1}	\cdots	y_2
w_2	y_2	y_1	y_0	\cdots	y_3
				⋮	
w_{N-1}	y_{N-1}	y_{N-2}	y_{N-3}	\cdots	y_0

Each w_r is obtained as the dot product of the corresponding row with the vector $(x_0, x_1, \dots, x_{N-1})$.

■ EXAMPLE 4.2

The two-point low-pass filtering operation from Section 4.2.1 can be cast as a convolution $\mathbf{w} = \mathbf{x} * \ell$ where $\ell \in \mathbb{C}^{128}$ has components $\ell_0 = \frac{1}{2}$, $\ell_1 = \frac{1}{2}$, and all other $\ell_k = 0$. To see this note that $\ell_{(r-k) \bmod N} = \frac{1}{2}$ when $k = r$ or $k = r - 1$ and 0 otherwise. Equation (4.2) then becomes $w_r = x_r/2 + x_{r-1}/2$, which is equation

(4.1). Indeed all the filtering operations with which we will be concerned can be implemented via an appropriate convolution.

Convolution Properties The following theorem summarizes some important algebraic properties of convolution.

Theorem 4.2.1 Let \mathbf{x} , \mathbf{y} and \mathbf{w} be vectors in \mathbb{C}^N . The following hold:

- i. *Linearity*: $\mathbf{x} * (ay + bw) = a(\mathbf{x} * \mathbf{y}) + b(\mathbf{x} * \mathbf{w})$ for any scalars a, b .
- ii. *Commutativity*: $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$.
- iii. *Matrix formulation*: If $\mathbf{w} = \mathbf{x} * \mathbf{y}$, then $\mathbf{w} = \mathbf{M}_y \mathbf{x}$, where \mathbf{M}_y is the $N \times N$ matrix

$$\mathbf{M}_y = \begin{bmatrix} y_0 & y_{N-1} & y_{N-2} & \cdots & y_1 \\ y_1 & y_0 & y_{N-1} & \cdots & y_2 \\ y_2 & y_1 & y_0 & \cdots & y_3 \\ & & & \vdots & \\ y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_0 \end{bmatrix}.$$

In particular, the row k , column m entry of \mathbf{M}_y is $y_{(k-m) \bmod N}$, rows and columns indexed from 0. Moreover $\mathbf{M}_x \mathbf{M}_y = \mathbf{M}_{\mathbf{x} * \mathbf{y}}$.

The matrix \mathbf{M}_y is called the circulant matrix for \mathbf{y} . Note that the rows of \mathbf{M}_y , or the columns, can be obtained by the circular shifting procedure described after equation (4.2).

- iv. *Associativity*: $\mathbf{x} * (\mathbf{y} * \mathbf{w}) = (\mathbf{x} * \mathbf{y}) * \mathbf{w}$.
- v. *Periodicity*: If x_k and y_k are extended to be defined for all k with period N , then the quantity w_r defined by equation (4.2) is defined for all r and satisfies $w_r = w_{r \bmod N}$.

Proof The linearity relation has a very straightforward proof; see Exercise 4.5. To prove commutativity (ii), let $\mathbf{w} = \mathbf{x} * \mathbf{y}$ so that from equation (4.2) we have

$$w_r = \sum_{k=0}^{N-1} x_k y_{r-k}$$

in which we've suppressed the modulo N arithmetic on indexes. The sum is defined for any r , since y_m is defined for any index m . The change of index $k = r - j$ (so $j = r - k$) yields

$$w_r = \sum_{j=r}^{r-N+1} x_{r-j} y_j.$$

The sequence $x_{r-j} y_j$ is periodic in j with period N and the sum for w_r above involves N consecutive terms of this sequence. By Remark 4.1, we know that we can instead

sum over the N consecutive indexes $j = 0$ to $j = N - 1$ to find

$$w_r = \sum_{j=0}^{N-1} x_{r-j} y_j,$$

which is precisely the definition of the r th component of the vector $\mathbf{y} * \mathbf{x}$.

To prove the matrix formulation (iii) start with the formal summation definition of the matrix-vector product $\mathbf{w} = \mathbf{M}_y \mathbf{x}$, which is

$$w_r = \sum_{k=0}^{N-1} (\mathbf{M}_y)_{rk} x_k,$$

where $(\mathbf{M}_y)_{rk}$ denotes the row r , column k entry of \mathbf{M}_y , defined to be $y_{(r-k) \bmod N}$. We then have

$$w_r = \sum_{k=0}^{N-1} x_k y_{(r-k) \bmod N}.$$

This is precisely the definition of $\mathbf{w} = \mathbf{x} * \mathbf{y}$ from equation (4.2).

To show that $\mathbf{M}_x \mathbf{M}_y = \mathbf{M}_{x * y}$, we start from the definition of matrix multiplication

$$\begin{aligned} (\mathbf{M}_x \mathbf{M}_y)_{k,m} &= \sum_{r=0}^{N-1} (\mathbf{M}_x)_{k,r} (\mathbf{M}_y)_{r,m} \\ &= \sum_{r=0}^{N-1} x_{k-r} y_{r-m}. \end{aligned} \tag{4.3}$$

For fixed k and m , the sequence $x_{k-r} y_{r-m}$ (considered as a sequence in r) is periodic with period N . In view of Remark 4.1 on page 141, we can shift the index of summation r in equation (4.3) (as $r \rightarrow r + m$) to find

$$(\mathbf{M}_x \mathbf{M}_y)_{k,m} = \sum_{r=0}^{N-1} x_{k-m-r} y_r.$$

The quantity on the right is by definition the $k - m$ component of the vector $\mathbf{x} * \mathbf{y}$, and by definition, this is precisely the row k , column m element of $\mathbf{M}_{x * y}$.

Associativity (iv) is easy to prove from the matrix formulation of convolution and the associativity of matrix multiplication, for

$$\begin{aligned} \mathbf{x} * (\mathbf{y} * \mathbf{w}) &= \mathbf{M}_x (\mathbf{M}_y \mathbf{w}) \\ &= (\mathbf{M}_x \mathbf{M}_y) \mathbf{w} \\ &= \mathbf{M}_{x * y} \mathbf{w} \\ &= (\mathbf{x} * \mathbf{y}) * \mathbf{w}. \end{aligned}$$

Finally, property (v) is left as Exercise 4.6. ■

4.3 CONVOLUTION THEOREM AND FILTERING

4.3.1 The Convolution Theorem

Computing the convolution of two vectors in the time domain may look a bit complicated, but the frequency domain manifestation of convolution is very simple.

Theorem 4.3.1 (The Convolution Theorem) *Let \mathbf{x} and \mathbf{y} be vectors in \mathbb{C}^N with DFT's \mathbf{X} and \mathbf{Y} , respectively. Let $\mathbf{w} = \mathbf{x} * \mathbf{y}$ have DFT \mathbf{W} . Then*

$$W_k = X_k Y_k \quad (4.4)$$

for $0 \leq k \leq N - 1$.

Proof This is a simple and direct computation. From the definition of the DFT we have

$$W_k = \sum_{m=0}^{N-1} e^{-2\pi i km/N} w_m$$

Since $\mathbf{w} = \mathbf{x} * \mathbf{y}$, we have $w_m = \sum_{r=0}^{N-1} x_r y_{m-r}$. Substitute this into the formula for W_k above and interchange the summation order to find

$$W_k = \sum_{r=0}^{N-1} \sum_{m=0}^{N-1} e^{-2\pi i km/N} x_r y_{m-r}.$$

Make a change of index in the m sum by substituting $n = m - r$ (so $m = n + r$). With the appropriate change in the summation limits and a bit of algebra we obtain

$$\begin{aligned} W_k &= \sum_{r=0}^{N-1} \sum_{n=-r}^{N-1-r} e^{-2\pi i k(n+r)/N} x_r y_n \\ &= \left(\sum_{r=0}^{N-1} e^{-2\pi i kr/N} x_r \right) \left(\sum_{n=-r}^{N-1-r} e^{-2\pi i kn/N} y_n \right) \end{aligned}$$

where in the last line we've broken apart the exponential and grouped disjoint sums together. Now note that for any fixed r the sum in n above represents N consecutive terms of the sequence $e^{-2\pi i kn/N} y_n$, which is periodic in n . By Remark 4.1 on page 141, we can shift the summation limits to $n = 0$ to $n = N - 1$ (also N consecutive terms) and obtain

$$\begin{aligned} W_k &= \left(\sum_{r=0}^{N-1} e^{-2\pi i kr/N} x_r \right) \left(\sum_{n=0}^{N-1} e^{-2\pi i kn/N} y_n \right) \\ &= X_k Y_k. \end{aligned}$$

This proves the theorem. ■

Theorem 4.3.1 will provide essential insight into the operation of filtering signals.

4.3.2 Filtering and Frequency Response

Let $\mathbf{x} \in \mathbb{C}^N$ have DFT \mathbf{X} . We will think of \mathbf{x} as a sampled signal, in which case it will have real components, but that doesn't matter at the moment. Let $\mathbf{h} \in \mathbb{C}^N$ denote a "filter" vector whose components we choose, with DFT \mathbf{H} . Define $\mathbf{w} = \mathbf{x} * \mathbf{h}$. The operation of convolving \mathbf{x} with \mathbf{h} is a type of *filtering*, with \mathbf{x} as the input to the filter and \mathbf{w} as the output. Theorem 4.3.1 quantifies exactly how the input \mathbf{x} and output \mathbf{w} are related in the frequency domain.

Below we look a little more closely at the effect of filtering in the time domain, especially how filtering affects the basic waveforms that make up the input signal. We also show how Theorem 4.3.1 can be used to design filters that predictably and systematically alter the frequency content of the input signal, entirely eliminating some while accentuating others.

Filtering Effect on Basic Waveforms It's important to note that filtering is a linear process, for it is a convolution. As such we can analyze the effect of filtering a signal by considering the effect of the filter on each of the basic waveforms $\mathbf{E}_{N,m}$ that make up the input signal, one waveform at a time. Filtering has a very simple effect on a basic waveform—it merely multiplies it by a constant!

To see this, suppose that a basic waveform $\mathbf{E}_m \in \mathbb{C}^N$ is used as the input to the filter \mathbf{h} and let $\mathbf{w} = \mathbf{E}_m * \mathbf{h}$ be the output; we suppress the dependence of \mathbf{E}_m on N for now. The DFT of the input in this case is the vector $\mathbf{X} \in \mathbb{C}^N$ with a single nonzero component, namely $X_m = N$ and $X_k = 0$ if $k \neq m$. This follows immediately from the definition of the DFT, equation (2.6) in Chapter 3. According to Theorem 4.3.1, $W_k = H_k X_k$, so the output \mathbf{w} also has a DFT \mathbf{W} with a single nonzero component $W_m = NH_m$ while $W_k = 0$ for $k \neq m$. If we apply the inverse DFT to \mathbf{W} , we find that $\mathbf{w} = H_m \mathbf{E}_m$, which follows easily from the definition of the inverse DFT (equation (2.7)). Thus

$$\mathbf{E}_m * \mathbf{h} = H_m \mathbf{E}_m. \quad (4.5)$$

In summary, the effect of filtering a basic waveform \mathbf{E}_m is to multiply the waveform by the (probably complex) scalar H_m .

The equation $\mathbf{E}_m * \mathbf{h} = H_m \mathbf{E}_m$ can be cast in matrix form as $\mathbf{M}_h \mathbf{E}_m = H_m \mathbf{E}_m$, where \mathbf{M}_h is the circulant matrix for \mathbf{h} and \mathbf{E}_m is written as a column vector. Equation (4.5) immediately yields the following theorem.

Theorem 4.3.2 *Let $\mathbf{h} \in \mathbb{C}^N$ have DFT \mathbf{H} . The eigenvectors of the circulant matrix \mathbf{M}_h are the basic waveform vectors $\mathbf{E}_{N,m}$, with corresponding eigenvalues H_m .*

Remark 4.2 The basic waveforms are complex exponential, with real and imaginary parts that are sines and cosines. It's worth looking at how filtering affects the amplitude and phase of $\mathbf{E}_{N,m}$. Recall that the k th component of $\mathbf{E}_{N,m}$ is $e^{2\pi i km/N}$. In particular, we may consider the vector $\mathbf{E}_{N,m}$ as a sampled version of the function

$$\phi_m(t) = e^{2\pi i mt}$$

TABLE 4.1 Effect of Filtering on Basic Waveforms

Input		Output
Basic waveform \mathbf{E}_m	Filter $\mathbf{E}_m \rightarrow \mathbf{h} * \mathbf{E}_m$	Magnified, shifted waveform $\mathbf{h} * \mathbf{E}_m = H_m \mathbf{E}_m$

at sample points $t = k/N$, $0 \leq k \leq N - 1$. Note that $|\phi_m(t)| = 1$ for all t and $\phi_m(t)$ oscillates with period $1/m$. The filtered version of the waveform is $H_m \mathbf{E}_{N,m}$, and this may be considered as the sampled version of the function

$$\begin{aligned} H_m \phi_m(t) &= H_m e^{2\pi i m t} \\ &= |H_m| e^{i \arg(H_m)} e^{2\pi i m t} \\ &= |H_m| e^{2\pi i m [t + (\arg(H_m)/2\pi m)]} \end{aligned} \quad (4.6)$$

at the same times $t = k/N$, where we've used the fact that $A = |A|e^{i \arg(A)}$ for any complex A . The function $H_m \phi_m(t)$ satisfies $|H_m \phi_m(t)| = |H_m|$ for all t and is advanced in time (graphically, shifted to the left) by an amount $\arg(H_m)/2\pi m$, which is a fraction $\arg(H_m)/2\pi$ of the period of $\phi_m(t)$.

In summary, when filtered, the basic waveform $\mathbf{E}_{N,m}$ is amplified in magnitude by $|H_m|$ and phase-shifted in time by a fraction $\arg(H_m)/2\pi$ of one oscillation period, as summarized in Table 4.1.

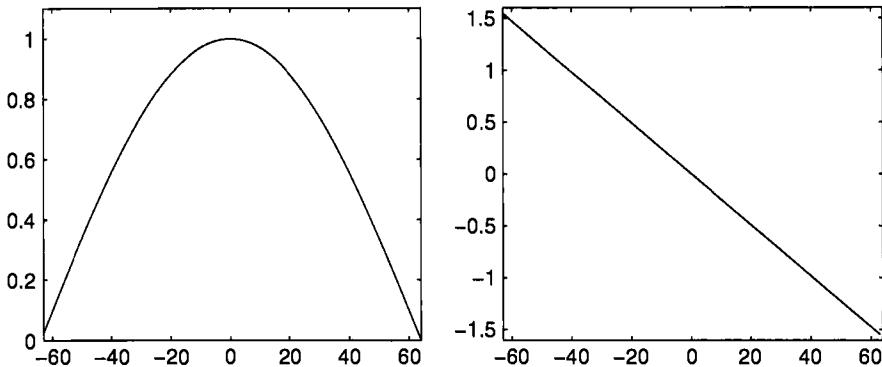
A general input \mathbf{x} is a superposition of such basic waveforms. When passed through the filter, this amplification and phase shifting occurs frequency by frequency.

One can similarly analyze the effect of filtering on the basic trigonometric waveforms obtained by sampling $\cos(2\pi m t)$ and $\sin(2\pi m t)$; see Exercise 4.20.

■ EXAMPLE 4.3

To illustrate the ideas above, let's revisit the two-point moving average filter from Section 4.2.1. We can filter any vector $\mathbf{x} \in \mathbb{R}^{128}$ with a two-point moving average as $\mathbf{w} = \mathbf{x} * \ell$, where $\ell \in \mathbb{R}^{128}$ with $\ell_0 = \frac{1}{2}$, $\ell_1 = \frac{1}{2}$, and all other $\ell_k = 0$. In Figure 4.3 we plot the magnitude and complex argument of $\mathbf{L} = DFT(\ell)$ on the range $-63 \leq k \leq 64$, so lower frequencies are near the center, high frequencies at the edges. The magnitude is on the left, argument on the right.

From the plot on the left we can see that frequencies near zero in any input vector \mathbf{x} have their magnitude largely unchanged by the filtering operation, for $L_k \approx 1$ in this region; hence $W_k = L_k X_k \approx X_k$. On the other hand, frequencies near the edges $k = \pm 64$ (corresponding to the Nyquist frequency) have their magnitude attenuated to nearly zero, for in this region $L_k \approx 0$ and so $W_k = L_k X_k \approx 0$. Thus ℓ acts as a crude low-pass filter, with no sharp delineation between frequencies that are allowed to pass and those that are blocked.

FIGURE 4.3 Amplitude and argument plot of \mathbf{L} .

The plot of the complex argument of the components of \mathbf{L} on the right shows that input waveforms with frequency near zero have their phase shifted very little, for here $\arg(L_k) \approx 0$. On the other hand, frequencies near $k = 64$ will have their phase shifted by a fraction of about $-1.6/2\pi$ (really $(-\pi/2)/2\pi = -\frac{1}{4}$) period (graphically, to the right), while those near $k = -64$ will be shifted about $\frac{1}{4}$ period to the left.

Let's look at how the basic waveforms that comprise the specific input \mathbf{x} from the example in Section 4.2.1 are altered by the filter. Table 4.2 shows the value of L_k , as well as $|L_k|$ and $\arg(L_k)$ for $k = -5, 5, -12, 12, -47$, and 47 . Of course, since ℓ has real components, Proposition 2.5.1 implies that $L_{-k} = \overline{L}_k$, so $|L_k| = |L_{-k}|$ and $\arg(-L_k) = -\arg(L_k)$, as is evident in Table 4.2. The data show that the 5 Hertz components pass at over 99% of original magnitude, the 12 Hertz pass at about 95.7%, and the 47 Hertz are diminished to about 40% of the original magnitude. Compare the left and right panels in Figure 4.2.

4.3.3 Filter Design

The design of digital filters is a vast subject. The primary goal is to design a filter with a given *frequency response*, that is, with given DFT coefficients H_k , often with additional conditions on the filter. What little we need to know about filter design

TABLE 4.2 Moving Average Effect on Basic Waveforms

k	L_k	$ L_k $	$\arg(L_k)$
-5	$0.9850 + 0.1215i$	0.9925	0.1227
5	$0.9850 - 0.1215i$	0.9925	-0.1227
-12	$0.9157 + 0.2778i$	0.9569	0.2945
12	$0.9157 - 0.2778i$	0.9569	-0.2945
-47	$0.1642 + 0.3705i$	0.4052	1.1536
47	$0.1642 - 0.3705i$	0.4052	-1.1536

we'll develop later, though we already have the mathematical means to design a filter with any desired frequency response if efficiency is not a big concern.

Specifically, suppose that we want a filter \mathbf{h} so that if $\mathbf{w} = \mathbf{h} * \mathbf{x}$, then $W_k = H_k X_k$, where we choose each H_k (possibly complex). By choosing the H_k suitably, we can make the filter do whatever we want to each frequency in the input. By Theorem 4.3.1, we can take the filter vector \mathbf{h} as the inverse DFT of the vector \mathbf{H} . The primary drawback to this approach is that all filter coefficients h_k will almost certainly be nonzero. This isn't a big deal if we're working with vectors of length 10, but it is if the vectors are of length 10^6 . The convolution $\mathbf{x} * \mathbf{h}$ would be pretty slow, of order N^2 for vectors of length N ; see Exercise 4.23. Indeed it would probably be faster to perform an FFT on \mathbf{x} , multiply each X_k in \mathbf{X} by H_k , and then perform an inverse FFT. If we really want to filter strictly in the time domain, then it would be helpful if we could come close to the desired frequency response with a filter that has few nonzero coefficients.

Definition 4.3.1 *The nonzero coefficients h_k in a filter \mathbf{h} are called the “taps” for the filter.*

There are a variety of strategies for designing filters with a given frequency response and few taps. One simple (not always ideal) approach is to specify the filter frequency response \mathbf{H} , take \mathbf{h} as the inverse DFT of \mathbf{H} , and then zero out coefficients in \mathbf{h} that lie below some threshold.

■ EXAMPLE 4.4

We will again work with input signals $\mathbf{x} \in \mathbb{C}^{128}$, but this time let us design a high-pass filter that will pass unchanged all basic waveforms with frequency in the range 30 to 64 and block all waveforms with frequency 29 and below. If we index from 0 to 127, then these lower frequencies correspond to waveforms $\mathbf{E}_{128,k}$ with $0 \leq k \leq 29$ or $99 \leq k \leq 127$ (recall that $\mathbf{E}_{128,N-k} = \mathbf{E}_{128,-k}$, so the range $99 \leq k \leq 127$ corresponds frequencies 1 to 29).

The DFT of the desired filter is the vector $\mathbf{H} \in \mathbb{C}^{128}$ with components $H_k = 0$ for $0 \leq k \leq 29$ and $99 \leq k \leq 127$, and $H_k = 1$ otherwise. The inverse DFT of \mathbf{H} yields a filter vector \mathbf{h} , which is real-valued due to the $H_{N-k} = H_k$ symmetry, with components h_k plotted for $0 \leq k \leq 127$ in Figure 4.4.

All the h_k turn out to be nonzero. We can eliminate some filter taps by thresholding. Define the vector \mathbf{h}_ϵ for $\epsilon > 0$ by zeroing out all components of \mathbf{h} that have magnitude less than ϵ . We'll use $\epsilon = 0.01, 0.05$, and 0.1 , which yields filters with 43, 7, and 3 taps, respectively. Increasing ϵ decreases the number of taps, but the resulting DFT of \mathbf{h}_ϵ deviates more and more from the ideal high-pass frequency response. In Figure 4.5 we show the magnitude of the DFT coefficients for each filter \mathbf{h}_ϵ , in each case superimposed over the ideal high-pass filter's DFT. Whether this degradation in the frequency response is acceptable depends on the situation. Phase distortion (not shown here) may also be a concern.

We'll return to filter design in the next chapter, in the context of “filter banks.”

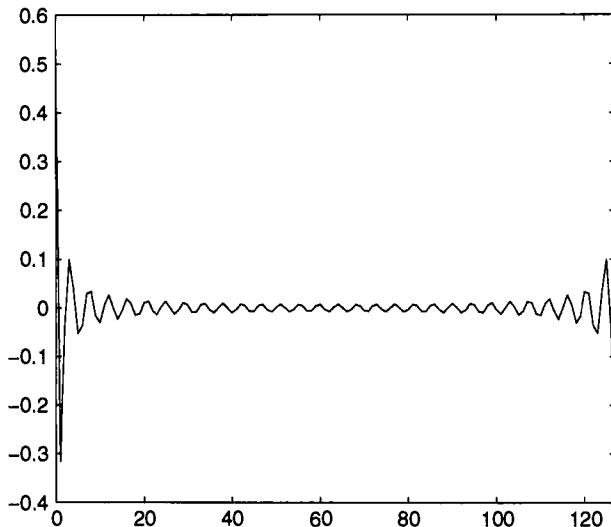


FIGURE 4.4 High-pass filter coefficients.

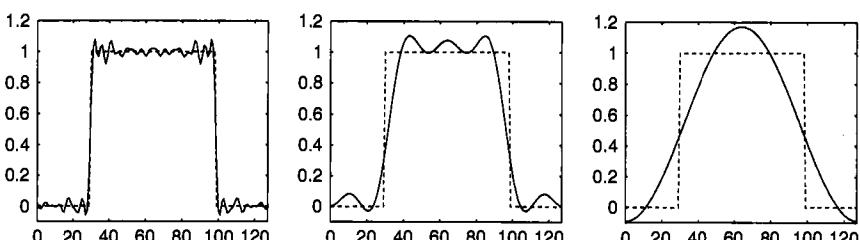
4.4 2D CONVOLUTION—FILTERING IMAGES

Let $\mathbf{A} \in M_{m,n}(\mathbb{C})$ with components $a_{r,s}$ indexed on the range $0 \leq r \leq m-1, 0 \leq s \leq n-1$. As in the one-dimensional case it will be helpful to think of any such array as having been extended periodically via $a_{r,s} = a_{r \bmod m, s \bmod n}$ so that the array “tiles” the plane. It’s also worth noting that in this case, the same as noted in Remark 4.1 on page 141, we have

$$\sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r+M, s+N} = \sum_{r=M}^{M+m-1} \sum_{s=N}^{N+n-1} a_{r,s} = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s}$$

for any M, N . Each sum involves exactly the same mn summands.

Convolution in $M_{m,n}(\mathbb{C})$ is quite analogous to the one-dimensional case.

FIGURE 4.5 DFT's for $h_\epsilon, \epsilon = 0.01, 0.05, 0.1$.

Definition 4.4.1 Let \mathbf{A} and \mathbf{B} be elements of $M_{m,n}(\mathbb{C})$. The circular convolution $\mathbf{A} * \mathbf{B}$ is defined as the element $\mathbf{C} \in M_{m,n}(\mathbb{C})$ with components

$$c_{p,q} = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} a_{r,s} b_{p-r, q-s}. \quad (4.7)$$

Like its one-dimensional counterpart, two-dimensional circular convolution has various useful algebraic properties: it is linear, commutative, associative, and if $\mathbf{C} = \mathbf{A} * \mathbf{B}$, then $c_{r,s} = c_{r \bmod m, s \bmod n}$. There is no convenient matrix formulation for two-dimensional convolution, however, except in certain special cases; see Exercise 4.24.

We also have a two-dimensional version of the convolution theorem:

Theorem 4.4.1 (The Convolution Theorem) Let \mathbf{A} and \mathbf{B} be elements of $M_{m,n}(\mathbb{C})$ with (two-dimensional) DFT's $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$, respectively (components $\hat{a}_{k,l}, \hat{b}_{k,l}$). Let $\mathbf{C} = \mathbf{A} * \mathbf{B}$ have two-dimensional DFT $\widehat{\mathbf{C}}$, components $\hat{c}_{k,l}$. Then

$$\hat{c}_{k,l} = \hat{a}_{k,l} \hat{b}_{k,l}$$

for $0 \leq k \leq m - 1, 0 \leq l \leq n - 1$.

The proof of this is just a “double-sum” version of the proof of Theorem 4.3.1 and is left as Exercise 4.17.

4.4.1 Two-dimensional Filtering and Frequency Response

We can filter two-dimensional images in a manner similar to one-dimensional signals. Specifically, given an image array $\mathbf{A} \in M_{m,n}(\mathbb{C})$ and element $\mathbf{D} \in M_{m,n}(\mathbb{C})$, we filter by convolving \mathbf{A} with \mathbf{D} . If $\mathbf{B} = \mathbf{A} * \mathbf{D}$, then, by Theorem 4.4.1, the DFT components satisfy

$$\hat{b}_{k,l} = \hat{a}_{k,l} \hat{d}_{k,l}.$$

In the context of image filtering, the array \mathbf{D} is often called a *mask*.

It's easy to see the effect of filtering a two-dimensional basic waveform with a mask \mathbf{D} . Let $\mathcal{E}_{m,n,k,l}$ be a two-dimensional basic waveform (recall the definition of these waveforms from equation (1.25), Section 1.7.2); note that k and l here indicate which basic waveform we are considering and are fixed. Let $\mathcal{G} \in M_{m,n}(\mathbb{C})$ denote the two-dimensional DFT of $\mathcal{E}_{m,n,k,l}$. The definition of the two-dimensional DFT shows that the $m \times n$ array \mathcal{G} has a single nonzero component:

$$\mathcal{G}_{r,s} = \begin{cases} mn, & \text{if } r = k \text{ and } s = l, \\ 0, & \text{else.} \end{cases}$$

By Theorem 4.4.1, if we filter $\mathcal{E}_{m,n,k,l}$ as $\mathbf{B} = \mathcal{E}_{m,n,k,l} * \mathbf{D}$, then $\hat{b}_{r,s} = \hat{d}_{r,s} \mathcal{G}_{r,s}$. The DFT $\widehat{\mathbf{B}}$ will have the same form as \mathcal{G} , with a single nonzero component $\hat{b}_{r,s} = mnd_{r,s}$ when $r = k$ and $s = l$, while $\hat{b}_{r,s} = 0$ for all other indexes r, s . An inverse DFT applied to $\widehat{\mathbf{B}}$ then shows that

$$\mathcal{E}_{m,n,k,l} * \mathbf{D} = \hat{d}_{k,l} \mathcal{E}_{m,n,k,l}, \quad (4.8)$$

the two-dimensional counterpart to equation (4.5). As in the one-dimensional case, when a basic 2D waveform is convolved with a mask, the effect is to multiply the waveform by a (complex) scalar.

Although two-dimensional convolution has no convenient matrix formulation, equation (4.8) shows that each of the mn basic waveforms $\mathcal{E}_{m,n,k,l}$ is a kind of “eigenvector” for the mapping $\phi : M_{m,n}(\mathbb{C}) \rightarrow M_{m,n}(\mathbb{C})$ defined by $\phi(\mathbf{A}) = \mathbf{A} * \mathbf{D}$. The corresponding eigenvalues are the constants $\hat{d}_{k,l}$.

4.4.2 Applications of 2D Convolution and Filtering

Noise Removal and Blurring Define a mask $\mathbf{D} \in M_{m,n}(\mathbb{C})$ with components $d_{r,s}$ given by

$$d_{r,s} = \begin{cases} d_{r,s} = \frac{1}{9}, & r, s = -1, 0, 1 \\ d_{r,s} = 0, & \text{otherwise} \end{cases}$$

Recall we extend \mathbf{D} periodically so, for example, $d_{m-1,n-1} = d_{-1,-1}$. The effect of convolving an image array \mathbf{A} with \mathbf{D} is to average each pixel in \mathbf{A} with its eight nearest neighbors, analogous to the two-point moving average used to denoise one-dimensional signals. Specifically, the (k, l) entry of $\mathbf{A} * \mathbf{D}$ is

$$\begin{aligned} (\mathbf{A} * \mathbf{D})_{k,l} = & \frac{1}{9}(a_{k-1,l-1} + a_{k-1,l} + a_{k-1,l+1} \\ & + a_{k,l-1} + a_{k,l} + a_{k,l+1} \\ & + a_{k+1,l-1} + a_{k+1,l} + a_{k+1,l+1}). \end{aligned}$$

The magnitude of the DFT of \mathbf{D} is plotted in Figure 4.6 in the case where $m = n = 100$, for the frequency range $-m/2 < k \leq m/2, -n/2 < l \leq n/2$. Low frequencies are near the center, higher frequencies near the edges. In this case the dc component is passed with magnitude unchanged. In general, lower frequencies pass with slightly diminished magnitude, while higher frequencies are strongly attenuated.

This mask can be used in a simple scheme for denoising images. The rationale is that noise (especially random noise) contains a lot of high-frequency energy, so the filter should substantially remove it while allowing more slowly varying image features to pass. The images in Figure 4.7 provide an example. On the left is a grayscale image without noise. On the right is the same image with random noise added. In Figure 4.8 we show on the left the result of applying the 9-point averaging

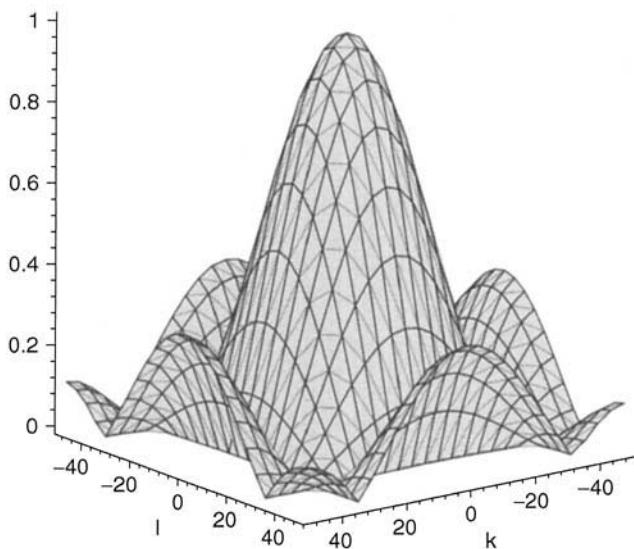


FIGURE 4.6 Magnitude of 9-point averaging mask DFT.

mask to the noisy image. It's clear the noise is somewhat reduced. On the right we show the result of applying the mask five times. The noise is much reduced, but at the expense of substantial blurring of the image. This illustrates an essential difficulty in the use of averaging for noise reduction. Sharp edges in a photograph require high

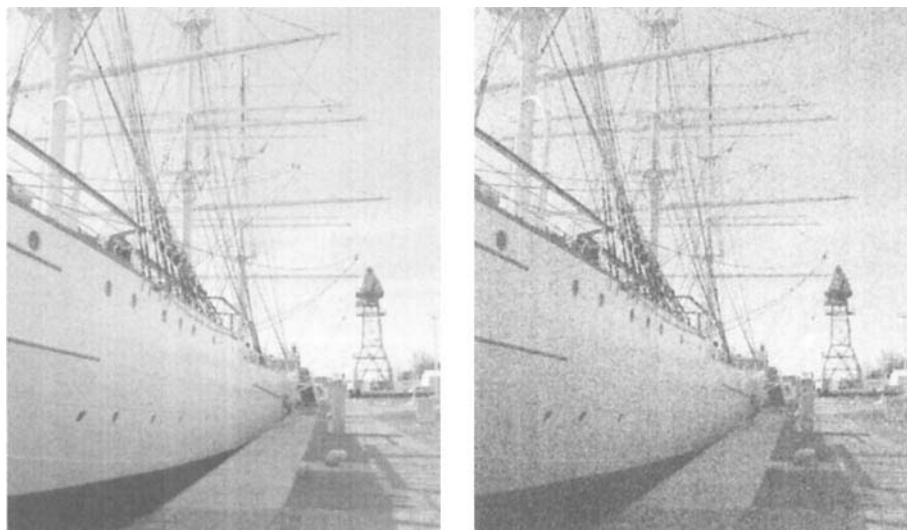


FIGURE 4.7 Original and noisy image.

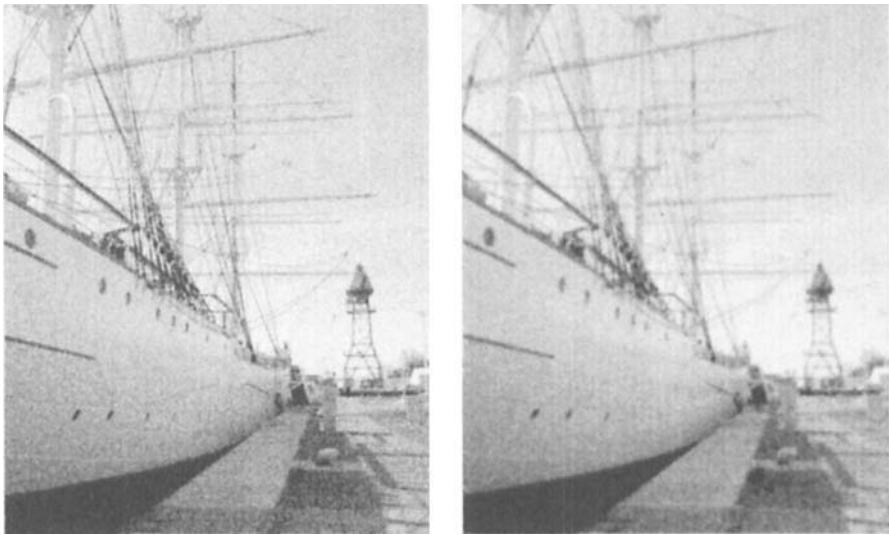


FIGURE 4.8 Noisy image filtered once and five times.

frequencies to synthesize, so the same averaging or low-pass filter that “smears out” the noise also smears the edges.

Edge Detection Let’s consider the problem of detecting sharp edges in a grayscale image, an important task in image processing. If we think of the analog case where the image intensity is given by a function $f(x, y)$, such an edge corresponds to an abrupt change in the value of f in some direction. In this case either $\partial f / \partial x$ and/or $\partial f / \partial y$ will likely be large in magnitude; more generally, $\|\nabla f\|$ should be large, where ∇f denotes the gradient of f and $\|\nabla f\|$ the magnitude of this vector.

To detect such edges in the discrete case, let $\mathbf{D} \in M_{m,n}(\mathbb{C})$ be the mask defined by $d_{0,0} = 1$, $d_{1,0} = -1$, and all other $d_{r,s} = 0$. For a sampled image $\mathbf{A} \in M_{m,n}(\mathbb{R})$ the row r , column s entry of $(\mathbf{A} * \mathbf{D})$ is given by

$$(\mathbf{A} * \mathbf{D})_{r,s} = a_{r,s} - a_{r-1,s}. \quad (4.9)$$

If we think of \mathbf{A} as a sampled analog image $f(x, y)$ via $a_{r,s} = f(x_s, y_r)$ on a rectangular grid $x_s = s(\Delta x)$, $y_r = r(\Delta y)$, then

$$a_{r,s} - a_{r-1,s} = f(x_s, y_r) - f(x_s, y_{r-1}) \approx (\Delta y) \frac{\partial f}{\partial y}(x_s, y_r),$$

where $\Delta y = y_r - y_{r-1}$. From the equation above we conclude that $(\mathbf{A} * \mathbf{D})_{r,s}$ is approximately proportional to $\partial f(x_s, y_r)/\partial y$, and so $|(\mathbf{A} * \mathbf{D})_{r,s}|$ should be large if the image has an abrupt change in the y direction at this location. This corresponds to a horizontal edge in the image.

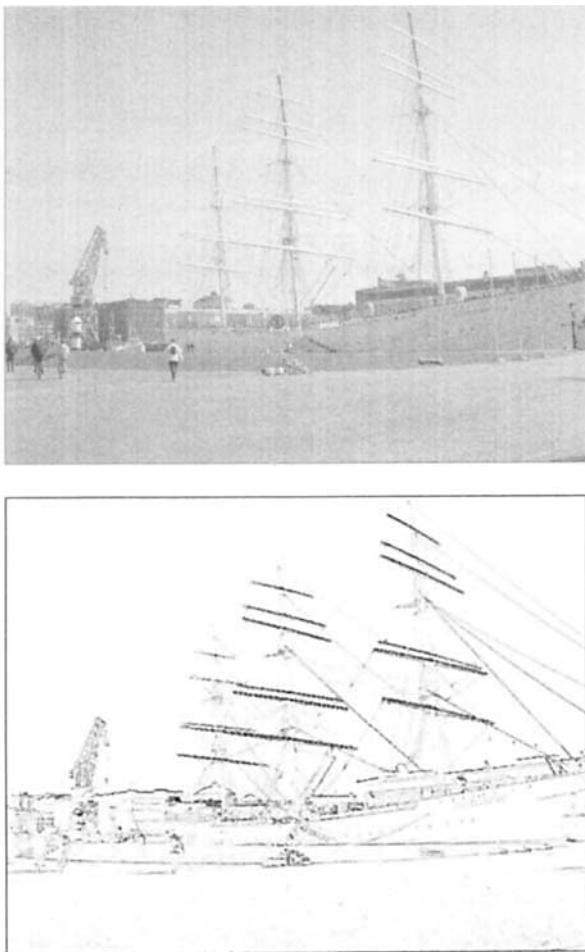


FIGURE 4.9 Original and horizontal edge-filtered image.

The mask \mathbf{V} with components $v_{0,0} = 1$, $v_{0,1} = -1$, and all other $v_{r,s} = 0$ does the same for vertical edges (abrupt changes in $\partial f / \partial x$).

At the top in Figure 4.9 is shown a grayscale image \mathbf{A} and at the bottom the filtered version $|\mathbf{A} * \mathbf{D}|$, with the absolute value taken component by component, to highlight horizontal edges. The image $|\mathbf{A} * \mathbf{D}|$ is displayed in an 8-bit “reverse” grayscale scheme, with 0 as white and 255 as black, which makes it a bit easier to see the edges (since most of the image does not consist of edges). In Figure 4.10 we show the similarly scaled image $|\mathbf{A} * \mathbf{V}|$ (top) as well as the quantity

$$\sqrt{|\mathbf{A} * \mathbf{V}|^2 + |\mathbf{A} * \mathbf{D}|^2}$$

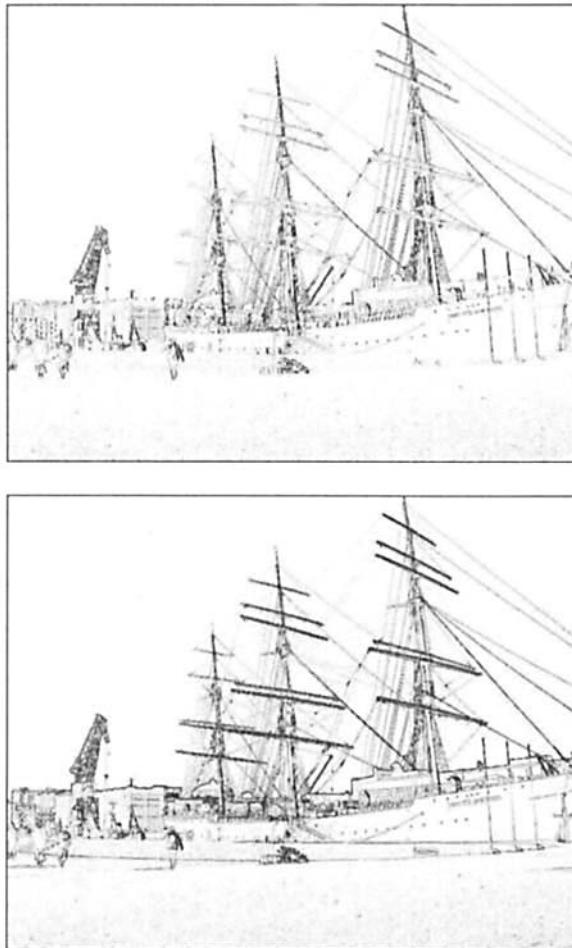


FIGURE 4.10 Vertical edge-filtered (*top*) and composite edge image (*bottom*).

with squaring and square roots performed component by component. The latter quantity effectively amalgamates the vertical and horizontal edge images into a single image that picks out all of the edges, and is a discrete approximation to $\|\nabla f\|$, at least if $\Delta x = \Delta y$. See also Exercise 4.25.

4.5 INFINITE AND BI-INFINITE SIGNAL MODELS

The one-dimensional discrete signals considered have so far consisted of vectors in \mathbb{R}^N or \mathbb{C}^N , extended by periodicity when convenient. We may think of these vectors as having been obtained by sampling a function on some finite time interval $[0, T]$.

In the same spirit our two-dimensional discrete images are $m \times n$ matrices obtained by sampling a function of two variables on a rectangle.

But it will be useful, especially in later chapters, to consider signals that are infinite or bi-infinite in extent. For example, in Example 1.3 in Chapter 1 we considered an analog function $f(t)$ sampled at times $t_k = a + k(\Delta T)$ for $k = 0, 1, 2, \dots$, to yield a sampled signal $\mathbf{x} = (x_0, x_1, x_2, \dots)$. We might also consider sampling a function from the indefinite past to indefinite future, at $t_k = a + k(\Delta T)$ for all $k \in \mathbb{Z}$, to yield a sample vector $\mathbf{x} = (\dots, x_{-1}, x_0, x_1, x_2, \dots)$.

In this section we discuss how the mathematics of transforms, convolution, and filtering extend to these settings. It very closely parallels the finite case.

4.5.1 $L^2(\mathbb{N})$ and $L^2(\mathbb{Z})$

The Inner Product Space $L^2(\mathbb{N})$ The vector space $L^2(\mathbb{N})$ consists of elements $\mathbf{x} = (x_0, x_1, x_2, \dots)$ of complex numbers x_k such that

$$\sum_{k=0}^{\infty} |x_k|^2 < \infty$$

and was introduced in Example 1.5. Let us now verify that

$$(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{\infty} x_k \overline{y_k} \quad (4.10)$$

defines an inner product on this vector space; the algebraic properties 1 through 3 of Definition 1.8.1 are straightforward verifications. The harder part is the verification that the sum on the right in equation (4.10) actually converges for any \mathbf{x} and \mathbf{y} in $L^2(\mathbb{N})$.

The sum in equation (4.10) will converge if the real and imaginary parts on the right, $\sum_k \operatorname{Re}(x_k \overline{y_k})$ and $\sum_k \operatorname{Im}(x_k \overline{y_k})$, each converge. To show this we use the fact that for any two complex numbers z and w we have the inequalities $|w||z| \leq \frac{1}{2}(|w|^2 + |z|^2)$, as well as $|\operatorname{Re}(zw)| \leq |z||w|$ and $|\operatorname{Im}(zw)| \leq |z||w|$; see Exercise 4.1. To show the sum for the real part in equation (4.10) converges note that for any $n > 0$,

$$\begin{aligned} \sum_{k=0}^n |\operatorname{Re}(x_k \overline{y_k})| &\leq \sum_{k=0}^n |x_k| |\overline{y_k}| \quad (\text{use } |\operatorname{Re}(zw)| \leq |z||w|), \\ &\leq \frac{1}{2} \sum_{k=0}^n (|x_k|^2 + |y_k|^2) \quad \left(\text{use } |z||w| \leq \frac{(|w|^2 + |z|^2)}{2} \right). \end{aligned}$$

If we take the limit of both sides above as $n \rightarrow \infty$, we find that

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n |\operatorname{Re}(x_k \overline{y_k})| \leq \frac{1}{2} \lim_{n \rightarrow \infty} \sum_{k=0}^n (|x_k|^2 + |y_k|^2) < \infty, \quad (4.11)$$

since $\sum_{k=0}^{\infty} |x_k|^2$ and $\sum_{k=0}^{\infty} |y_k|^2$ both converge by assumption. From equation (4.11) we conclude that the real series $\sum_{k=0}^{\infty} \operatorname{Re}(x_k \bar{y}_k)$ converges absolutely, and hence converges. A similar argument works for $\sum_{k=0}^{\infty} \operatorname{Im}(x_k \bar{y}_k)$. The series defining the inner product (\mathbf{x}, \mathbf{y}) in equation (4.10) converges to some complex number.

The space $L^2(\mathbb{N})$ is thus an inner product space with inner product defined by (4.10). The norm on $L^2(\mathbb{N})$ is defined by

$$\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})} = \left(\sum_{k=0}^{\infty} |x_k|^2 \right)^{1/2}.$$

The Inner Product Space $L^2(\mathbb{Z})$ The vector space $L^2(\mathbb{Z})$ consists of elements $\mathbf{x} = (\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots)$, where each $x_k \in \mathbb{C}$ and

$$\sum_{k=-\infty}^{\infty} |x_k|^2 < \infty.$$

This is an inner product space with the inner product

$$(\mathbf{x}, \mathbf{y}) = \sum_{k=-\infty}^{\infty} x_k \bar{y}_k.$$

The verification is almost identical to that for $L^2(\mathbb{N})$.

We can impose the additional requirement that the components x_k of vectors in $L^2(\mathbb{N})$ or $L^2(\mathbb{Z})$ be real numbers. The resulting spaces are still inner product spaces with the inner products as above, but of course no conjugation is needed.

In what follows we'll work for the most part in the space $L^2(\mathbb{Z})$, bi-infinite sequences. Results for $L^2(\mathbb{N})$ can be obtained by considering elements $\mathbf{x} \in L^2(\mathbb{Z})$ with $x_k = 0$ for $k < 0$.

4.5.2 Fourier Analysis in $L^2(\mathbb{Z})$ and $L^2(\mathbb{N})$

The DFT/IDFT pair allows us to decompose a vector in \mathbb{C}^N into basic complex exponential waveforms. As we've seen, this facilitates a lot of useful analysis. We'll now look at the analogue of the DFT in $L^2(\mathbb{Z})$.

The Discrete Time Fourier Transform in $L^2(\mathbb{Z})$ What follows in this paragraph is intended as “inspiration” for a reasonable definition for the Fourier transform in $L^2(\mathbb{Z})$, to be followed by the precise definition. In the finite-dimensional case we were able to write each component of a vector $\mathbf{x} \in \mathbb{C}^N$ as a sum

$$x_k = \frac{1}{N} \sum_{m=0}^{N-1} X_m e^{2\pi i k m / N}$$

for $0 \leq k \leq N - 1$ and where \mathbf{X} is the DFT of \mathbf{x} . The sum above is just the inverse DFT. Since $e^{2\pi i km/N}$ is periodic in both k and m with period N , we may “shift” the indexes k and m and write

$$x_k = \frac{1}{N} \sum_{m=-N/2+1}^{N/2} X_m e^{2\pi i km/N} \quad (4.12)$$

for $-N/2 + 1 \leq k \leq N/2$, where we’ve effectively used the periodic extensions $x_k = x_{k \bmod N}$ and $X_m = X_{m \bmod N}$. Now the sum in equation (4.12) is precisely a right Riemann sum appropriate to integrating a function $X(f)e^{2\pi i fk}$ on the interval $-\frac{1}{2} \leq f \leq \frac{1}{2}$ where $X_m = X(m/N)$; we use “ f ” for the integrand variable since X should have frequency as its independent variable. Taking the limit as $N \rightarrow \infty$ in (4.12) strongly suggests that we should try to write

$$x_k = \int_{-1/2}^{1/2} X(f) e^{2\pi i kf} df \quad (4.13)$$

for an appropriate function $X(f)$. Presumably the function $X(f)$ would be computed linearly from the samples x_k with a summation in k .

We’ve already done this computation in Section 1.10.3! In particular, equations (1.47) and (1.48), which we reproduce here for convenience in the case $T = \frac{1}{2}$, state that for a continuous function h defined on the interval $[-\frac{1}{2}, \frac{1}{2}]$, we have

$$h(f) = \sum_{k=-\infty}^{\infty} x_k e^{2\pi i kf}, \quad (4.14)$$

where

$$x_k = \int_{-1/2}^{1/2} h(f) e^{-2\pi i kf} df \quad (4.15)$$

and we’ve used f for the independent variable instead of t . Compare equation (4.13) and (4.15)—they’re quite similar. If we define $X(f) = h(-f)$ (so $h(f) = X(-f)$), then equation (4.14) defines the transform we want, and equation (4.15) (after a compensating change of variable $f \rightarrow -f$) defines a corresponding inverse transform.

Definition 4.5.1 For an element $\mathbf{x} \in L^2(\mathbb{Z})$ the “discrete time Fourier transform” (DTFT) is the function $X(f)$ on the interval $-\frac{1}{2} \leq f \leq \frac{1}{2}$ defined by

$$X(f) = \sum_{k=-\infty}^{\infty} x_k e^{-2\pi i kf}. \quad (4.16)$$

The "inverse discrete time Fourier transform" (IDTFT) is defined by

$$x_k = \int_{-1/2}^{1/2} X(f) e^{2\pi i k f} df. \quad (4.17)$$

Equation (4.17) shows how to synthesize the sequence $\mathbf{x} \in L^2(\mathbb{Z})$ as an integral superposition of basic waveforms $e^{2\pi i k f}$. Equation (4.16) indicates how much $X(f)$ of the waveform $e^{2\pi i k f}$ is required. Here f plays the role of a frequency variable while the index k is essentially a time variable. The time domain in this case is \mathbb{Z} and the frequency domain is the real interval $[-\frac{1}{2}, \frac{1}{2}]$. In essence the DTFT/IDTFT pair is just Fourier series turned around, with the roles of the time and frequency domains reversed, and the forward and inverse transforms reversed.

Aliasing and the Nyquist Frequency in $L^2(\mathbb{Z})$ We might think of the vector \mathbf{x} in Definition 4.5.1 as having been obtained by sampling an analog signal $x(t)$ as $x_k = x(k)$, a sampling rate of one Hertz. As such the Nyquist frequency is $\frac{1}{2}$ Hertz, which is why the domain of the DTFT $X(f)$ is $-\frac{1}{2} \leq f \leq \frac{1}{2}$. However, just as in the finite-dimensional case, frequencies outside this range will be aliased when sampled.

The forward/inverse transforms of Definition 4.5.1 are valid for any $\mathbf{x} \in L^2(\mathbb{Z})$, of course, even if \mathbf{x} was obtained by sampling at a different rate, say $x_k = x(kT)$, (sampling interval T , sampling frequency $F = 1/T$ Hertz, Nyquist rate $1/2T$). However, in this case it is sometimes helpful to rescale the transforms in (4.16) and (4.17) to reflect the different sampling rate. Specifically, if we sample at $F = 1/T$ Hertz then we would expect the DTFT to be defined on the range $-1/2T \leq f \leq 1/2T$. The formulas (4.16) and (4.17) can be adapted to reflect this via a simple change of variable. Let $\tilde{f} = f/T$ (so $f = T\tilde{f}$ and $df = T d\tilde{f}$) in the integral on the right in (4.17) to obtain

$$x_k = \int_{-1/2T}^{1/2T} X(\tilde{f}T) e^{2\pi i k \tilde{f}T} T d\tilde{f}, \quad (4.18)$$

where the function X is still defined by (4.16). From (4.16) we have

$$X(\tilde{f}T) = \sum_{k=-\infty}^{\infty} x_k e^{-2\pi i k \tilde{f}T}. \quad (4.19)$$

Define a rescaled DTFT $\tilde{X}(\tilde{f}) := X(\tilde{f}T)$. Then from (4.18) we have

$$x_k = \int_{-1/2T}^{1/2T} \tilde{X}(\tilde{f}) e^{2\pi i k \tilde{f}T} T d\tilde{f}, \quad (4.20)$$

where from (4.19) $\tilde{X}(\tilde{f})$ is defined as

$$\tilde{X}(\tilde{f}) = \sum_{k=-\infty}^{\infty} x_k e^{-2\pi i k \tilde{f}T}. \quad (4.21)$$

Equations (4.20) and (4.21) comprise a rescaled version of the forward/inverse transforms (4.16) and (4.17). Equation (4.20) shows how to synthesize the signal $x(t)$ at $t = kT$ from the waveforms $e^{2\pi i \tilde{f} t}$ sampled at $t = kT$, using frequencies $\tilde{f} \in [-1/2T, 1/2T]$ (the Nyquist range).

In general, however, we won't worry about how \mathbf{x} was sampled, or whether it was even obtained via sampling. We will simply use formulas (4.16) and (4.17).

Remark 4.3 If the number of nonzero components in \mathbf{x} is finite, then the sum in (4.16) is finite and defines a continuous (indeed, infinitely differentiable) function $X(f)$. However, if \mathbf{x} is a general vector in $L^2(\mathbb{Z})$, then the convergence of the sum (4.16) is slightly more problematic, and precisely the same analytic issues are raised as those in Section 1.10.4 with regard to Fourier series. It can be shown nevertheless that for any $\mathbf{x} \in L^2(\mathbb{Z})$ the sum (4.16) converges to a function in the space $L^2(-\frac{1}{2}, \frac{1}{2})$ as defined in Example 1.8, that is, to a function that satisfies $\int_{-1/2}^{1/2} |X(f)|^2 df < \infty$. The function $X(f)$ does not need to be continuous in this case, and indeed not even Riemann integrable.

The Fourier Transform on $L^2(\mathbb{N})$ We can consider an element $\mathbf{x} \in L^2(\mathbb{N})$ as an element of $L^2(\mathbb{Z})$ by extending the vector \mathbf{x} as $x_k = 0$ for all $k < 0$. In this case we obtain a transform/inverse transform pair from equations (4.16) and (4.17), which become

$$X(f) = \sum_{k=0}^{\infty} x_k e^{-2\pi i kf}$$

and

$$x_k = \int_{-1/2}^{1/2} X(f) e^{2\pi i kf} df$$

for $k \geq 0$.

4.5.3 Convolution and Filtering in $L^2(\mathbb{Z})$ and $L^2(\mathbb{N})$

We now discuss convolution and filtering for infinite and bi-infinite sequences, a common conceptual operation in digital signal processing. There are two-dimensional analogs as well, which we discuss briefly at the end of this section.

The convolution of two elements \mathbf{x} and \mathbf{y} in $L^2(\mathbb{Z})$ is defined by

$$(\mathbf{x} * \mathbf{y})_m = \sum_{k=-\infty}^{\infty} x_k y_{m-k}, \quad (4.22)$$

where $m \in \mathbb{Z}$ and $(\mathbf{x} * \mathbf{y})_m$ denotes m th component of the convolution. As with circular convolution in \mathbb{C}^N , a diagram is helpful in understanding equation (4.22). When

$m = 0$, the sum may be computed as the “dot product” of the infinite row vectors

$$\begin{array}{ccccccc} \cdots & x_{-2} & x_{-1} & x_0 & x_1 & x_2 & \cdots \\ \cdots & y_2 & y_1 & y_0 & y_{-1} & y_{-2} & \cdots \end{array}$$

For $m \neq 0$ the quantity $(\mathbf{x} * \mathbf{y})_m$ is computed as the dot product of

$$\begin{array}{ccccccc} \cdots & x_{-2} & x_{-1} & x_0 & x_1 & x_2 & \cdots & x_m \cdots \\ \cdots & y_{m+2} & y_{m+1} & y_m & y_{m-1} & y_{m-2} & \cdots & y_0 \cdots \end{array}$$

in which the bottom row above has been shifted m steps to the right (a left shift if $m < 0$).

Convergence of the infinite sum in (4.22) is not obvious, but in fact the sum converges absolutely (and hence converges), since

$$\sum_{k=-\infty}^{\infty} |x_k||y_{m-k}| \leq \frac{1}{2} \sum_{k=-\infty}^{\infty} (|x_k|^2 + |y_{m-k}|^2) = \frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2) < \infty$$

and \mathbf{x} and \mathbf{y} are in $L^2(\mathbb{Z})$. Note that $\sum_{k=-\infty}^{\infty} |y_{m-k}|^2 = \sum_{k=-\infty}^{\infty} |y_k|^2$ for any m .

Remark 4.4 The argument just given demonstrates that if $\mathbf{w} = \mathbf{x} * \mathbf{y}$ then $|w_m| \leq \frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2)$ for all $m \in \mathbb{Z}$. We conclude that the components w_m all satisfy a common bound, so \mathbf{w} is in the vector space $L^\infty(\mathbb{Z})$ defined in Example 1.4 of Chapter 1. However, it need not be the case that \mathbf{w} itself lies in $L^2(\mathbb{Z})$, so convolution is not well-defined as an operation from $L^2(\mathbb{Z}) \times L^2(\mathbb{Z})$ to $L^2(\mathbb{Z})$; see Exercise 4.30. Under some circumstances, however, we can conclude that $\mathbf{w} \in L^2(\mathbb{Z})$, for example, if either \mathbf{x} or \mathbf{y} has only finitely many nonzero components; see Exercise 4.29.

To convolve vectors \mathbf{x} and \mathbf{y} in $L^2(\mathbb{N})$ just treat them as vectors in $L^2(\mathbb{Z})$ with all components equal to zero for negative indexes. In this case convolution can be written

$$(\mathbf{x} * \mathbf{y})_m = \sum_{k=0}^m x_k y_{m-k},$$

since $x_k = 0$ for $k < 0$ and $y_{m-k} = 0$ for $k > m$.

As with convolution in \mathbb{C}^N we have

Proposition 4.5.1 *Let \mathbf{x} , \mathbf{y} and \mathbf{w} all be elements in either $L^2(\mathbb{Z})$ or $L^2(\mathbb{N})$. Then*

- *Commutativity:* $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$.
- *Associativity:* $\mathbf{x} * (\mathbf{y} * \mathbf{w}) = (\mathbf{x} * \mathbf{y}) * \mathbf{w}$.
- *Linearity:* $\mathbf{x} * (a\mathbf{y} + b\mathbf{w}) = a\mathbf{x} * \mathbf{y} + b\mathbf{x} * \mathbf{w}$ for any scalars a and b .

The proof of each point is a straightforward manipulation of the relevant summations.

The Convolution Theorem Convolution in \mathbb{C}^N became point-by-point multiplication in the frequency domain. The same thing happens for $L^2(\mathbb{Z})$.

Theorem 4.5.1 *Let \mathbf{x} and \mathbf{y} be elements of $L^2(\mathbb{Z})$ with discrete time Fourier transforms $X(f)$ and $Y(f)$, respectively. If $\mathbf{w} = \mathbf{x} * \mathbf{y}$, then the discrete time Fourier transform $W(f)$ of \mathbf{w} is*

$$W(f) = X(f)Y(f).$$

The equation $W(f) = X(f)Y(f)$ is just the frequency-by-frequency product of the transforms, analogous to $W_k = X_k Y_k$ for the DFT.

Proof The proof of Theorem 4.5.1 in the most general context requires a bit of analysis. We'll content ourselves with a proof for the case of primary interest, in which \mathbf{x} and \mathbf{y} contain only finitely many nonzero components. We thus avoid some delicate issues concerning the convergence of the relevant sums and integrals.

Let us write the discrete time Fourier transforms of the vectors \mathbf{x} and \mathbf{y} in the form

$$\begin{aligned} X(f) &= \sum_{k=-\infty}^{\infty} x_k z^{-k}, \\ Y(f) &= \sum_{m=-\infty}^{\infty} y_m z^{-m}, \end{aligned}$$

for $-\frac{1}{2} \leq f \leq \frac{1}{2}$, where $z = e^{2\pi i f}$ (note z depends on the frequency f , but we won't explicitly indicate this). Since \mathbf{x} and \mathbf{y} are assumed to have only finitely many nonzero components, the sums above involve only finitely many terms and the convergence of any sum is not in question. The product $X(f)Y(f)$ is given by

$$X(f)Y(f) = \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} x_k y_m z^{-k-m},$$

which also involves only finitely many nonzero terms. We collect together like powers of z by setting $s = k + m$ (so $m = s - k$) and performing a change of summation index in the inner sum, to obtain

$$X(f)Y(f) = \sum_{k=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} x_k y_{s-k} z^{-s} = \sum_{s=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x_k y_{s-k} \right) z^{-s}, \quad (4.23)$$

where we also interchange the summation order. Since $\mathbf{w} = \mathbf{x} * \mathbf{y}$, the parenthesized sum on the right in equation (4.23) is exactly w_s (recall equation (4.22)). Equation (4.23) yields

$$X(f)Y(f) = \sum_{k=-\infty}^{\infty} w_s z^{-s},$$

and the right side is precisely $W(f)$. ■

■ EXAMPLE 4.5

Consider the vector $\ell \in L^2(\mathbb{Z})$ with components

$$\ell_0 = \frac{1}{2}, \quad \ell_1 = \frac{1}{2}, \quad \ell_k = 0, \text{ otherwise.}$$

In Example 4.3 it was shown that the finite length (\mathbb{C}^N) version of ℓ is a crude low-pass filter. The same holds true in $L^2(\mathbb{Z})$.

To see this, consider the convolution $y = \ell * x$ for a general $x \in L^2(\mathbb{R})$. In the frequency domain we have

$$Y(f) = L(f)X(f).$$

By Definition 4.5.1, $L(f)$ is given by

$$L(f) = \sum_{k=-\infty}^{\infty} \ell_k e^{-2\pi i k f} = \frac{1}{2}(1 + e^{-2\pi i f}).$$

The magnitude of $L(f)$ is $|L(f)| = \cos(\pi f)$ on the Nyquist range $-\frac{1}{2} \leq f \leq \frac{1}{2}$ (this follows from $L(f) = e^{-\pi i f} \frac{1}{2}(e^{\pi i f} + e^{-\pi i f}) = e^{-\pi i f} \cos(\pi f)$). We have $L(0) = 1$. So the dc component of x passes unchanged, $L(f)$ tapers to zero at $f = \pm \frac{1}{2}$, and frequencies in x near the Nyquist rate are attenuated to zero.

4.5.4 The z-Transform

We can remove the restriction that $z = e^{2\pi i f}$ in the proof of Theorem 4.5.1 and consider $X(z)$ as a function of a “general” variable z . The resulting expression, a type of infinite series

$$X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k}, \tag{4.24}$$

is called the *bilateral z-transform* of the sequence x , though we’ll usually leave off the adjective “bilateral.” The *unilateral z transform* is applied to sequences $x = (x_0, x_1, \dots)$ and won’t concern us. Also many authors capitalize the “z,” that is, they call it the “Z-transform.”

Two Points of View But this definition of the z-transform is a bit ambiguous—what can z be? There are two distinct ways we can approach the z-transform:

1. We can treat the z-transform *formally*, by considering z as an indeterminate “placeholder.” We don’t think about plugging in real or complex numbers for z and then summing the resulting series. In this approach we can write down the z-transform of any bi-infinite sequence.
2. We can treat the z-transform *analytically*, by considering $X(z)$ as a function of a complex variable z . In this case we must concern ourselves with the convergence of the infinite series. On the bright side, it means we can apply powerful tools from complex analysis to z-transforms.

■ EXAMPLE 4.6

Let \mathbf{x} be the sequence in $L^2(\mathbb{Z})$ with components $x_{-1} = 2, x_0 = 1, x_1 = 3, x_2 = -2$, and all other components equal to zero. The z -transform of \mathbf{x} is

$$X(z) = 2z + 1 + 3z^{-1} - 2z^{-2}.$$

We can consider $X(z)$ formally, as a *Laurent polynomial* in z (a finite sum of positive and negative powers of z). Or we can treat $X(z)$ analytically, by considering it as a function defined on the complex plane. In this case $X(z)$ is well-defined for any $z \neq 0$.

■ EXAMPLE 4.7

Let \mathbf{x} be the sequence in $L^2(\mathbb{Z})$ with components $x_0 = 0$ and $x_k = 1/|k|$ for $k \neq 0$. In this case the z -transform is

$$X(z) = \cdots + \frac{1}{2}z^2 + z + 0 + z^{-1} + \frac{1}{2}z^{-2} + \cdots$$

Formally, $X(z)$ is merely another way to write \mathbf{x} . Analytically $X(z)$ is problematic. A simple ratio test shows that the sum of the positive powers of z diverges for $|z| > 1$, while the series consisting of the negative powers diverges for $|z| < 1$. As a function of a complex variable $X(z)$ is defined—at best—only on the unit circle $|z| = 1$. This greatly limits what we can do with complex analysis.

■ EXAMPLE 4.8

Let \mathbf{x} be the sequence in $L^2(\mathbb{Z})$ with $x_k = 1/2^k$ for $k \geq 0, x_k = 0$ for $k < 0$. The z -transform of \mathbf{x} is, formally,

$$X(z) = \sum_{k=0}^{\infty} \frac{1}{2^k} z^{-k} = 1 + \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2} + \cdots.$$

If we consider $X(z)$ as a function of a complex variable z , then a ratio test show that the series defined by $X(z)$ converges for $|z| > \frac{1}{2}$ and diverges for $|z| < \frac{1}{2}$. In the region of convergence we find $X(z) = 1/(1 - (2z)^{-1})$ (it's a simple geometric sum).

Now consider the sequence \mathbf{y} with components $y_k = 0$ for $k \geq 0$ and $y_k = -2^{-k}$ for $k < 0$ (the components y_k grow as k decreases, so this vector is not in $L^2(\mathbb{Z})$, but let us press on). Then

$$Y(z) = - \sum_{k=-\infty}^{-1} \frac{1}{2^k} z^{-k} = \cdots + 8z^3 + 4z^2 + 2z.$$

From the formal perspective $X(z)$ and $Y(z)$ are different. Considered as a function of z , however, it's not hard to show (see Exercise 4.33) that $Y(z) = 1/(1 - z^{-1}/2)$ for $|z| < \frac{1}{2}$ and diverges for $|z| > \frac{1}{2}$. Superficially it appears that from the analytical point of view we have $X(z) = Y(z)$, but this is not correct! A function always comes with a domain of definition. Since $X(z)$ and $Y(z)$ don't have the same domain of definition, they are distinct functions. Without a domain of definition $1/(1 - z^{-1}/2)$ is just an expression, not a function.

In this text we will take the formal point of view with regard to z -transforms, for the simple reason that this is sufficient for all of our needs. In particular, most of the signals we z -transform, though ostensibly in $L^2(\mathbb{Z})$, have only finitely many nonzero components. An analytic treatment won't be necessary.

Algebra of z -Transforms; Convolution The z -transform is a useful tool in signal processing. As with the other transforms we've studied, many operations in the time domain have simple parallels in the z -frequency domain. In particular, it's easy to see that the z -transform is linear: if \mathbf{x} and \mathbf{y} are any bi-infinite sequences with z -transforms $X(z)$ and $Y(z)$, respectively, and $\mathbf{w} = c\mathbf{x} + d\mathbf{y}$, then

$$W(z) = cX(z) + dY(z).$$

Note also that the proof of Theorem 4.5.1 works perfectly well for "any" z , so we've already shown

Theorem 4.5.2 *Let \mathbf{x} and \mathbf{y} be elements of $L^2(\mathbb{Z})$ with z -transforms $X(z)$ and $Y(z)$, respectively. Let $\mathbf{w} = \mathbf{x} * \mathbf{y}$ (note $\mathbf{w} \in L^\infty(\mathbb{Z})$). The z -transform $W(z)$ of \mathbf{w} is*

$$W(z) = X(z)Y(z).$$

Theorem 4.5.1 becomes a special case of Theorem 4.5.2 where $z = e^{2\pi i f}$.

From the formal perspective the z -transform thus provides a convenient way to do algebra—especially addition and convolution—for signals in $L^2(\mathbb{Z})$.

4.5.5 Convolution in \mathbb{C}^N versus $L^2(\mathbb{Z})$

If \mathbf{x} and \mathbf{y} are elements of \mathbb{C}^N , then we can also consider each as an element of $L^2(\mathbb{Z})$ by extending each with zeros. However, it is not generally the case that the convolution of \mathbf{x} and \mathbf{y} in $L^2(\mathbb{Z})$ will equal the circular convolution of \mathbf{x} and \mathbf{y} in \mathbb{C}^N , even if we interpret indexes modulo N .

■ EXAMPLE 4.9

Let $\mathbf{x} = (x_0, x_1, x_2)$ and $\mathbf{y} = (y_0, y_1, y_2)$ be vectors in \mathbb{C}^3 , where we also consider

$$\mathbf{x} = (\dots, 0, x_0, x_1, x_2, 0, \dots) \quad \text{and} \quad \mathbf{y} = (\dots, 0, y_0, y_1, y_2, 0, \dots)$$

as elements of $L^2(\mathbb{Z})$. The circular convolution $\mathbf{w} = \mathbf{x} * \mathbf{y}$ in \mathbb{C}^3 has components

$$\begin{aligned} w_0 &= x_0 y_0 + x_1 y_2 + x_2 y_1, & w_1 &= x_0 y_1 + x_1 y_0 + x_2 y_2, \\ w_2 &= x_0 y_2 + x_1 y_1 + x_2 y_0. \end{aligned}$$

But the vector $\tilde{\mathbf{w}} = \mathbf{x} * \mathbf{y} \in L^2(\mathbb{Z})$ has five nonzero components given by

$$\begin{aligned} \tilde{w}_0 &= x_0 y_0, & \tilde{w}_1 &= x_0 y_1 + x_1 y_0, & \tilde{w}_2 &= x_0 y_2 + x_1 y_1 + x_2 y_0, \\ \tilde{w}_3 &= x_1 y_2 + x_2 y_1, & \tilde{w}_4 &= x_2 y_2. \end{aligned}$$

There is a simple relation between \mathbf{w} and $\tilde{\mathbf{w}}$, however, and this relation is most elegantly understood in terms of the z -transform. The following ideas and notation will also be very useful in Chapter 6.

Some Notation Consider a Laurent polynomial $p(z)$ of the form

$$p(z) = \sum_{k=-m}^n a_k z^k$$

consisting of (possibly) negative and positive powers of z . We will use the notation “ $p(z) \bmod z^{-N}$ ” with $N > 0$ to denote that unique polynomial in the variable z^{-1} obtained by identifying in $p(z)$ all powers of z that are congruent modulo N . More formally,

$$p(z) \bmod z^{-N} := \sum_{j=0}^{N-1} \left(\sum_{k \equiv j \pmod N} a_k \right) z^{-j}.$$

Thus $p(z) \bmod z^{-N}$ is a sum of powers $z^{-(N-1)}, \dots, z^{-2}, z^{-1}, z^0$.

■ EXAMPLE 4.10

Let

$$p(z) = 2z^{-2} - z^{-1} + 3 + \frac{1}{2}z + z^2 + 5z^3.$$

Then

$$p(z) \bmod z^{-3} = (2 + \frac{1}{2})z^{-2} + (-1 + 1)z^{-1} + (3 + 5)z^0 = \frac{5}{2}z^{-2} + 8,$$

since we identify the powers $z^{-2} \equiv z^1$, $z^{-1} \equiv z^2$, and $z^0 \equiv z^3$.

Circular Convolution and z -Transforms To relate convolution in \mathbb{C}^N and $L^2(\mathbb{Z})$, let \mathbf{x} and \mathbf{y} be vectors in \mathbb{C}^N , and define $X(z)$ and $Y(z)$ as

$$X(z) = \sum_{k=0}^{N-1} x_k z^{-k}, \quad Y(z) = \sum_{k=0}^{N-1} y_k z^{-k},$$

polynomials in z^{-1} . These are merely the z -transforms of \mathbf{x} and \mathbf{y} , respectively, if we consider \mathbf{x} and \mathbf{y} as elements of $L^2(\mathbb{Z})$ via extension by zero. Let $\mathbf{w} = \mathbf{x} * \mathbf{y}$ (circular convolution in \mathbb{C}^N), and let

$$W(z) = \sum_{k=0}^{N-1} w_k z^{-k}.$$

Theorem 4.5.3 *If $\mathbf{x}, \mathbf{y} \in \mathbb{C}^N$ and $\mathbf{w} = \mathbf{x} * \mathbf{y}$ (circular convolution in \mathbb{C}^N), then*

$$W(z) = X(z)Y(z) \bmod z^{-N}.$$

Proof The proof is easy enough to write out directly, and is similar to the proof of Theorem 4.3.1 or Theorem 4.5.1. However, a slightly shorter proof is obtained by making use of Theorem 4.3.1.

For each integer k in the range $0 \leq k \leq N - 1$ let $\omega_k = e^{2\pi i k/N}$, an N th root of unity; we'll suppress the dependence of ω_k on N . Let \mathbf{X} , \mathbf{Y} , and \mathbf{W} denote the DFT's of \mathbf{x} , \mathbf{y} , \mathbf{w} , respectively. From the definition of the discrete Fourier transform we have $X_k = X(\omega_k)$, $Y_k = Y(\omega_k)$, and $W_k = W(\omega_k)$. From Theorem 4.3.1, we have $W_k = X_k Y_k$, or

$$W(\omega_k) = X(\omega_k)Y(\omega_k) \quad (4.25)$$

for $0 \leq k \leq N - 1$. Now define the function $P(z)$ as

$$P(z) := X(z)Y(z) \bmod z^{-N} \quad (4.26)$$

(so $P(z)$ consists of powers $z^{-(N-1)}, \dots, z^{-2}, z^{-1}, z^0$, while the full product $X(z)Y(z)$ consists of powers $z^{-2(N-1)}, \dots, z^{-2}, z^{-1}, z^0$). Then

$$X(\omega_k)Y(\omega_k) = P(\omega_k) \quad (4.27)$$

for $0 \leq k \leq N - 1$, since $\omega_k^m = \omega_k^n$ whenever $m \equiv n \pmod{N}$. We conclude from (4.25) and (4.27) that

$$W(\omega_k) = P(\omega_k) \quad (4.28)$$

for $0 \leq k \leq N - 1$.

From equation (4.28) the polynomials $z^{N-1}W(z)$ and $z^{N-1}P(z)$ of degree $N - 1$ agree at N distinct points $z = \omega_k$, $0 \leq k \leq N - 1$. Equivalently, the polynomial $z^{N-1}(W(z) - P(z))$ of degree $N - 1$ has N distinct roots. Since a nonzero n th degree polynomial has at most n complex roots, this forces $z^{N-1}(W(z) - P(z))$ to be the zero polynomial, or $W(z) = P(z)$. The theorem follows from equation (4.26). ■

Convolution in \mathbb{C}^N from Convolution in $L^2(\mathbb{Z})$ Theorem 4.5.3 makes it easy to see how convolution in \mathbb{C}^N and $L^2(\mathbb{Z})$ are related. Let \mathbf{x} and \mathbf{y} be vectors in \mathbb{C}^N (indexed from 0) and $\mathbf{w} = \mathbf{x} * \mathbf{y} \in \mathbb{C}^N$, circular convolution. Let $\tilde{\mathbf{w}} = \mathbf{x} * \mathbf{y}$, the convolution in $L^2(\mathbb{Z})$ (so $\tilde{\mathbf{w}} \in L^\infty(\mathbb{Z})$). We already know by Theorem 4.5.1 that \tilde{w}_m is the coefficient of z^{-m} in the product $X(z)Y(z)$, consisting of powers $z^{-2(N-1)}, \dots, z^{-1}, z^0$. By Theorem 4.5.3, we know that w_m is the coefficient of z^{-m} in $X(z)Y(z) \bmod z^{-N}$, $0 \leq m \leq N - 1$. Now the coefficient of z^{-m} in $X(z)Y(z) \bmod z^{-N}$ comes from adding the coefficients of the powers z^{-m} and z^{-m-N} in the product $X(z)Y(z)$. These coefficients are just \tilde{w}_m and \tilde{w}_{m+N} , and we conclude that

$$w_m = \tilde{w}_m + \tilde{w}_{m+N} \quad (4.29)$$

for $0 \leq m \leq N - 1$. Equation (4.29) shows how to convert convolution in $L^2(\mathbb{Z})$ into circular convolution in \mathbb{C}^N .

■ EXAMPLE 4.11

Let \mathbf{x} and \mathbf{y} be as in Example 4.9. From equation (4.29) we expect $w_0 = \tilde{w}_0 + \tilde{w}_3$, $w_1 = \tilde{w}_1 + \tilde{w}_4$, and $w_2 = \tilde{w}_2 + \tilde{w}_5$. These are easily seen to be true (note $\tilde{w}_5 = 0$).

Remark 4.5 The convolution command “conv” in Matlab is not circular convolution, although it acts on finite vectors. The conv command produces the convolution of vectors \mathbf{x} and \mathbf{y} in \mathbb{C}^N by considering them as elements of $L^2(\mathbb{Z})$ via zero-extension. We thus need to construct a circular convolution command “manually.” The simplest approach is to use the built-in conv command and then to employ equation (4.29). The Signal Processing Toolbox in Matlab also contains a command cconv that performs circular convolution.

We may also circularly convolve vectors $\mathbf{x} \in \mathbb{C}^M$ and $\mathbf{y} \in \mathbb{C}^N$ for $M \neq N$ by “padding” the shorter vector with zeros, and then performing a circular convolution in $\mathbb{C}^{\max(M,N)}$.

4.5.6 Some Filter Terminology

Let $\mathbf{x} = \{\dots x_{-1}, x_0, x_1, \dots\}$ be a bi-infinite signal and \mathbf{h} another bi-infinite vector such that

$$h_k = 0 \quad \text{for } k < L$$

and

$$h_k = 0 \quad \text{for } k > M$$

for some integers L, M . We will consider \mathbf{h} to be a filter. The vector $\mathbf{x} * \mathbf{h}$ ($= \mathbf{h} * \mathbf{x}$) has components given by

$$(\mathbf{x} * \mathbf{h})_m = \sum_{k=L}^M h_k x_{m-k},$$

so the value of $(\mathbf{x} * \mathbf{h})_m$ only depends on x_k for $m - M \leq k \leq m - L$. Only finitely many values of x_k are needed to compute $(\mathbf{x} * \mathbf{h})_m$, and since the sum is finite, there are no convergence issues.

We say that \mathbf{h} as above is a *finite impulse response* (FIR) filter. If $L \geq 0$, then the computation of $(\mathbf{x} * \mathbf{h})_m$ involves only values of x_k for $k \leq m$ (no “future” values, $k > m$). We then say that the filter \mathbf{h} is *causal*. If, on the other hand, we have $M \leq 0$, then

$$h_k = 0 \quad \text{for } k > 0$$

and \mathbf{h} is an *acausal* filter; the computation of $(\mathbf{x} * \mathbf{h})_m$ depends on future times. If

$$h_k = h_{-k},$$

then \mathbf{h} is a *symmetric filter*, and it is neither causal or acausal. In the real-time filtering of data the use of causal filters is obviously preferred because future samples are not available. For images all the data are known at one time and the concept of causality is generally irrelevant. In this setting many filters are in fact symmetric.

4.5.7 The Space $L^2(\mathbb{Z} \times \mathbb{Z})$

Although we won’t actually need them, the results of Section 4.5 extend in a straightforward way to the vector space $L^2(\mathbb{Z} \times \mathbb{Z})$ consisting of “arrays” \mathbf{A} with real or complex components $a_{r,s}$ for $-\infty < r, s < \infty$ such that

$$\sum_{r,s=-\infty}^{\infty} |a_{r,s}|^2 < \infty.$$

This set has an obvious vector space structure and inner product

$$(\mathbf{A}, \mathbf{B}) = \sum_{r,s=-\infty}^{\infty} a_{r,s} \overline{b_{r,s}}.$$

The Fourier transform of \mathbf{A} is the function $\widehat{A}(f_1, f_2)$ defined by

$$\widehat{A}(f_1, f_2) = \sum_{r,s=-\infty}^{\infty} a_{r,s} e^{-2\pi i(rf_1+sf_2)}$$

for $-\frac{1}{2} \leq f_1, f_2 \leq \frac{1}{2}$. The inverse transform is

$$a_{r,s} = \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} \widehat{A}(f_1, f_2) e^{2\pi i(rf_1+sf_2)} df_1 df_2.$$

We define convolution (which is commutative) as

$$(\mathbf{A} * \mathbf{B})_{r,s} = \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} a_{k,m} b_{r-k,s-m}. \tag{4.30}$$

If $\mathbf{C} = \mathbf{A} * \mathbf{B}$, then

$$\widehat{\mathbf{C}}(f_1, f_2) = \widehat{\mathbf{A}}(f_1, f_2)\widehat{\mathbf{B}}(f_1, f_2), \quad (4.31)$$

where $\widehat{\mathbf{C}}$ denotes the transform of \mathbf{C} .

The proof of these facts is analogous to the one-dimensional case.

4.6 MATLAB PROJECT

4.6.1 Basic Convolution and Filtering

1. Use Matlab to construct a one-second audio signal of the form

$$y(t) = 0.5 \sin(2\pi(200t)) + 0.2 \sin(2\pi(455t)) - 0.3 \cos(2\pi(672t))$$

sampled at 8192 Hertz with the commands

```
t = [0:8191]/8192;
y = 0.5*sin(2*pi*200*t) + 0.2*sin(2*pi*455*t)
- 0.3*cos(2*pi*672*t);
```

Play the signal with `sound(y)`. Because of the choice of 8192 for the length of y , the coefficient Y_k (for $0 \leq k \leq 4096$) in the DFT \mathbf{Y} corresponds to exactly k Hertz. However, since Matlab indexes vectors starting with index 1, Y_k corresponds to $\mathbf{Y}(k+1)$ in Matlab.

2. Design a high-pass filter \mathbf{h} of length 8192 that, when convolved with another vector $\mathbf{y} \in \mathbb{C}^{8192}$, filters out all DFT components Y_k with $0 \leq k \leq 500$ and the corresponding conjugate coefficients, while leaving all other frequencies unchanged. To do this, recall that if $\mathbf{w} = \mathbf{y} * \mathbf{h}$, then $W_k = Y_k H_k$, where \mathbf{H} is the DFT of \mathbf{h} . Thus we want $H_k = 0$ for $0 \leq k \leq 500$ and $7692 \leq k \leq 8191$ (the conjugate frequencies for $1 \leq k \leq 500$), as well as $H_k = 1.0$ for $501 \leq k \leq 7691$. Accounting for the fact that Matlab indexes from 1 we can construct \mathbf{H} with

```
H = zeros(1,8192);
H(502:7692)=1.0;
```

The (real-valued) filter itself is obtained as

```
h = real(ifft(H));
```

We need to take the real part, since `ifft` will likely leave some vestigial imaginary part due to round-off.

Plot the filter vector \mathbf{h} with `plot(h)`. Most of the coefficients should be very close to zero. It might be more illuminating to plot only $\mathbf{h}(1:100)$.

Count the number of nonzero coefficients with `sum(abs(h)>0)`; almost all coefficients should be nonzero.

3. Filter y as $w = y * h$. You can use the supplied routine `circconv` as

```
w = circconv(y,h);
```

(The same effect can be achieved in the frequency domain by zeroing out all appropriate Y_k .) Play the audio signal w . It should consist of a pure 672 Hertz tone.

Note that since h has essentially all components nonzero, the straightforward circular convolution of $y * h$ in the time domain requires on the order of N^2 operations where $N = 8192$. It would actually be more efficient to implement the convolution via an FFT/IFFT.

4. Determine the smallest threshold δ so that the number of coefficients in h which exceed δ is 10 or fewer. For example, the command

```
sum(abs(h)>0.01)
```

will count how many coefficients in h exceed 0.01 in magnitude.

Once you have an appropriate δ , construct a filter g consisting only of those coefficients of h that exceed the threshold, as

```
g = h .* (abs(h)>delta);
```

where `delta` is the appropriate threshold. The filter g should be an “economized” version of h that provides approximately the same frequency response with many fewer coefficients.

Plot the frequency response of the filter g by computing $G = fft(g)$; and executing

```
plot(abs(G(1:4097)))
```

(dc to the Nyquist frequency). Compare to the ideal frequency response given by the filter h .

5. Construct the filtered vector $w = y * g$ with the `circconv` command. Play w with the `sound` command and compare to the perfectly high-pass filtered version from question 2. Also plot the DFT of w on the range from dc to the Nyquist frequency. Are the frequencies below 500 Hertz still present?

Experiment with various values for the threshold δ , and note the compromise between filter length and high-pass performance.

4.6.2 Audio Signals and Noise Removal

1. Load in the “train whistle” signal with `load train`; (which loads the signal into a vector y). Recall that the audio signal is a vector of double-precision

floating point numbers in the range -1 to 1 . Add random noise to the signal with

$$yn = y + 0.2 * (\text{rand}(\text{size}(y)) - 0.5);$$

The noise added is uniformly distributed in the range -0.1 to 0.1 , though we could just as well use Gaussian noise. Play the noisy signal with `sound(yn)`. You may wish to multiply `yn` by a suitable constant (0.9 will work) to make sure the noisy signal is still in the range -1 to 1 . This avoids “clipping,” whereby a signal that exceeds its assigned range is clipped to the maximum or minimum permissible value. The audio signal should have a noticeable background hiss.

2. Use the techniques from the previous Section 4.6.1 to design a low-pass filter with cutoff frequency 2500 Hertz and no more than 40 filter coefficients. It may be helpful to refer back to Section 2.8.1 in the Matlab portion of Chapter 2, to recall the relation between frequency in Hertz and the Fourier coefficients Y_k . Plot the magnitude of the filter vector’s DFT. Use `circconv` to filter the noisy train whistle signal, and play the resulting audio signal.

Experiment with the cutoff frequency and/or number of filter coefficients. Is there any good compromise between fidelity to the original audio signal and effective noise removal?

4.6.3 Filtering Images

1. Load your favorite image into Matlab with `z = imread('myimage.jpg');`. Let us suppose that the image is $m \times n$ pixels. Construct a grayscale version `zg` of the image as in previous chapters, and then construct a noisy version of `zg` with

$$zn = \text{double}(zg) + 50 * (\text{rand}(\text{size}(zg)) - 0.5);$$

The image `zn` is just `zg` with each pixel corrupted by a uniformly distributed random variable in the -25 to 25 range; vary the 50 multiplier to get more or less noise. Set up an appropriate color map for viewing grayscale images (refer back to Section 1.11 in Chapter 1) with

$$\begin{aligned} L &= 255; \\ \text{colormap}([(0:L)/L; (0:L)/L; (0:L)/L]'); \end{aligned}$$

and view the image with `image(zn)`. Some pixels in `zn` will likely lie outside the range 0 to 255 , but will be clipped to black or white when displayed.

2. Try low-pass filtering the corrupted image `zn` to get rid of the noise, by circularly convolving `zn` with the mask `h` that has (indexing starting at zero) $h_{0,0} = 0.2$, $h_{0,1} = 0.2$, $h_{1,0} = 0.2$, $h_{m-1,0} = 0.2$, and $h_{0,n-1} = 0.2$. This mask has the effect of averaging each pixel with its four nearest neighbors. Set up the mask with `h = zeros(m, n)`, and then manually set `h(1, 1) = 0.2`, and so forth (remember, you now index from $1!$).

Performing the convolution in the “image” domain can be rather slow. Instead, apply the Fourier transform to both h and zn with the `fft2` command to obtain transforms H and Zn , multiply “point by point” as $Zdn = Zn.*H$; in the frequency domain, then apply the inverse transform as $zdn=real(ifft2(Zdn))$; . Display the denoised image zdn . Is this approach to noise removal effective?

3. To see that the mask h is indeed a low-pass filter, image the magnitude of the DFT H (use a 0 to 255 grayscale colormap) with `image(c*abs(H))` where c is a suitably chosen scaling constant. You should find that the components of H that correspond to high frequencies are small and so display as darker shades of gray or black, while the components that correspond to low frequencies are large and display as lighter shades or white. Recall also that in the indexing of $H_{j,k}$ with $0 \leq j \leq m - 1, 0 \leq k \leq n - 1$, high frequencies are near the center, and low frequencies near the corners.
4. Try filtering the noisy image zn repeatedly. This can be done painlessly with the command

$$Zdn = Zn.* (H.^N);$$

which applies the filter to zn N times in the frequency domain (think about why). Then set $zdn=real(ifft2(Zdn))$; . What happens when you filter repeatedly?

5. Let’s perform some edge detection as in Section 4.4.2. Use the same steps as in the last exercise (operate on the noiseless image $z1$, though) to construct and apply an $m \times n$ mask h with components $h_{0,0} = 1, h_{1,0} = -1$ and all other components zero to the image array $z1$. As remarked prior to equation (4.9), the convolution $z * h$ should detect horizontal edges. Display $|z * h|$ as a grayscale image; it may be helpful to rescale first.

Similarly construct an $m \times n$ mask v with $v_{0,0} = 1, v_{0,1} = -1$ and all other components zero. The quantity $|z1 * v|$ should detect vertical edges. Display this quantity as a grayscale image.

Display also the quantity $\sqrt{|z1 * v|^2 + |z1 * h|^2}$, a discrete version of the magnitude of the image function gradient. As above, it may be helpful to rescale before displaying, or altering the colormap so that 255 corresponds to black and 0 to white.

6. Redo item 3 above to compute and display as images the magnitude of the DFT’s H and V of the masks from item 5. Correlate the appearance of these DFT’s with the job that each mask performs.

EXERCISES

Convolution and Circulant Matrices

- 4.1 Prove that for complex numbers w, z we have the inequalities $|w||z| \leq \frac{1}{2}(|w|^2 + |z|^2)$, $|\operatorname{Re}(zw)| \leq |z||w|$, and $|\operatorname{Im}(zw)| \leq |z||w|$.

- 4.2** Write out the circulant matrix for $\mathbf{g} = (1, 5, 7, 2)^T$, and use it to compute the circular convolution of \mathbf{g} with the vector $\mathbf{x} = (1, -1, 1, 2)^T$.
- 4.3** Let $\mathbf{w} = (1, 0, \dots, 0) \in \mathbb{C}^N$ (indexed from 0, so $w_0 = 1$), and let \mathbf{x} be any vector in \mathbb{C}^N . Show that $\mathbf{x} * \mathbf{w} = \mathbf{x}$ by
- Using Definition 4.2.1 directly.
 - Computing the DFT \mathbf{W} of \mathbf{w} and using the Convolution Theorem 4.3.1.
- 4.4** Prove the assertion of Remark 4.1 on page 141.
- 4.5** Prove item (i) in Theorem 4.2.1 (the linearity of convolution).
- 4.6** Prove item (v) of Theorem 4.2.1, that if x_k and y_k are defined for all $k \in \mathbb{Z}$ via $x_k = x_{k \bmod N}$ and $y_k = y_{k \bmod N}$, then w_r as defined in equation (4.2) is defined for all r and satisfies $w_r = w_{r \bmod N}$.
- 4.7** Suppose that circulant matrices satisfy $\mathbf{M}_g = \mathbf{M}_h$ for vectors \mathbf{g} and \mathbf{h} in \mathbb{C}^N . Show $\mathbf{g} = \mathbf{h}$.
- 4.8** Let $\mathbf{h} \in \mathbb{C}^N$ and \mathbf{M}_h the corresponding circulant matrix. Show that the DFT matrix \mathbf{F}_N of Theorem 2.5.1 diagonalizes \mathbf{M}_h , that is,

$$\mathbf{M}_h = \mathbf{F}_N \mathbf{D}_H \mathbf{F}_N^{-1},$$

where \mathbf{D}_H denotes the diagonal matrix with diagonal entries H_m , $0 \leq m \leq N - 1$, the components of the DFT \mathbf{H} of the vector \mathbf{h} . Assume that the diagonal entries are ordered so that H_m corresponds to the basic waveform $\mathbf{E}_{N,m}$. Hint: Use Theorem 4.3.2.

- 4.9** a. Define the *time reversal* (also called the *adjoint*) of a filter $\mathbf{h} \in \mathbb{R}^N$ as the vector $\mathbf{h}' \in \mathbb{R}^N$ with components

$$h'_k = h_{-k \bmod N}$$

for $0 \leq k \leq N - 1$ (though we may extend \mathbf{h}' periodically in the index k as well). Show that the circulant matrices for \mathbf{h} and \mathbf{h}' are related as $\mathbf{M}_{\mathbf{h}'} = \mathbf{M}_{\mathbf{h}}^T$.

- b. We say that a filter \mathbf{h} is symmetric if $\mathbf{h} = \mathbf{h}'$. Show that \mathbf{h} is symmetric if and only if the circulant matrix \mathbf{M}_h is a symmetric matrix.
- c. For a filter $\mathbf{h} \in \mathbb{C}^N$ let us define by adjoint filter $\mathbf{h}' \in \mathbb{C}^N$ via

$$h'_k = \overline{h_{-k \bmod N}}$$

(the overline denotes complex conjugation). Show that the circulant matrices for \mathbf{h} and \mathbf{h}' are related as $\mathbf{M}_{\mathbf{h}'} = \mathbf{M}_{\mathbf{h}}^*$, where the superscript “ $*$ ” denotes the conjugate transpose.

Filtering

- 4.10** (Refer to Exercise 4.9 for notation.) What is the relationship among \mathbf{g} , \mathbf{h} , and $(\mathbf{g}' * \mathbf{h}')'$? Assume that the filters are real-valued. *Hint:* Use circulant matrices; part (iii) of Theorem 4.2.1 may be useful, and also Exercise 4.7.
- 4.11** *Matlab experiment:* Use Matlab to construct a sampled version \mathbf{f} of the function

$$f(t) = \begin{cases} \sin(10\pi t), & 0 \leq t < \frac{1}{2}, \\ \frac{1}{2}, & \frac{1}{2} \leq t < 1, \end{cases}$$

for $t \in [0, 1]$, say at $N = 256$ or more points. Let \mathbf{v} be the vector in \mathbb{R}^N with components

$$v_k = \begin{cases} \frac{1}{M}, & 0 \leq k \leq M - 1, \\ 0, & M \leq k \leq N - 1, \end{cases}$$

where M is to be chosen. Compute and plot the circular convolution $\mathbf{v} * \mathbf{f}$ for $M = N, 50, 20, 10, 5, 1$. Explain what you see.

- 4.12** *Matlab experiment:* Find out what a “Toeplitz” matrix is, and then read the help page for and experiment with Matlab’s “`toeplitz`” command to find a simple way to construct the circulant matrix for a filter $\mathbf{h} \in \mathbb{C}^N$. It may be helpful to note that the first column of \mathbf{M}_h is \mathbf{h} , whereas the first row of \mathbf{M}_h is the time reversal $\mathbf{h}' = (h_0, h_{N-1}, h_{N-2}, \dots, h_1)$ as in Exercise 4.9. Note also that if \mathbf{h} is an N -dimensional vector in Matlab (indexed from 1 to N), then the time reversal of \mathbf{h} is the vector $[h(1) \ h(N:-1:2)]$. Show your code and verify that it works on a vector in \mathbb{R}^4 .
- 4.13** *Matlab experiment:* Let \mathbf{M}_h be the circulant matrix for $\mathbf{h} \in \mathbb{C}^N$ and let $\mathbf{E}_{N,k}$ denote the basic exponential waveform defined by equation (1.23) in Chapter 1. Theorem 4.3.2 says that $\mathbf{E}_{N,k}$ an eigenvector of \mathbf{M}_h with eigenvalue H_k , where \mathbf{H} denotes the DFT of \mathbf{h} .
- Choose a nontrivial vector $\mathbf{h} \in \mathbb{C}^4$, and compute its DFT. Also construct the circulant matrix for \mathbf{h} (see Exercise 4.12), and compute the eigenvalues of this matrix. Verify that the eigenvalues are the H_k .
- 4.14** For symmetric filters (real coefficients) show that the eigenvalues discussed in 4.13 are real-valued. The result of Exercise 4.9 and Theorem 4.3.2 may help.
- 4.15** Consider a filter \mathbf{h} with coefficients

$$h_0 = \frac{1}{2}, \quad h_1 = -\frac{1}{2}, \quad h_r = 0 \quad \text{otherwise.}$$

This is called a “two-point differencing” filter. Compute the discrete Fourier transform of \mathbf{h} and plot the magnitude and phase of the DFT, as was done for the two-point averaging filter in Figure 4.3. Why is the two-point differencing filter an example of a “high-pass” filter?

- 4.16** Consider the convolutional equation $\mathbf{f} = \mathbf{x} * \mathbf{g}$ in which \mathbf{f} and \mathbf{g} are given vectors in \mathbb{C}^N and we are to solve for $\mathbf{x} \in \mathbb{C}^N$.
- Let $\mathbf{g} = (2, 1, 0, 3)$ and $\mathbf{f} = (4, 8, 20, 16)$. Find a vector $\mathbf{x} \in \mathbb{C}^4$ that satisfies $\mathbf{f} = \mathbf{x} * \mathbf{g}$. *Hint:* Use the DFT and the Convolution Theorem. You can use Matlab or do the DFT with the matrix \mathbf{F}_4 from equation (2.8) in Chapter 3.
 - For a general vector $\mathbf{g} \in \mathbb{C}^N$, find a simple condition on \mathbf{g} that guarantees that \mathbf{x} can be solved for uniquely for any $\mathbf{f} \in \mathbb{C}^N$.
 - If your condition on \mathbf{g} in part (b) is not met, might we still be able to find some \mathbf{x} that satisfies $\mathbf{f} = \mathbf{x} * \mathbf{g}$? Would \mathbf{x} be unique?
- 4.17** Prove Theorem 4.4.1. *Hint:* Mimic the proof of the one-dimensional case, Theorem 4.3.1.
- 4.18** Consider the two-point averaging process of equation (4.1) applied to a basic complex exponential vector $\mathbf{x} = \mathbf{E}_{N,m}$ to produce output \mathbf{w} . Show that $\mathbf{w} = A_m \mathbf{x}$ for some complex constant A_m . Compute $|A_m|$ and in particular, $|A_0|$ and $|A_{N/2}|$. Explain how this demonstrates that the two-point averaging filter is a low-pass filter.
Also compute $\arg(A_m)$, and use this to explain the linear appearance of the graph on the right in Figure 4.3.
- 4.19** In each case take vectors of length $N = 8$; use Matlab liberally.
- Use the inverse Fourier transform approach of Section 4.3.3 to design a filter \mathbf{h} that leaves any input \mathbf{x} unchanged. Write out \mathbf{h} explicitly, and note why it makes perfect sense.
 - Use the inverse Fourier transform approach design a filter \mathbf{h} that leaves the dc component of the input signal unchanged, but zeros out all other frequencies. Again, write out \mathbf{h} explicitly and note why it makes perfect sense.
 - Design a filter \mathbf{h} that leaves the Fourier coefficients X_2 and X_6 of the input \mathbf{x} unchanged, flips the sign of X_1 and X_7 , and zeros out all other frequencies. The filter \mathbf{h} should have real coefficients—why?
- 4.20** Let $\mathbf{x} \in \mathbb{R}^N$ be obtained by sampling the function $\phi(t) = \cos(2\pi mt)$ at times $t = k/N$ for $0 \leq k \leq N - 1$. Let $\mathbf{h} \in \mathbb{R}^N$ be a filter vector with DFT \mathbf{H} , and let $\mathbf{w} = \mathbf{x} * \mathbf{h}$. All vectors have real-valued components.
Show that \mathbf{w} is a sampled version of the function $|H_m|\phi(t + d)$ where $d = \arg(H_m)/(2\pi m)$. (This is just an amplified and phase-shifted version of $\phi(t)$). *Hint:* Use $\phi(t) = (e^{2\pi i m t} + e^{-2\pi i m t})/2$, and employ the reasoning that led to equation (4.6). Note also that $H_{N-k} = H_{-k} = \overline{H_k}$.
Show that the same conclusion holds for $\phi(t) = \sin(2\pi mt)$.
- 4.21** Let $\mathbf{h} \in \mathbb{C}^N$ be a filter vector and $\mathbf{w} = \mathbf{x} * \mathbf{h}$ (circular convolution in \mathbb{C}^N). Show that

$$\|\mathbf{w}\| \leq \|\mathbf{x}\|$$

for all inputs \mathbf{x} if and only if \mathbf{H} satisfies $|H_k| \leq 1$ for all k . Thus filtering by \mathbf{h} conserves or attenuates the energy of all input signals if and only if \mathbf{h} does not increase the energy at any given frequency. *Suggestion:* Use the Convolution Theorem and the result of Exercise 2.10.

Two-dimensional Convolution and Masks

- 4.22** Write out explicitly and plot (analogous to Figure 4.6) the magnitude of the 2D DFT for the vertical and horizontal edge detection masks used to produce Figures 4.9 and 4.10, for the case of 100×100 pixel images.
- 4.23** A filter or mask \mathbf{h} is called “sparse” if the number of nonzero elements in \mathbf{h} is small in comparison to the size of the vectors or arrays with which \mathbf{h} is convolved. Here the term “small” is somewhat subjective.
- Let $\mathbf{h} \in \mathbb{C}^N$ be a filter vector with only r nonzero components and $\mathbf{x} \in \mathbb{C}^N$ a vector with possibly all components nonzero. Determine the maximum number of operations required to compute the circular convolution $\mathbf{h} * \mathbf{x}$ in the time domain. Count each complex add, subtract, multiply, or divide as a single operation.
 - Suppose that we have an FFT algorithm that can compute each of \mathbf{H} and \mathbf{X} with $2N \log_2(N)$ operations, and we have a correspondingly efficient IFFT algorithm. How many operations will be needed to compute $\mathbf{h} * \mathbf{x}$ using these FFT/IFFT algorithms and the convolution theorem?
 - What value for r will make a direct circular convolution more efficient than using the FFT? (The answer will depend on N .)
 - Repeat the analysis above for the problem of convolving an $m \times n$ mask \mathbf{D} with an $m \times n$ array \mathbf{A} . Assume that \mathbf{D} has r nonzero elements and that the FFT/IFFT algorithms have $2mn \log_2(mn)$ efficiency.
- 4.24** A mask $\mathbf{D} \in M_{m,n}(\mathbb{C})$ with components $d_{r,s}$ is called “factorizable” if

$$d_{r,s} = f_r g_s$$

for vectors $\mathbf{f} \in \mathbb{C}^m$, $\mathbf{g} \in \mathbb{C}^n$. Equivalently $\mathbf{D} = \mathbf{f}\mathbf{g}^T$ where \mathbf{f} and \mathbf{g} are treated as column vectors.

Show that if $\mathbf{A} \in M_{m,n}(\mathbb{C})$, then

$$\mathbf{A} * \mathbf{D} = \mathbf{M}_f \mathbf{A} \mathbf{M}_g^T,$$

where \mathbf{M}_f and \mathbf{M}_g are the circulant matrices for \mathbf{f} and \mathbf{g} .

- 4.25** To detect all edges in an image as in Section 4.4.2 and Figure 4.10, we could simply filter for both simultaneously as

$$\mathbf{A} \rightarrow (\mathbf{A} * \mathbf{V}) * \mathbf{D},$$

where \mathbf{V} and \mathbf{D} are the vertical and horizontal edge detection masks from Section 4.4.2. Why won’t this work well?

The Space $L^2(\mathbb{Z})$

- 4.26** Let $\mathbf{x} \in L^2(\mathbb{Z})$ have components

$$x_k = \begin{cases} 0, & k < 0, \\ e^{-k}, & k \geq 0. \end{cases}$$

- a. Verify that \mathbf{x} is a member of $L^2(\mathbb{Z})$.
 - b. Compute the discrete time Fourier transform $X(f)$ of \mathbf{x} . Hint: Use $1 + z + z^2 + \dots = 1/(1 - z)$ for $|z| < 1$.
- 4.27** Let $\mathbf{x} \in L^2(\mathbb{Z})$, and define $\mathbf{y} \in L^2(\mathbb{Z})$ with components $y_k = x_{k-m}$ for some fixed integer m . How are the discrete time Fourier transforms $X(f)$ and $Y(f)$ related? If we define $y_k = x_{-k}$, how are $X(f)$ and $Y(f)$ related?
- 4.28** Let $\mathbf{x} = (x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N$, and let $\tilde{\mathbf{x}}$ denote that element of $L^2(\mathbb{Z})$ defined by $\tilde{x}_k = x_k$ for $0 \leq k \leq N-1$, $\tilde{x}_k = 0$ otherwise (so $\tilde{\mathbf{x}}$ is just \mathbf{x} extended by zeros).
- a. Show that the DFT coefficients X_m of \mathbf{x} can be computed from $X(f)$, the discrete time Fourier transform of $\tilde{\mathbf{x}}$.
 - b. Show that $X(f)$ can be expressed in terms of the X_m as

$$X(f) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} X_m e^{2\pi i k(m/N-f)}.$$

Hint: First use the IDFT to express x_k in terms of the X_m .

- c. Show that if $f \neq m/N$ for any integer m in the range $0 \leq m \leq N-1$, then

$$X(f) = \frac{1}{N} \sum_{m=0}^{N-1} \left(\frac{1 - e^{2\pi i (m-fN)}}{1 - e^{2\pi i (m/N-f)}} \right) X_m.$$

Hint: Use part (b) and the identity (1.31) from Chapter 1.

- 4.29** Suppose that \mathbf{x} and \mathbf{y} are elements of $L^2(\mathbb{Z})$ and that \mathbf{x} has only finitely many nonzero components. Let $\mathbf{w} = \mathbf{x} * \mathbf{y}$. Show that $\mathbf{w} \in L^2(\mathbb{Z})$. Hint: Start by supposing that \mathbf{x} has only one nonzero component x_r . Show in this case that $\|\mathbf{x} * \mathbf{y}\| = |x_r| \|\mathbf{y}\|$. Then suppose that \mathbf{x} has finitely many nonzero components and make use of linearity and the triangle inequality.
- 4.30** To see that the convolution of two vectors in $L^2(\mathbb{N})$ (or $L^2(\mathbb{Z})$) need not be in $L^2(\mathbb{N})$, let \mathbf{x} be the vector with components $x_k = 1/k^\alpha$ for $k \geq 1$, where $\alpha \in (1/2, 3/4)$. We'll show $\mathbf{x} * \mathbf{x}$ is not in $L^2(\mathbb{N})$.
- a. Show that $\mathbf{x} \in L^2(\mathbb{N})$. Hint: Recall that a “ p -series”

$$\sum_{k=1}^{\infty} \frac{1}{k^p}$$

converges if and only if $p > 1$.

- b. Let $\mathbf{w} = \mathbf{x} * \mathbf{x}$, so

$$w_r = \sum_{k=1}^{r-1} \frac{1}{k^\alpha} \frac{1}{(r-k)^\alpha}.$$

All terms are positive, as is w_r . Show that

$$w_r \geq \frac{1}{(r-1)^\alpha} \sum_{k=1}^{r-1} \frac{1}{k^\alpha}. \quad (4.32)$$

- c. Show that

$$\sum_{k=1}^{r-1} \frac{1}{k^\alpha} \geq \int_1^{r-1} \frac{dx}{x^\alpha}. \quad (4.33)$$

Hint: Show that the left side above is a left Riemann sum for the integral on the right (with rectangle base width 1, and one “extra” rectangle); note also that $1/x^\alpha$ is decreasing in x . A sketch may be helpful.

- d. Evaluate the integral in (4.33), and combine this with (4.32) to show that

$$w_r \geq \frac{1}{(1-\alpha)(r-1)^\alpha} ((r-1)^{1-\alpha} - 1). \quad (4.34)$$

- e. Show that if $r \geq r_0$ where r_0 is chosen as $r_0 \geq 2^{1/(1-\alpha)} + 1$, then $(r-1)^{1-\alpha} - 1 \geq \frac{1}{2}(r-1)^{1-\alpha}$. Conclude that for $r \geq r_0$,

$$w_r \geq \frac{1}{2(1-\alpha)(r-1)^\alpha} (r-1)^{1-\alpha} = \frac{1}{2(1-\alpha)} \frac{1}{(r-1)^{2\alpha-1}}.$$

Why is $2^{1/(1-\alpha)} + 1$ necessarily finite?

- f. Based on part (e), explain why if $\frac{1}{2} < \alpha < \frac{3}{4}$ the sum

$$\sum_{k=r_0}^{\infty} w_r^2$$

cannot converge (and hence the sum starting at $r = 1$ can't converge either).
Hint: Use the p -series test again.

- 4.31 Prove the convolution theorem analogous to Theorem 4.5.1 for the space $L^2(\mathbb{Z} \times \mathbb{Z})$ (refer to equation (4.30) and Section 4.5.7) for the case in which the Fourier transforms of \mathbf{A} and \mathbf{B} have only finitely many nonzero coefficients (it holds more generally).

The z-Transform and Convolution

- 4.32 Let $\mathbf{x} = (-1, 2, 0, 4)$ and $\mathbf{y} = (1, 1, 2, -1)$ be vectors in \mathbb{C}^4 . We will also consider \mathbf{x} and \mathbf{y} as elements of $L^2(\mathbb{Z})$, via extension by zeros.

- a. Compute the z -transforms $X(z)$ and $Y(z)$, and the product $X(z)Y(z)$.
 - b. Use the result of part (a) to write out the convolution of \mathbf{x} and \mathbf{y} in $L^2(\mathbb{Z})$.
 - c. Use part (a) to compute the circular convolution of \mathbf{x} and \mathbf{y} in \mathbb{R}^4 by using Theorem 4.5.3.
 - d. Use part (b) to compute the circular convolution of \mathbf{x} and \mathbf{y} in \mathbb{R}^4 by using equation (4.29).
- 4.33** Show that the z -transforms $X(z)$ and $Y(z)$ in Example 4.8 are equal to $1/(1 - z^{-1}/2)$ on the domains asserted.
- 4.34** Adapt Example 4.5 to show that the vector $\mathbf{h} \in L^2(\mathbb{Z})$ with components $h_0 = 1/2$, $h_1 = -\frac{1}{2}$ and all other $h_k = 0$ acts as a high-pass filter under convolution.
- 4.35** Find a vector $\mathbf{x} \in L^2(\mathbb{Z})$ such that $\mathbf{x} * \mathbf{x} = \mathbf{y}$, where

$$y_0 = 1, \quad y_1 = 0, \quad y_2 = -4, \quad y_3 = 0, \quad y_4 = 4$$

and all other $y_k = 0$. *Hint:* Use the z -transform and Theorem 4.5.1.

- 4.36** Suppose that $\mathbf{x} \in L^2(\mathbb{Z})$ with z -transform $X(z)$. Let \mathbf{y} be defined by $y_k = x_{k+r}$ for some integer r . Show that $Y(z) = z^r X(z)$.

CHAPTER 5

WINDOWING AND LOCALIZATION

5.1 OVERVIEW: NONLOCALITY OF THE DFT

Let's begin with an example.

■ EXAMPLE 5.1

Consider two different audio signals $f(t)$ and $g(t)$ defined on $0 \leq t \leq 1$, given by

$$f(t) = 0.5 \sin(2\pi(96)t) + 0.5 \sin(2\pi(235)t)$$

and

$$g(t) = \begin{cases} \sin(2\pi(96)t), & 0 \leq t < \frac{1}{2}, \\ \sin(2\pi(235)t), & \frac{1}{2} \leq t \leq 1. \end{cases}$$

Both are composed of the same two basic waveforms, $\sin(2\pi(96)t)$ and $\sin(2\pi(235)t)$. In f the waveforms are present throughout, while in g each waveform is present for exactly half of the time interval $[0, 1]$.

Let us sample each at 1000 Hertz to produce sample vectors \mathbf{f} and \mathbf{g} , and then compute the DFT of each sampled signal. The magnitude of each is plotted in Figure 5.1, DFT(\mathbf{f}) on the left and DFT(\mathbf{g}) on the right, where in each graph the horizontal index k corresponds to k Hertz. It's obvious from the plots that

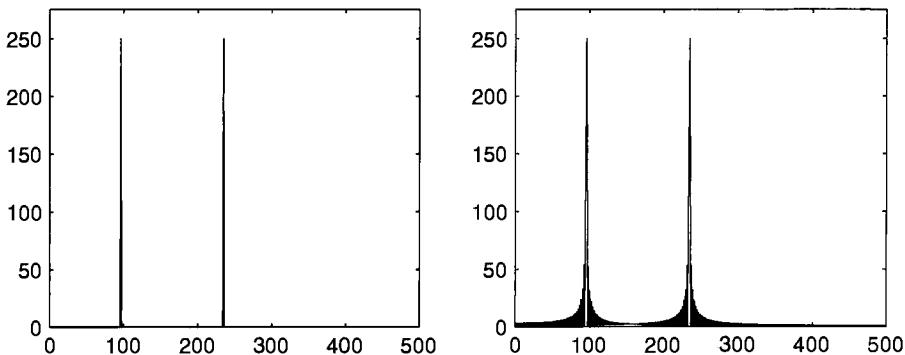


FIGURE 5.1 Magnitude of DFT for f (left) and g (right).

each signal contains dominant frequencies near 96 and 235 Hertz, and the magnitude of the DFT's are otherwise fairly similar. But the signals $f(t)$ and $g(t)$ are quite different in the time domain, a fact that is difficult to glean from the DFT graphs.

This example illustrates one of the shortcomings of traditional Fourier transforms: the coefficient X_k in $\mathbf{X} = \text{DFT}(\mathbf{x})$ quantifies the amount of the corresponding waveform in \mathbf{x} as if that waveform is present throughout the entire signal at constant amplitude. We can see this clearly in the inverse DFT formula

$$\mathbf{x} = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathbf{E}_{N,k}$$

in which the basic waveform $\mathbf{E}_{N,k}$ appears multiplied by X_k throughout. However, it's quite common that a frequency appears in one part of a signal and not another, as in $g(t)$ above. The DFT doesn't make it easy to recognize or quantify this. The essence of the problem is the "nonlocality" or global nature of the basis vectors $\mathbf{E}_{N,k}$ or underlying analog waveforms $e^{2\pi i k t / T}$: they are nonzero at almost all points.

Discontinuities are particularly troublesome. For example, consider a function that is identically 1 for $t \in [0, \frac{1}{2}]$ and 0 for $t \in [\frac{1}{2}, 1]$. In any discretization of the signal the discontinuity excites "most" of the coefficients X_k in the DFT (see Exercise 5.2), but this is in some sense a misrepresentation of the signal. The signal consists of pure dc for both halves of the time interval, with a single highly localized discontinuity. However, the excitation of many X_k gives the impression that the entire signal is highly oscillatory.

What we'd like is the ability to *localize* our frequency analysis to smaller portions of the signal. We've already employed this strategy for both audio signals and images in the Matlab project of Section 3.9. Indeed we saw that certain frequencies may be present in some parts of a signal and not others, which is what led to the "JPEG idea"

of breaking the image up into blocks and compressing each independently. The notion of “localizing” the analysis of a signal by performing block-by-block transforms dates back to (at least) 1946 (see [13]). In conjunction with Fourier methods this leads to “windowed Fourier transforms,” which we’ll take a look at in this chapter, as well as “spectrograms,” a way to present such data.

We’ll focus on windowing discrete signals of finite length, but analogous results hold for signals in $L^2(\mathbb{Z})$ and analog signals (functions). See Exercise 5.9.

5.2 LOCALIZATION VIA WINDOWING

5.2.1 Windowing

Consider a sampled signal $\mathbf{x} \in \mathbb{C}^N$, indexed from 0 to $N - 1$ as usual. We wish to analyze the frequencies present in \mathbf{x} , but only within a certain index (time) range. To accomplish this, we choose integers $m \geq 0$ and M such that $m + M \leq N$ and define a vector $\mathbf{w} \in \mathbb{C}^N$ as

$$w_k = \begin{cases} 1, & m \leq k \leq m + M - 1, \\ 0, & \text{otherwise.} \end{cases} \quad (5.1)$$

Define a vector \mathbf{y} with components

$$y_k = w_k x_k \quad (5.2)$$

for $0 \leq k \leq N - 1$. We use the notation $\mathbf{y} = \mathbf{w} \cdot \mathbf{x}$ to denote the component-by-component product defined by equation (5.2) and refer to the vector \mathbf{w} as the *window*.

The reason for this terminology is simple: the operation $\mathbf{x} \rightarrow \mathbf{w} \cdot \mathbf{x}$ zeros out all components x_k with $k < m$ and $k > m + M$ while leaving the other components unchanged, and thus it provides a “window” on \mathbf{x} in the range $m \leq k \leq m + M - 1$. We localize our frequency analysis of \mathbf{x} by analyzing \mathbf{y} . By changing m and/or M , we can systematically localize the analysis to different portions of the signal \mathbf{x} . The window vector \mathbf{w} defined by equation (5.1) is called a *rectangular* window. We’ll discuss others later.

The relation between the DFTs of \mathbf{x} and \mathbf{y} requires a bit of analysis, however. The operation of windowing induces a specific type of distortion into the process, even in the case where \mathbf{x} consists of only a few basic waveforms.

■ EXAMPLE 5.2

Let us consider the analog signal $f(t) = 0.5 \sin(2\pi(96)t) + 0.5 \sin(2\pi(235)t)$ from Example 5.1 at the start of the chapter, sampled at times $t = k/1000$ for $0 \leq k \leq 999$ to produce sample vector $\mathbf{f} \in \mathbb{C}^{1000}$. The DFT of \mathbf{f} was shown on the left in Figure 5.1. Now consider a windowed version $\tilde{\mathbf{f}} \in \mathbb{C}^{1000}$ obtained as $\tilde{\mathbf{f}} = \mathbf{w} \cdot \mathbf{f}$ where $\mathbf{w} \in \mathbb{C}^{1000}$ has components $w_k = 1$ for $100 \leq k \leq 149$, and $w_k = 0$

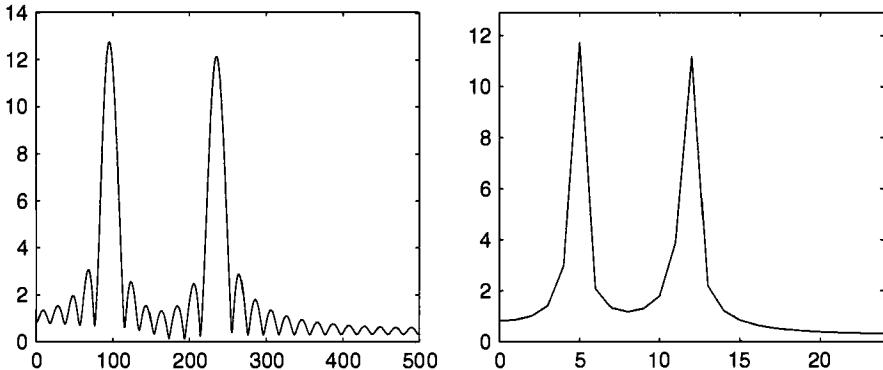


FIGURE 5.2 DFT magnitude for two windowed versions of \tilde{f} .

otherwise. The magnitude of the DFT of \tilde{f} is shown on the left in Figure 5.2, in the frequency range 0 to 500 Hertz (the Nyquist frequency).

The windowed signal \tilde{f} is 95% zeros, however, it seems inefficient to perform a 1000-point DFT on a vector with only 50 nonzero components. On the right in Figure 5.2 we show the 50-point DFT of the nonzero portion (f_{100}, \dots, f_{149}) of the windowed vector. The frequency range remains dc to 500 Hertz, but now the DFT index k corresponds to frequency $20k$ Hertz. Hence the index range is 0 to 25.

It's clear that both DFTs in Figure 5.2 are closely related to that on the left in Figure 5.1. We want to understand this relation quantitatively.

5.2.2 Analysis of Windowing

Suppose that we have a sampled signal $\mathbf{x} \in \mathbb{C}^N$ and produce a windowed version $\mathbf{y} \in \mathbb{C}^N$ as

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{w}, \quad (5.3)$$

where $\mathbf{w} \in \mathbb{C}^N$. We previously chose \mathbf{w} according to equation (5.1), but more general choices are possible. For the moment let's assume that as in (5.1), \mathbf{w} has components $w_k = 0$ for $k < m$ and $k \geq m + M$, but leave w_k unspecified in the range $m \leq k \leq m + M - 1$. This will allow greater flexibility in windowing later. Windowing according to equation (5.3) will still zero out every component of \mathbf{x} that is not in the range $m \leq k \leq m + M - 1$. After windowing, we perform an M -point DFT on just the nonzero portion $\tilde{\mathbf{x}} = (w_m x_m, w_{m+1} x_{m+1}, \dots, w_{m+M-1} x_{m+M-1})$ of the windowed signal \mathbf{y} .

Our goal is to determine the relationship between the M -point DFT of $\tilde{\mathbf{x}}$ and the original N -point DFT of the full signal \mathbf{x} . In what follows we will assume that M divides N so that $N = qM$ for some integer q .

The following two propositions will be helpful in answering this question. The first is a counterpart to the convolution theorem in which the roles of the time and frequency domains are reversed, as well as the roles of the DFT and IDFT.

Proposition 5.2.1 *Let \mathbf{x} and \mathbf{w} be vectors in \mathbb{C}^N with DFT's \mathbf{X} and \mathbf{W} , respectively. Let $\mathbf{y} = \mathbf{x} \cdot \mathbf{w}$ as defined by equation (5.2) have DFT \mathbf{Y} . Then*

$$\mathbf{Y} = \frac{1}{N} \mathbf{X} * \mathbf{W}, \quad (5.4)$$

where $*$ is circular convolution in \mathbb{C}^N .

In short, a component-by-component product in the time domain becomes a convolution in the frequency domain. The proof is almost identical to that of Theorem 4.3.1 and is left to Exercise 5.3.

From Exercise 2.16 in Chapter 3 we also have

Proposition 5.2.2 *Let $\mathbf{y} \in \mathbb{C}^N$ have DFT \mathbf{Y} . Let $\tilde{\mathbf{y}} \in \mathbb{C}^N$ be the vector obtained by circularly shifting \mathbf{y} as*

$$\tilde{y}_k = y_{(k+m) \bmod N}.$$

Then the DFT of $\tilde{\mathbf{y}}$ has components $\tilde{Y}_r = e^{2\pi i rm/N} Y_r$.

We'll determine the relationship between the original N -point DFT \mathbf{F} and the M -point DFT of the windowed signal in three steps:

1. Quantify the relation between the N -point DFT of \mathbf{x} and N -point DFT of the windowed version \mathbf{y} of equation (5.3).
2. Determine the frequency domain effect of shifting the windowed signal \mathbf{y} circularly as in Proposition 5.2.2, to the vector

$$\tilde{\mathbf{y}} = (w_m x_m, w_{m+1} x_{m+1}, \dots, w_{m+M-1} x_{m+M-1}, 0, \dots, 0) \in \mathbb{C}^N. \quad (5.5)$$

3. Determine the relation between the N -point DFT of $\tilde{\mathbf{y}}$ in (5.5) and the M -point DFT of $\tilde{\mathbf{x}} = (w_m x_m, w_{m+1} x_{m+1}, \dots, w_{m+M-1} x_{m+M-1}) \in \mathbb{C}^M$.

Step 1: Relation of \mathbf{X} and \mathbf{Y} Proposition 5.2.1 provides the answer: if we perform an N -point DFT on \mathbf{y} as defined by equation (5.3), then equation (5.4) shows

$$\mathbf{Y} = \frac{1}{N} (\mathbf{X} * \mathbf{W}). \quad (5.6)$$

Equation (5.6) makes it obvious that the N -point DFT \mathbf{Y} is not identical to \mathbf{Y} but is distorted by convolution with \mathbf{W} , the Fourier transform of the window vector.

■ EXAMPLE 5.3

In the case where \mathbf{w} is the rectangular window defined by equation (5.1), then \mathbf{W} can be worked out in closed form. It's easy to check that $W_0 = M$ while

$$\begin{aligned} W_k &= \sum_{j=m}^{m+M-1} e^{-2\pi i j k / N} \\ &= e^{-2\pi i m k / N} \sum_{j=0}^{M-1} e^{-2\pi i j k / N} \\ &= \frac{e^{-2\pi i m k / N} (1 - e^{-2\pi i M k / N})}{1 - e^{-2\pi i k / N}} \end{aligned}$$

for $k \neq 0$, where we yet again make use of the identity $1 + z + z^2 + \dots + z^{M-1} = (1 - z^M)/(1 - z)$. One can also check that

$$|W_k| = \left(\frac{1 - \cos(2\pi M k / N)}{1 - \cos(2\pi k / N)} \right)^{1/2}$$

for $k \neq 0$, while $|W_0| = M$. Note the value of m is irrelevant.

Figure 5.3 shows the magnitude of \mathbf{W} for the window vector \mathbf{w} used in Example 5.2. The many ripples away from the central peak are called the “side lobes” for the window. The operation of windowing $\mathbf{f} \in \mathbb{C}^{1000}$ in that example with \mathbf{w} in the time domain effectively convolves the DFT \mathbf{F} (Figure 5.1, left) with \mathbf{W} . This gives some indication of why the plot on the left in Figure 5.2 looks the way it does.

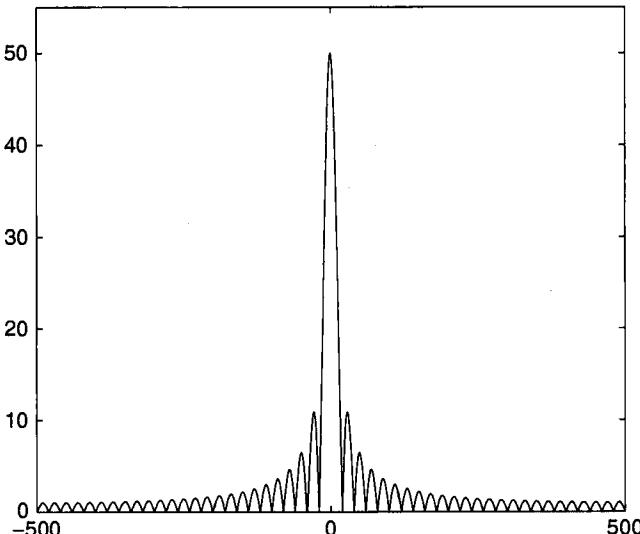


FIGURE 5.3 Magnitude for DFT \mathbf{W} .

Step 2: Effect of Index Shift The shifted vector $\tilde{\mathbf{y}} \in \mathbb{C}^N$ in equation (5.5) has components $\tilde{y}_k = y_{(k+m) \bmod N}$ with \mathbf{y} defined by (5.3). According to Proposition 5.2.2,

$$\begin{aligned}\tilde{Y}_r &= e^{2\pi i m r / N} Y_r \\ &= \frac{e^{2\pi i m r / N}}{N} (\mathbf{X} * \mathbf{W})_r,\end{aligned}\quad (5.7)$$

where $(\mathbf{X} * \mathbf{W})_r$ is the r th component of $\mathbf{X} * \mathbf{W}$ and the last line follows from equation (5.6).

Step 3: N-Point versus M-Point DFT The N -point DFT of the vector $\tilde{\mathbf{y}}$ is given by

$$\begin{aligned}\tilde{Y}_r &= \sum_{k=0}^{N-1} \tilde{y}_k e^{-2\pi i k r / N} \\ &= \sum_{k=0}^{M-1} \tilde{y}_k e^{-2\pi i k r / N}.\end{aligned}\quad (5.8)$$

for $0 \leq r \leq N - 1$, where the last line follows since $\tilde{y}_k = 0$ for $k > M - 1$. However, the M -point DFT of the vector $\tilde{\mathbf{x}} = (\tilde{y}_0, \tilde{y}_1, \dots, \tilde{y}_{M-1}) \in \mathbb{C}^M$ is given by

$$\tilde{X}_s = \sum_{k=0}^{M-1} \tilde{y}_k e^{-2\pi i k s / M}\quad (5.9)$$

for $0 \leq s \leq M - 1$, where we use $\tilde{\mathbf{X}} \in \mathbb{C}^M$ to denote this DFT.

Now recall that we are assuming that M divides N , so $N = qM$ for some integer q . If we substitute $M = N/q$ into the exponential on the right in equation (5.9), we obtain

$$\tilde{X}_s = \sum_{k=0}^{M-1} \tilde{y}_k e^{-2\pi i k (qs) / N}.$$

But from equation (5.8) this is exactly \tilde{Y}_r in the case of $r = qs$. In short, the relation between the N -point DFT $\tilde{\mathbf{Y}}$ and M -point DFT $\tilde{\mathbf{X}}$ is

$$\tilde{X}_s = \tilde{Y}_{qs}.\quad (5.10)$$

If we combine equations (5.7) and (5.10), we can summarize our results in the following theorem:

Theorem 5.2.1 Let $\mathbf{x} \in \mathbb{C}^N$ be windowed according to equation (5.3), where $\mathbf{w} \in \mathbb{C}^N$ with $w_k = 0$ for $k < m$ and $k \geq m + M$. Assume that $N = qM$. The relation between the N -point DFT \mathbf{X} of \mathbf{x} and the M -point DFT $\tilde{\mathbf{X}}$ of $\tilde{\mathbf{x}} =$

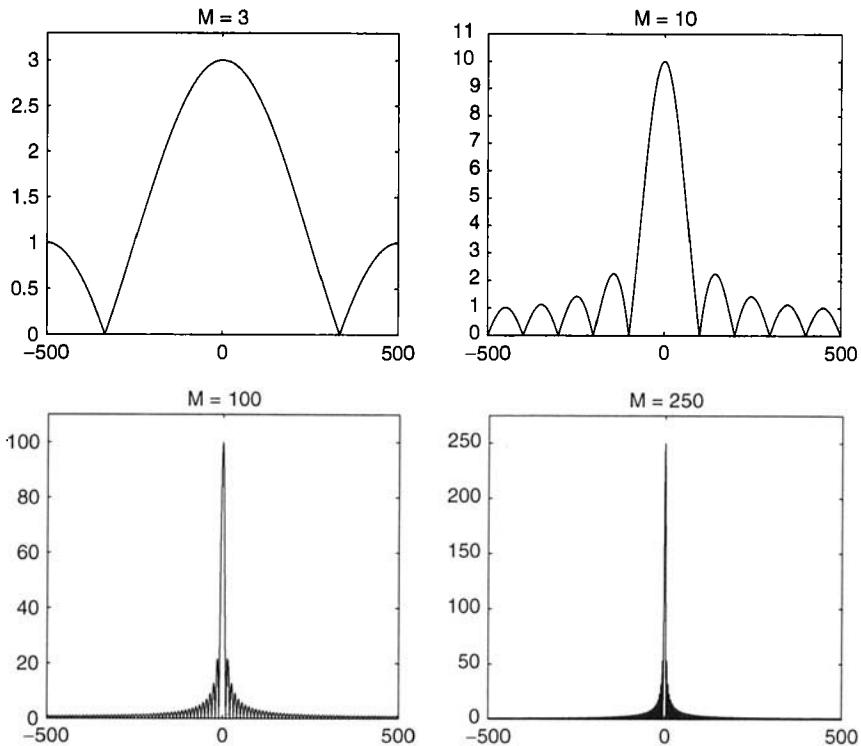


FIGURE 5.4 DFT magnitude $|W|$ for rectangular windows of width $M = 3, 10, 100, 250$.

$(w_m x_m, w_{m+1} x_{m+1}, \dots, w_{m+M-1} x_{m+M-1})$ is given by

$$\tilde{X}_s = \frac{e^{2\pi i m q s / N}}{N} (\mathbf{X} * \mathbf{W})_{qs} = \frac{e^{2\pi i m s / M}}{N} (\mathbf{X} * \mathbf{W})_{qs} \quad (5.11)$$

for $0 \leq s \leq M - 1$.

The last equality in (5.11) follows from $q/N = 1/M$.

■ EXAMPLE 5.4

Theorem 5.2.1 quantifies the distortion induced by windowing a signal. It's worth looking at this effect in some specific cases. Below are plots showing the magnitude of the DFT \mathbf{W} for rectangular windows of the form (5.1) for window widths $M = 3, 10, 100, 250$ (all with $m = 0$, but m doesn't influence the plot), in the case of $N = 1000$. It's apparent that as the window \mathbf{w} gets narrower (and so encompasses a more and more localized portion of the signal), the DFT \mathbf{W} gets wider. In the frequency domain the windowing process $\mathbf{x} \rightarrow \mathbf{w} \cdot \mathbf{x}$ is $\mathbf{X} \rightarrow \mathbf{X} * \mathbf{W}$,

and this frequency domain convolution tends to “smear out” the DFT of \mathbf{X} . As we’ll see in examples below, this can hamper our ability to distinguish closely spaced frequencies in a signal, but this is the price we pay for localizing the frequency analysis.

In Exercise 5.4 you are asked to analyze the extreme cases where $M = 1$ and $M = N$.

5.2.3 Spectrograms

A common approach to analyzing a signal with changing frequency content is to choose some window of length ΔT in the time domain, short enough so that the frequency content is relatively stable over the window length, and then to perform a frequency analysis on many windowed sub-blocks of length ΔT of the original signal. A typical windowed sub-block would be $(x_m, x_{m+1}, \dots, x_{m+M-1})$, where M samples corresponds to a time interval of length ΔT ; this also assumes that we use the rectangular window of (5.1). By changing the starting position m , we obtain a moving window on the signal. Analysis of the data in this window yields local frequency information on the signal. The sub-blocks into which we break the original signal may or may not overlap. This process of isolating different time slices of a signal and performing a Fourier transform on each is called a *windowed Fourier transform* or *short-time Fourier transform*.

Dividing the signal into many sub-blocks and computing the DFT of each produces a lot of data. The data can be summarized efficiently with a plot called a *spectrogram*. Suppose that the block size is M samples, with the k th block of data (starting with $k = 0$) given by

$$(x_{kn}, x_{kn+1}, \dots, x_{kn+M-1}), \quad (5.12)$$

where $n \geq 1$. The integer n controls the distance from the start of one block to the start of the next block. If $n = M$, the k th block is $(x_{kM}, \dots, x_{(k+1)M-1})$ and the k th and $k + 1$ sub-blocks do not overlap. By taking $n < M$, we obtain overlapping blocks, which can be useful. The magnitude of the M -point DFT of the k th block is stored as the k th column of a matrix, and this matrix is then displayed as an image, perhaps grayscale or color-coded. Choosing n so adjacent blocks overlap 50% to 80% is common.

This is best illustrated with a specific example.

■ EXAMPLE 5.5

Consider the signal $g(t)$ from Example 5.1 at the beginning of the chapter, sampled at 1000 Hz. The DFT of the full sampled signal vector \mathbf{g} was shown on the right in Figure 5.1, and it gives the mistaken impression that energy at frequencies 96 and 235 Hertz is present throughout the signal. Let us break the signal vector \mathbf{g} into sub-blocks of length $\Delta T = 0.05$ seconds (size $M = 50$ samples) and take $n = 20$ in (5.12), so the k and $k + 1$ blocks will overlap by 30 samples, or 60%.

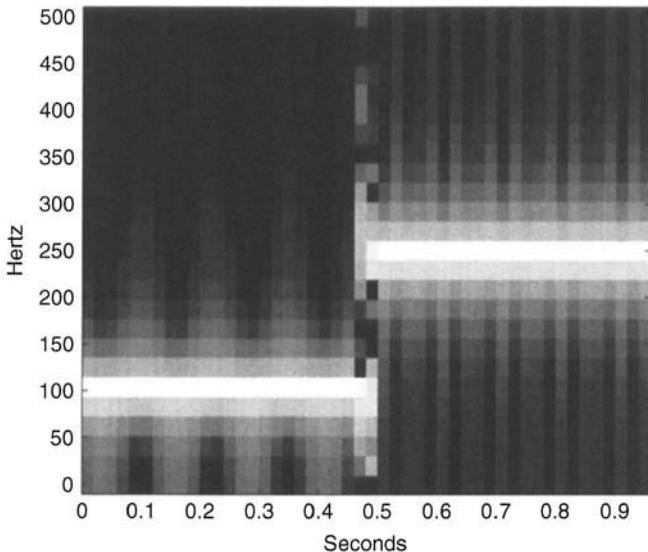


FIGURE 5.5 Spectrogram for g .

Each block is transformed and becomes a “column” in the grayscale spectrogram of Figure 5.5. The DFTs are displayed on a logarithmic scale, with lighter shades indicating greater magnitude.

Figure 5.5 makes it easy to see that the 96 Hertz energy is present for only the first 0.5 seconds, and then the 235 Hertz sinewave kicks in. Of course, there is some untidiness at the transition.

In Figure 5.6 we illustrate the effect of using narrower or wider windows. On the left, we use a rectangular window of width $M = 20$ ($\Delta T = 0.02$) and on the right, we use a window of length $M = 200$ ($\Delta T = 0.2$), both with 60% overlap between successive transforms. The narrower window more clearly picks up the

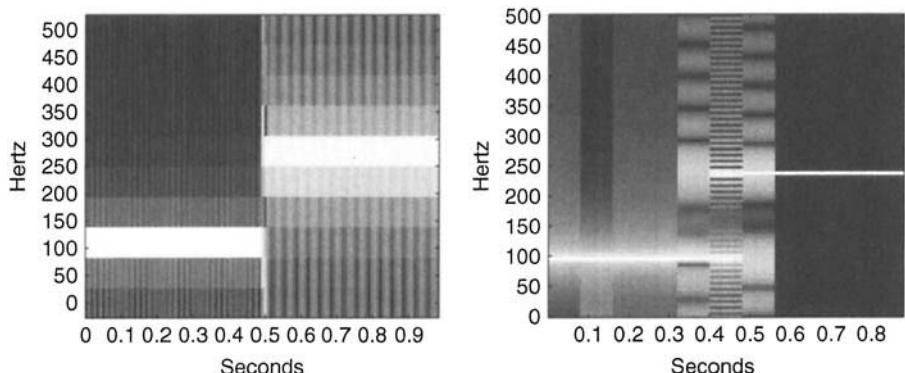


FIGURE 5.6 Two different window widths in the spectrogram for g .

abrupt frequency transition at $t = 0.5$, but it has less overall resolution in the frequency (vertical) domain. Conversely, the wider window resolves the change at $t = 0.5$ less crisply but gives better frequency resolution, at least when the frequency is stable. This is illustrated by the plots of Figure 5.4: the narrower the window, the broader the window's DFT. Convolution of this broader window DFT with the original signal's DFT tends to smear adjacent frequencies together.

■ EXAMPLE 5.6

Consider a signal consisting of two closely spaced frequencies and a third frequency that varies over time, namely

$$f(t) = 1.0 \sin(2\pi \cdot 111t) + 0.5 \sin(2\pi \cdot 123t) + 0.5 \sin(2\pi \omega(t)t)$$

for $0 \leq t \leq 1$, where $\omega(t) = 150 + 50 \cos(2\pi t)$. We sample $f(t)$ at 1000 Hertz to produce sample vector \mathbf{f} with components $f_k = f(k/1000)$, $0 \leq k \leq 999$. The DFT \mathbf{F} of \mathbf{f} is shown in Figure 5.7, scaled as $\log(1 + |\mathbf{F}_k|)$ over the Nyquist range 0 to 500 Hertz: The stable frequencies at 111 and 123 Hertz are obvious, but the plot makes it clear that the signal also contains energy at many other frequencies. Unfortunately, the plot doesn't show how this energy is distributed in time. Figure 5.8 is a spectrogram with sub-blocks of length $M = 100$ ($\Delta T = 0.1$), with 80% overlap between blocks ($n = 20$ in (5.12)). Now it's easy to see the portion $0.5 \sin(2\pi \cdot \omega(t)t)$ of the signal. However, the localization to sub-blocks of length $M = 100$ has blurred together the stable 111 and 123 Hertz components, though the presence of these stable frequencies is clear.

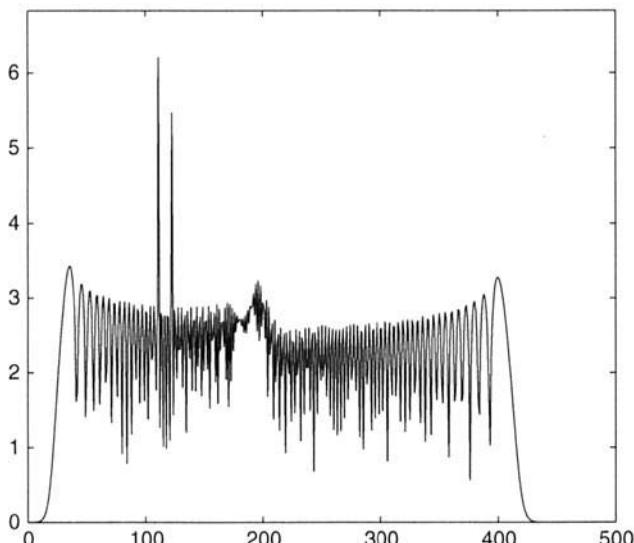


FIGURE 5.7 Plot of $\log(1 + |\mathbf{F}|)$ for signal \mathbf{f} .

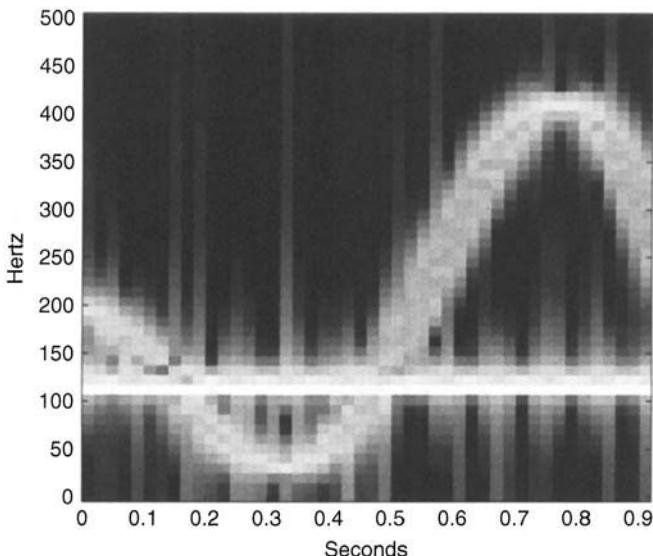


FIGURE 5.8 Spectrogram of signal f , $M = 100$.

Unfortunately, there is no perfect compromise that allows us to maintain the highest frequency resolution (to distinguish the closely spaced 111 and 123 Hertz components) while tracking the changing frequency. On the left in Figure 5.9 is the spectrogram obtained by taking $M = 500$, which provides better resolution of frequencies but much less localization in time. On the right in Figure 5.9 is the spectrogram obtained by taking $M = 20$, which provides better localization, but less frequency resolution. The horizontal axis is indexed to the time at which the relevant sub-block starts. For example, with $M = 500$, the last block starts at $t = 0.5$. See also Exercise 5.6.

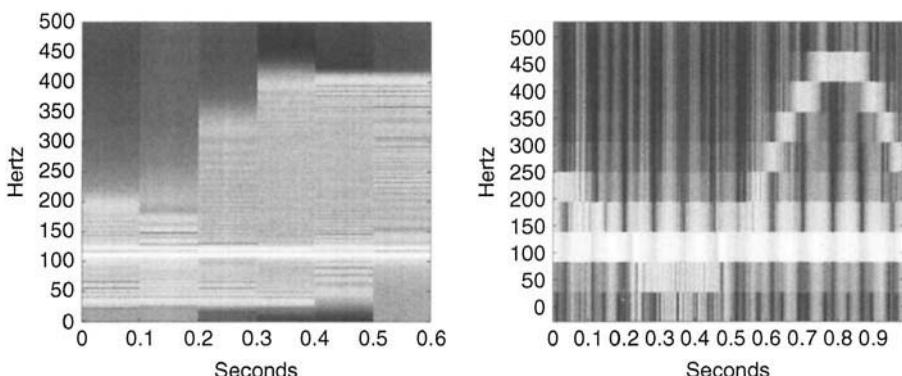


FIGURE 5.9 Spectrograms of signal f , $M = 500$ (left) and $M = 20$ (right).

5.2.4 Other Types of Windows

The specific window used above and as defined in (5.1) is called a *rectangular window*. As shown in Figure 5.3, the DFT of a rectangular window is rather oscillatory, with prominent side lobes, since the rectangular window is to good approximation a sampled version of a discontinuous function. When we use a rectangular window for localization, we end up convolving the “true” spectrum of the signal with the DFT of the window. This introduces the type of distortion shown in Figure 5.2.

But the rectangular window isn’t the only option. We may choose something other than 1 for the nonzero components of w . Since the prominent side lobes in the DFT of a rectangular window stem from the underlying discontinuity, it may be helpful to choose w so that the nonzero components taper gradually to zero at the ends of the window. This typically helps suppress the magnitude of the side lobes.

For example, we might take w to be defined as

$$w_j = \begin{cases} \frac{2j}{M}, & m \leq j \leq m + M/2, \\ \frac{M+m-1-j}{M/2+m-1}, & m + M/2 < j < m + M, \\ 0, & \text{else.} \end{cases}$$

the so-called *triangular window* (or “Bartlett” window). The case $N = 1000$, $m = 100$, $M = 49$ (comparable to the rectangular window with the DFT shown in Figure 5.3) is shown in Figure 5.10, along with the full 1000-point DFT of the window. Compare the DFT on the right with that of Figure 5.3. It’s clear that the side lobes are diminished. If we use this window for a signal x , then the energy at each frequency isn’t dissipated into oscillatory sidelobes quite so much. However, this benefit comes at the price of “fattening” the central lobe, an issue explored in the Matlab project of Section 5.3.

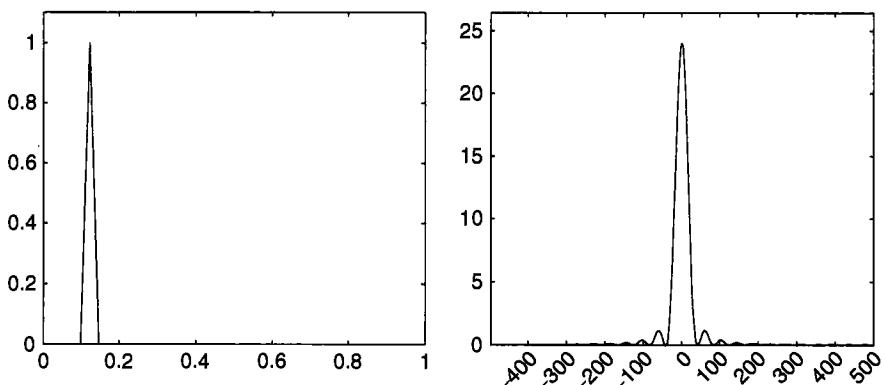


FIGURE 5.10 Triangular window and DFT.

Many other types of windows have been developed in an attempt to balance various competing needs: suppress side lobe energy, keep the central peak narrow, obtain good localization in time, and so forth. Some examples are the Gaussian window

$$w_j = \phi\left(\frac{j - (m + (M - 1)/2)}{M/2}\right), \quad (5.13)$$

where $\phi(t) = Ce^{-\alpha t^2}$ and C and α are constants that can be adjusted to alter the shape of and/or normalize the window. As defined, w_j peaks at $j = m + M/2$ and is symmetric about this point; very small values of w_j might be truncated to zero. Another option is the *Hamming window*, defined by

$$w_j = \begin{cases} \phi((j - m)/M), & m \leq j \leq m + M, \\ 0, & \text{else.} \end{cases} \quad (5.14)$$

with $\phi(t) = 0.54 - 0.46 \cos(2\pi t)$.

Many other windows—Hanning, Blackwell, Chebyshev, Kaiser, and the like—also exist in many variations. Any can be used to perform localized frequency analysis and construct spectrograms. Some of the trade-offs are illustrated in the Matlab exercises below.

5.3 MATLAB PROJECT

5.3.1 Windows

Let $f(t) = \sin(2\pi(137)t) + 0.4 \sin(2\pi(147)t)$ for $0 \leq t \leq 1$.

1. a. Use Matlab to sample $f(t)$ at 1000 points $t_k = k/1000$, $0 \leq k \leq 999$, as

```
t = [0:999]/1000;
f = sin(2*pi*137*t) + 0.4*sin(2*pi*147*t);
```

- b. Let $F = fft(f)$. Plot the magnitude of the first 501 components of F .
- c. Construct a (rectangular) windowed version of f (but still of length 1000) with $fw = f$; followed by $fw(201:1000) = 0.0$. Compute and display the magnitude of the first 501 components of the DFT of fw . Can you distinguish the two constituent frequencies? Be careful—is it really obvious that the second frequency isn't just side lobe leakage?
- d. Construct a windowed version of f of length 200, with $fw2 = f(1:200)$. Compute and display the magnitude of the first 101 components of the DFT of $fw2$. Can you distinguish the two constituent frequencies? Compare to the plot of the DFT of fw .

- e. Repeat parts (c) and (d) using shorter windows, such as 100 or 50, (and longer windows too). How short can the time window be and still allow resolution of the two separate frequencies? Does it seem to matter whether we treat the windowed signal as a vector of length 1000 as in part (c) or as a shorter vector as in part (d)? Does the side lobe energy confuse the issue?
2. Repeat problem 1, but use a triangular window. You can, for example, construct a window vector w of length 201 as

```
w = zeros(1,201);
w(1:101) = [0:100]/100;
w(102:201) = [99:-1:0]/100;
```

Then construct a windowed signal of length 1000 as $fw = zeros(size(f));$ and $fw(1:201) = f(1:201) .* w;$.

Try varying the window length. What is the shortest window that allows you to clearly distinguish the two frequencies?

3. Repeat problems 1 and 2 for the Hamming window.

5.3.2 Spectrograms

1. Construct a signal f with varying frequency content on the time interval $0 \leq t \leq 1$ sampled at 1 kHz as

```
t = [0:999]/1000;
p = 100 + 10 * cos(2 * pi * t);
f = sin(2 * pi * p.*t);
```

Compute the DFT of f and display its magnitude up to the Nyquist frequency 500 Hertz.

2. Use the supplied “spectrogram” routine `spectro` to compute a spectrogram of the signal f , using a rectangular window. You might start with

```
spectro(f, 1.0, 0.1, 50, 'rect');
```

(signal of length 1.0 seconds, rectangular windows of length 0.1 second, 50 percent overlap between windows).

Experiment with the window length and overlap.

3. Repeat problem 2 using a triangular window (string argument ‘tri’ to the `spectrogram` command).
4. Repeat problem 2 using a Gaussian window (string argument ‘gauss’ to the `spectrogram` command).

5. Load in the “splat” sound with `load splat` (recall the sampling rate is 8192 Hertz). Create an informative spectrogram, and comment on its relation to what you hear when the sound is played.

EXERCISES

- 5.1** Let $\mathbf{x} = (x_0, x_1, x_2, x_3)$ be a signal and $\mathbf{X} = (X_0, X_1, X_2, X_3)$ the DFT of \mathbf{x} . Let $\mathbf{w} = (0, 1, 1, 0)$ be the window vector and $\mathbf{y} = \mathbf{w} \cdot \mathbf{x}$.
- Compute the 4-point DFT \mathbf{X} explicitly/symbolically (the matrix \mathbf{F}_4 in equation (2.8) might be helpful).
 - Compute the 4-point DFT’s \mathbf{W} and \mathbf{Y} , and verify that equation (5.6) holds.
 - Compute the 2-point DFT $\tilde{\mathbf{X}}$ of just the nonzero portion $\tilde{\mathbf{x}} = (x_1, x_2)$ of the windowed vector, and verify that Theorem 5.2.1 holds.
- 5.2** Let $x(t)$ be defined on the interval $[0, 1]$ as

$$x(t) = \begin{cases} 1, & t < \frac{1}{2}, \\ 0, & t \geq \frac{1}{2}. \end{cases}$$

Let $\mathbf{x} \in \mathbb{R}^N$ with components $x_m = x(m/N)$ for $0 \leq m \leq N - 1$ be a sampled version of $x(t)$ and assume, for simplicity, that N is even. Show that the DFT coefficient X_k of \mathbf{x} is given by $X_0 = N/2$, $X_k = 0$ if k is even and positive, and

$$X_k = \frac{2}{1 - e^{-2\pi i mk/N}}$$

if k is odd. Hence over half of the DFT coefficients are nonzero, even though $x(t)$ consists of two “dc pieces.” *Hint:* Write out X_k explicitly from the definition of the DFT and use the geometric summation formula $1 + z + \cdots + z^{r-1} = (1 - z^r)/(1 - z)$ for $z \neq 1$.

- 5.3** Prove Proposition 5.2.1.
- 5.4** Consider analyzing the local frequency content of a signal $\mathbf{x} \in \mathbb{R}^N$ by taking (rectangular) windows of length M , of the form

$$\tilde{\mathbf{x}} = (x_m, x_{m+1}, \dots, x_{m+M-1})$$

for some value of m , then performing an M -point DFT of $\tilde{\mathbf{x}}$.

- Examine the extreme case where $M = 1$. In particular, compute the DFT of $\tilde{\mathbf{x}}$. Argue that $M = 1$ provides perfect localization in time, but essentially no frequency information except “local dc.”

- b. Examine the extreme case $M = N$ (with $m = 0$). In particular, compute the DFT of $\tilde{\mathbf{x}}$. Argue that $M = N$ provides no localization in time but undistorted frequency information.
- 5.5**
- Use Matlab to plot the Gaussian window defined by equation (5.13), with $N = 1000$, $m = 100$, $M = 49$, $C = 1$, and $\alpha = 5$. Compute and plot the magnitude of the DFT of this window.
 - Change α in part (a). Verify that as α gets larger, the window \mathbf{w} narrows and the DFT \mathbf{W} widens, and vice versa.
 - Use Matlab to plot the Hamming window defined by equation (5.14), with $N = 1000$, and $m = 100$, and $M = 49$. Vary M and observe the effect on the window and its DFT.
- 5.6** In Example 5.6 we considered a signal that contains a time-varying frequency component $\sin(2\pi\omega(t)t)$ with $\omega(t) = 150 + 50\cos(2\pi t)$. One might expect that the “local” frequency at any time t_0 is just $\omega(t_0)$, so in Example 5.6 we’d expect a low frequency of 100 Hertz and a high of 200 Hertz. However, Figure 5.8 indicates this isn’t quite the case—we see local frequencies in the spectrogram as low as (roughly) 30 Hertz at time $t = 0.32$ seconds and as high as 400 Hertz, around $t = 0.75$.

To understand this, consider a signal $f(t) = \sin(\omega(t)t)$ in a time interval $t_0 \leq t \leq t_0 + \Delta t$. Assume that $\omega(t)$ is differentiable.

- a. Justify the approximation

$$\omega(t)t \approx \omega(t_0)t_0 + (\omega(t_0) + \omega'(t_0)t_0)(t - t_0)$$

for $t_0 \leq t \leq t_0 + \Delta t$ if Δt is small. *Hint:* Apply the tangent line approximation to the function $\omega(t)t$.

- b. Show that to a good approximation we have

$$f(t) = \sin(a + \omega_0 t),$$

where $\omega_0 = \omega(t_0) + \omega'(t_0)t_0$ and a is some constant. Thus ω_0 is the “local” frequency of f near $t = t_0$.

- c. Use part (b) to explain the minimum and maximum frequencies seen in the spectrogram of Figure 5.8. In particular, plot ω_0 as a function of t_0 for $0 \leq t_0 \leq 1$.

- 5.7** Consider the following windowed or short-time Fourier transform. Given a signal vector $\mathbf{x} \in \mathbb{C}^N$, where N factors as $N = mn$, we compute the transform $\mathbf{X} \in \mathbb{C}^N$ as follows: for each integer k in the range $0 \leq k \leq m - 1$, let $\tilde{\mathbf{x}}_k$ denote the vector

$$\tilde{\mathbf{x}}_k = (x_{kn}, x_{kn+1}, \dots, x_{kn+n-1})$$

in \mathbb{C}^n . Let $\tilde{\mathbf{X}}_k$ denote the n -point DFT of $\tilde{\mathbf{x}}_k$, and let the full transform $\mathbf{X} \in \mathbb{C}^N$ of \mathbf{x} be obtained as the concatenation of the $\tilde{\mathbf{X}}_k$, that is,

$$\mathbf{X} = (\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \dots, \tilde{\mathbf{X}}_m)^T$$

if we treat \mathbf{X} as a column vector. In short, we transform \mathbf{x} by breaking the vector into nonoverlapping pieces of length n , take the DFT of each piece, and then concatenate the results. No windowing is performed prior to transforming each piece.

- a. What does the matrix representation of the transform $\mathbf{x} \rightarrow \mathbf{X}$ look like, in relation to \mathbf{F}_m (the m -point DFT matrix)? Show that $\mathbf{x} \rightarrow \mathbf{X}$ is invertible, and explain how to compute the inverse.
- b. Show that each component X_j , $0 \leq j \leq N - 1$, of \mathbf{X} can be computed as an inner product

$$X_j = (\mathbf{x}, \mathbf{v}_j)$$

for a suitable vector $\mathbf{v}_j \in \mathbb{C}^N$. (Note that X_j refers to a single component of \mathbf{X} , not the block \mathbf{X}_j). Show that the set \mathbf{v}_j , $0 \leq j \leq N - 1$ forms an orthogonal basis for \mathbb{C}^N .

- c. Suppose we window each piece prior to transforming; that is, take

$$\tilde{\mathbf{x}}_k = (w_0 x_{kn}, w_1 x_{kn+1}, \dots, w_{n-1} x_{kn+n-1})$$

for some vector $\mathbf{w} \in \mathbb{R}^n$. What conditions on \mathbf{w} are necessary if the transform $\mathbf{x} \rightarrow \mathbf{X}$ is to be invertible? Would you expect the corresponding vectors \mathbf{v}_j from part (b) to still be orthogonal?

- 5.8** Let \mathbf{x} and \mathbf{w} be vectors in $L^2(\mathbb{Z})$ with discrete time Fourier transforms $X(f)$ and $W(f)$ for $-\frac{1}{2} \leq f \leq \frac{1}{2}$, respectively (recall Definitions 4.5.1 and (4.17)). For simplicity, we'll assume that \mathbf{w} has only finitely many nonzero components. Define \mathbf{z} to have components $z_k = x_k w_k$. In this case $\mathbf{z} \in L^2(\mathbb{Z})$. Define the (circular) convolution of X and W as

$$(X * W)(f) = \int_{-1/2}^{1/2} X(g)W(f - g) dg,$$

where as in Chapter 4 we extend X and W periodically with period 1.

Show that $Z = X * W$, where Z is, of course, the discrete time Fourier transform of \mathbf{z} . Assume that the $X(f)$ and $W(f)$ are “well-behaved” functions, for example, double integrals in which these functions appear can be integrated in any order.

Thus just as in the finite case, convolution in the frequency domain corresponds to a pointwise product in the time domain.

5.9 Let $\mathbf{x} = (\dots, x_{-1}, x_0, x_1, \dots) \in L^2(\mathbb{Z})$ have discrete time Fourier transform

$$X(f) = \sum_{k=-\infty}^{\infty} x_k e^{-2\pi i k f},$$

where $-\frac{1}{2} \leq f \leq \frac{1}{2}$ (recall Definition (4.5.1)). Suppose that we window \mathbf{x} symmetrically about index 0 to construct a vector

$$\tilde{\mathbf{x}} = (\dots, 0, x_{-M}, x_{-M+1}, \dots, x_{M-1}, x_M, 0, \dots)$$

in $L^2(\mathbb{Z})$. Let $\tilde{X}(f)$ denote the discrete time Fourier transform of $\tilde{\mathbf{x}}$.

a. Show that $\tilde{X} = X * W$ where

$$W(f) = \frac{\sin((2M+1)\pi f)}{\sin(\pi f)}.$$

Hint: Use Exercise 5.8 with an appropriate choice for w . Thus windowing a signal in $L^2(\mathbb{Z})$ distorts its spectrum, but in a quantifiable way.

b. Use part (a) and the fact that

$$\lim_{R \rightarrow \infty} \int_{-1/2}^{1/2} \frac{\sin(R\pi x)}{\sin(\pi x)} \phi(x) dx = \phi(0)$$

(if ϕ is continuous at $x = 0$) to show that for any fixed f we have that $\tilde{X}(f)$ converges to $X(f)$ (if X is continuous at f). Thus the distortion disappears in the limit that the window width goes to infinity (as expected).

CHAPTER 6

FILTER BANKS

6.1 OVERVIEW

In previous chapters we encountered various signal models, continuous and discrete, for signals of both finite and infinite duration. We then developed notions of Fourier analysis appropriate to several of these models. Wavelet theory is similar in that it can be approached from both the continuous point of view and the discrete point of view, for signals of either finite or infinite length. As with Fourier analysis, wavelet analysis in the continuous setting involves more technical details and does not allow one to get to the computational applications very quickly. In this chapter we will thus focus on discrete signals only. We'll examine some continuous models in the next chapter.

However, it is easier to first carry out the analysis for signals that are bi-infinite in extent, for example, signals in $L^2(\mathbb{Z})$ or $L^\infty(\mathbb{Z})$ rather than \mathbb{R}^N . In specific computational examples where we have a finite length signal \mathbf{x} we will embed \mathbf{x} in $L^2(\mathbb{Z})$ by zero extension or some other technique, and then apply techniques for bi-infinite signals.

Wavelet analysis is in part motivated by the need to analyze signals whose frequency content varies over the duration of the signal. This is one application where Fourier methods do not perform well, for the basic Fourier waveforms are global in nature. As a result standard Fourier techniques do not really recognize or exploit the fact that the frequency content of a signal or image can vary considerably from one point to another. Wavelet techniques are an alternative to the windowing approach of the last chapter, and they provide an elegant, flexible, computationally efficient

solution to the problem. The ideas have found use in many areas of mathematics, science, and engineering, and remain a very active area of research.

One simple approach to discrete wavelet analysis is through the mechanism of a *filter bank*. A properly designed filter bank naturally gives rise to a discrete wavelet transform (DWT) similar to the discrete Fourier transform. The “discrete wavelets” appear as basic waveforms associated to the DWT, just as the exponential basic waveforms arise from the DFT matrix. In order to get to applications and Matlab experimentation sooner, we defer the topic of filter design for filter banks until Section 6.7. Indeed, if the reader is content to use “off-the-shelf” filters, that section can be omitted, though the mathematics of filter design ties in nicely with the wavelet analysis of Chapter 7.

To build up some intuition, we’ll begin by examining the very simplest kind of filter bank, the Haar filter bank.

6.2 THE HAAR FILTER BANK

One of the key features of the Fourier transform is that it allows us to decompose a signal into constituent frequencies and then deal with the signal one frequency at a time. The filter bank method also adopts this approach by splitting the signal into various frequency bands. In the simplest case, that of a single-stage two-channel filter bank, we merely separate the signal into high and low frequencies by filtering with the methods of Chapter 4.

6.2.1 The One-stage Two-channel Filter Bank

In what follows we’ll consider bi-infinite signals in the space $L^2(\mathbb{Z})$, although many of the ideas extend to signals in $L^\infty(\mathbb{Z})$.

Consider a pair of vectors ℓ and \mathbf{h} in $L^2(\mathbb{Z})$, to be used as low-and high-pass filters for a signal $\mathbf{x} \in L^2(\mathbb{Z})$. The vectors ℓ and \mathbf{h} act by convolution with \mathbf{x} , via equation (4.22). We will assume that ℓ and \mathbf{h} are finite impulse response (FIR) filters as defined in Section 4.5.6. In this case the infinite sum defining the convolution of either filter with $\mathbf{x} \in L^2(\mathbb{Z})$ converges, and in fact both $\mathbf{x} * \ell$ and $\mathbf{x} * \mathbf{h}$ will also be elements of $L^2(\mathbb{Z})$ (recall Exercise 4.29).

For simplicity, let us first consider the familiar two-point averaging and differencing filters with components

$$\begin{aligned}\ell_0 &= \frac{1}{2}, & \ell_1 &= \frac{1}{2}, & \ell_r &= 0, \text{ otherwise,} \\ h_0 &= \frac{1}{2}, & h_1 &= -\frac{1}{2}, & h_r &= 0, \text{ otherwise.}\end{aligned}\tag{6.1}$$

It was shown in Example 4.5 that ℓ acts as a low-pass filter, whereas Exercise 4.34 shows that \mathbf{h} is a high-pass filter. In the context of filter banks these are called the *Haar filters*.

■ EXAMPLE 6.1

In Figure 6.1 the signal in the upper pane is a portion of the vector $\mathbf{x} \in L^2(\mathbb{Z})$ with components

$$x_k = \frac{1}{2} \sin\left(\frac{2\pi \cdot 3k}{128}\right) + \frac{1}{2} \sin\left(\frac{2\pi \cdot 49k}{128}\right)$$

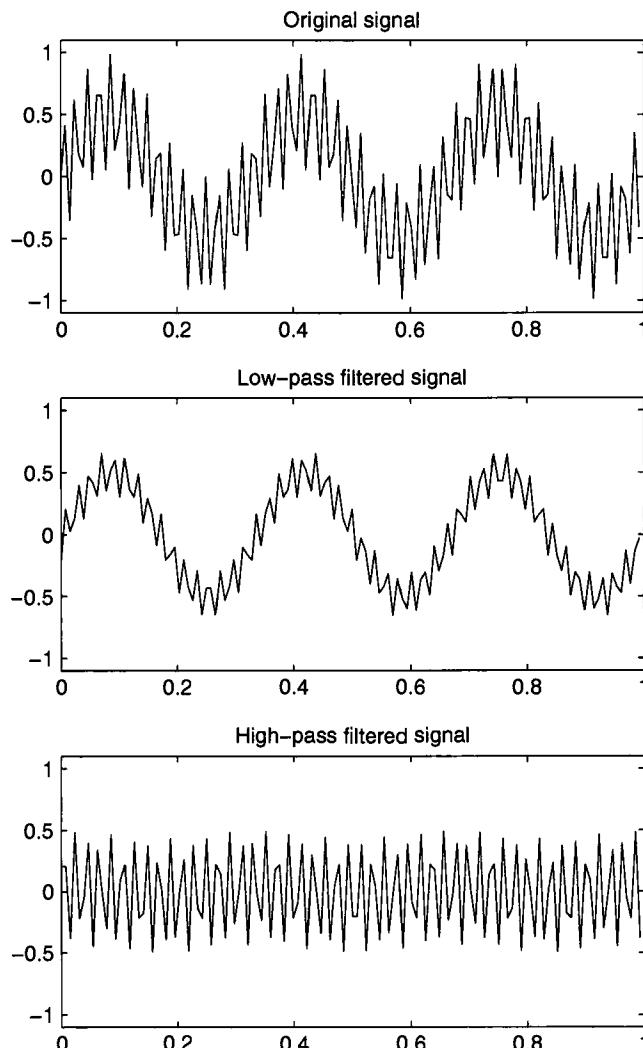


FIGURE 6.1 Original signal with low- and high-pass filtered versions.

for $0 \leq k \leq 128$; that is, we sample $\frac{1}{2} \sin(2\pi \cdot 3t) + \frac{1}{2} \sin(2\pi \cdot 49t)$ at 128 points $t = k/128$ for $0 \leq k \leq 127$. Outside this range we extend \mathbf{x} by zero, although this of no particular importance at the moment.

The lower panes show the nonzero portion of the convolutions $\mathbf{x} * \ell$ and $\mathbf{x} * \mathbf{h}$. The low-pass filtered version $\mathbf{x} * \ell$ is a smoothed approximation to the original signal. It's worth noting that because $\ell + \mathbf{h} = \mathbf{e}$, where $e_0 = 1$ and $e_k = 0$ for all other k , we have $\mathbf{x} = \ell * \mathbf{x} + \mathbf{h} * \mathbf{x}$. We can thus recover \mathbf{x} by adding the low- and high-pass filtered vectors. No information has been lost in the course of filtering.

To continue with our analysis of filter banks, let's look carefully at how an arbitrary vector $\mathbf{x} \in L^2(\mathbb{Z})$ is affected by this process. We will write the vectors in the same column format that we use for vectors in \mathbb{C}^N , as

$$\mathbf{x} = \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}.$$

After low-and high-pass filtering we find

$$\ell * \mathbf{x} = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-2} + x_{-3} \\ x_{-1} + x_{-2} \\ x_0 + x_{-1} \\ x_1 + x_0 \\ x_2 + x_1 \\ \vdots \end{bmatrix}, \quad \mathbf{h} * \mathbf{x} = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-2} - x_{-3} \\ x_{-1} - x_{-2} \\ x_0 - x_{-1} \\ x_1 - x_0 \\ x_2 - x_1 \\ \vdots \end{bmatrix}. \quad (6.2)$$

In general the components are given by $(\ell * \mathbf{x})_k = \frac{1}{2}(x_k + x_{k-1})$ and $(\mathbf{h} * \mathbf{x})_k = \frac{1}{2}(x_k - x_{k-1})$. Moreover the vectors $\ell * \mathbf{x}$ and $\mathbf{h} * \mathbf{x}$ are themselves in the space $L^2(\mathbb{Z})$.

We could actually think of the operation $\mathbf{x} \rightarrow (\ell * \mathbf{x}, \mathbf{h} * \mathbf{x})$ as an invertible transform that converts any input vector $\mathbf{x} \in L^2(\mathbb{Z})$ into a pair of vectors. However, there is considerable redundancy in the process; we don't need all components of both $\ell * \mathbf{x}$ and $\mathbf{h} * \mathbf{x}$ to recover \mathbf{x} .

We can economize by *downsampling*, that is, throwing out every odd-indexed component in the low-pass and high-pass filtered vectors. We will still be able to reconstruct \mathbf{x} . In the present case we obtain from equations (6.2) the vectors \mathbf{X}_ℓ and \mathbf{X}_h defined by

$$\mathbf{X}_\ell := D(\ell * \mathbf{x}) = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-2} + x_{-3} \\ x_0 + x_{-1} \\ x_2 + x_1 \\ x_4 + x_3 \\ \vdots \end{bmatrix}, \quad (6.3)$$

$$\mathbf{X}_h := D(\mathbf{h} * \mathbf{x}) = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-2} - x_{-3} \\ x_0 - x_{-1} \\ x_2 - x_1 \\ x_4 - x_3 \\ \vdots \end{bmatrix}. \quad (6.4)$$

where D is the *downsampling* operator:

Definition 6.2.1 *The downsampling operator $D : L^2(\mathbb{Z}) \rightarrow L^2(\mathbb{Z})$ is defined by*

$$D : (\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots) \rightarrow (\dots, x_{-2}, x_0, x_2, \dots),$$

that is, $(D(\mathbf{x}))_k = x_{2k}$.

Let us define the transform $W(\mathbf{x}) = \mathbf{X}$, where $\mathbf{X} = (\mathbf{X}_\ell, \mathbf{X}_h)$. The transform W is a linear mapping from $L^2(\mathbb{Z})$ to $L^2(\mathbb{Z}) \times L^2(\mathbb{Z})$. The components of \mathbf{X}_ℓ are called the *approximation coefficients*, and the components of \mathbf{X}_h are called the *detail coefficients*. The vector \mathbf{X}_ℓ is a low-pass filtered (hence smoothed) version of \mathbf{x} , but with every other sample omitted. The vector \mathbf{X}_h is similar, but high-pass filtered.

The transform W is represented by the schematic in Figure 6.2 and is referred to as an *analysis filter bank*. In Figure 6.2 we've subscripted the low- and high-pass filters with the letter “a” to indicate that these are the *analysis* filters, used to break the input signal into frequency subbands. The blocks $[\ell_a]$ and $[\mathbf{h}_a]$ in the diagram represent $L^2(\mathbb{Z})$ convolution with the low- and high-pass filters respectively, while the $[\downarrow 2]$ blocks represent the operation of downsampling. We'll refer to the transform W as the *Haar filter bank transform* for now.

As it turns out, there is a simple and elegant method for inverting this transform.

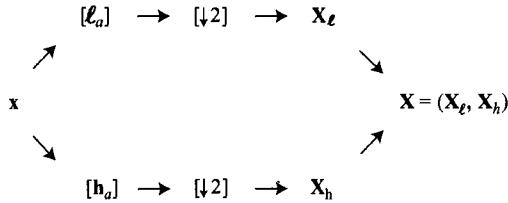


FIGURE 6.2 Schematic of one-stage analysis filter bank.

6.2.2 Inverting the One-stage Transform

The inverse transform can be computed in a structured way that is quite similar to the forward transform. We begin by *upsampling* each piece \mathbf{X}_ℓ and \mathbf{X}_h of the transform \mathbf{X} .

Definition 6.2.2 *The upsampling operator $U : L^2(\mathbb{Z}) \rightarrow L^2(\mathbb{Z})$ is defined by*

$$U : (\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots) \rightarrow (\dots, x_{-2}, 0, x_{-1}, 0, x_0, 0, x_1, 0, x_2, \dots).$$

The components of $U(\mathbf{x})$ are given by

$$(U(\mathbf{x}))_k = \begin{cases} x_{k/2}, & k \text{ even,} \\ 0, & k \text{ odd.} \end{cases}$$

If we upsample the vectors \mathbf{X}_ℓ and \mathbf{X}_h in equations (6.3) and (6.4), we obtain

$$U(\mathbf{X}_\ell) = \frac{1}{2} \begin{bmatrix} \vdots \\ 0 \\ x_{-2} + x_{-3} \\ 0 \\ x_0 + x_{-1} \\ 0 \\ x_2 + x_1 \\ 0 \\ \vdots \end{bmatrix}, \quad U(\mathbf{X}_h) = \frac{1}{2} \begin{bmatrix} \vdots \\ 0 \\ x_{-2} - x_{-3} \\ 0 \\ x_0 - x_{-1} \\ 0 \\ x_2 - x_1 \\ 0 \\ \vdots \end{bmatrix}. \quad (6.5)$$

The next step in inverting the transform is to convolve $U(\mathbf{X}_\ell)$ and $U(\mathbf{X}_h)$ with appropriate *synthesis filters*. In this example we convolve $U(\mathbf{X}_\ell)$ with the low-pass synthesis filter \mathbf{l}_s , which has components $(\mathbf{l}_s)_{-1} = 1, (\mathbf{l}_s)_0 = 1, (\mathbf{l}_s)_k = 0$ for all other indexes k . We convolve $U(\mathbf{X}_h)$ with the high-pass synthesis filter \mathbf{h}_s , which has

components $(\mathbf{h}_s)_{-1} = -1$, $(\mathbf{h}_s)_0 = 1$, and $(\mathbf{h}_s)_k = 0$ for all other k . Note that ℓ_s and \mathbf{h}_s are not causal, but let's not worry about that for the moment. This filtering process yields vectors $\mathbf{v}_\ell = \ell_s * (U(\mathbf{X}_\ell))$ and $\mathbf{v}_h = \mathbf{h}_s * (U(\mathbf{X}_h))$ given by

$$\mathbf{v}_\ell = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-2} + x_{-3} \\ x_{-2} + x_{-3} \\ x_0 + x_{-1} \\ x_0 + x_{-1} \\ x_2 + x_1 \\ x_2 + x_1 \\ x_4 + x_3 \\ \vdots \end{bmatrix}, \quad \mathbf{v}_h = \frac{1}{2} \begin{bmatrix} \vdots \\ x_{-3} - x_{-2} \\ x_{-2} - x_{-3} \\ x_{-1} - x_0 \\ x_0 - x_{-1} \\ x_1 - x_2 \\ x_2 - x_1 \\ x_3 - x_4 \\ \vdots \end{bmatrix}.$$

In general, the k th component of \mathbf{v}_ℓ is $\frac{1}{2}(x_{k-1} + x_k)$ if k is even and $\frac{1}{2}(x_k + x_{k+1})$ if k is odd. The k th component of \mathbf{v}_h is $\frac{1}{2}(x_{k-1} - x_k)$ if k is even and $\frac{1}{2}(x_k - x_{k+1})$ if k is odd. The synthesis filters ℓ_s and \mathbf{h}_s must be chosen to satisfy certain constraints that, not surprisingly, depend on the analysis filters.

Finally, we recover the original vector as $\mathbf{x} = \mathbf{v}_\ell + \mathbf{v}_h$. All operations above are linear. The combination of upsampling and convolution with the synthesis filters is called the *synthesis filter bank*.

The inverse transform/synthesis process can be summarized graphically as in Figure 6.3. In contrast to the analysis filter bank, the downsampling operator is replaced by the upsampling operator $[\uparrow 2]$, followed by convolution with the synthesis filters ℓ_s and \mathbf{h}_s .

Remark 6.1 In the discussion above the synthesis filters are not causal. If causal filters are required, we can simply shift the filter vectors ℓ_s and \mathbf{h}_s to obtain causal filters. Specifically, let us define the shift operator $S : L^2(\mathbb{Z}) \rightarrow L^2(\mathbb{Z})$ as

$$(S(\mathbf{y}))_k = y_{k-1}.$$

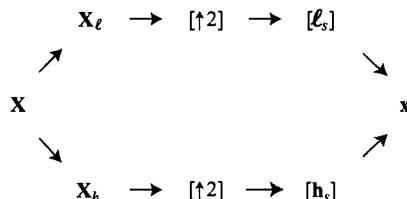


FIGURE 6.3 Schematic of one-stage synthesis filter bank.

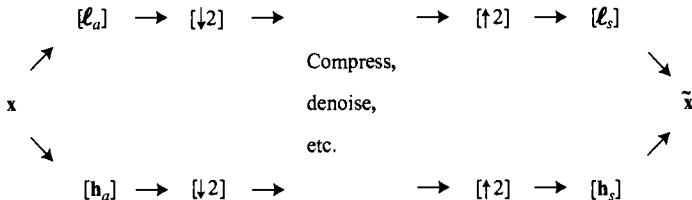


FIGURE 6.4 Analysis/synthesis filter bank.

The S operator shifts y one index to the right, which we can interpret as delaying y one index in time (yes, delaying; e.g., $S(y)$ attains at time index 1 the value y attained one unit earlier, since $(S(y))_1 = y_0$). If we replace the synthesis filters ℓ_s and \mathbf{h}_s by the causal filters $S(\ell_s)$ and $S(\mathbf{h}_s)$ in the computations above, then the output of the synthesis filter bank is $S(\mathbf{x})$, that is, a one-index delayed version of \mathbf{x} . We consider this shifted output a legitimate inversion of the analysis filter bank.

More generally, if \mathbf{g} is any filter, then it's easy to show that $(S^m(\mathbf{g})) * \mathbf{x} = S^m(\mathbf{g} * \mathbf{x})$ for any m and \mathbf{x} ; see Exercise 6.7. Given any noncausal FIR filter vector \mathbf{g} , we can always shift \mathbf{g} so that it gives a causal filter, if we are willing to accept the resulting delay in the output. However, in image processing causality is not a concern.

6.2.3 Summary of Filter Bank Operation

The overall operation of a filter bank in a typical application is illustrated in Figure 6.4. The output $\tilde{\mathbf{x}}$ denotes an altered version (de-noised, compressed, etc.) of the input vector \mathbf{x} . In the event that no compression, de-noising, or other alteration is performed between the analysis and synthesis banks, we would like $\tilde{\mathbf{x}} = S^m(\mathbf{x})$ for some m . This means that the output of the filter bank is a perfect reconstruction of the input, with a possible m -index delay.

The analysis-synthesis filter bank pair is yet another specific instance of the ubiquitous “transform, process, reverse transform” operation in mathematics as was illustrated in Figure 2.1.

■ EXAMPLE 6.2

Before proceeding to more general filter banks, it will be useful to see the Haar filter bank applied to a specific signal. We also examine how this approach might be used in a crude compression application, and why its more local nature can give superior performance over a DFT-based approach.

Let $\mathbf{x} \in L^2(\mathbb{Z})$ denote the sampled piecewise constant signal shown in Figure 6.5 for the index range $0 \leq k \leq 1023$, with $x_k = 0$ outside this range. The signal consists of an oscillatory section followed by large stretches of constant values interspersed with abrupt jumps; the jumps contain a lot of (localized) high-frequency energy. This is just the kind of signal on which a global DFT will perform poorly when used for compression. It is also similar to two-dimensional images, which often have large areas of constant value with abrupt edges.

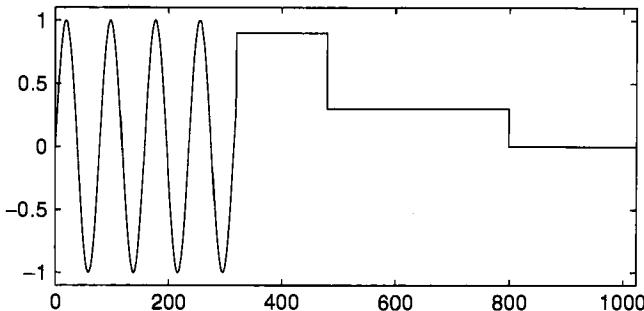


FIGURE 6.5 Signal to be filtered.

We run the signal through a one-stage Haar filter bank with analysis filters defined by equation (6.1), to produce approximation coefficients \mathbf{X}_ℓ and detail coefficients \mathbf{X}_h , both in $L^2(\mathbb{Z})$. Since \mathbf{x} is zero outside a finite range of indexes, so are \mathbf{X}_ℓ and \mathbf{X}_h (Exercise 6.20). In the present case it suffices to display only components indexed in the range $0 \leq k \leq 512$ for \mathbf{X}_ℓ and \mathbf{X}_h . Plots of both are shown in Figure 6.6. The vector \mathbf{X}_ℓ is in the top panel. Note that the beginning of \mathbf{X}_ℓ corresponds to the beginning of the signal \mathbf{x} and the end of \mathbf{X}_ℓ corresponds to the end of \mathbf{x} . The same holds for \mathbf{X}_h in the bottom panel of Figure 6.6. The approximation coefficients strongly resemble the original signal. The detail coefficients

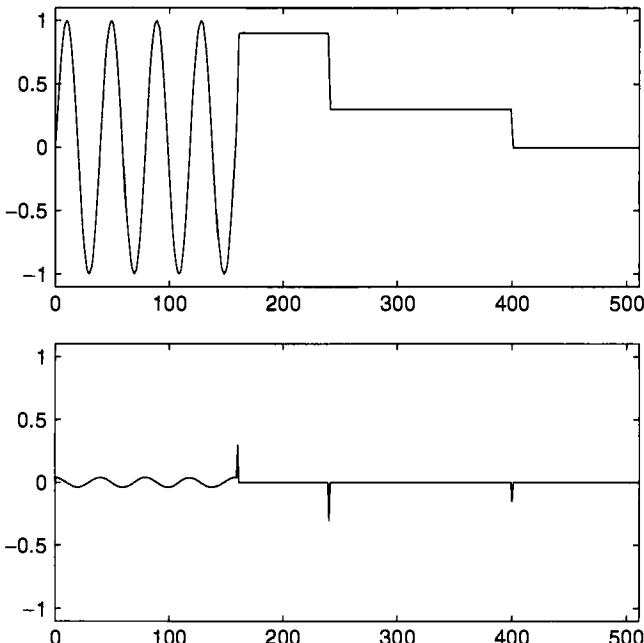


FIGURE 6.6 Approximation (top) and detail (bottom) coefficients for sample signal.

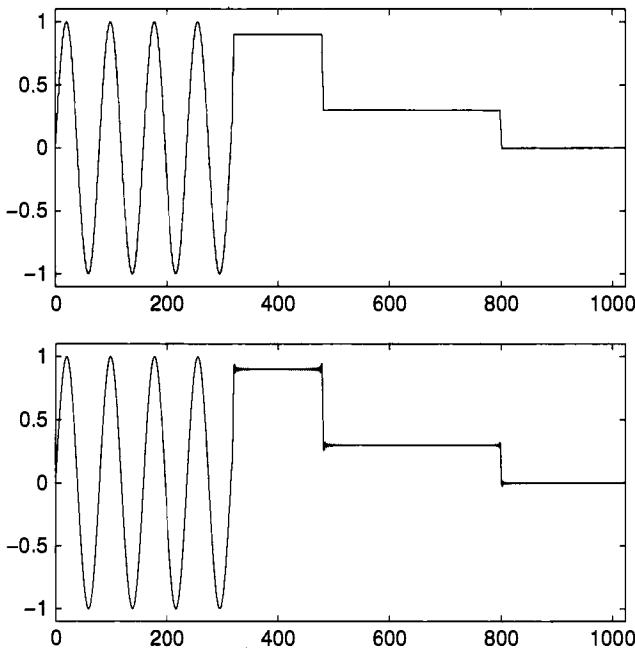


FIGURE 6.7 Signal \mathbf{x} compressed via Haar transform (*top*) and DFT (*bottom*).

are large where the original signal \mathbf{x} is highly oscillatory (during the first third of the signal, and at the jumps) but otherwise relatively small.

We can obtain an “easy” 50% compression of \mathbf{x} by zeroing out all components of \mathbf{X}_h and then inverse transforming with the appropriate synthesis filter bank. The resulting vector $\tilde{\mathbf{x}}$ is a compressed version of \mathbf{x} and is shown in the top panel of Figure 6.7.

Contrast this to the compression obtained using a DFT on the nonzero portion of \mathbf{x} . Specifically, we compute the DFT of \mathbf{x} and zero out the top half of the frequencies (alternatively, a threshold at a level that zeros out half the frequencies; it doesn’t make much difference). We then inverse transform. The resulting vector is shown in the bottom panel of Figure 6.7. The first third of the signal is very local in the frequency domain and compresses well, but the jumps require many frequencies to synthesize. As a result the DFT deals with the jumps less gracefully, and the difficulty is somewhat evident. In a compressed image this phenomenon can be pretty objectionable.

6.3 THE GENERAL ONE-STAGE TWO-CHANNEL FILTER BANK

6.3.1 Formulation for Arbitrary FIR Filters

The procedure above can be generalized beyond the simple two-point averaging filters. For an input signal vector $\mathbf{x} \in L^2(\mathbb{Z})$ we begin with a pair ℓ_a and \mathbf{h}_a of

FIR *analysis filters* (conventionally low-pass/high-pass, but not necessarily so). To compute the transform pair \mathbf{X}_ℓ and \mathbf{X}_h we proceed as illustrated by Figure 6.2. Specifically, we proceed in two steps:

1. Compute $\mathbf{x} * \boldsymbol{\ell}_a$ and $\mathbf{x} * \mathbf{h}_a$.
2. Downsample each filtered signal to obtain

$$\mathbf{X}_\ell = D(\mathbf{x} * \boldsymbol{\ell}_a), \quad \mathbf{X}_h = D(\mathbf{x} * \mathbf{h}_a).$$

Aside from the downsampling, \mathbf{X}_h is just a high-pass filtered version of the original signal. Since the filters are FIR, each component of \mathbf{X}_h is computed from a relatively small number of components of \mathbf{x} . Specifically, $(\mathbf{X}_h)_k$ is computed from x_{2k} and nearby samples (where “nearby” is dictated by the filter length). Similar remarks apply to \mathbf{X}_ℓ . Contrast this to the DFT, where each transform coefficient depends on ALL of the sample values. In this sense the filter bank transform is more local than Fourier-based methods.

To invert the process we essentially just reverse the steps:

1. Upsample each of \mathbf{X}_ℓ and \mathbf{X}_h , to obtain signals $U(\mathbf{X}_\ell)$ and $U(\mathbf{X}_h)$.
2. Compute $\mathbf{v}_\ell = \boldsymbol{\ell}_s * (U(\mathbf{X}_\ell))$ and $\mathbf{v}_h = \mathbf{h}_s * (U(\mathbf{X}_h))$, where $\boldsymbol{\ell}_s$ and \mathbf{h}_s are certain low- and high-pass filters, called the *synthesis filters*.
3. Add the vectors \mathbf{v}_ℓ and \mathbf{v}_h to recover \mathbf{x} , or perhaps $S^m(\mathbf{x})$, the signal \mathbf{x} shifted m indexes as per Remark 6.1 on page 207.

The synthesis filters depend on the analysis filters. There are other hidden constraints too—we can’t take just anything for the analysis filters, for there may not be appropriate matching synthesis filters. And even if there are, the synthesis filters may be too complicated or have too many taps.

Remark 6.2 The scheme above is called a *two-channel* filter bank, since the incoming signal is split into two channels and each is subjected to a different filter, to isolate a different frequency portion of the signal. Generally, we can split a signal into M channels, filter each to extract information in a different frequency range, and then downsample (downsampling each channel here usually consists of keeping only every M th filtered component). This is an example of an M -channel filter bank. We won’t examine such filter banks in this text, but see [24] for more information. This general strategy of splitting a signal into various frequency bands and processing each separately is also known as *subband coding*.

Below we examine a bit more carefully what is required of the analysis/synthesis filter pair in the time domain if the process above is to perfectly reconstruct \mathbf{x} . Later it will be more convenient to look at the filter requirements via the z -transform.

6.3.2 Perfect Reconstruction

The requirement that the filter bank output be a (possibly delayed) copy of the input signal \mathbf{x} can be expressed as

$$S^m(\mathbf{x}) = \ell_s * (U(D(\ell_a * \mathbf{x}))) + \mathbf{h}_s * (U(D(\mathbf{h}_a * \mathbf{x}))). \quad (6.6)$$

We want to choose the various filters so that equation (6.6) holds for all $\mathbf{x} \in L^2(\mathbb{Z})$.

Definition 6.3.1 A “perfect reconstruction” or “biorthogonal” filter bank is one for which analysis followed by synthesis on any signal \mathbf{x} yields $S^m(\mathbf{x})$ for some m .

The terminology “perfect reconstruction” is very sensible; the term “biorthogonal” will be explained later.

Unfortunately, equation (6.6) isn’t particularly helpful when it comes to actually designing filters, even when written out explicitly in summation notation. The equation does, however, make it clear that if one specifies the analysis filters, then the synthesis filters are determined from an infinite set of linear equations in the filter coefficients. Let’s look at a simple example.

■ EXAMPLE 6.3

Let’s consider the case $m = 0$ where the analysis filters have already been chosen according to equation (6.1), the Haar filters. Are the synthesis filters uniquely determined? To find out, we’ll make use of equation (6.6).

Let \mathbf{x} be an arbitrary vector in $L^2(\mathbb{Z})$. The result of filtering, downsampling, and then upsampling \mathbf{x} was worked out in equation (6.5). In general, the vector $\mathbf{v} = U(D(\ell_a * \mathbf{x}))$ has components $v_k = (x_k + x_{k-1})/2$ for k even, and $v_k = 0$ for k odd. Similarly the vector $\mathbf{w} = U(D(\mathbf{h}_a * \mathbf{x}))$ has components $w_k = (x_k - x_{k-1})/2$ for k even, and $w_k = 0$ for k odd.

We convolve \mathbf{v} with the synthesis low-pass filter ℓ_s and convolve \mathbf{w} with the synthesis high-pass filter \mathbf{h}_s ; then we require that the n th component of the sum $\mathbf{v} + \mathbf{w}$ equal x_n , for perfect reconstruction. This yields

$$\frac{1}{2} \sum_{k \text{ even}} ((x_k + x_{k-1})(\ell_s)_{n-k} + (x_k - x_{k-1})(\mathbf{h}_s)_{n-k}) = x_n,$$

since odd components of \mathbf{v} and \mathbf{w} are zero. A little regrouping yields

$$\frac{1}{2} \sum_{k \text{ even}} [((\ell_s)_{n-k} + (\mathbf{h}_s)_{n-k})x_k + ((\ell_s)_{n-k} - (\mathbf{h}_s)_{n-k})x_{k-1}] = x_n, \quad (6.7)$$

which must hold for all $\mathbf{x} \in L^2(\mathbb{Z})$ and all n .

Choose \mathbf{x} to have components $x_0 = 1$ and $x_k = 0$ for $k \neq 0$. Equation (6.7) yields (only the $k = 0$ term contributes)

$$(\ell_s)_n + (\mathbf{h}_s)_n = \begin{cases} 2, & n = 0, \\ 0, & n \neq 0. \end{cases} \quad (6.8)$$

Next choose \mathbf{x} to have components $x_1 = 1$ and $x_k = 0$ for $k \neq 1$. In this case equation (6.7) yields (only $k = 2$ contributes here)

$$(\ell_s)_{n-2} - (\mathbf{h}_s)_{n-2} = \begin{cases} 2, & n = 1, \\ 0, & n \neq 1, \end{cases} \quad (6.9)$$

for all n . From equation (6.8) with $n = 0$ and (6.9) with $n = 2$, we obtain a pair of linear equations $(\ell_s)_0 + (\mathbf{h}_s)_0 = 2$ and $(\ell_s)_0 - (\mathbf{h}_s)_0 = 0$ for $(\ell_s)_0$ and $(\mathbf{h}_s)_0$, from which we can deduce that $(\ell_s)_0 = (\mathbf{h}_s)_0 = 1$. Choosing $n = -1$ in (6.8) and $n = 1$ in (6.9) leads to $(\ell_s)_{-1} = 1$ and $(\mathbf{h}_s)_{-1} = -1$. By considering $n = n_0$ in equation (6.8) and $n = n_0 + 2$ in equation (6.9) for $n_0 \neq -1, 0$ it can be shown that all other synthesis filter coefficients are zero. These are, of course, the synthesis filters we used in Section 6.2.2. We now know they are the only filters that will work with the chosen analysis filters (for zero output delay).

■ EXAMPLE 6.4

Another set of biorthogonal filters is the Le Gall 5/3 filters, with coefficients

$$\begin{aligned} \ell_a &= (\dots, 0, -\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}, 0, \dots), & \mathbf{h}_a &= (\dots, 0, -\frac{1}{2}, 1, -\frac{1}{2}, 0, \dots), \\ \ell_s &= (\dots, 0, \frac{1}{2}, 1, \frac{1}{2}, 0, \dots), & \mathbf{h}_s &= (\dots, 0, -\frac{1}{8}, -\frac{1}{4}, \frac{3}{4}, -\frac{1}{4}, -\frac{1}{8}, 0, \dots), \end{aligned}$$

used in the JPEG 2000 compression standard. The derivation of these filters is given in Section 6.7, where we look at a variety of techniques for designing filters.

6.3.3 Orthogonal Filter Banks

As we'll see, there are infinitely many analysis/synthesis FIR filters that satisfy equation (6.6). It's somewhat helpful to narrow the types of filters under consideration by imposing additional constraints. For example, in the simple Haar two-point averaging/differencing filters above the synthesis filters were merely time-reversed and rescaled versions of the analysis filters. More specifically, $(\ell_s)_k = C(\ell_a)_{-k}$ and $(\mathbf{h}_s)_k = C(\mathbf{h}_a)_{-k}$ for some constant C . This turns out to be a common and useful arrangement, and in fact by properly rescaling, we can also arrange $C = 1$. In this case the analysis and synthesis filters satisfy

$$(\ell_s)_k = (\ell_a)_{-k}, \quad (\mathbf{h}_s)_k = (\mathbf{h}_a)_{-k}. \quad (6.10)$$

Filters in this relation are said to be *adjoints* or *time-reversals* of each other (previously discussed in Exercise 4.9). The resulting filter bank is said to be *orthogonal*, for reasons that will be explained in Section 6.5.3. In this case if the analysis filters are causal, then the synthesis filters cannot be, but by Remark 6.1 on page 207, if desired, we can always shift the synthesis filters (by the same amount) to obtain causality at the expense of introducing a delay in the filter bank output. In this case equations (6.10) won't hold, but we'll still call the filter bank "orthogonal."

■ EXAMPLE 6.5

A simple rescaling of the Haar filters above yields an orthogonal filter bank. Specifically, multiply both analysis filters by $\sqrt{2}$ and compensate by dividing the synthesis filters by $\sqrt{2}$. We obtain filters with nonzero coefficients $(\ell_a)_0 = 1/\sqrt{2}$, $(\ell_a)_1 = 1/\sqrt{2}$ and $(\mathbf{h}_a)_0 = 1/\sqrt{2}$, $(\mathbf{h}_a)_1 = -1/\sqrt{2}$. The synthesis filters become $(\ell_s)_0 = 1/\sqrt{2}$, $(\ell_s)_{-1} = 1/\sqrt{2}$ and $(\mathbf{h}_s)_0 = 1/\sqrt{2}$, $(\mathbf{h}_s)_{-1} = -1/\sqrt{2}$, merely the time reversals of the analysis filters. All other filter coefficients are zero.

6.4 MULTISTAGE FILTER BANKS

It can be extremely useful to iterate or nest filter banks. For example, if an input vector \mathbf{x} is passed through an analysis filter bank to produce filtered and downsampled vectors $\mathbf{X}_\ell = D(\ell_a * \mathbf{x})$ and $\mathbf{X}_h = D(\mathbf{h}_a * \mathbf{x})$, it can be useful to pass \mathbf{X}_ℓ through another filter bank to further separate low- and high-pass components. Or we might choose to pass both \mathbf{X}_ℓ and \mathbf{X}_h through another filter bank, or even just \mathbf{X}_h . A couple of possibilities are illustrated in Figures 6.8 and 6.9.

The subscripts indicate the order in which the filtering operations are applied; for example, $\mathbf{X}_{\ell h}$ means the low-pass filtering is done first and then the high-pass filtering. In other words, $\mathbf{X}_{\ell h} = (\mathbf{X}_\ell)_h$. The filter bank in Figure 6.8 and others similarly obtained by successively refining the low-pass output are the filter banks usually associated to wavelets. This is the case that we will focus on for the rest of the chapter. The filter bank of Figure 6.9 is the two-stage full filter bank tree, that leads to

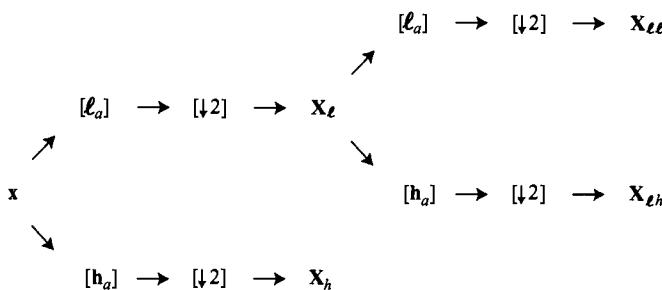


FIGURE 6.8 Filter bank possibility (leads to wavelets).

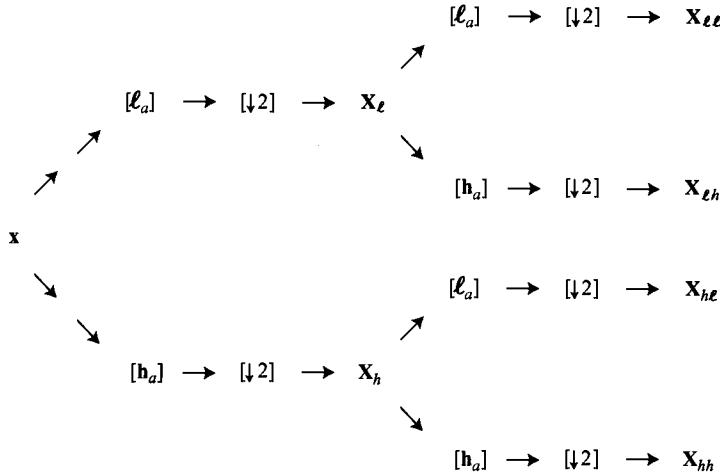


FIGURE 6.9 Full filter bank tree.

the theory of wavelet packets. We do not pursue this here, though it is implemented in Matlab. It is possible to use different filter pairs at each stage in a filter bank, although this is not common.

Let us consider more carefully the process of filtering a signal using the structure of the first two-stage filter bank above, in which we repeatedly put the low-pass portion through a filter bank. The two-stage filter bank operation might also be illustrated as below:

$$\mathbf{x} \rightarrow \begin{bmatrix} \mathbf{X}_\ell \\ \mathbf{X}_h \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{X}_{\ell\ell} \\ \mathbf{X}_{\ell h} \\ \mathbf{X}_h \end{bmatrix}.$$

Each of \mathbf{X}_ℓ , \mathbf{X}_h , $\mathbf{X}_{\ell\ell}$, $\mathbf{X}_{\ell h}$, and \mathbf{X}_h is an element of the same space, $L^2(\mathbb{Z})$, as \mathbf{x} .

Of course, we can iterate again, by passing $\mathbf{X}_{\ell\ell}$ through the analysis bank to produce vectors $\mathbf{X}_{\ell\ell\ell}$ and $\mathbf{X}_{\ell\ell h}$. The process can be continued in an obvious manner, repeatedly passing the $\mathbf{X}_{\ell\cdots\ell}$ sub-vector through the filter bank to produce $\mathbf{X}_{\ell\cdots\ell\ell}$ and $\mathbf{X}_{\ell\cdots\ell h}$. For example, if we use four stages, the net result can be written in the form

$$\begin{bmatrix} \mathbf{X}_{\ell\ell\ell\ell} \\ \mathbf{X}_{\ell\ell\ell h} \\ \mathbf{X}_{\ell\ell h h} \\ \mathbf{X}_{\ell h} \\ \mathbf{X}_h \end{bmatrix}.$$

It's clear how this multistage analysis filter bank can be inverted. We upsample and apply the appropriate synthesis filters to $\mathbf{X}_{\ell\ell\ell\ell}$ and $\mathbf{X}_{\ell\ell\ell h}$, to produce $\mathbf{X}_{\ell\ell\ell}$. We repeat with $\mathbf{X}_{\ell\ell\ell}$ and $\mathbf{X}_{\ell\ell h}$ to produce $\mathbf{X}_{\ell\ell}$, and then use $\mathbf{X}_{\ell\ell}$ and $\mathbf{X}_{\ell h}$ to synthesize \mathbf{X}_ℓ . Finally, we synthesize the output of the filter bank with \mathbf{X}_ℓ and \mathbf{X}_h .

■ EXAMPLE 6.6

Let $\mathbf{x} \in L^2(\mathbb{Z})$ be the signal in the upper left of Figure 6.5, considered in Example 6.2. We pass \mathbf{x} through a three-stage orthogonal Haar filter bank with analysis filters $\ell = (\dots, 0, 1/\sqrt{2}, 1/\sqrt{2}, 0, \dots)$ and $\mathbf{h} = (\dots, 0, 1/\sqrt{2}, -1/\sqrt{2}, 0, \dots)$ as computed in Example 6.5. The quantity \mathbf{X}_h was shown in Figure 6.6. In Figure 6.10 we show $\mathbf{X}_{\ell h}$ in the top panel and $\mathbf{X}_{\ell \ell}$ in the bottom panel. In each case only the nonzero components of the relevant signal are shown, so $\mathbf{X}_{\ell h}$ and $\mathbf{X}_{\ell \ell}$ span index range 0 to 256.

For the three-stage transform, \mathbf{X}_h and $\mathbf{X}_{\ell h}$ remain unchanged, while $\mathbf{X}_{\ell \ell}$ is passed through the filter bank to produce $\mathbf{X}_{\ell \ell h}$ and $\mathbf{X}_{\ell \ell \ell}$. The latter two vectors are shown in Figure 6.11. The nonzero components span index range 0 to 128.

At each stage the signal $\mathbf{X}_{\ell \dots \ell}$ somewhat resembles the original signal but is low-pass filtered and downsampled. After the one-stage transform the high-frequency portion \mathbf{X}_h is nonzero only in those parts of the signal where high-frequency energy is present, such as the sinusoidal portion at the start and the two discontinuities. To a large extent the same observation applies to $\mathbf{X}_{\ell h}$ and even $\mathbf{X}_{\ell \ell h}$.

Let's examine what happens if we compress the signal by "zeroing out" selective high-frequency portions of the three-stage filter bank output and then applying the appropriate synthesis filters. For example, if we zero out \mathbf{X}_h and then re-synthesize the signal with the remaining components (by re-synthesizing $\mathbf{X}_{\ell \ell h}$ and $\mathbf{X}_{\ell \ell \ell}$ into $\mathbf{X}_{\ell h}$, then $\mathbf{X}_{\ell h}$ and $\mathbf{X}_{\ell \ell}$ to \mathbf{X}_{ℓ} , and finally use \mathbf{X}_{ℓ} with $\mathbf{X}_h = \mathbf{0}$ to synthesize the output), we obtain the signal that was shown in the top panel in Figure 6.7. The upper panel on the right in Figure 6.12 shows the reconstruction from $\mathbf{X}_{\ell \ell}$ alone

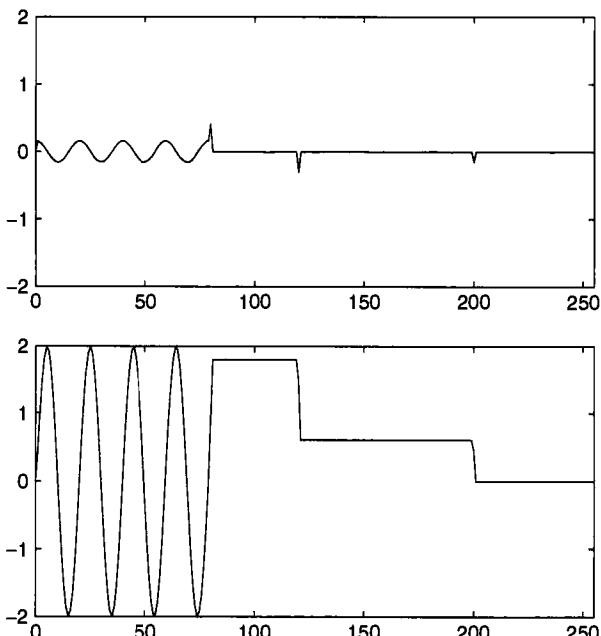


FIGURE 6.10 Quantities $\mathbf{X}_{\ell h}$ (top) and $\mathbf{X}_{\ell \ell}$ (bottom) of two-stage Haar transform.

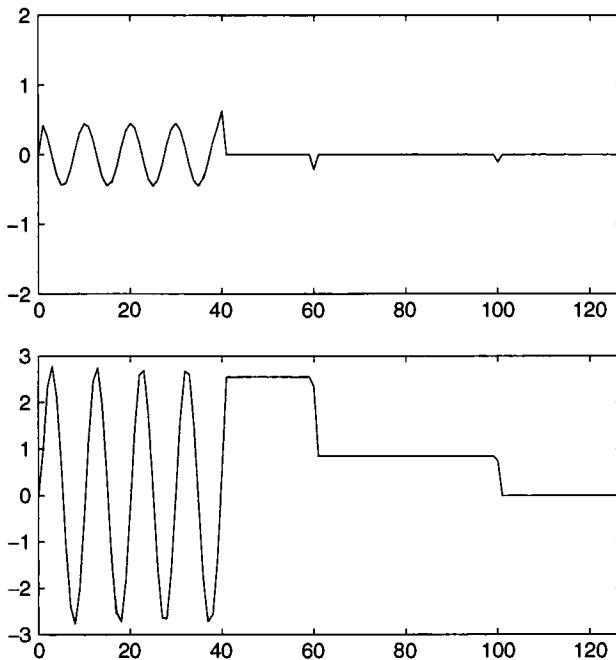


FIGURE 6.11 Portions \mathbf{X}_{eeh} (top) and \mathbf{X}_{eee} (bottom) of three-stage Haar transform.

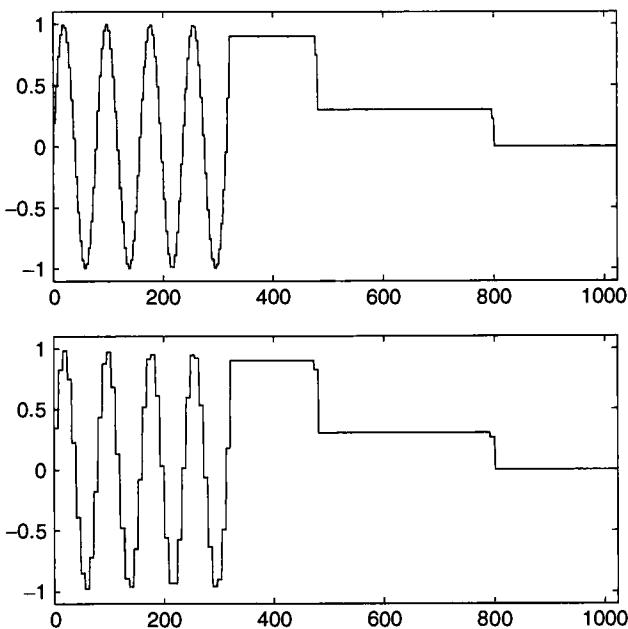


FIGURE 6.12 Reconstruction of \mathbf{x} using only \mathbf{X}_{ee} (top) and \mathbf{X}_{eee} (bottom).

(implicitly setting \mathbf{X}_h and $\mathbf{X}_{\ell h}$ to zero). The vector $\mathbf{X}_{\ell \ell}$ has at most 257 nonzero coefficients. Finally, the lower panel in Figure 6.12 shows the reconstruction from $\mathbf{X}_{\ell \ell \ell}$ alone (at most 129 nonzero coefficients).

6.5 FILTER BANKS FOR FINITE LENGTH SIGNALS

6.5.1 Extension Strategy

The theory developed thus far is for signals of infinite length, for example, signals in $L^2(\mathbb{Z})$. Signals of finite length present a certain difficulty that we must overcome: the convolution of a filter $\mathbf{g} \in L^2(\mathbb{Z})$ and a finite length signal $\mathbf{x} \in \mathbb{C}^N$ is undefined. The obvious fix is to extend \mathbf{x} to be an element of $L^2(\mathbb{Z})$, and then convolve. This is exactly what we'll do.

Specifically, we use the following conceptual approach: we first extend \mathbf{x} to $L^2(\mathbb{Z})$ in some manner, filter the extended signal, and then truncate the filtered signal to some finite length. We call this a “conceptual approach,” since we won’t actually perform numerical computations on signals in $L^2(\mathbb{Z})$.

There are many ways to extend a signal $\mathbf{x} \in \mathbb{C}^N$ to a signal $\tilde{\mathbf{x}} \in L^2(\mathbb{Z})$. Some common choices are listed below.

1. We can extend \mathbf{x} by zero-padding; that is, set $\tilde{x}_k = x_k$ for $0 \leq k \leq N - 1$ and $\tilde{x}_k = 0$ for k outside this range.
2. We can extend \mathbf{x} periodically, by setting $\tilde{x}_k = x_{k \bmod N}$. But note that if we do this for all k , then $\tilde{\mathbf{x}}$ won’t be in $L^2(\mathbb{Z})$. In fact, since we’re working with FIR filters, we’ll only need to extend \mathbf{x} a few cycles in either direction, and then we can extend by zero. In this case the extension is in $L^2(\mathbb{Z})$. As it turns out, this results in a simple modification of the $L^2(\mathbb{Z})$ theory in which all convolutions in $L^2(\mathbb{Z})$ are replaced by circular convolutions in \mathbb{C}^N . We’ll look at this more closely below.
3. Periodic extension as in scheme 2 above has caused difficulty in using the DFT for compression, since this extension introduces artificial discontinuities at the signal edges. The same thing can happen here. For most analyses it’s better to extend \mathbf{x} by even reflection as in Chapter 3, first to \mathbb{C}^{2N} as

$$\tilde{x}_k = \begin{cases} x_k, & 0 \leq k \leq N - 1, \\ x_{2N-k-1}, & N \leq k \leq 2N - 1, \end{cases} \quad (6.11)$$

and then via $\tilde{x}_k = \tilde{x}_{k \bmod 2N}$. This type of extension duplicates the boundary points x_0 and x_{N-1} , so the extended version $\tilde{\mathbf{x}}$ looks like

$$\dots, x_3, x_2, x_1, x_0, x_0, x_1, \dots, x_{N-1}, x_{N-1}, x_{N-2}, \dots$$

and is periodic with period $2N$. In conjunction with the DFT this is the extension that led to the discrete cosine transform. This type of extension is called a *half-point symmetric* extension.

Alternatively, we may extend \mathbf{x} to \mathbb{C}^{2N-2} as

$$\tilde{x}_k = \begin{cases} x_k, & 0 \leq k \leq N-1, \\ x_{2N-k-2}, & N \leq k \leq 2N-3, \end{cases} \quad (6.12)$$

and then via $\tilde{x}_k = \tilde{x}_{k \bmod 2N-2}$. This type of extension does not duplicate the boundary points x_0 and x_{N-1} , so the extended version $\tilde{\mathbf{x}}$ looks like

$$\dots, x_3, x_2, x_1, x_0, x_1, \dots, x_{N-2}, x_{N-1}, x_{N-2}, \dots$$

and is periodic with period $2N-2$. This is called a *whole-point symmetric* extension.

As with simple periodic extension, whatever method we use only requires that we extend a few cycles in each direction. We can then extend by zero and obtain an extension in $L^2(\mathbb{Z})$.

There are other methods for extending signals (many built into Matlab's wavelet commands), but we won't concern ourselves with them here.

After extending the signal \mathbf{x} to $\tilde{\mathbf{x}}$ and filtering, we obtain a vector $\mathbf{y} = \tilde{\mathbf{x}} * \mathbf{g}$ in $L^2(\mathbb{Z})$. We then truncate \mathbf{y} to a finite length vector. It's not clear how this last step should be done, or how it will affect the theory we've developed thus far. Thus to better understand this approach, let's take a closer look at extension approach (2) above to see how the computations proceed, and obtain a "discrete filter bank transform" for signals in \mathbb{R}^N .

6.5.2 Analysis of Periodic Extension

Recall the overall operation of a filter bank for bi-infinite signals: A signal $\mathbf{x} \in L^2(\mathbb{Z})$ is filtered with analysis low-pass and high-pass filters, and then downsampled to produce vectors $\mathbf{X}_\ell = D(\mathbf{x} * \ell_a)$ and $\mathbf{X}_h = D(\mathbf{x} * \mathbf{h}_a)$. To invert the process, we upsample, convolve with appropriate synthesis filters, and add. With perfect reconstruction we have $U(\mathbf{X}_\ell) * \ell_s + U(\mathbf{X}_h) * \mathbf{h}_s = S^m(\mathbf{x})$ for some m .

The punchline of the analysis in this section is that this same procedure works for signals of finite length provided that the filters involved are part of a perfect reconstruction filter bank for signals in $L^2(\mathbb{Z})$. Specifically, we have

Theorem 6.5.1 *Let $\mathbf{x} \in \mathbb{C}^N$. Let ℓ_a , \mathbf{h}_a , ℓ_s , and \mathbf{h}_s be the analysis and synthesis FIR filters for a perfect reconstruction filter bank, each with N or fewer taps. If we interpret these filters as elements of \mathbb{C}^N by taking all indexes modulo N and all convolutions as circular convolutions, then the linear transform $\mathbf{x} \rightarrow \mathbf{X} := (\mathbf{X}_\ell, \mathbf{X}_h)$ from \mathbb{C}^N to \mathbb{C}^N is invertible. The inverse is given by the mapping $\mathbf{X} \rightarrow U(\mathbf{X}_\ell) * \ell_s + U(\mathbf{X}_h) * \mathbf{h}_s$, with all convolutions as circular convolutions in \mathbb{C}^N .*

The next couple subsections explain why this works.

Adapting the Analysis Transform to Finite Length Given a signal $\mathbf{x} \in \mathbb{C}^N$, we begin by extending to a signal $\tilde{\mathbf{x}} \in L^2(\mathbb{Z})$ according to scheme 2 above. Specifically, suppose that we set

$$\tilde{x}_k = x_{k \bmod N}$$

in the range $-qN \leq k \leq qN$ for some “large” integer q (thus extending \mathbf{x} periodically with period N , to at least $2q$ cycles); we set $\tilde{x}_k = 0$ for k outside this range. We can take q as large as we like, so for all practical purposes \tilde{x}_k is periodic in k for “all” k .

In what follows we assume that N is even. We assume that all filters involved have N or fewer taps, but no other special properties. We’ll assume that the filters (elements of $L^2(\mathbb{Z})$) have nonzero coefficients indexed in the range $-N/2$ to $N/2$, though this is primarily for convenience. Because we’ll ultimately interpret the filters as elements of \mathbb{C}^N by considering indices modulo N , the range doesn’t matter. In the periodic case the notion of “causal filter” isn’t really meaningful.

We begin by examining the action of the filter bank on the signal $\tilde{\mathbf{x}} \in L^2(\mathbb{Z})$. Let $\mathbf{y} = \tilde{\mathbf{x}} * \ell_a \in L^2(\mathbb{Z})$ so that

$$y_m = \sum_{k=-N/2}^{N/2} \tilde{x}_{m-k} (\ell_a)_k.$$

The component y_m depends on \tilde{x}_j only for the range $m - N/2 \leq j \leq m + N/2$. It’s easy to see that y_m is periodic in the index m with period N , that is,

$$y_m = y_{m+N}$$

for any integer m sufficiently close to 0, such as $-(q-1)N \leq m \leq (q-1)N$. The reason is that

$$y_{m+N} = \sum_{k=-N/2}^{N/2} \tilde{x}_{m+N-k} (\ell_a)_k = \sum_{k=-N/2}^{N/2} \tilde{x}_{m-k} (\ell_a)_k = y_m,$$

since $\tilde{x}_{m+N-k} = \tilde{x}_{m-k}$ (because \tilde{x}_j is periodic with period N). Thus knowledge of y_m for $0 \leq m \leq N-1$ determines y_m in the larger range $-(q-1)N \leq m \leq (q-1)N$.

Indeed, by Definition 4.2.1, it’s clear that the vector $(y_0, y_1, \dots, y_{N-1})$ is just the circular convolution of \mathbf{x} and ℓ_a as vectors in \mathbb{C}^N , with all indexes interpreted modulo N .

The next step in the filter bank is downsampling. Let

$$\tilde{\mathbf{X}}_\ell = D(\tilde{\mathbf{x}} * \ell_a),$$

where D is the downsampling operator on $L^2(\mathbb{Z})$. It’s easy to see that $(\tilde{\mathbf{X}}_\ell)_k$ is periodic in the index k with period $N/2$, for at least $q-1$ cycles in either direction from $k=0$. Thus knowledge of $(\tilde{\mathbf{X}}_\ell)_k$ for the range $0 \leq k \leq N/2-1$ determines $(\tilde{\mathbf{X}}_\ell)_k$ on the larger range $-(q-1)N/2 \leq k \leq (q-1)N/2$.

Similar remarks apply to the high-pass analysis filter \mathbf{h}_a , and we can produce the analogous quantity

$$\tilde{\mathbf{X}}_h = D(\tilde{\mathbf{x}} * \mathbf{h}_a).$$

Define truncated versions \mathbf{X}_ℓ and \mathbf{X}_h of the downsampled filtered signals as

$$(\mathbf{X}_\ell)_k = (\tilde{\mathbf{X}}_\ell)_k, \quad (\mathbf{X}_h)_k = (\tilde{\mathbf{X}}_h)_k,$$

for $0 \leq k \leq N/2 - 1$, so both \mathbf{X}_ℓ and \mathbf{X}_h are elements of $\mathbb{C}^{N/2}$. Concatenate these vectors by defining $\mathbf{X} = (\mathbf{X}_\ell, \mathbf{X}_h) \in \mathbb{C}^N$. We will refer to the mapping

$$\mathbf{x} \rightarrow \mathbf{X} \tag{6.13}$$

obtained via this procedure as a *discrete wavelet transform*, abbreviated “DWT.” This transform is LINEAR (because the extension procedure, convolution, and truncation are all linear). As such, the mapping in (6.13) is represented by an $N \times N$ matrix.

■ EXAMPLE 6.7

Let us consider signals $\mathbf{x} = (x_0, x_1, x_2, x_3) \in \mathbb{C}^4$, with the orthogonal Haar filter bank ($\ell_a = (1/\sqrt{2}, 1/\sqrt{2}, 0, 0)$, $\mathbf{h}_a = (1/\sqrt{2}, -1/\sqrt{2}, 0, 0)$) constructed in Examples 6.5 and 6.15. The extension $\tilde{\mathbf{x}}$ looks like

$$\tilde{\mathbf{x}} = (\dots, x_0, x_1, x_2, x_3, x_0, x_1, x_2, x_3, x_0, x_1, x_2, x_3, \dots)$$

with period 4, at least for some distance. The convolution $\tilde{\mathbf{x}} * \ell_a$ is periodic with period 4 and looks like

$$\left(\dots, \frac{x_0}{\sqrt{2}} + \frac{x_3}{\sqrt{2}}, \frac{x_1}{\sqrt{2}} + \frac{x_0}{2}, \frac{x_2}{\sqrt{2}} + \frac{x_1}{\sqrt{2}}, \frac{x_3}{\sqrt{2}} + \frac{x_2}{\sqrt{2}}, \dots \right).$$

Downsampling this and then truncating (take only the first $N/2 = 2$ components) produces

$$\mathbf{X}_\ell = \left(\frac{x_0 + x_3}{\sqrt{2}}, \frac{x_2 + x_1}{\sqrt{2}} \right).$$

Similar computations show that

$$\mathbf{X}_h = \left(\frac{x_0 - x_3}{\sqrt{2}}, \frac{x_2 - x_1}{\sqrt{2}} \right).$$

All in all, the DWT looks like

$$(x_0, x_1, x_2, x_3) \rightarrow \left(\frac{x_0 + x_3}{\sqrt{2}}, \frac{x_2 + x_1}{\sqrt{2}}, \frac{x_0 - x_3}{\sqrt{2}}, \frac{x_2 - x_1}{\sqrt{2}} \right).$$

This linear transform is embodied by the matrix

$$\mathbf{W}_4^a = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}. \quad (6.14)$$

The \mathbf{W} is for “wavelet,” the 4 for the size of the transform, and a for “analysis.”

Adapting the Synthesis Transform to Finite Length To invert the discrete wavelet transform, we proceed as follows. First, recall that we already know how to invert the transform for signals in $L^2(\mathbb{Z})$. Specifically, given the vectors $\tilde{\mathbf{X}}_\ell$ and $\tilde{\mathbf{X}}_h$ in $L^2(\mathbb{Z})$, we upsample each and convolve with the synthesis filters, to obtain vectors $\tilde{\mathbf{v}}_\ell = \mathbf{\ell}_s * U(\tilde{\mathbf{X}}_\ell)$ and $\tilde{\mathbf{v}}_h = \mathbf{h}_s * U(\tilde{\mathbf{X}}_h)$ in $L^2(\mathbb{Z})$ with components

$$(\tilde{\mathbf{v}}_\ell)_m = \sum_{k=-N/2}^{N/2} U(\tilde{\mathbf{X}}_\ell)_{m-k} (\mathbf{\ell}_s)_k, \quad (\tilde{\mathbf{v}}_h)_m = \sum_{k=-N/2}^{N/2} U(\tilde{\mathbf{X}}_h)_{m-k} (\mathbf{h}_s)_k. \quad (6.15)$$

We then recover $\tilde{\mathbf{x}} = \tilde{\mathbf{v}}_\ell + \tilde{\mathbf{v}}_h$. Since $x_m = \tilde{x}_m$ for $0 \leq m \leq N - 1$, we also recover the vector \mathbf{x} . However, in the finite length case we have only the truncated versions \mathbf{X}_ℓ and \mathbf{X}_h in $\mathbb{C}^{N/2}$ to work with, not the full $L^2(\mathbb{Z})$ versions $\tilde{\mathbf{X}}_\ell$ and $\tilde{\mathbf{X}}_h$.

But within the range $0 \leq m \leq N - 1$ equations (6.15) only require knowledge of $U(\tilde{\mathbf{X}}_\ell)_j$ and $U(\tilde{\mathbf{X}}_h)_j$ on at most the range $-N/2 \leq j \leq 3N/2$. Since both $\tilde{\mathbf{X}}_\ell$ and $\tilde{\mathbf{X}}_h$ are periodic with period $N/2$ on the index range $-(q-1)N/2 \leq k \leq (q-1)N/2$ for some “large” q , knowledge of the truncated versions \mathbf{X}_ℓ and \mathbf{X}_h allows us to reconstruct $\tilde{\mathbf{X}}_\ell$ and $\tilde{\mathbf{X}}_h$ on this range. As a result we can reconstruct $U(\tilde{\mathbf{X}}_\ell)$ and $U(\tilde{\mathbf{X}}_h)$ (both period N) on the index range $-(q-1)N \leq k \leq (q-1)N$ from knowledge of $U(\mathbf{X}_\ell)$ and $U(\mathbf{X}_h)$, by extending the latter two vectors periodically with period N . Indeed both $U(\tilde{\mathbf{X}}_\ell)_j$ and $U(\tilde{\mathbf{X}}_h)_j$ are just the truncated versions $U(\mathbf{X}_\ell)_j$ and $U(\mathbf{X}_h)_j$ with indexes interpreted modulo N .

With these observations equations (6.15) can be written

$$(\tilde{\mathbf{v}}_\ell)_m = \sum_{k=-N/2}^{N/2} U(\mathbf{X}_\ell)_{m-k} (\mathbf{\ell}_s)_k, \quad (\tilde{\mathbf{v}}_h)_m = \sum_{k=-N/2}^{N/2} U(\mathbf{X}_h)_{m-k} (\mathbf{h}_s)_k \quad (6.16)$$

on (at least) the index range $0 \leq m \leq N - 1$. In particular, define truncated vectors \mathbf{v}_ℓ and \mathbf{v}_h in \mathbb{C}^N as

$$\mathbf{v}_\ell = ((\tilde{\mathbf{v}}_\ell)_0, (\tilde{\mathbf{v}}_\ell)_1, \dots, (\tilde{\mathbf{v}}_\ell)_{N-1}),$$

$$\mathbf{v}_h = ((\tilde{\mathbf{v}}_h)_0, (\tilde{\mathbf{v}}_h)_1, \dots, (\tilde{\mathbf{v}}_h)_{N-1}).$$

Equations (6.16) are just

$$\mathbf{v}_\ell = U(\mathbf{X}_\ell) * \mathbf{\ell}_s, \quad \mathbf{v}_h = U(\mathbf{X}_h) * \mathbf{h}_s,$$

where “*” denotes circular convolution in \mathbb{C}^N . But $\mathbf{v}_\ell + \mathbf{v}_h = \mathbf{x}$, and thus the linear operation $\mathbf{X} \rightarrow U(\mathbf{X}_\ell) * \mathbf{\ell}_s + U(\mathbf{X}_h) * \mathbf{h}_s$ inverts the original transform.

The discussion above is a proof of Theorem 6.5.1.

■ EXAMPLE 6.8

Let's work out the inverse transform corresponding to Example 6.7. Let $\mathbf{X} = (X_0, X_1, X_2, X_3) \in \mathbb{C}^4$ so that $\mathbf{X}_\ell = (X_0, X_1)$ and $\mathbf{X}_h = (X_2, X_3)$. Recall that the orthogonal Haar synthesis filters in $L^2(\mathbb{Z})$ had coefficients $(\mathbf{\ell}_s)_0 = 1/\sqrt{2}$, $(\mathbf{\ell}_s)_{-1} = 1/\sqrt{2}$, $(\mathbf{h}_s)_0 = 1/\sqrt{2}$, and $(\mathbf{h}_s)_{-1} = -1/\sqrt{2}$. However, if we interpret these as vectors in \mathbb{C}^4 by considering indexes modulo 4 we have

$$\mathbf{\ell}_s = \left(\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}} \right), \quad \mathbf{h}_s = \left(\frac{1}{\sqrt{2}}, 0, 0, \frac{-1}{\sqrt{2}} \right).$$

Upsampling produces the vectors $U(\mathbf{X}_\ell) = (X_0, 0, X_1, 0)$ and $U(\mathbf{X}_h) = (X_2, 0, X_3, 0)$. Convolution with the synthesis filters yields

$$U(\mathbf{X}_\ell) * \mathbf{\ell}_s = \left(\frac{X_0}{\sqrt{2}}, \frac{X_1}{\sqrt{2}}, \frac{X_1}{\sqrt{2}}, \frac{X_0}{\sqrt{2}} \right),$$

$$U(\mathbf{X}_h) * \mathbf{h}_s = \left(\frac{X_2}{\sqrt{2}}, \frac{-X_3}{\sqrt{2}}, \frac{X_3}{\sqrt{2}}, \frac{-X_2}{\sqrt{2}} \right).$$

Adding produces the linear transform $\mathbf{X} \rightarrow U(\mathbf{X}_\ell) * \mathbf{\ell}_s + U(\mathbf{X}_h) * \mathbf{h}_s$ or

$$\mathbf{X} \rightarrow \left(\frac{X_0 + X_2}{\sqrt{2}}, \frac{X_1 - X_3}{\sqrt{2}}, \frac{X_1 + X_3}{\sqrt{2}}, \frac{X_0 - X_2}{\sqrt{2}} \right).$$

This transformation is embodied in the matrix

$$\mathbf{W}_4^s = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix}. \quad (6.17)$$

Of course $\mathbf{W}_4^a = (\mathbf{W}_4^s)^{-1}$, as is easy to check directly. In fact both \mathbf{W}_4^a and \mathbf{W}_4^s are orthogonal matrices, so $(\mathbf{W}_4^a)^T = \mathbf{W}_4^s$. This is one reason behind the terminology “orthogonal” for certain filter banks—the analysis and synthesis matrices obtained in the finite-length filter bank are orthogonal, which we show more generally in Section 6.5.3.

Other Extensions It is also possible to construct a discrete wavelet transform and corresponding inverse for vectors in \mathbb{C}^N based on the symmetric extension technique defined by equation (6.11) or (6.12). This is especially easy when the filters involved possess even or odd symmetries, such as the Le Gall filters of Example 6.4. Similar transforms can be constructed for other extension techniques too, and many are built into Matlab’s Wavelet Toolbox. See [24] and [5] for more information, and other approaches to adapting filter banks to finite-length signals.

6.5.3 Matrix Formulation of the Periodic Case

If all convolutions in the finite case are circular then $\mathbf{X}_\ell = \mathbf{DM}_{\ell_a}$ and $\mathbf{X}_h = \mathbf{DM}_{h_a}$, where \mathbf{M}_{ℓ_a} and \mathbf{M}_{h_a} are the circulant matrices for the analysis filters and \mathbf{D} is a matrix that represents downsampling. The matrix \mathbf{D} has dimensions $N/2 \times N$ with entries

$$D_{j,k} = \begin{cases} 1, & k = 2j, \quad 0 \leq j \leq N/2 - 1, \\ 0, & \text{else,} \end{cases} \quad (6.18)$$

with rows and columns indexed from 0. It’s not hard to check that $(\mathbf{D}\mathbf{x})_k = x_{2k}$ for any $\mathbf{x} \in \mathbb{C}^N$; see Exercise 6.13. In this case the N -point filter bank analysis transform has the form $\mathbf{x} \rightarrow \mathbf{W}_N^a \mathbf{x}$, where

$$\mathbf{W}_N^a = \begin{bmatrix} \mathbf{DM}_{\ell_a} \\ \mathbf{DM}_{h_a} \end{bmatrix} \quad (6.19)$$

is the $N \times N$ matrix obtained by “stacking” the $N/2 \times N$ matrices \mathbf{DM}_{ℓ_a} and \mathbf{DM}_{h_a} .

By Theorem 6.5.1, the inverse transform is obtained by upsampling each of \mathbf{X}_ℓ and \mathbf{X}_h , and then circularly convolving each with the synthesis filters and adding. Let us define the $N \times N/2$ upsampling matrix \mathbf{U} with entries

$$U_{j,k} = \begin{cases} 1, & 2k = j, \quad 0 \leq k \leq N/2 - 1, \\ 0, & \text{else.} \end{cases} \quad (6.20)$$

The synthesis bank operation can be then expressed as $\mathbf{X} \rightarrow \mathbf{W}_N^s \mathbf{X}$, where

$$\mathbf{W}_N^s = [\mathbf{M}_{\ell_s} \mathbf{U} | \mathbf{M}_{h_s} \mathbf{U}] \quad (6.21)$$

is the $N \times N$ matrix obtained by concatenating the $N \times N/2$ matrices $\mathbf{M}_{\ell_s} \mathbf{U}$ and $\mathbf{M}_{h_s} \mathbf{U}$. Here \mathbf{M}_{ℓ_s} and \mathbf{M}_{h_s} are the circulant matrices for the synthesis filters. It can be shown that $\mathbf{U} = \mathbf{D}^T$; again, see Exercise 6.13.

For a perfect reconstruction filter bank with zero delay we have $\mathbf{W}_N^s \mathbf{W}_N^a = \mathbf{I}_N$, where \mathbf{I}_N is the $N \times N$ identity matrix. In the case that the filter bank is orthogonal (so that the filters satisfy equations (6.10)) a little matrix algebra yields

$$\begin{aligned} (\mathbf{W}_N^a)^T &= \left[\begin{array}{c|c} \mathbf{D}\mathbf{M}_{\ell_a} & \\ \hline \mathbf{D}\mathbf{M}_{h_a} & \end{array} \right]^T \\ &= [(\mathbf{D}\mathbf{M}_{\ell_a})^T \mid (\mathbf{D}\mathbf{M}_{h_a})^T] \\ &= [\mathbf{M}_{\ell_a}^T \mathbf{D}^T \mid \mathbf{M}_{h_a}^T \mathbf{D}^T] \\ &= [\mathbf{M}_{\ell_a}^T \mathbf{U} \mid \mathbf{M}_{h_a}^T \mathbf{U}] \\ &= [\mathbf{M}_{\ell_s} \mathbf{U} \mid \mathbf{M}_{h_s} \mathbf{U}] \\ &= \mathbf{W}_N^s, \end{aligned} \quad (6.22)$$

where we've used $\mathbf{M}_{\ell_a}^T = \mathbf{M}_{\ell_s}$ and $\mathbf{M}_{h_a}^T = \mathbf{M}_{h_s}$, both of which are consequences of equations (6.10) and Exercise 4.9.

From equation (6.22) and $\mathbf{W}_N^s \mathbf{W}_N^a = \mathbf{I}_N$ we conclude that for a perfect reconstruction orthogonal filter bank we have $(\mathbf{W}_N^a)^T \mathbf{W}_N^a = \mathbf{W}_N^s (\mathbf{W}_N^s)^T = \mathbf{I}_N$. Thus both the analysis and synthesis transform matrices are in fact orthogonal. This is one motivation for the terminology “orthogonal filter bank,” at least in the finite-dimensional case.

6.5.4 Multistage Transforms

The ideas that follow don't depend on the particular method by which we obtain a wavelet transform for finite length signals, such as periodic extension or symmetric extension. All we need is a well-defined method for performing a discrete wavelet transform on a signal of a given length. We thus assume that for signals of any (even)

length k we can construct matrices \mathbf{W}_k^a and \mathbf{W}_k^s that govern a forward and inverse discrete wavelet transform, for example, as equations (6.19) and (6.21) do in the periodic extension case.

Iterating the One-stage Transform Just as for signals in $L^2(\mathbb{Z})$ we can iterate discrete wavelet transforms. The typical procedure is to transform a signal $\mathbf{x} \in \mathbb{C}^N$ to produce $\mathbf{X} = (\mathbf{X}_\ell, \mathbf{X}_h) \in \mathbb{C}^N$ consisting of approximation and detail coefficients \mathbf{X}_ℓ and \mathbf{X}_h , each elements of $\mathbb{C}^{N/2}$, and then to pass the approximation coefficient vector \mathbf{X}_ℓ through another filter bank to produce $N/4$ -dimensional vectors $\mathbf{X}_{\ell\ell}$ and $\mathbf{X}_{\ell h}$. The entire two-stage transform can be amalgamated into the single vector $(\mathbf{X}_{\ell\ell}, \mathbf{X}_{\ell h}, \mathbf{X}_h) \in \mathbb{C}^N$.

The process can be iterated further. For example, we can pass $\mathbf{X}_{\ell\ell}$ through the filter bank again to produce $N/8$ -dimensional vectors $\mathbf{X}_{\ell\ell\ell}$ and $\mathbf{X}_{\ell\ell h}$, and so obtain a three-stage discrete wavelet transform

$$\mathbf{X} = (\mathbf{X}_{\ell\ell\ell}, \mathbf{X}_{\ell\ell h}, \mathbf{X}_{\ell h}, \mathbf{X}_h),$$

an element of \mathbb{C}^N . The appropriate synthesis filters can be used to reconstruct $\mathbf{X}_{\ell\ell}$ from the vectors $\mathbf{X}_{\ell\ell\ell}$ and $\mathbf{X}_{\ell\ell h}$. We can then reconstruct \mathbf{X}_ℓ from $\mathbf{X}_{\ell\ell}$ and $\mathbf{X}_{\ell h}$. Finally, we recover \mathbf{x} from \mathbf{X}_ℓ and \mathbf{X}_h .

This process can be carried out to any stage provided that N is divisible by a suitably large power of 2.

■ EXAMPLE 6.9

Consider the 1024-point signal from Example 6.2, shown in Figure 6.5. In Figure 6.13 we show the one-, two-, four-, and six-stage orthogonal Haar transforms of that signal. The one-stage transform yields vectors \mathbf{X}_ℓ and \mathbf{X}_h , each an element of \mathbb{R}^{512} , which we display as the vector $(\mathbf{X}_\ell, \mathbf{X}_h) \in \mathbb{R}^{1024}$ on the upper left in Figure 6.13. The two-stage transform consists of the 256-dimensional vectors $\mathbf{X}_{\ell\ell}$, $\mathbf{X}_{\ell h}$ and the same vector \mathbf{X}_h from the one-stage transform. The overall two-stage transform is displayed as the 1024-dimensional vector $(\mathbf{X}_{\ell\ell}, \mathbf{X}_{\ell h}, \mathbf{X}_h)$. The same general scheme is used to plot the other transforms. For example, the six-stage transform is a plot of the 1024-component vector

$$(\mathbf{X}_{\ell\ell\ell\ell\ell\ell}, \mathbf{X}_{\ell\ell\ell\ell\ell h}, \mathbf{X}_{\ell\ell\ell\ell h}, \mathbf{X}_{\ell\ell\ell h}, \mathbf{X}_{\ell\ell h}, \mathbf{X}_{\ell h}, \mathbf{X}_h)$$

consisting of subvectors with 16, 16, 32, 64, 128, 256, and 512 components. At each stage the previously high-pass filtered portions remain unchanged. The portion $\mathbf{X}_{\ell\ldots\ell}$ is a progressively smoother and smoother (and downsampled) version of the original signal.

Matrix Formulation of Multistage Transform The multistage transform for finite length signals has a simple matrix formulation. Let \mathbf{W}_m^a denote the analysis

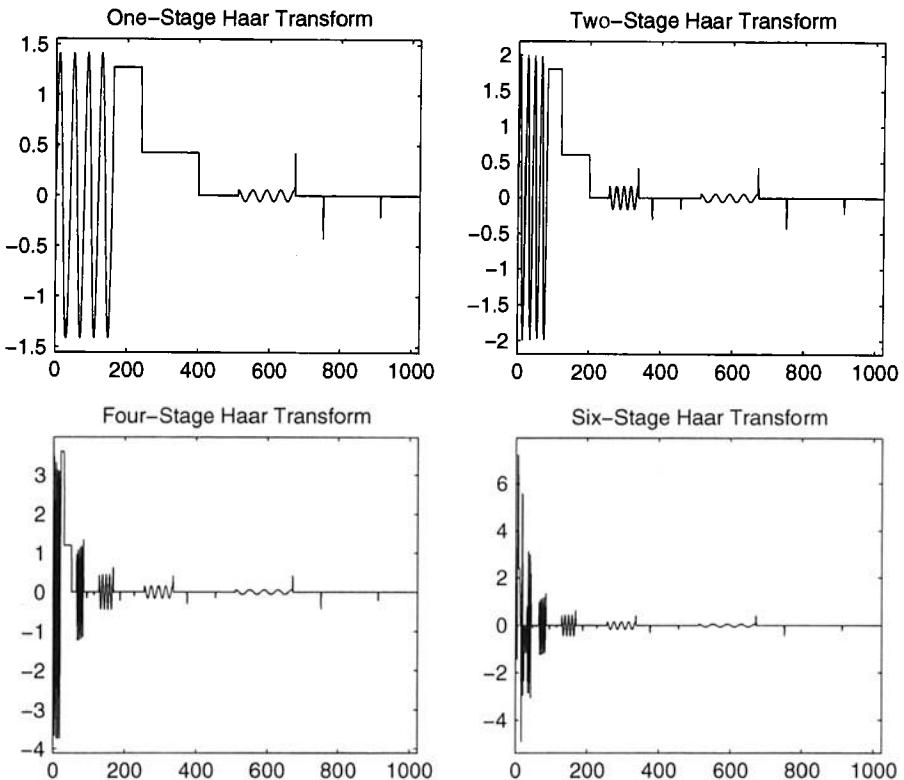


FIGURE 6.13 One-, two-, four- and six-stage Haar discrete wavelet transforms.

matrix for a single-stage m -point DWT. A two-stage transform consisting of an N -point transform of \mathbf{x} followed by an $N/2$ point transform of \mathbf{X}_ℓ can be implemented as $\mathbf{x} \rightarrow \mathcal{W}_2^a \mathbf{x}$, where

$$\mathcal{W}_2^a = \begin{bmatrix} \mathbf{W}_{N/2}^a & 0 \\ 0 & \mathbf{I}_{N/2} \end{bmatrix} \mathbf{W}_N^a. \quad (6.23)$$

The subscript “2” in \mathcal{W}_2^a denotes a two-stage transform. Notice that \mathcal{W}_2^a also depends on N , but we do not explicitly note this. Of course, this construction requires that N be divisible by 4 in order to perform the $N/2$ point transform. The corresponding inverse transform $\mathbf{X} \rightarrow \mathcal{W}_2^s \mathbf{X}$ is implemented via the matrix

$$\mathcal{W}_2^s = \mathbf{W}_N^s \begin{bmatrix} \mathbf{W}_{N/2}^s & 0 \\ 0 & \mathbf{I}_{N/2} \end{bmatrix}, \quad (6.24)$$

which inverts each transform in reverse order.

Generally, we can perform an r -stage DWT, governed by the matrix product of the r matrices

$$\mathcal{W}_r^a = \begin{bmatrix} \mathbf{W}_{N/2^{r-1}}^a & 0 \\ 0 & \mathbf{I}_{N(1-1/2^{r-1})} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{W}_{N/2}^a & 0 \\ 0 & \mathbf{I}_{N/2} \end{bmatrix} \mathbf{W}_N^a,$$

where each factor is an $N \times N$ matrix of the form

$$\begin{bmatrix} \mathbf{W}_{N/2^{k-1}}^a & 0 \\ 0 & \mathbf{I}_{N(1-1/2^{k-1})} \end{bmatrix}$$

for $1 \leq k \leq r$. All this requires is that N be divisible by 2^r . The one-stage transform matrix \mathcal{W}_1^a is just the matrix \mathbf{W}_N^a .

The inverse transform is governed by the matrix

$$\mathcal{W}_r^s = \mathbf{W}_N^s \begin{bmatrix} \mathbf{W}_{N/2}^s & 0 \\ 0 & \mathbf{I}_{N/2} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{W}_{N/2^{r-1}}^s & 0 \\ 0 & \mathbf{I}_{N(1-1/2^{r-1})} \end{bmatrix},$$

where each factor is an $N \times N$ matrix of the form

$$\begin{bmatrix} \mathbf{W}_{N/2^{k-1}}^s & 0 \\ 0 & \mathbf{I}_{N(1-1/2^{k-1})} \end{bmatrix}$$

Reconstruction from Approximation Coefficients A crude form of compression would be to store only the approximation coefficient \mathbf{X}_ℓ vector, which is half as long as \mathbf{x} . We could then approximately reconstruct \mathbf{x} from \mathbf{X}_ℓ alone. Specifically, we can write the one-stage DWT of a signal \mathbf{x} as

$$\begin{bmatrix} \mathbf{X}_\ell \\ \mathbf{X}_h \end{bmatrix} = \mathcal{W}_1^a(\mathbf{x}).$$

Since the DWT and its inverse are linear, we have

$$\begin{aligned} \mathbf{x} &= \mathcal{W}_1^s(\mathcal{W}_1^a(\mathbf{x})) = \mathcal{W}_1^s \left(\begin{bmatrix} \mathbf{X}_\ell \\ \mathbf{X}_h \end{bmatrix} \right) \\ &= \mathcal{W}_1^s \left(\begin{bmatrix} \mathbf{X}_\ell \\ 0 \end{bmatrix} \right) + \mathcal{W}_1^s \left(\begin{bmatrix} 0 \\ \mathbf{X}_h \end{bmatrix} \right) \\ &:= \alpha_1(\mathbf{x}) + \delta_1(\mathbf{x}). \end{aligned} \tag{6.25}$$

The vector $\alpha_1(\mathbf{x})$ is called the *level 1* or *stage 1 approximation* to \mathbf{x} and $\delta_1(\mathbf{x})$ is called the *stage 1 detail*. Note that $\alpha_1(\mathbf{x})$ and $\delta_1(\mathbf{x})$ are vectors of the same size as \mathbf{x} , even

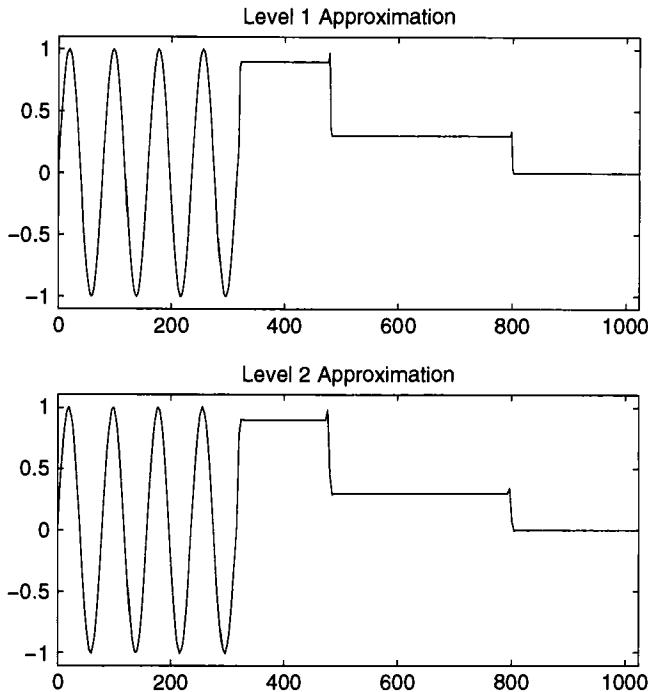


FIGURE 6.14 Reconstructions $\alpha_1(\mathbf{x})$ (top) and $\alpha_2(\mathbf{x})$ (bottom).

though \mathbf{X}_ℓ and \mathbf{X}_h are half that size. The vector $\alpha_1(\mathbf{x})$ is an approximation to the original signal \mathbf{x} and $\delta_1(\mathbf{x})$ constitutes the (probably small) corrections necessary to perfectly reconstruct the original signal. The top panel in Figure 6.14 shows $\alpha_1(\mathbf{x})$ for the 1024-point signal of Examples 6.2 and 6.9, using the Le Gall 5/3 filters and circular convolution.

In the two-stage case we have

$$\begin{aligned} \mathbf{x} &= \mathcal{W}_2^s(\mathcal{W}_2^a(\mathbf{x})) = \mathcal{W}_2^s \left(\begin{bmatrix} \mathbf{X}_{\ell\ell} \\ \mathbf{X}_{\ell h} \\ \mathbf{X}_h \end{bmatrix} \right) \\ &= \mathcal{W}_2^s \left(\begin{bmatrix} \mathbf{X}_{\ell\ell} \\ 0 \\ 0 \end{bmatrix} \right) + \mathcal{W}_2^s \left(\begin{bmatrix} 0 \\ \mathbf{X}_{\ell h} \\ 0 \end{bmatrix} \right) + \mathcal{W}_2^s \left(\begin{bmatrix} 0 \\ 0 \\ \mathbf{X}_h \end{bmatrix} \right). \end{aligned} \quad (6.26)$$

Now by the construction of the second stage of the DWT we have

$$\mathcal{W}_2^s \left(\begin{bmatrix} 0 \\ 0 \\ \mathbf{X}_h \end{bmatrix} \right) = \mathcal{W}_1^s \left(\begin{bmatrix} 0 \\ 0 \\ \mathbf{X}_h \end{bmatrix} \right)$$

(look back at equation (6.24) to see why), and thus we can express equation (6.26) as

$$\mathbf{x} = \alpha_2(\mathbf{x}) + \delta_2(\mathbf{x}) + \delta_1(\mathbf{x}), \quad (6.27)$$

where $\alpha_2(\mathbf{x})$ and $\delta_2(\mathbf{x})$ are called the *level 2 approximation and details* for \mathbf{x} , respectively; the vector $\delta_1(\mathbf{x})$ is unchanged from the one-stage case. The quantity $\alpha_2(\mathbf{x})$ is computable from $\mathbf{X}_{\ell\ell}$ alone. The bottom panel in Figure 6.14 shows $\alpha_2(\mathbf{x})$ for the 1024-point signal of Examples 6.2 and 6.9, again with the Le Gall 5/3 filters and circular convolution.

We can continue this process to level r and obtain

$$\begin{aligned} \mathbf{x} &= \alpha_3(\mathbf{x}) + \delta_3(\mathbf{x}) + \delta_2(\mathbf{x}) + \delta_1(\mathbf{x}) \\ &= \alpha_4(\mathbf{x}) + \delta_4(\mathbf{x}) + \delta_3(\mathbf{x}) + \delta_2(\mathbf{x}) + \delta_1(\mathbf{x}) \\ &= \vdots \\ &= \alpha_r(\mathbf{x}) + \delta_r(\mathbf{x}) + \delta_{r-1}(\mathbf{x}) + \cdots + \delta_2(\mathbf{x}) + \delta_1(\mathbf{x}). \end{aligned} \quad (6.28)$$

It's easy to see that

$$\alpha_r(\mathbf{x}) = \alpha_{r-1}(\mathbf{x}) - \delta_r(\mathbf{x}),$$

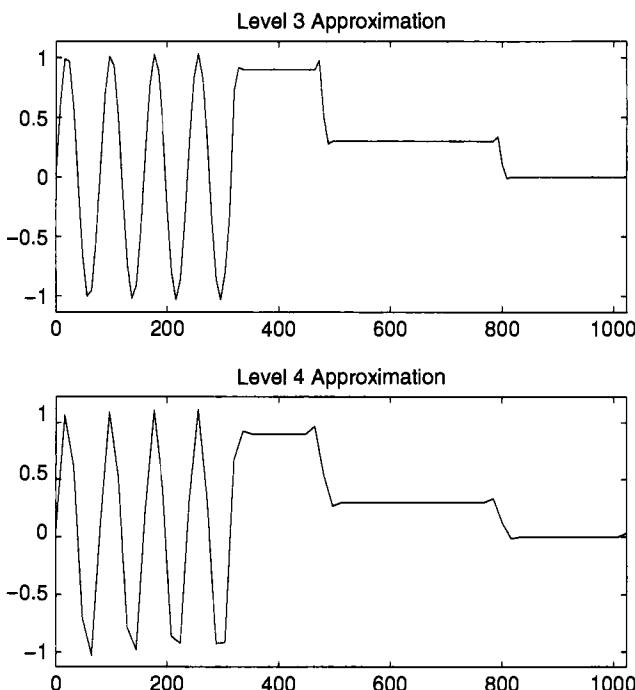


FIGURE 6.15 Reconstructions $\alpha_3(\mathbf{x})$ (top) and $\alpha_4(\mathbf{x})$ (bottom).

that is, the level r approximation $\alpha_r(\mathbf{x})$ is just the more detailed level $r - 1$ approximation $\alpha_{r-1}(\mathbf{x})$ minus the level r detail $\delta_r(\mathbf{x})$. Thus we get cruder and cruder approximations of \mathbf{x} by successively peeling off coarser and coarser details from the original \mathbf{x} . In Figure 6.15 we show $\alpha_3(\mathbf{x})$ and $\alpha_4(\mathbf{x})$ for the 1024-point signal of Examples 6.2 and 6.9, again with the Le Gall 5/3 filters and circular convolution.

If \mathbf{x} is a finite-dimensional vector, we can continue this for only finitely many stages. However, in the $L^2(\mathbb{Z})$ case it can be continued indefinitely.

6.5.5 Matlab Implementation of Discrete Wavelet Transforms

Matlab's wavelet decomposition command `dwt` implements the discrete wavelet transform for finite length signals. The command can be tailored to use any of the extension methods described above, as well as many more. We'll explore this in the Matlab project at the end of this chapter.

6.6 THE 2D DISCRETE WAVELET TRANSFORM AND JPEG 2000

6.6.1 Two-dimensional Transforms

We'll restrict our wavelet analysis in two dimensions to the case of finite-extent signals or images rather than signals in $L^2(\mathbb{Z} \times \mathbb{Z})$. Let (ℓ_a, \mathbf{h}_a) be the low- and high-pass analysis filters for a filter bank for $L^2(\mathbb{Z})$ signals. We assume that the analysis filters have been adapted for use in a filter bank for finite length signals, either via periodic extension as in Section 6.5.2 or some other technique, and hence we can define a corresponding analysis matrix \mathbf{W}_k^a for signals of any length k . We also assume that suitable perfect reconstruction synthesis filters exist with corresponding matrix \mathbf{W}_k^s .

Let \mathbf{A} be an $M \times N$ matrix or grayscale image, where we assume that both M and N are divisible by reasonably large powers of 2. There are many ways we can use a filter bank for one-dimensional signals to define a 2D wavelet transform. The simplest is to do precisely what we did with the discrete Fourier and cosine transforms: transform each column of \mathbf{A} and then transform each row of the resulting matrix. We thus make the following definition:

Definition 6.6.1 *Let \mathbf{A} , M , N , ℓ_a , \mathbf{h}_a be as above. The one-stage, two-dimensional discrete wavelet transform (DWT) determined by (ℓ_a, \mathbf{h}_a) is the linear mapping given by*

$$\mathcal{W}_1^a(\mathbf{A}) = \mathbf{W}_M^a \mathbf{A} (\mathbf{W}_N^a)^T,$$

where \mathbf{W}_M^a and \mathbf{W}_N^a are the $M \times M$ and $N \times N$ analysis matrices determined by (ℓ_a, \mathbf{h}_a) . We define the inverse one-stage transform as

$$\mathcal{W}_1^s(\widehat{\mathbf{A}}) = \mathbf{W}_M^s \widehat{\mathbf{A}} (\mathbf{W}_N^s)^T.$$

TABLE 6.1 Arrangement of Low- and High-pass Components

	Low	High
Low	<i>LL</i> (approximation)	<i>HL</i> (vertical details)
High	<i>LH</i> (horizontal details)	<i>HH</i> (diagonal details)

To graphically present a one-stage DWT for an image we'll use the following scheme: if the image \mathbf{A} is $M \times N$, then the transform $\mathcal{W}_1^a(\mathbf{A})$ will consist of four distinct parts, each an array of size $M/2 \times N/2$. One such part is that in which each row and column of the original image has been low-pass filtered and downsampled (downsampling the rows cuts them to length $N/2$ and downsampling columns cuts them to length $M/2$). Another part of the transform consists of rows that are low-pass filtered with columns that are high-pass filtered, both downsampled. Similarly we obtain high-pass row, low-pass column and high-pass high-pass parts. We present the transform as a grayscale image according to the scheme of Table 6.1. The *LL* piece (low-low) is in the upper left-hand corner and comes from low-pass filtering in both directions. Of the four pieces it is the most like the original picture and so is called the *stage 1 approximation coefficients*. The remaining three pieces are called *stage 1 detail components*. The upper right corner comes from high-pass filtering in the horizontal direction (rows) and low-pass filtering in the vertical direction (columns), and so has the label *HL*. The visible details in this sub-image, such as edges, have an overall vertical orientation, since their alignment is orthogonal to the direction of the high pass filtering. The remaining components have analogous explanations.

In what follows if $\mathbf{C} = \mathcal{W}_1^a(\mathbf{A})$ (so that \mathbf{C} is an $M \times N$ matrix), then we write

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_A & \mathbf{C}_V \\ \mathbf{C}_H & \mathbf{C}_D \end{bmatrix} \quad (6.29)$$

for the various portions of the transform, where each of the sub-matrices is an $M/2 \times N/2$ matrix.

■ EXAMPLE 6.10

Let us show before and after pictures for an image using the 1-stage DWT. The original picture is in the top panel in Figure 6.16, 600 by 800 pixels, and the one-stage DWT in the bottom panel, displayed according to the scheme of equation (6.29). The analysis filters are the Le Gall 5/3 filters of Example 6.4. The DWT is displayed on a logarithmic scale. In the upper right portion of the transform the tree trunks are most visible, since this is the vertical detail section of the transform. The relatively flat and featureless portions of the image (e.g., the water) ends up appearing black in all but the approximation coefficients, since high-pass filtering in either the vertical or horizontal direction zeros out dc components.

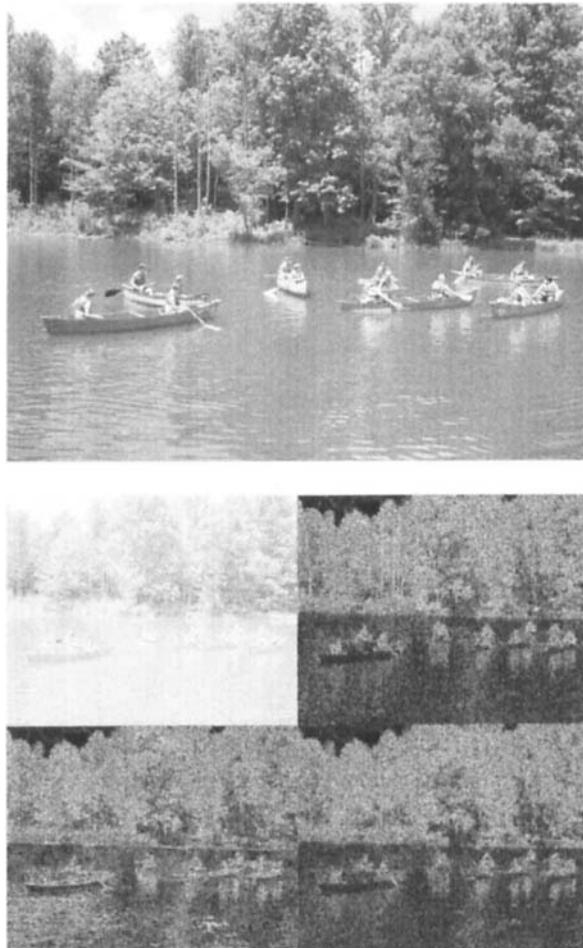


FIGURE 6.16 Original image (*top*) and one-stage DWT (*bottom*).

6.6.2 Multistage Transforms for Two-dimensional Images

We may of course do a multistage transform. In Figure 6.17 we show two-stage and three-stage transforms of the image from Example 6.10, both displayed on a logarithmic scale.

Remark 6.3 For image analysis one usually goes to $r = 4$ or 5 levels. There is always one approximation sub-matrix and $3r$ detail sub-matrices.

Let's examine the multistage procedure more carefully. Let (ℓ_a, \mathbf{h}_a) be the analysis filters and (ℓ_s, \mathbf{h}_s) the synthesis filters for a biorthogonal filter bank with DWT analysis

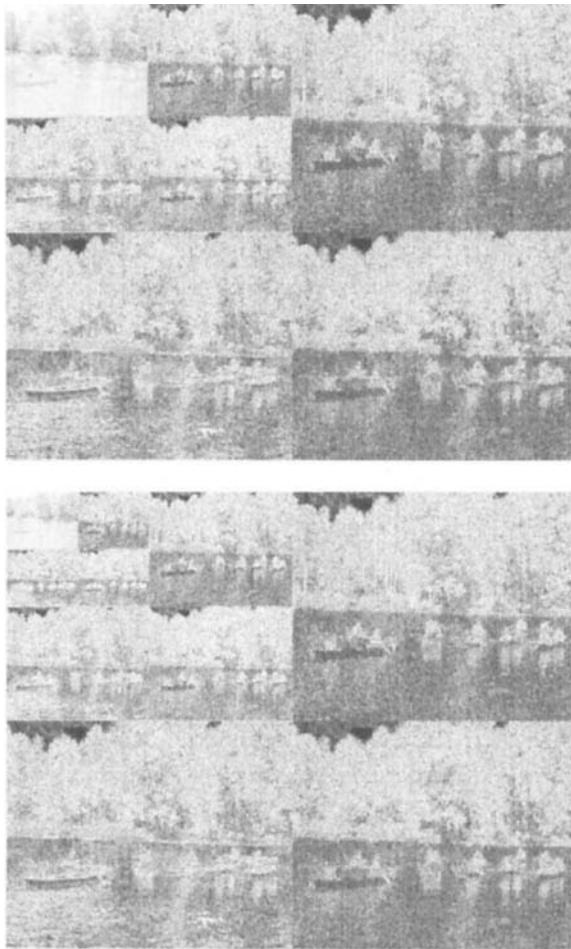


FIGURE 6.17 Two-stage (*top*) and three-stage (*bottom*) DWT of image from Example 6.10.

and synthesis matrices \mathbf{W}_k^a or \mathbf{W}_k^s for one-dimensional transforms of size k . Let \mathbf{A} be an $M \times N$ array. Let $\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_r$ be the matrices formed as follows: the matrix \mathbf{C}_1 is obtained by computing the DWT of \mathbf{A} via

$$\mathbf{C}_1 = \mathbf{W}_M^a \mathbf{A} (\mathbf{W}_N^a)^T = \begin{bmatrix} \mathbf{C}_{1,A} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix},$$

where $\mathbf{C}_{1,A}$ denotes the $M/2 \times N/2$ upper left quadrant of \mathbf{C}_1 (the approximation sub-image; this is the same as equation (6.29)). Let \mathbf{C}_2 be the matrix obtained from

\mathbf{C}_1 by replacing $\mathbf{C}_{1,A}$ by the DWT of $\mathbf{C}_{1,A}$,

$$\begin{aligned}\mathbf{C}_2 &= \begin{bmatrix} \mathbf{W}_{M/2}^a \mathbf{C}_{1,A} (\mathbf{W}_{N/2}^a)^T & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} (\mathbf{C}_{1,A})_A & (\mathbf{C}_{1,A})_V \\ (\mathbf{C}_{1,A})_H & (\mathbf{C}_{1,A})_D \end{bmatrix} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} \mathbf{C}_{2,A} & \mathbf{C}_{2,V} \\ \mathbf{C}_{2,H} & \mathbf{C}_{2,D} \end{bmatrix} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix}. \quad (6.30)\end{aligned}$$

Compare the array in (6.30) with the picture at the top in Figure 6.17, to build up some intuition about the various pieces of \mathbf{C}_2 .

In general, having defined \mathbf{C}_r , let $\mathbf{C}_{r,A}$ be the $M/2^r \times N/2^r$ upper left approximation sub-matrix in the DWT of \mathbf{C}_r . Then \mathbf{C}_{r+1} is the matrix obtained from \mathbf{C}_r by replacing $\mathbf{C}_{r,A}$ by its one-stage DWT, which has four component pieces:

$$\begin{bmatrix} \mathbf{C}_{r+1,A} & \mathbf{C}_{r+1,V} \\ \mathbf{C}_{r+1,H} & \mathbf{C}_{r+1,D} \end{bmatrix}.$$

For example, at stage 3 we have

$$\mathbf{C}_3 = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} \mathbf{C}_{3,A} & \mathbf{C}_{3,V} \\ \mathbf{C}_{3,H} & \mathbf{C}_{3,D} \end{bmatrix} & \mathbf{C}_{2,V} \\ \mathbf{C}_{2,H} & \mathbf{C}_{2,D} \end{bmatrix} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix}.$$

Compare this result with the image on the bottom in Figure 6.17.

Definition 6.6.2 We define the r -stage (or r -level) two-dimensional DWT of \mathbf{A} as the matrix \mathbf{C}_r . Symbolically we denote this r -stage transform as

$$\mathbf{A} \rightarrow \mathcal{W}_r^a(\mathbf{A}).$$

The transform is linear.

Analogously the r -stage synthesis transform is denoted

$$\widehat{\mathbf{A}} \rightarrow \mathcal{W}_r^s(\widehat{\mathbf{A}})$$

and is obtained by reversing the above steps with the synthesis filters and corresponding matrices. The inverse transform is linear. For a biorthogonal (perfect reconstruction) filter bank we have $\mathcal{W}_r^s(\mathcal{W}_r^a(\mathbf{A})) = \mathbf{A}$.

Matlab implements DWTs in two dimensions using the command `dwt2`. As in the one-dimensional case there are a variety of options for extending the images past their natural boundaries in order to apply the analysis filters, including, periodic extension, whole and half-point symmetric extension, and many others. We explore them in the project below.

6.6.3 Approximations and Details for Images

As for one-dimensional signals, we can perform a multistage decomposition into approximation and details for images. For an image \mathbf{A} , for example, we can write

$$\begin{aligned}\mathbf{A} &= \mathcal{W}_1^s(\mathcal{W}_1^a(\mathbf{A})) \\ &= \mathcal{W}_1^s \left(\begin{bmatrix} \mathbf{C}_{1,A} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \right) \\ &= \mathcal{W}_1^s \left(\begin{bmatrix} \mathbf{C}_{1,A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right) + \mathcal{W}_1^s \left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \right) \\ &:= \alpha_1(\mathbf{A}) + \delta_1(\mathbf{A}),\end{aligned}$$

analogous to equation (6.25). The quantity $\alpha_1(\mathbf{A})$ is the *level 1 approximation* to \mathbf{A} and $\delta_1(\mathbf{A})$ is the *level 1 detail*. Note that $\alpha_1(\mathbf{A})$ is a full $M \times N$ image constructed from the $M/2 \times N/2$ approximation coefficient matrix $\mathbf{C}_{1,A}$. In Figure 6.18 on the top we show $\alpha_1(\mathbf{A})$ for the image from Figure 6.16.

As with signals, the decomposition process can be iterated, for example, as

$$\begin{aligned}\mathbf{A} &= \mathcal{W}_2^s(\mathcal{W}_2^a(\mathbf{A})) \\ &= \mathcal{W}_2^s \left(\begin{bmatrix} \begin{bmatrix} \mathbf{C}_{2,A} & \mathbf{C}_{2,V} \\ \mathbf{C}_{2,H} & \mathbf{C}_{2,D} \end{bmatrix} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \right) \\ &= \mathcal{W}_2^s \left(\begin{bmatrix} \begin{bmatrix} \mathbf{C}_{2,A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right) + \mathcal{W}_2^s \left(\begin{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{C}_{2,V} \\ \mathbf{C}_{2,H} & \mathbf{C}_{2,D} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right) \\ &\quad + \mathcal{W}_1^s \left(\begin{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} & \mathbf{C}_{1,V} \\ \mathbf{C}_{1,H} & \mathbf{C}_{1,D} \end{bmatrix} \right) \\ &:= \alpha_1(\mathbf{A}) + \delta_2(\mathbf{A}) + \delta_1(\mathbf{A}),\end{aligned}$$

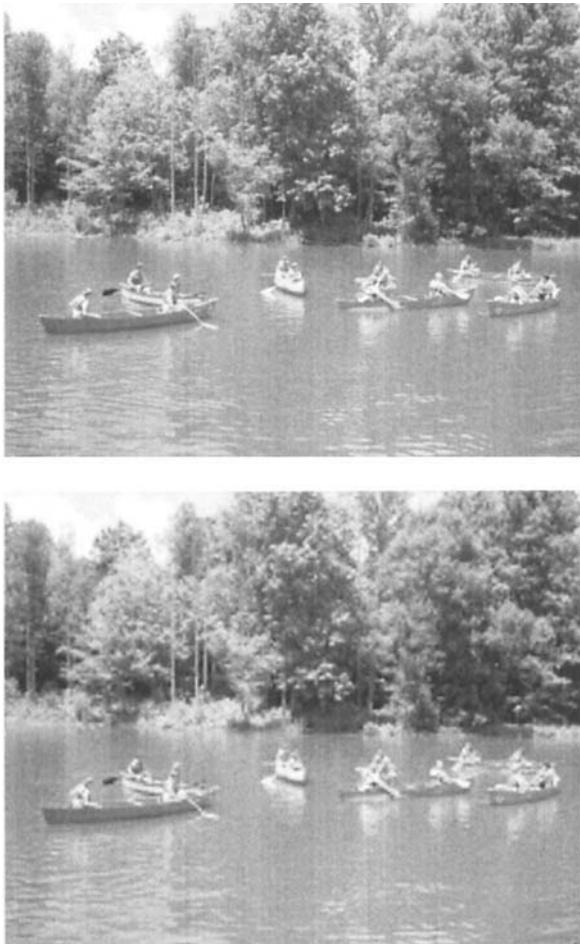


FIGURE 6.18 Reconstructions $\alpha_1(\mathbf{A})$ (*top*) and $\alpha_2(\mathbf{A})$ (*bottom*).

analogous to equation (6.28), where we've made use of equation (6.30) and the fact that $\mathbf{C}_2 = \mathcal{W}_2^a(\mathbf{A})$. We can thus approximate \mathbf{A} "crudely," as $\alpha_2(\mathbf{A})$, or with slightly more detail, as $\alpha_2(\mathbf{A}) + \delta_2(\mathbf{A})$ (which is just $\alpha_1(\mathbf{A})$). The level-2 approximation $\alpha_2(\mathbf{A})$ for the image from Figure 6.16 is shown at the bottom in Figure 6.18; the approximation $\alpha_3(\mathbf{A})$ is shown in Figure 6.19. Not surprisingly, the level- r approximation degrades as r increases. The decomposition can be carried to any level as long as M and N are divisible by suitable powers of two.

6.6.4 JPEG 2000

The JPEG 2000 image compression standard has been under development, in several stages, since 1997. The goal is to improve upon the traditional JPEG standard,



FIGURE 6.19 Reconstruction $\alpha_3(A)$.

not merely by increasing compression performance but by also providing greater flexibility and a wider range of options in compressing images. Thus, for example, the possibility exists for compressing different parts of an image with different parameters, or specifying that part (or all) of an image be compressed with no information loss. Many other features are built into the standard.

We don't intend to give a comprehensive or even partial survey of the many facets of the standard, but merely to indicate how the discrete wavelet transform comes into play. See [23] for additional information.

We should mention that image compression is important for more than just personal photos and transmission of Internet images. Image compression, and more generally image processing, plays an important role in printing, medical imaging, nondestructive testing and remote sensing, image archiving, and other areas. The JPEG 2000 standard is an attempt to serve many of these needs with a coherent, flexible standard.

The overall structure of the JPEG 2000 standard is similar to the traditional JPEG standard, but at the core of the JPEG 2000 is the discrete wavelet transform. Preprocessing steps similar to those of the traditional JPEG algorithm take place, such as, subtraction of dc levels or transformations to separate color components. After this the relevant data are subjected to a DWT using either the Daubechies 9/7 filters or the Le Gall 5/3 filters (the latter are used for lossless compression). The transformed image components are then quantized using one or more schemes like those described in Chapter 1, and then the results are compressed using an entropy encoding algorithm. Reconstruction of the image consists of applying the appropriate decoding algorithm and inverse quantization. The appropriate synthesis filter (inverse wavelet transform) is then applied to reconstruct the image.

The Matlab project at the end of this chapter provides the opportunity to use Matlab's Wavelet Toolbox to process some signals and grayscale images.

6.7 FILTER DESIGN

This section focuses on filter design for perfect reconstruction filter banks. It's not all that essential for using filter banks if you are content to use off-the-shelf filters. But it is useful for those readers who seek to understand the essential quantitative relationship between filter banks and wavelets.

It's natural to ask whether we can choose any analysis filters we want and then design appropriate synthesis filters for perfect reconstruction. A moment's thought should show that this is not the case—taking both analysis filters to be identically zero provides an obvious counterexample. So what constraints exist on the analysis filters? If suitable synthesis filters exist, how do we find them? Some information can be gleaned from the time-domain relations that come from equation (6.6), but it's easier to carry out the analysis via z -transforms. Our goal here is not an exhaustive treatment of filter design techniques, but a few representative and important examples.

6.7.1 Filter Banks in the z -Domain

We begin by considering the effect of downsampling, upsampling and filtering in the “frequency domain” of the z -transform. Let \mathbf{x} be a signal in $L^2(\mathbb{Z})$, and define

$$X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k},$$

the z -transform of \mathbf{x} . We suppose that \mathbf{x} is passed through a two-channel filter bank with FIR analysis filters ℓ_a, \mathbf{h}_a and synthesis filters ℓ_s, \mathbf{h}_s as in Figure 6.4 (with no intermediate compression, de-noising, etc.) We seek perfect reconstruction.

Downsampling and Upsampling in the z -Domain The first and last operations performed in a filter bank are convolutions with the filters. In between, each filtered portion of the signal is downsampled and then upsampled. The operation of downsampling and immediately upsampling a signal is simple to quantify with z -transforms.

Let \mathbf{y} be the result of downsampling and then upsampling some signal \mathbf{w} . It's easy to see that $y_j = w_j$ for j even and $y_j = 0$ for j odd. This observation leads to the following proposition:

Proposition 6.7.1 *Let $W(z)$ denote the z -transform of $\mathbf{w} \in L^2(\mathbb{Z})$ and $Y(z)$ the z -transform of the vector $\mathbf{y} = U(D(\mathbf{w}))$ obtained by downsampling and then upsampling \mathbf{w} . The function $Y(z)$ is given by*

$$Y(z) = \frac{1}{2}W(z) + \frac{1}{2}W(-z).$$

The proof is left as Exercise 6.21.

Filtering in the Frequency Domain If $\mathbf{y} = \mathbf{x} * \mathbf{g}$ for some FIR filter \mathbf{g} , then the equivalent frequency domain statement is the Convolution Theorem 4.3.1, which asserts that $Y(z) = X(z)G(z)$. By assumption \mathbf{g} is FIR, so there is no difficulty in summing the terms involved in the product $X(z)G(z)$: the sums will converge for any $\mathbf{x} \in L^2(\mathbb{Z})$ (or even $L^\infty(\mathbb{Z})$).

6.7.2 Perfect Reconstruction in the z -Frequency Domain

Let \mathbf{x} be passed through a two-channel filter bank with analysis filters ℓ_a and \mathbf{h}_a . Let L_a and H_a denote the z -transform of these filters. By the Convolution Theorem 4.5.1, the z -transform of $\mathbf{x} * \ell_a$ is $L_a(z)X(z)$. By Proposition 6.7.1, when $\mathbf{x} * \ell_a$ is downsampled and then upsampled, the resulting signal $U(D(\mathbf{x} * \ell_a))$ has z -transform

$$\frac{1}{2}(X(z)L_a(z) + X(-z)L_a(-z)).$$

Again, by the Convolution Theorem 4.5.1, if we convolve $U(D(\mathbf{x} * \ell_a))$ with the synthesis low-pass filter, we obtain a signal with z -transform

$$\frac{1}{2}(X(z)L_a(z) + X(-z)L_a(-z))L_s(z).$$

This is the z -transform of the output from the low-pass portion of the filter bank. Performing the analogous operation with the high-pass portion of the filter bank yields $\frac{1}{2}(X(z)H_a(z) + X(-z)H_a(-z))H_s(z)$. The final step in the filter bank is to add the results, which yields an output vector $\tilde{\mathbf{x}}$ with z -transform

$$\begin{aligned}\tilde{X}(z) &= \frac{1}{2}(X(z)L_a(z) + X(-z)L_a(-z))L_s(z) \\ &\quad + \frac{1}{2}(X(z)H_a(z) + X(-z)H_a(-z))H_s(z) \\ &= \frac{1}{2}[L_a(z)L_s(z) + H_a(z)H_s(z)]X(z) \\ &\quad + [L_a(-z)L_s(z) + H_a(-z)H_s(z)]X(-z).\end{aligned}\tag{6.31}$$

Perfect reconstruction dictates that the output signal should be a copy of the input signal, possibly delayed, so that $\tilde{X}(z) = z^{-m}X(z)$ for some $m \geq 0$; recall Exercise 4.36. (We could also allow $m < 0$, corresponding to an advance; it won't affect the algebra to follow). In light of equation (6.31), if $\tilde{X}(z) = z^{-m}X(z)$ and if we require

$$L_a(z)L_s(z) + H_a(z)H_s(z) = 2z^{-m},\tag{6.32}$$

$$L_a(-z)L_s(z) + H_a(-z)H_s(z) = 0,\tag{6.33}$$

then we will obtain perfect reconstruction. Equations (6.32) and (6.33) are thus sufficient conditions on the analysis and synthesis filter z -transforms for a perfect reconstruction filter bank with m -step delay. As it turns out, they are also necessary; see Exercise 6.26.

There are many approaches to designing suitable filters for use in a biorthogonal or orthogonal filter bank. We'll focus on a few illustrative examples. See [24] for more information.

Remark 6.4 For any perfect reconstruction filter bank we can make numerous minor alterations to the filters, and equations (6.32) and (6.33) will still hold. We can, for example, multiply one low-pass filter by a constant c while dividing the other by c , or do the same to the high-pass filters. We can also introduce shifts or delays of the same magnitude to both analysis or both synthesis filters.

6.7.3 Filter Design I: Synthesis from Analysis

Suppose that the analysis filters are given and we seek suitable synthesis filters. Equations (6.32) and (6.33) are linear in the variables $L_s(z)$, $H_s(z)$ if we consider the analysis filters as given. We can solve as

$$L_s(z) = \frac{2z^{-m}H_a(-z)}{L_a(z)H_a(-z) - L_a(-z)H_a(z)}, \quad (6.34)$$

$$H_s(z) = \frac{-2z^{-m}L_a(-z)}{L_a(z)H_a(-z) - L_a(-z)H_a(z)}. \quad (6.35)$$

However, this is a bit suspect: *we have not defined what it means to divide z-transforms, in the formal sense*. Of course, if the relevant transforms converge as functions of the complex variable z in some common domain, then we can carry out the algebra in the usual way. The quotients above are just rational functions (ratios of polynomials). So let's continue.

In principle, we can choose any analysis filters for which the common right side denominator in equations (6.34) and (6.35) is not zero and then read off the synthesis filters. The only catch is that the resulting z-transforms L_s and H_s may not correspond to FIR filters. However, if the denominator $L_a(z)H_a(-z) - L_a(-z)H_a(z)$ is a monomial z^q , then division by z^q is formally well defined: $z^q X(z)$ just corresponds to a shift of \mathbf{x} , q indexes to the right. In this case L_s and H_s will correspond to FIR filters.

It's worth noting that the z^{-m} factor in each numerator on the right in (6.34) and (6.35) merely corresponds to an m -index translate of the filters obtained using $m = 0$. We will thus assume $m = 0$ for now.

■ EXAMPLE 6.11

Let the analysis filters be given by the Haar filters $\ell_a = (\dots, 0, \frac{1}{2}, \frac{1}{2}, 0, \dots)$ and $\mathbf{h}_a = (\dots, 0, \frac{1}{2}, -\frac{1}{2}, 0, \dots)$ as in Example 6.3, with z-transforms

$$L_a(z) = \frac{1}{2} + \frac{1}{2}z^{-1}, \quad H_a(z) = \frac{1}{2} - \frac{1}{2}z^{-1}.$$

In this case the denominator in equations (6.34) and (6.35) turns out to be z^{-1} , and we obtain (with $m = 0$) that

$$L_s(z) = 1 + z, \quad H_s(z) = 1 - z,$$

the same filters we obtained in Example 6.3. These correspond to filters that are not causal, but by shifting the filter coefficients 1 index (equivalently, use $m = 1$ in (6.34)–(6.35)), we obtain causal filters with transforms $L_s(z) = 1 + z^{-1}$, $H_s(z) = 1 - z^{-1}$. The resulting filter bank perfectly reconstructs the input, but with a one index delay.

■ EXAMPLE 6.12

Let the analysis filters be FIR with coefficients $(\ell_a)_0 = \frac{1}{2}$, $(\ell_a)_1 = 0$, $(\ell_a)_2 = \frac{1}{2}$ and all other coefficients zero, and take \mathbf{h}_a as in the last example. In this case we have $L_a(z) = \frac{1}{2} + \frac{1}{2}z^{-2}$, $H_a(z) = \frac{1}{2} - \frac{1}{2}z^{-1}$, and equations (6.34) and (6.35) (with $m = 0$) yield

$$L_s(z) = \frac{2z^2(z+1)}{z^2+1}, \quad H_s(z) = -2z.$$

In this case the denominator is not a monomial. The transform $H_s(z)$ corresponds to an FIR filter, but $L_s(z)$ does not. Why? If $L_s(z)$ did correspond to an FIR filter, then $z^n L_s(z)$ would be a polynomial in z for n sufficiently large, which is not the case; see Exercise 6.23.

Remark 6.5 One way to guarantee that the required filters are FIR is to obtain all the filters as simple permutations of one or more specified FIR filters, a technique we use in the sections that follow. In this case it's helpful to note the relation between certain filter coefficient permutations in the time domain and the resulting effect in the z -domain. In particular, let \mathbf{g} be a causal N -tap FIR filter with nonzero coefficients g_k only in the range $0 \leq k \leq N-1$, and with z -transform $G(z) = \sum_{k=0}^{N-1} g_k z^{-k}$. If $\tilde{\mathbf{g}}$ is one of the permutations of \mathbf{g} in Table 6.2, then the z -transform of $\tilde{\mathbf{g}}$ is given in the rightmost column. The proofs are left to Exercise 6.19.

TABLE 6.2 Filter Permutations and Corresponding z -Transforms

$\tilde{\mathbf{g}}$ Coefficients	z -Transform $\tilde{G}(z)$
$\tilde{g}_k = (-1)^k g_k$	$G(-z)$
$\tilde{g}_k = g_{-k}$	$G(z^{-1})$
$\tilde{g}_k = g_{N-1-k}$	$z^{-N+1} G(z^{-1})$
$\tilde{g}_k = (-1)^k g_{N-1-k}$	$(-z)^{-N+1} G(-z^{-1})$

Remark 6.6 A good “low-pass” filter \mathbf{g} ought to at least pass dc (perhaps scaled) but zero out the Nyquist frequency. Thus we want $\mathbf{g} * \mathbf{x} = C\mathbf{x}$ for some nonzero constant C if $x_k = 1$ for all k , and $\mathbf{g} * \mathbf{x} = \mathbf{0}$ when $x_k = (-1)^k$. If \mathbf{g} is an FIR filter with coefficients g_k for $-m \leq k \leq n$, this means that

$$\sum_{k=-m}^n g_k \neq 0 \quad \text{and} \quad \sum_{k=-m}^n g_k(-1)^k = 0.$$

The equations above can be stated even more compactly: if $G(z)$ is the z -transform of \mathbf{g} , then $G(1) \neq 0$ and $G(-1) = 0$. This condition will be our rough characterization of low-pass filters. We call \mathbf{g} a “high-pass filter” if convolution with \mathbf{g} zeros out dc and passes the Nyquist frequency, again perhaps rescaled or shifted. Similar reasoning to the low-pass case shows that this is equivalent to $G(1) = 0$ and $G(-1) \neq 0$.

These characterizations make it easy to see that the permutations of the first and fourth rows in Table 6.2 turn low-pass filters into high-pass filters, and vice versa. It also is clear that if the analysis filters really are low- and high-pass, then the synthesis filters obtained from equations (6.34) and (6.35) will be of the appropriate type, after possible rescaling.

6.7.4 Filter Design II: Product Filters

The fact that we have four filters to play with leaves considerable leeway in designing a filter bank. One reasonable way to add more structure and ensure that the synthesis filters are FIR is to obtain them as simple modifications of the analysis filters. In particular, if we choose the synthesis filters so that

$$L_s(z) = H_a(-z), \quad H_s(z) = -L_a(-z), \quad (6.36)$$

then equation (6.33) is automatically satisfied. From Remark 6.6 on page 243 this will also ensure that ℓ_s is low-pass if \mathbf{h}_a is high-pass, and similarly that \mathbf{h}_s is high-pass if ℓ_a is low-pass.

According to Remark 6.5 on page 242, equations (6.36) correspond to taking the synthesis filters as

$$(\ell_s)_k = (-1)^k (\mathbf{h}_a)_k, \quad (\mathbf{h}_s)_k = (-1)^{k+1} (\ell_a)_k, \quad (6.37)$$

so that, for example, the synthesis low-pass filter is just the analysis high-pass filter with alternating signs. Clearly, the synthesis filters are FIR if the analysis filters are FIR.

If we use (6.36) to substitute $H_a(z) = L_a(-z)$ and $H_s(z) = -L_a(-z)$ in equation (6.32), we obtain

$$L_a(z)L_s(z) - L_a(-z)L_s(-z) = 2z^{-m}. \quad (6.38)$$

Define $P(z) = L_a(z)L_s(z)$ so that (6.38) becomes

$$P(z) - P(-z) = 2z^{-m}. \quad (6.39)$$

The expression $P(z) - P(-z)$ is an odd function of z ; if filters with this structure exist, m must be odd. Also $P(z)$ will itself correspond to a low-pass filter, called the *product filter*, since $P(z)$ corresponds to concatenating the low-pass filters ℓ_a and ℓ_s . This yields a prospective approach for finding suitable filters: choose an odd value for m and design a low-pass filter \mathbf{p} with z -transform $P(z)$ satisfying equation (6.39); then factor

$$P(z) = L_a(z)L_s(z) \quad (6.40)$$

to obtain low-pass analysis and synthesis filters. We obtain the high-pass filters from (6.37).

It's easy to see that equation (6.39) requires that the function $P(z)$ contain no odd powers of z except for z^{-m} . However, the equation imposes no conditions at all on the coefficients of the even powers of z in $P(z)$.

■ EXAMPLE 6.13

Let's design a causal filter \mathbf{p} with nonzero coefficients p_0 , p_1 , p_2 , and p_3 whose z -transform $P(z)$ satisfies equation (6.39) with $m = 1$. Start with

$$P(z) = p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}.$$

Equation (6.39) becomes

$$P(z) - P(-z) = 2p_1 z^{-1} + 2p_3 z^{-3} = 2z^{-1},$$

which forces $p_1 = 1$ and $p_3 = 0$. The coefficients p_0 and p_2 can be "anything." However, according to Remark 6.6 on page 243 a low-pass filter ought to satisfy $P(-1) = 0$. This condition forces

$$p_0 - 1 + p_2 = 0.$$

One obvious choice is $p_0 = p_2 = \frac{1}{2}$, which gives

$$P(z) = \frac{1}{2} + z^{-1} + \frac{1}{2} z^{-2}.$$

We can then factor

$$P(z) = \frac{(1+z)^2}{2z^2}$$

and obtain $L_a(z) = L_s(z) = \frac{1}{\sqrt{2}}(1 + z^{-1})$. This is only one possible factorization; we could, for example, take $L_a(z) = 1 + z^{-1}$ and $L_s(z) = (1 + z^{-1})/2$. The analysis filter transforms are, from equation (6.36), $H_a(z) = \frac{1}{\sqrt{2}}(-1 + z^{-1})$ and $H_s(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$, which correspond to the orthogonal Haar filters from Example 6.5. The low-pass filters really are low-pass since $L_a(-1) = L_s(-1) = 0$.

6.7.5 Filter Design III: More Product Filters

In this subsection we'll continue with the choice of equation (6.36) and (6.37) that led to the design strategy based on equation (6.40).

In Example 6.13 above we could just as well have chosen $p_0 = \frac{3}{4}$ and $p_2 = \frac{1}{4}$, to obtain $P(z) = \frac{3}{4} + z^{-1} + \frac{1}{4}z^{-2}$. Then we can factor to obtain, for example, $L_a(z) = \frac{1}{2} + \frac{1}{2}z^{-1}$ and $L_s(z) = \frac{3}{2} + \frac{1}{2}z^{-1}$. In this case, however, $L_s(-1) = 1$, so the synthesis filter ℓ_s isn't a low-pass filter (though the filter bank will still give perfect reconstruction). As a result the analysis filter \mathbf{h}_a isn't high-pass. If it is desired to obtain true low-pass filters, then we need to make sure we can factor $P(z)$ into pieces $L_a(z)$ and $L_s(z)$ such that $L_a(-1) = L_s(-1) = 0$. One way to do this is to build a factor $(1 + z^{-1})^n$ with $n \geq 2$ into $P(z)$ so that we can arrange for both L_a and L_s to contain some positive power of $(1 + z^{-1})$ and obtain $L_a(-1) = L_s(-1) = 0$.

To accomplish this, we take

$$P(z) = (1 + z^{-1})^{2r} Q(z), \quad (6.41)$$

where $Q(z)$ is a polynomial to be chosen and $r \geq 1$. We can determine the appropriate degree for Q as follows: The binomial $(1 + z^{-1})^{2r}$ contains powers of z from z^{-2r} to z^0 . So if Q is of n th degree, then $P(z)$ will contain powers of z from z^{-2r} to z^n , $2r + n + 1$ powers in all. Thus $P(z) - P(-z)$ will contain all odd powers of z between z^{-2r} to z^n . If n is even, this is a total of $r + n/2$ powers of z . Then the equation $P(z) - P(-z) = 2z^{-m}$ yields $r + n/2$ linear conditions on the $n + 1$ coefficients of Q . In order to solve uniquely, we require $r + n/2 = n + 1$, or $n = 2r - 2$. We thus seek a polynomial Q of degree $2r - 2$ to use in equation (6.41) so that $P(z) - P(-z) = 2z^{-m}$ is satisfied.

■ EXAMPLE 6.14

Let $m = 1$ in $P(z) - P(-z) = 2z^{-m}$ and take $r = 2$. We want a quadratic polynomial $Q(z) = q_0 + q_1z + q_2z^2$ so that if $P(z) = (1 + z^{-1})^4 Q(z)$, then $P(z) - P(-z) = 2z^{-1}$. This last equation, when expanded, leads to

$$(2q_1 + 8q_2)z + (8q_0 + 12q_1 + 8q_2)z^{-1} + (8q_0 + 2q_1)z^{-3} = 2z^{-1}.$$

Matching powers of z leads to three linear equations, $2q_1 + 8q_2 = 0$, $8q_0 + 12q_1 + 8q_2 = 2$, $8q_0 + 2q_1 = 0$ in three unknowns q_0, q_1, q_2 . The solution is

$q_0 = -\frac{1}{16}$, $q_1 = \frac{1}{4}$, $q_2 = -\frac{1}{16}$. We find that

$$P(z) = \frac{1}{16}(-z^2 + 9 + 16z^{-1} + 9z^{-2} - z^{-4}).$$

From (6.41) we see that $P(z)$ factors completely as

$$P(z) = -\frac{1}{16}(1+z^{-1})^4(z^2 - 4z + 1) = -\frac{1}{16}(1+z^{-1})^4(z-r_1)(z-r_2) \quad (6.42)$$

where $r_1 = 2 - \sqrt{3}$, $r_2 = 1/r_1 = 2 + \sqrt{3}$.

There are now many possible ways to allocate the various linear factors (and constant out front) to $L_a(z)$ and $L_s(z)$ so that $L_a(z)L_s(z) = P(z)$ in equation (6.42). For example, we might take

$$\begin{aligned} L_a(z) &= -\frac{1}{8}(1+z^{-1})^2(z^2 - 4z + 1) = -\frac{1}{8}z^2 + \frac{1}{4}z + \frac{3}{4} + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}, \\ L_s(z) &= \frac{1}{2}(1+z^{-1})^2 = \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2}. \end{aligned}$$

From equation (6.36) then

$$\begin{aligned} H_a(z) &= L_s(-z) = \frac{1}{2} - z^{-1} + \frac{1}{2}z^{-2}, \\ H_s(z) &= -L_a(-z) = \frac{1}{8}z^2 + \frac{1}{4}z - \frac{3}{4} + \frac{1}{4}z^{-1} + \frac{1}{8}z^{-2}. \end{aligned}$$

The corresponding filter coefficients are given in Table 6.3. The function $L_s(z)$ (and hence $H_a(z)$) does not correspond to a causal filter, but if we redefine $L_s(z)$ by multiplying z^{-2} (which also multiplies $H_a(z)$ by z^{-2}), we obtain causal filters, and also see that now $P(z) - P(-z) = z^{-3}$. With this alteration the filters yield a perfect reconstruction filter bank with delay $m = 3$ in equation (6.32). It's also convenient to multiply both high-pass filters by -1 . These filters are known as the "Le Gall 5/3" filters and are used in the JPEG 2000 compression standard.

Another way we might allocate the factors in equation (6.42) is as

$$\begin{aligned} L_a(z) &= \frac{(\sqrt{3}+1)\sqrt{2}}{8}(1+z^{-1})^2(z-r_1), \\ L_s(z) &= -\frac{(\sqrt{3}-1)\sqrt{2}}{8}(1+z^{-1})^2(z-r_2) \end{aligned}$$

TABLE 6.3 Filter Coefficients for Le Gall 5/3 Filters

Index	-2	-1	0	1	2
ℓ_a	$-\frac{1}{8}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{4}$	$-\frac{1}{8}$
\mathbf{h}_a	0	0	$\frac{1}{2}$	-1	$\frac{1}{2}$
ℓ_s	0	0	$\frac{1}{2}$	1	$\frac{1}{2}$
\mathbf{h}_s	$\frac{1}{8}$	$\frac{1}{4}$	$-\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{8}$

TABLE 6.4 Filter Coefficients for Daubechies 4-Tap Filters

Index	0	1	2	3
ℓ_a	$\frac{1 + \sqrt{3}}{4\sqrt{2}}$	$\frac{3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{3 - \sqrt{3}}{4\sqrt{2}}$	$\frac{1 - \sqrt{3}}{4\sqrt{2}}$
\mathbf{h}_a	$\frac{1 - \sqrt{3}}{4\sqrt{2}}$	$\frac{-3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{-1 - \sqrt{3}}{4\sqrt{2}}$
ℓ_s	$\frac{1 - \sqrt{3}}{4\sqrt{2}}$	$\frac{3 - \sqrt{3}}{4\sqrt{2}}$	$\frac{3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{1 + \sqrt{3}}{4\sqrt{2}}$
\mathbf{h}_s	$\frac{-1 - \sqrt{3}}{4\sqrt{2}}$	$\frac{3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{-3 + \sqrt{3}}{4\sqrt{2}}$	$\frac{1 - \sqrt{3}}{4\sqrt{2}}$

(the product of the constants in front is in fact $-1/16$). When expanded, these yield

$$L_a(z) = \frac{1}{4\sqrt{2}}((1 + \sqrt{3})z + (3 + \sqrt{3}) + (3 - \sqrt{3})z^{-1} + (1 - \sqrt{3})z^{-2}),$$

$$L_s(z) = \frac{1}{4\sqrt{2}}((1 - \sqrt{3})z + (3 - \sqrt{3}) + (3 + \sqrt{3})z^{-1} + (1 + \sqrt{3})z^{-2}).$$

We obtain $H_a(z)$ and $H_s(z)$ from equation (6.36). The corresponding filters are not causal, but as in the previous example if causality is desired, we can shift all of the filters. In this case multiply each z -transform by z^{-1} , which is equivalent to shifting the filter vector one index to the right. Then $P(z)$ satisfies $P(z) - P(-z) = 2z^{-3}$, and the resulting filter bank has delay 3. The filters have the coefficients shown in Table 6.4 and yield a perfect reconstruction filter bank with delay $m = 3$ in equation (6.32). These are known as the “Daubechies 4-tap” filters. The resulting analysis and synthesis filters are reversals of each other, if we shift the synthesis filters three indexes to the left (and note equation (6.10)), thus the Daubechies 4-tap filters yield an orthogonal filter bank.

Exercise 6.24 looks at another filter obtained by an alternate splitting of $P(z)$ in equation (6.42).

6.7.6 Orthogonal Filter Banks

Design Equations for an Orthogonal Bank In an orthogonal filter bank the synthesis filters ℓ_s and \mathbf{h}_s satisfy the relations (6.10). If the analysis low-pass filter ℓ_a has nonzero components $(\ell_0, \dots, \ell_{N-1})$, then ℓ_s has nonzero components $(\ell_{N-1}, \dots, \ell_0)$ (indexes $-N+1$ to 0) and so from Table 6.2 in Remark 6.5 on page 242 we have $L_s(z) = L_a(z^{-1})$ and $H_s(z) = H_a(z^{-1})$. In this case equations

(6.32) and (6.33) become

$$L_a(z)L_a(z^{-1}) + H_a(z)H_a(z^{-1}) = 2z^{-m}, \quad (6.43)$$

$$L_a(-z)L_a(z^{-1}) + H_a(-z)H_a(z^{-1}) = 0. \quad (6.44)$$

Note that we are no longer requiring condition (6.36). Equations (6.43) and (6.44) are the z -transform frequency domain conditions for a perfect reconstruction orthogonal filter bank.

When the filter length N of ℓ_a is even, there is an easy choice for the high-pass analysis filter, which automatically ensures that equation (6.44) is satisfied. Specifically, we choose the filter coefficients of \mathbf{h}_a in terms of ℓ_a as

$$(\mathbf{h}_a)_k = (-1)^k (\ell_a)_{N-1-k}. \quad (6.45)$$

In short, \mathbf{h}_a is a reversed version of ℓ_a with alternating signs. Since ℓ_a is assumed to have nonzero entries $(\ell_a)_k$ only for $0 \leq k \leq N-1$, the same is true for \mathbf{h}_a . With this choice for \mathbf{h}_a we have from the Table 6.2 that $H_a(z) = (-z)^{-(N-1)} L_a(-z^{-1}) = -z^{-(N-1)} L_a(-z^{-1})$, since N is even. In this case equation (6.44) is satisfied identically for

$$L_a(-z)L_a(z^{-1}) - z^{-(N-1)} L_a(z^{-1})(z^{-1})^{-(N-1)} L_a(-z) = 0.$$

The choice (6.45) will work for any ℓ_a with an even number of taps. In this case the filters ℓ_a and \mathbf{h}_a are an example of *quadrature mirror filters*; see Exercise 6.25.

The design of the filter bank now comes down to just $L_a(z)$ and equation (6.43), which then reads

$$L_a(z)L_a(z^{-1}) + L_a(-z^{-1})L_a(-z) = 2z^{-m} \quad (6.46)$$

(since $(-z)^{-(N-1)}(-z^{-1})^{-(N-1)} = 1$).

The Product Filter in the Orthogonal Case Define $P(z)$ as

$$P(z) = L_a(z)L_a(z^{-1}) \quad (6.47)$$

so that (6.46) becomes

$$P(z) + P(-z) = 2z^{-m}. \quad (6.48)$$

(Compare this to the reasoning which led up to equation (6.39).) The function $P(z)$ corresponds to the *product filter* obtained by convolving ℓ_a with its own time reversal.

Since the left side of (6.48) is an even function of z , m must be even. Indeed, from (6.47), we see that $P(z) = P(z^{-1})$, so

$$P(z) + P(-z) = P(z^{-1}) + P(-z^{-1}) = 2(z^{-1})^{-m} = 2z^m.$$

Comparison with (6.48) shows that m must be zero, and hence

$$P(z) + P(-z) = 2. \quad (6.49)$$

Filters that come from $P(z)$ and satisfy condition (6.49) are called *halfband filters*; see [24].

From equation (6.49) we can also conclude that all coefficients of the even powers of z in $P(z)$ must be zero, except for the z^0 (constant) term, which is 1. Equation (6.49) imposes no conditions on the coefficients of the odd powers of z . Now from the beginning we have assumed that ℓ_a is causal, so $L_a(z)$ will be of the form $L_a(z) = \ell_0 + \ell_1 z^{-1} + \cdots + \ell_{N-1} z^{-(N-1)}$. Our goal is thus to produce such an $L_a(z)$ so that $P(z) = L_a(z)L_a(z^{-1})$ satisfies equation (6.49).

We also desire that $L_a(-1) = 0$, so that ℓ_a is a low-pass filter, and this forces $P(-1) = 0$. From equation (6.49) this implies that $P(1) = 2$, and so $L_a(1) = \sqrt{2}$. Equivalently, we require

$$\sum_{k=0}^{N-1} \ell_k = \sqrt{2}. \quad (6.50)$$

The converse is also true; that is, if equation (6.50) holds, then $L(1) = \sqrt{2}$, which forces $P(-1) = 0$ and $L_a(-1) = 0$.

We could seek to impose other constraints, for example, on the length or frequency response of ℓ_a .

Restrictions on $P(z)$; Spectral Factorization One might attempt to simply start with a function

$$P(z) = p_{N-1} z^{-(N-1)} + \cdots + p_1 z^{-1} + 1 + p_1 z + \cdots p_{N-1} z^{N-1} \quad (6.51)$$

of the required form, involving only odd powers of z (e.g., with $P(-1) = 0$) and then obtain ℓ_a by factoring $P(z) = L_a(z)L_a(z^{-1})$ where L_a has real coefficients. However, this is not generally possible; as it turns out, the function P must satisfy a certain condition. Specifically, let $z = e^{i\omega}$ for $\omega \in [-\pi, \pi]$. Observe that $\bar{z} = z^{-1}$ in this case so that

$$L_a(z^{-1}) = L_a(\bar{z}) = \overline{L_a(z)}$$

(the last equality follows since L_a has real coefficients). We thus find that $P(z) = L_a(z)L_a(z^{-1}) = L_a(z)\overline{L_a(z)} = |L_a(z)|^2 \geq 0$ if $z = e^{i\omega}$. In short, $P(e^{i\omega})$ must be real and nonnegative if a factorization $P(z) = L_a(z)L_a(z^{-1})$ exists.

The converse is also true. A function $P(z)$ of the form (6.51) (which may also contain even powers of z) that satisfies $P(e^{i\omega}) \geq 0$ for $\omega \in [-\pi, \pi]$ can be factored as $P(z) = L_a(z)L_a(z^{-1})$ where L_a has real coefficients. We sketch a proof in Exercise 6.28, and show how to actually perform the factorization constructively for modest values of N (see also [11], p. 172, or [24], sec. 5.4). The task of determining $L_a(z)$ from $P(z)$ is often referred to as *spectral factorization*. The factorization is not unique.

Our overall goal then is to design a function $P(z)$ of the form (6.51) with $P(e^{i\omega}) \geq 0$ and $P(-1) = 0$, and perhaps other desirable properties, and then to obtain L_a by performing a spectral factorization.

Daubechies Filters Rather than look at general techniques for designing orthogonal filters, we'll content ourselves with an examination of an important subset of orthogonal filters, the *Daubechies* filters. For these filters we have, for any even $N \geq 2$,

$$P(z) = Q\left(\frac{z + z^{-1}}{2}\right), \quad (6.52)$$

where Q is the polynomial defined by

$$Q(x) = c \int_{-1}^x (1 - y^2)^{N/2-1} dy \quad (6.53)$$

and c is chosen according to $1/c = \int_{-1}^0 (1 - y^2)^{N/2-1} dy$ (this choice for c yields $Q(0) = 1$). For the derivation of this formula see Exercise 6.29. In particular, $P(-1) = Q(-1) = 0$; indeed $Q(x)$ is designed to vanish to the highest order possible at $x = -1$. Such filters are called “maxflat” filters. The function P inherits this property. It's also not hard to check that Q and P possess only odd powers of the input argument (except for a constant term), and that $P(e^{i\omega}) \geq 0$.

■ EXAMPLE 6.15

When $N = 2$, we obtain $Q(x) = x + 1$ (here $c = 1$) from equations (6.52) and (6.53) and

$$P(z) = \frac{1}{2}z^{-1} + 1 + \frac{1}{2}z.$$

It's simple to see $P(z) + P(-z) = 2$; also $P(-1) = 0$. We can factor $P(z) = L_a(z)L_a(z^{-1})$, where

$$L_a(z) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}z^{-1},$$

corresponding to the Haar low-pass analysis filter $\ell_a = (1/\sqrt{2}, 1/\sqrt{2})$. We obtain $\mathbf{h}_a = (1/\sqrt{2}, -1/\sqrt{2})$ from equation (6.45), and the synthesis filters from equations (6.10).

■ EXAMPLE 6.16

When $N = 4$, we obtain from equations (6.52) and (6.53) that $Q(x) = 1 + \frac{3}{2}x - \frac{1}{2}x^3$ (here $c = \frac{3}{2}$) and

$$P(z) = -\frac{1}{16}z^{-3} + \frac{9}{16}z^{-1} + 1 + \frac{9}{16}z - \frac{1}{16}z^3,$$

previously found in Example 6.14 (but multiplied by z^{-1} there). We can then compute $P(z) = L_a(z)L_a(z^{-1})$, where

$$\begin{aligned} L_a(z) &= \frac{1+\sqrt{3}}{4\sqrt{2}}(1+z^{-1})^2(1-(2-\sqrt{3})z^{-1}) \\ &\approx 0.483z^{-3} + 0.837z^{-2} + 0.224z^{-1} - 0.129. \end{aligned}$$

The other analysis and synthesis filter coefficients are obtained as in the previous example.

Equations (6.52) and (6.53) yield an infinite family of orthogonal filters, one for each even $N \geq 2$. There are many other families of orthogonal filters as well, many of which are built into Matlab's Wavelet Toolbox.

6.8 MATLAB PROJECT

This section contains Matlab explorations involving the use of filter banks/DWTs for analyzing signals and images.

6.8.1 Basics

1. Start Matlab and create an artificial piecewise constant signal with

```
y = zeros(256,1);
y(100:160) = 1.0; y(161:256) = 4.0;
```

As mentioned in the text, Matlab can perform DWTs under a variety of extension modes. For the moment we'll use the default extension mode, symmetric periodic extension according to equation (6.11).

We can perform a one-stage wavelet decomposition of the signal with

```
[cA, cD] = dwt(y, 'bior2.2');
```

Here “cA” is a vector of the approximation coefficients (what we called \mathbf{X}_l in the text) and “cD” the detail coefficients (what we called \mathbf{X}_h). The vectors will be slightly longer than half the length of the input signal, since Matlab keeps some extra coefficients, but it doesn’t matter. The “bior2.2” argument specifies the wavelets/filters to be used, which in this case correspond to the Le Gall 5/3 wavelet from the text. You can execute `waveinfo('bior')` to see other biorthogonal wavelets that are available, or `waveinfo()` to see information on all available wavelets/filters.

Plot the vectors `cA` and `cD`. Note that `cA` is the low-pass filtered portion of the transform. Since the signal is mostly constant, `cA` looks like an approximately half-length version of the original signal with some distortion at the jumps. The vector `cD` should be mostly zeros except at the discontinuity, which contains high-frequency energy that makes it through the high-pass filter. It’s also worth noting that since the signal is implicitly extended at the boundary by symmetric reflection, there are no discontinuities there.

2. We can reconstruct an approximation to the original full length using only the coefficients `cA` by executing

```
y2 = upcoef('a', cA, 'bior2.2', 1);
```

The first “a” indicates that we are reconstructing from approximation coefficients only and the final “1” indicates `cA` comes from a one-stage transform. In this case `y2` is computed by applying the synthesis filter bank to `cA` alone, with `cD` implicitly set to zero. Of course, you should plot `y2`.

3. Change the extension mode to periodic extension ($\tilde{x}_k = x_{k \bmod N}$) with the command

```
dwtmode('per');
```

Repeat steps 2 through 3 above. Periodic extension introduces discontinuities at the signal boundaries, and this should be obvious in the plots of `cA`, `cD`, and the reconstruction `y2` because the values from each end of the signal now “pollute” the values from the other end.

4. Repeat the steps 2 through 4 for a few other wavelets.

6.8.2 Audio Signals

1. First, set the extension mode back to symmetric extension with the command

```
dwtmode('sym');
```

Load in the “splat” signal with the command `load('splat')`; Recall that the audio signal is loaded into a variable “y” and the sampling rate into “Fs”. You can play the sound with `sound(y)`.

2. Use the `dwt` command and the “`bior2.2`” wavelets as above to perform a one-stage transform of the audio signal, and then reconstruct from the vector `cA` alone using the `upcoef` command. Play the sound through the speaker.
3. Let’s try some thresholding for signal compression. The `wavedec` and `waverec` commands perform multistage wavelet analysis and reconstructions, but the format of the output of `wavedec` isn’t so easy to interpret if we want to threshold. Instead we can use the `wdencmp` command. Execute

```
[ycomp,cxc,lxc,perfo,perf12]
= wdencmp('gbl',y,'bior2.2',5,0.05,'h',1);
```

The input arguments are “`gbl`”, which is global thresholding (“all” portions of the DWT coefficients will be subject to thresholding), “`y`” is the input signal, “`bior2.2`” dictates the wavelet/filters, “`5`” is the number of stages in the transform, and “`0.05`” is the threshold parameter. The “`h`” is for “hard” thresholding (any value not exceeding the threshold in magnitude is zeroed out). The final “`1`” indicates that the lowest level approximation coefficients are not subject to thresholding (change it to “`0`” to subject all coefficients to the threshold). The outputs are “`ycomp`”, which is the signal reconstructed from the thresholded DWT, “`cxc`” and “`lxc`” are certain bookkeeping details we don’t need to worry about, “`perfo`” specifies what percentage of the wavelet coefficients from the analysis bank output were zeroed out (a measure of how much the signal was compressed), and “`perf12`” is the L^2 distortion of the thresholded image, as a percentage.

Play around with the thresholding parameter; try the range 0.0 to 1.0. Make a table that shows in each case the compression and distortion parameters, and your subjective rating of the sound quality.

4. Experiment with other wavelets!

6.8.3 Images

1. Load an image into Matlab with

```
z = imread('myimage.jpg');
```

Construct an artificial grayscale image `zg` as in previous chapters, along with the colormap

```
L = 255;
colormap([(0:L)/L; (0:L)/L; (0:L)/L]);
```

View the image with `image(zg)`.

2. Execute the command

```
[cA, cH, cV, cD] = dwt2(zg, 'bior2.2');
```

that performs a two-dimensional DWT. The arrays `cA`, `cH`, `cV`, `cD` now hold the low- and high-frequency components of the transform; `cA` contains the low/low elements in the horizontal/vertical directions, the approximation coefficients. The array `cH` is the horizontal detail, corresponding to high/low in the horizontal/vertical directions, while `cV` is vertical detail (low/high) and `cD` is the “diagonal” detail, corresponding to high/high. Each of these arrays is $M/2$ by $N/2$, if the original image is M by N , though they might be a bit larger since Matlab extends the image by a certain amount.

3. You can display `cA` or the other pieces directly, although they’re only half-sized. For a fairer picture, execute

```
zA = upcoef2('a', cA, 'bior2.2', 1);
```

to upsample `cA` to a full-sized array `zA` (reconstructing with `cH`, `cV`, `cD` implicitly set to zero). Display `zA` (you might need to scale it a bit if you want to display on a 0 to 255 grayscale, by adding a suitable constant and then multiplying by a scalar). Compare to the original image.

Do the same with the other three pieces of the transform, with the commands

```
zH = upcoef2('h', cH, 'bior2.2', 1);
zV = upcoef2('v', cV, 'bior2.2', 1);
zD = upcoef2('d', cD, 'bior2.2', 1);
```

In each case display the output. Again, rescaling may be helpful.

4. As with audio signals, we can perform multistage decompositions and thresholding. In particular, execute

```
[yz, cxc, lxc, perfo, perf12]
= wdencmp('gbl', zg, 'bior2.2', 5, 10.0, 'h', 1);
```

(input and output arguments as described in the audio signal section; the command is still called `wdencmp`, even if the argument is 2D). We’ve began with the threshold parameter at 10.0. Record the L^2 distortion parameter `perf12`, and the percentage of zeroed coefficients `perfo`.

Try threshold parameters in the range 0 to 500. Table your results, showing in each case the quantities `perfo`, `perf12`, and your rating of the image quality.

Again, try different wavelets!

6.9 ALTERNATE MATLAB PROJECT

This section contains Matlab explorations in which we use filter banks/DWTs for analyzing signals and images, but with various supplied commands instead of Matlab's Wavelet Toolbox. This project closely parallels that of the previous Matlab project.

6.9.1 Basics

1. Start Matlab, and create an artificial piecewise constant signal with

```
y = zeros(256,1);  
y(100:160) = 1.0; y(161:256) = 4.0;
```

The supplied routines for DWTs assume periodic extension of all signals, which is sufficient for our purposes. We can perform a one-stage wavelet decomposition of the signal with

```
Y = wavel(y, 'd4');
```

Here Y consists of the approximation coefficients, stored in $Y(1:N/2)$ (where N is the length of the signal, in this case $N = 256$). The detail coefficients are stored in $Y(N/2+1:N)$. The "d4" argument specifies the Daubechies 4-tap orthogonal filters; other options are "haar", "orthhaar", and "legall." You can easily add your own.

Plot the vectors $Y(1:128)$ and $Y(129:256)$. Note that $Y(1:128)$ is the low-pass filtered portion of the transform, and since the signal is mostly constant, $Y(1:128)$ looks like a half-length version of the original signal, with some distortion at the jumps. The vector $Y(129:256)$ should be mostly zero except at the discontinuity, which contains high-frequency energy that makes it through the high-pass filter. It's also worth noting that since the signal is implicitly extended periodically, there is some "crosstalk" between the ends of the signal.

2. We can reconstruct an approximation to the original full length using only the approximation coefficients $Y(1:128)$ by executing

```
Y(129:256) = 0.0;  
y2 = invwave1(Y, 'd4');
```

Of course, you should plot $y2$.

3. Repeat the steps 2 and 3 for the other available wavelets.

6.9.2 Audio Signals

1. Load in the “splat” signal with the command `load ('splat');`. Recall that the audio signal is loaded into a variable “y” and the sampling rate into “Fs.” You can play the sound with `sound (y)`.
2. Use the `wave1` command (you need to truncate the signal to even length; e.g., use $N = 10000$ and vector `y (1 : 10000)`) and the “d4” wavelets as above to perform a one-stage transform of the audio signal, and then reconstruct from the vector `Y (1:N/2)` alone. Play the sound through the speaker.
3. Let’s try some thresholding for signal compression. Let `y1=y (1 : 10000)`; and perform a four-stage wavelet decomposition with

```
Y = fullwave (y, 'd4', 4);
```

Then threshold Y with threshold parameter 0.01 via

```
Y2 = Y .* (abs (Y) > 0.01);
```

Compute the percentage of wavelet coefficients zeroed out by using the command `sum (Y2>0) / 10000`. Inverse transform as

```
y2 = invfullwave (Y2, 'd4', 4);
```

and play the sound `y2`.

Play around with the thresholding parameter; try the range 0.0 to 1.0. Make a table that shows in each case the compression and distortion parameters, and your rating of the sound quality.

4. Experiment with other wavelets!

6.9.3 Images

1. Load an image into Matlab with

```
z = imread ('myimage.jpg');
```

If necessary, crop the image so that the number of rows and columns is divisible by 8. Construct an artificial grayscale image `zg`, and set the colormap to

```
L = 255;
colormap ([(0:L)/L; (0:L)/L; (0:L)/L]');
```

View the image with `image (zg)`.

2. Execute the command

```
Z = imwave1(zg, 'd4');
```

which performs a two-dimensional single-stage DWT. You can display Z directly, or logarithmically scaled, for example, `image(log(1+abs(Z))*30)`.

3. You can display the approximation coefficients or the other pieces directly, although they're only half-sized. For a fairer picture, execute

```
[m,n] = size(Z)
Z(m/2+1:m,:) = 0.0; Z(:,n/2+1:n) = 0.0;
z2 = invimwave1(Z, 'd4');
image(z2);
```

to zero out all but the lowest level approximation coefficients (c_A) and then inverse transform. This yields the level 1 approximation α_1 . Compare it to the original image.

Recompute Z, and do the same with the other three pieces of the transform, that is, the vertical, horizontal, and diagonal details. Rescaling prior to displaying may be helpful.

4. As with audio signals, we can perform multistage decompositions and thresholding. In particular, execute

```
Z = imwavefull(zg, 3, 'd4');
image(log(1 + abs(Z))*30);
```

Try zeroing out all but the lowest $\frac{1}{4}$ of the coefficients in each direction, with $Z(m/4+1:m,:) = 0$; and $Z(:,n/4+1:n) = 0.0$; then inverse transform with

```
z2 = invimwavefull(Z, 3, 'd4');
```

This will produce the level two approximation α_2 . Executing the command $Z(m/8+1:m,:) = 0$; and $Z(:,n/8+1:n) = 0.0$; and inverse transforming yields α_3 , and so on.

5. Try recomputing the three-stage transform Z, thresholding, and inverse transforming, for example,

```
Z = imwavefull(zg, 3, 'd4');
Z2 = Z.* (abs(Z) > 10);
z2 = invimwavefull(Z2, 3, 'd4');
```

then display \mathbf{z}_2 . Vary the threshold parameter. In each case record the percentage of wavelet coefficients zeroed out and the distortion.

Try different wavelets!

EXERCISES

Computation and the Haar Filter Bank

- 6.1 Let $\mathbf{x} \in L^2(\mathbb{Z})$ be a signal with components $x_0 = 1, x_1 = -2, x_2 = 2, x_3 = 4$, and all other components zero. Run \mathbf{x} through the Haar filter bank with filter coefficients $(\ell_a)_0 = (\ell_a)_1 = \frac{1}{2}, (\mathbf{h}_a)_0 = \frac{1}{2}, (\mathbf{h}_a)_1 = -\frac{1}{2}$ to compute \mathbf{X}_ℓ and \mathbf{X}_h explicitly. Then use the synthesis filters with coefficients $(\ell_s)_{-1} = (\ell_s)_0 = 1, (\mathbf{h}_s)_{-1} = -1, (\mathbf{h}_s)_0 = 1$ to reconstruct \mathbf{x} .
- 6.2 Consider a filter bank that uses the Haar analysis filters with coefficients $(\ell_a)_0 = (\ell_a)_1 = \frac{1}{2}, (\mathbf{h}_a)_0 = \frac{1}{2}, (\mathbf{h}_a)_1 = -\frac{1}{2}$, and corresponding causal synthesis filters $(\ell_s)_0 = (\ell_s)_1 = 1, (\mathbf{h}_s)_0 = -1, (\mathbf{h}_s)_1 = 1$ (yes, the $(\mathbf{h}_s)_0$ is -1 ; the synthesis filters here are shifted from the last problem). Show that this filter bank yields perfect reconstruction with output delay 1 by considering an input vector $\mathbf{x} = (\dots, x_{-1}, x_0, x_1, \dots)$ and computing the filter bank output $\tilde{\mathbf{x}}$ directly (filter, downsample, upsample, synthesis filter). In particular, show that $\tilde{x}_k = x_{k-1}$. Hint: Most of the work has already been done in equations (6.5).
- 6.3 Let \mathbf{x} be a signal with components

$$x_k = \begin{cases} 5 + 3(-1)^k, & k \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

The “5” portion is dc and $3(-1)^k$ is at the Nyquist frequency. Use the Haar analysis filters as defined by equation (6.1) to compute the components of the Haar transform vectors $\mathbf{X}_\ell = D(\mathbf{x} * \ell)$ and $\mathbf{X}_h = D(\mathbf{x} * \mathbf{h})$ explicitly. In particular, look at the components $(\mathbf{X}_\ell)_k$ and $(\mathbf{X}_h)_k$ and comment on why this makes perfect sense. Note that even though \mathbf{x} is not in $L^2(\mathbb{Z})$, the convolution with the FIR Haar filters is still well-defined.

- 6.4 Let $\mathbf{x} \in L^2(\mathbb{Z})$ be a signal with components

$$x_k = \begin{cases} 0, & k < 0, \\ 1, & 0 \leq k < 50, \\ 3, & 50 \leq k < 100, \\ 0, & 100 \leq k. \end{cases}$$

Use the Haar analysis filters as defined by equation (6.1) to compute the components of the Haar transform vectors $\mathbf{X}_\ell = D(\mathbf{x} * \ell)$ and $\mathbf{X}_h = D(\mathbf{x} * \mathbf{h})$ explicitly. Comment on what you see!

- 6.5** Write simple Matlab routines that implement the Haar analysis and synthesis filter banks (say with analysis filters $\ell_a = (\frac{1}{2}, \frac{1}{2})$, $\mathbf{h}_a = (\frac{1}{2}, -\frac{1}{2})$, synthesis filters $\ell_s = (1, 1)$, $\mathbf{h}_s = (-1, 1)$) for signals $\mathbf{x} \in L^2(\mathbb{Z})$ with $x_k = 0$ for $k < 0$, \mathbf{x} of finite length. Use Matlab's `conv` command for the convolutions.

General Filter Banks

- 6.6** Consider a filter bank for signals in $L^2(\mathbb{Z})$ with analysis low-pass filter coefficients $(\ell_a)_0 = 1$ and all other coefficients zero, while $(\mathbf{h}_a)_1 = 1$ and all other high-pass coefficients zero. These are not low-pass or high-pass—indeed ℓ_a is the “identity filter” and \mathbf{h}_a is simply a delay—but that doesn’t matter.
- Explicitly compute the components of $\mathbf{X}_\ell = D(\mathbf{x} * \ell_a)$ and $\mathbf{X}_h = D(\mathbf{x} * \mathbf{h}_a)$ in terms of the components x_j .
 - Explicitly compute the components of the upsampled vectors $U(\mathbf{X}_\ell)$ and $U(\mathbf{X}_h)$.
 - Determine, by inspection, suitable synthesis filters ℓ_s and \mathbf{h}_s so that

$$\ell_s * (U(\mathbf{X}_\ell)) + \mathbf{h}_s * (U(\mathbf{X}_h)) = \mathbf{x}.$$

- d.** Suppose that we take the analysis filter \mathbf{h}_a to have $(\mathbf{h}_a)_2 = 1$ and all other coefficients zero. Repeat parts (a) and (b), and then show in this case that no synthesis filters can give perfect reconstruction.
- 6.7** Let $\mathbf{x} \in L^2(\mathbb{Z})$. Let $\mathbf{g} \in L^2(\mathbb{Z})$ be an FIR filter, and S the shift operator defined in Remark 6.1 on page 207. Show that

$$(S^m(\mathbf{g})) * \mathbf{x} = S^m(\mathbf{g} * \mathbf{x}).$$

- 6.8** Suppose that we choose analysis filters ℓ_a and \mathbf{h}_a with coefficients $(\ell_a)_0 = \frac{3}{4}$, $(\ell_a)_1 = \frac{1}{2}$, $(\mathbf{h}_a)_0 = \frac{2}{3}$, $(\mathbf{h}_a)_1 = -\frac{2}{3}$, all other coefficients zero (ℓ_a isn’t a low-pass filter, but that doesn’t matter). Find appropriate synthesis filters for perfect reconstruction with zero delay. *Hint:* look at Example 6.3. Also 2-tap filters will work.
- 6.9** Consider a “one-channel” filter bank: A signal $\mathbf{x} \in L^2(\mathbb{Z})$ is convolved with some FIR analysis filter \mathbf{f} . Since the filter bank is one-channel, there is no downsampling—we keep every component of the filtered signal (see Remark 6.2 on page 211), so that $\mathbf{X} = \mathbf{x} * \mathbf{f}$. To reconstruct we then apply a synthesis FIR filter \mathbf{g} . We seek perfect reconstruction (possibly with some output delay); that is, we want $\mathbf{g} * (\mathbf{f} * \mathbf{x}) = S^m(\mathbf{x})$ for all inputs \mathbf{x} , where S is the shift operator.
- Show that the only such one-channel filter banks have filters of the form $f_n = a$ for some fixed n and $f_k = 0$ for all $k \neq n$, where a is some nonzero constant. Similarly \mathbf{g} has components the form $g_r = 1/a$ for some fixed r and $g_k = 0$ for all $k \neq r$. Thus the only appropriate filters are simple delays or advances multiplied by a scalar. *Hint:* Use the Convolution Theorem

4.5.2 to show that $F(z)G(z) = z^{-m}$. Why is this impossible for FIR filters of length greater than 1?

- b. Show that if we work with signals in \mathbb{C}^N (instead of $L^2(\mathbb{Z})$) and interpret the shift operator S circularly, then such an appropriate synthesis filter will exist for any analysis filter \mathbf{f} provided that the analysis filter's discrete Fourier transform $\mathbf{F} = DFT(\mathbf{f})$ has all nonzero components. Hint: Look back at Exercise 4.16.

Finite Length Signals

- 6.10** Write out the matrix \mathbf{W}_6^a in equation (6.19) for the Le Gall 5/3 filters from Example 6.4, with coefficients as tabled below in the indicated positions. (It's easy if you notice that the first three rows of \mathbf{W}^6 are just rows 0, 2, and 4 of the circulant matrix for the analysis low-pass filter; a similar observation holds for the other rows). Also write out \mathbf{W}_6^s (equation (6.21)). Compute a few entries of the product $\mathbf{W}_6^a \mathbf{W}_6^s$ to check $\mathbf{W}_6^a \mathbf{W}_6^s = \mathbf{I}_6$.

Index	0	1	2	3	4	5
ℓ_a	$\frac{3}{4}$	$\frac{1}{4}$	$-\frac{1}{8}$	0	$-\frac{1}{8}$	$\frac{1}{4}$
\mathbf{h}_a	$-\frac{1}{2}$	1	$-\frac{1}{2}$	0	0	0
ℓ_s	1	$\frac{1}{2}$	0	0	0	$\frac{1}{2}$
\mathbf{h}_s	$-\frac{1}{4}$	$-\frac{1}{8}$	0	$-\frac{1}{8}$	$-\frac{1}{4}$	$\frac{3}{4}$

- 6.11** Compute and plot the magnitude of the DTFT on the frequency range $-\frac{1}{2} \leq f \leq \frac{1}{2}$ for each of the Le Gall filters tabled in Example 6.14 (use equation (4.16)).
- 6.12** Compute and plot the DTFT for each of the Daubechies 4-tap filters tabled in Section 6.7.5 (use equation (4.16)).
- 6.13** a. Show that the downsampling matrix \mathbf{D} defined by equation (6.18) does in fact downsample vectors $\mathbf{x} \in \mathbb{C}^N$, for example, $(\mathbf{D}\mathbf{x})_k = x_{2k}$.
b. Show that the upsampling matrix \mathbf{U} defined by equation (6.20) does in fact upsample vectors $\mathbf{x} \in \mathbb{C}^N$, for example, $(\mathbf{D}\mathbf{x})_k = x_{k/2}$ if k is even and $(\mathbf{D}\mathbf{x})_k = 0$ if k is odd.
c. Show that $\mathbf{U}^T = \mathbf{D}$. Also, work out the products \mathbf{DU} and \mathbf{UD} explicitly.
- 6.14** a. The matrix \mathbf{W}_4^a for the Haar analysis transform on four-point signals (periodic extension) is given in Example 6.7, more specifically by equation (6.14). Write out the corresponding matrix \mathbf{W}_2^a , and then use equation (6.23) to form the matrix \mathcal{W}_2^a that governs the two-stage Haar transform for four-point signals with periodic extension.
b. Use an analogous procedure and equations (6.17) and (6.24) to find \mathcal{W}_2^s , and then verify directly that $\mathcal{W}_2^a \mathcal{W}_2^s = \mathbf{I}_4$.

- c. Let $\mathbf{x} = (3, 4, 3, 2)$. Use the results of parts (a) and (b) above and equations (6.26) and (6.27) to compute the quantities $\alpha_2(\mathbf{x})$, $\delta_1(\mathbf{x})$, $\delta_2(\mathbf{x})$, and also $\alpha_1(\mathbf{x})$. Verify directly that equation (6.27) is satisfied. Comment on why $\alpha_2(\mathbf{x})$ has the appearance it does! (Suggestion: First compute \mathbf{X} using \mathcal{W}_2^q , and note that $\mathbf{X}_{\ell\ell}$ is just X_0 here, while $\mathbf{X}_{\ell h}$ is X_1 , etc.)

Filter Design

- 6.15** Redo Exercise 6.2 but use equations (6.32) and (6.33), rather than a “brute force” approach.
- 6.16** Solve equations (6.32) and (6.33) for $L_s(z)$ and $H_s(z)$ to obtain equations (6.34) and (6.35).
- 6.17** Let $\ell_a = (1, 1)$, $\mathbf{h}_a = (1, 1, -2)$, $\ell_s = (-\frac{1}{2}, \frac{1}{2}, 1)$, $\mathbf{h}_s = (\frac{1}{2}, -\frac{1}{2})$. Use equations (6.32) and (6.33) to show that these filters yield a perfect reconstruction filter bank. What is the output delay?
- 6.18** Show that in any biorthogonal filter bank the analysis and synthesis filters can be swapped (low-pass with low-pass, high-pass with high-pass) and the resulting filter bank is still biorthogonal. Hint: Look at the conditions for perfect reconstruction.
- 6.19** Prove the four relations in Table 6.2 in Remark 6.5 on page 242.
- 6.20** Suppose that \mathbf{x} and \mathbf{y} are elements of $L^2(\mathbb{Z})$ with x_k nonzero only for $M_1 \leq k \leq N_1$ and y_k nonzero only for $M_2 \leq k \leq N_2$. Show that $(\mathbf{x} * \mathbf{y})_k$ (the k th component of $\mathbf{x} * \mathbf{y}$) is zero outside the finite range $M_1 + M_2 \leq k \leq N_1 + N_2$. Hint: Do it via z -transforms.
- 6.21** Prove Proposition 6.7.1.
- 6.22** Let the analysis filter ℓ_a in a filter bank have coefficients $(\ell_a)_0 = \frac{1}{2}$, $(\ell_a)_1 = 1$, $(\ell_a)_2 = -\frac{1}{2}$, all other coefficients zero (not really low-pass, but that doesn’t matter). Let the high-pass filter have coefficients $(\mathbf{h}_a)_0 = \frac{1}{4}$, $(\mathbf{h}_a)_1 = \frac{1}{2}$, $(\mathbf{h}_a)_2 = \frac{1}{4}$, all other coefficients zero. Use equations (6.32) and (6.33) to find appropriate causal synthesis filters for perfect reconstruction. What is the delay in the filter bank output?
- 6.23** Let $G(z)$ be the z -transform of an FIR filter \mathbf{g} . Show that $z^m G(z)$ is a polynomial in z for sufficiently large m .
- 6.24** Consider splitting $P(z)$ in equation (6.42) as

$$P(z) = L_a(z)L_s(z),$$

where

$$L_a(z) = \frac{1}{4}(1 + z^{-1})^3, \quad L_s(z) = -\frac{1}{4}(1 + z^{-1})(z^2 - 4z + 1).$$

Work out the coefficients of the analysis and synthesis filters, both low and high-pass, the latter by making use of equation (6.36). Verify that the low-pass filters really are low-pass and the high-pass filters really are high-pass (recall Remark 6.6 on page 243).

- 6.25** Filters ℓ_a and \mathbf{h}_a (not necessarily low- and high-pass) are said to be *quadrature mirror filters* if their z-transforms satisfy

$$|L_a(e^{i\theta})| = |H_a(e^{i(\pi-\theta)})|$$

for $0 \leq \theta \leq \pi$, so that the frequency response of each filter is a “mirror” of the other about $\theta = \pi/2$. Show that filters related by the equation (6.45) are quadrature mirror filters.

- 6.26** Suppose that $Q_1(z)$ and $Q_2(z)$ are two Laurent polynomials, say

$$Q_1(z) = \sum_{k=-M}^M a_k z^{-k},$$

$$Q_2(z) = \sum_{k=-M}^M b_k z^{-k}$$

(not all a_k and b_k in this range need be nonzero). Suppose that Q_1 and Q_2 satisfy

$$X(z)Q_1(z) + X(-z)Q_2(z) = X(z)$$

for every z-transform $X(z) = z^q$, $q \in \mathbb{Z}$. All products above are formally well-defined.

- a. Show that $Q_1(z) = 1$ and $Q_2(z) = 0$. *Hint:* You really only need to consider the cases $q = 0$ and $q = 1$.
- b. Use part (a) to show that equations (6.32) and (6.33) are necessary if equation (6.31) is to hold.

- 6.27** Most filters that are used in biorthogonal filter banks are symmetric or antisymmetric. An FIR filter \mathbf{g} with coefficients g_0, \dots, g_{M-1} is said to be *symmetric* if $g_k = g_{M-1-k}$; \mathbf{g} is *antisymmetric* if $g_k = -g_{M-1-k}$.

- a. Show that a filter \mathbf{g} with M taps is symmetric if and only if its z-transform satisfies $G(z) = z^{1-M}G(z^{-1})$. Show that \mathbf{g} is antisymmetric if and only if $G(z) = -z^{1-M}G(z^{-1})$. *Hint:* Use Table 6.2, though it’s easy to write out directly too.
- b. Show that the low-pass filters in a perfect reconstruction filter bank cannot be antisymmetric. *Hint:* If \mathbf{g} is one of the filters, compute $G(1)$.

- c. Show that if both low-pass filters ℓ_a and ℓ_s in a perfect reconstruction filter bank of the form dictated by (6.36) are symmetric, with ℓ_a having M taps and ℓ_s having N taps, then M and N are both even or both odd. Hint: Suppose, in contradiction, that $M + N$ is odd. Use equation (6.38) to show that

$$P(z) = z^{2-M-N} P(z^{-1})$$

and

$$P(-z) = -z^{2-M-N} P(-z^{-1}).$$

Subtract these two equations, and use equation (6.38) to conclude that $P(z)$ has only a single even power of z , namely a constant term $c_0 z^0$. Conclude that if P is low-pass, then $P(z) = c(1 + z^{-m})$ for some odd integer m . Explicitly write out the solutions to $P(z) = 0$. Why does this contradict $L_a(-1) = 0$ and $L_s(-1) = 0$?

- d. Show that under the conditions of (c) above, if M and N are both even, then \mathbf{h}_a and \mathbf{h}_s are both antisymmetric. Show that if M and N are both odd, then \mathbf{h}_a and \mathbf{h}_s are both symmetric.

- 6.28** Let P be of the form in equation (6.51), where all p_k are real numbers and $p_{N-1} \neq 0$. The goal of this exercise is to show that if $P(e^{i\omega})$ is real and nonnegative for all $\omega \in [-\pi, \pi]$, then P can be factored as $P(z) = L(z)L(z^{-1})$ for a suitable function L of the form

$$L(z) = \ell_0 + \ell_1 z + \cdots + \ell_{N-1} z^{N-1}$$

in which the ℓ_k are real and $\ell_{N-1} \neq 0$. In fact the approach yields a computational method that works for modest values of N . However, L is not typically unique. A minor discrepancy should also be mentioned: here we take $L(z)$ as a polynomial in nonnegative powers of z , whereas the discussion of Section 6.7.6 involves nonpositive powers of z in $L_a(z)$; that doesn't matter, for we can take $L_a(z) = L(z^{-1})$.

In what follows it will be helpful to recall a couple elementary facts from algebra. First, if a polynomial $T(z)$ of degree n has roots z_1, \dots, z_m , then $T(z)$ factors as

$$T(z) = c(z - z_1)^{n_1}(z - z_2)^{n_2} \cdots (z - z_m)^{n_m} \quad (6.54)$$

for some constant c , where $n_1 + \cdots + n_m = n$. The exponent n_k is called the *multiplicity* of the root z_k . Moreover, if the coefficients of T are all real and z_k is a root of T of multiplicity n_k , then $\overline{z_k}$ is also a root of multiplicity n_k . Indeed, $T(z)$ has real coefficients if and only if the non-real roots of T come in conjugate pairs.

Finally, in working this problem, it might be helpful to keep in mind the specific $P(z)$ in part (f), as an example.

- a. Let $Q(z) = z^{N-1}P(z)$. Why must Q be a polynomial of degree $2N - 2$ rather than a lower degree? Show that $Q(z) = 0$ and $P(z) = 0$ have exactly the same solutions z_k , $1 \leq k \leq 2N - 2$, in the complex plane; the z_k need not be distinct. Conclude that $P(z)$ can be expressed as

$$P(z) = p_{N-1} \frac{(z - z_1)(z - z_2) \cdots (z - z_{2N-2})}{z^{N-1}}. \quad (6.55)$$

- b. Show that if z_k is a solution to $P(z) = 0$, then so are $1/z_k$, \bar{z}_k , and $1/\bar{z}_k$ (but not necessarily distinct).
c. Let z be a nonzero complex number with $|z| \neq 1$. Show that if z is not real, then z , \bar{z} , $1/z$, and $1/\bar{z}$ are all distinct, but if z is real, then the set z , \bar{z} , $1/z$, and $1/\bar{z}$ contains only two distinct elements.

- Conclude that if z_k is a non-real solution to $P(z_k) = 0$ with $|z_k| \neq 1$, then z_k , $1/z_k$, \bar{z}_k , and $1/\bar{z}_k$ are all distinct solutions to $P(z) = 0$, but if z_k is real the set z_k , $1/z_k$, \bar{z}_k , and $1/\bar{z}_k$ yields only two distinct solutions.
d. Suppose that z_k is a solution to $P(z) = 0$ (hence by part (a) also $Q(z_k) = 0$) and $|z_k| = 1$. Show that z_k must have even multiplicity. *Some hints:* If z_k has multiplicity n , then from (6.55),

$$P(z) = \frac{(z - z_k)^n R(z)}{z^{N-1}},$$

where $R(z)$ a polynomial with $R(z_k) \neq 0$. Suppose that $z_k = e^{i\theta}$ for some real θ , and let $z = e^{i\omega}$. Argue that for ω near θ we must have a series expansion of the form

$$P(e^{i\omega}) = A(\omega - \theta)^n + O((\omega - \theta)^{n+1}),$$

where A is some nonzero constant and $O((\omega - \theta)^{n+1})$ indicates powers $(\omega - \theta)^{n+1}$ and higher. Conclude that if n is odd, then $P(e^{i\omega})$ cannot be of one sign for all ω near θ .

- e. Let

$$S_1 = \bigcup_{k=1}^K \left\{ z_k, \frac{1}{z_k}, \bar{z}_k, \frac{1}{\bar{z}_k} \right\}$$

denote the set of non-real solutions to $P(z) = 0$, where $|z_k| < 1$ and z_k lies in the upper half plane. Note that $|\bar{z}_k| < 1$, while $|1/z_k| > 1$ and $|1/\bar{z}_k| > 1$. Let

$$S_2 = \bigcup_{m=1}^M \left\{ z_m, \frac{1}{z_m} \right\}$$

denote the set of real solutions to $P(z) = 0$, arranged so that $|z_m| < 1$ for each m . Finally, let

$$S_3 = \bigcup_{r=1}^R \{z_r, \bar{z}_r\}$$

denote the set of solutions to $P(z) = 0$ with $|z_r| = 1$ (if z_r is a solution, so is \bar{z}_r). Let n_r denote the multiplicity (even) of z_r . Let $L_0(z)$ be defined as

$$\begin{aligned} L_0(z) = & \left(\prod_{k=1}^K (z - z_k)(z - \bar{z}_k) \right) \left(\prod_{m=1}^M (z - z_m) \right) \\ & \times \left(\prod_{r=1}^R (z - z_r)^{n_r/2} (z - \bar{z}_r)^{n_r/2} \right) \end{aligned}$$

where the first product is over S_1 , the second over S_2 , and the last over S_3 . Show that the polynomial $L_0(z)$ has real coefficients, and that we can take $L(z)$ in the factorization $P(z) = L(z)L(z^{-1})$ as $L(z) = cL_0(z)$ where $c = \sqrt{p_{N-1}/L_0(0)}$. Hint: First show that $z^{N-1}L_0(z)L_0(z^{-1})$ is a polynomial with the same roots and multiplicities as $Q(z)$ so that $z^{N-1}L_0(z)L_0(z^{-1})$ is a multiple of $Q(z)$.

- f. Use this method to find an appropriate $L(z)$ for $P(z) = L(z)L(z^{-1})$, where

$$P(z) = -z^{-3} + \frac{11}{2}z^{-2} - 13z^{-1} + \frac{69}{4} - 13z + \frac{11}{2}z^2 - z^3.$$

Hint: One of the roots of P is $1 + i$. Another is $\frac{1}{2}$.

- 6.29** The purpose of this exercise is to show one method for obtaining the function $P(z)$ in equations (6.52) and (6.53). Recall that the goal is to find, for any even N , a function

$$P(z) = 1 + \sum_{k=-N+1, \text{ odd}}^{N-1} p_k z^k \quad (6.56)$$

so that $P(z)$ vanishes to high order at $z = -1$; that is, $P(z)$ is divisible by as high a power of $(1 + z)$ as possible (so that P will correspond to a low-pass filter). We'd also like $P(e^{i\omega}) \geq 0$ for $\omega \in [-\pi, \pi]$.

If we write P in the form

$$P(z) = \frac{(1 + z)^{2M} R(z)}{z^{N-1}}, \quad (6.57)$$

then the goal is to make M as large as possible with $R(z)$ a polynomial of degree $2N - 2 - 2M$ that is not divisible by $(1 + z)$. The “2” in the exponent of $1 + z$ is just for convenience.

- a. Let $P(z) = Q((z + z^{-1})/2)$, where Q is a polynomial of degree $N - 1$, of the form

$$Q(x) = 1 + \sum_{k=1, \text{ odd}}^{N-1} q_k x^k \quad (6.58)$$

and each q_k is real. Show that P will then be of the form (6.56).

- b. Suppose that $Q(x)$ is a polynomial that has a root of multiplicity M at $x = -1$ so that $Q(x) = (1 + x)^M S(x)$ for some polynomial $S(x)$ with $S(-1) \neq 0$. Show that $P(z) = Q((z + z^{-1})/2)$ will be of the form (6.57).

Our goal is thus to produce a polynomial $Q(x)$ of the form (6.58) such that Q is divisible by as high a power of $(1 + x)$ as possible.

- c. Observe that $Q'(x)$ (a polynomial of degree $N - 2$) should consist entirely of even powers of x and (like Q) be divisible by as high a power of $(1 + x)^{M-1}$ as possible. Explain why taking $Q'(x)$ as an $N - 2$ degree polynomial of the form

$$Q'(x) = c(1 - x^2)^{N/2-1} \quad (6.59)$$

for some constant c is consistent with these requirements. Specifically, show that no polynomial of degree $N - 2$ consisting of even powers of x can be divisible by any higher power of $(1 + x)$.

- d. According to part (c) we should take

$$Q(x) = c \int_{-1}^x (1 - y^2)^{N/2-1} dy. \quad (6.60)$$

Show that $Q(x)$ of this form must be divisible by $(1 + x)^{N/2}$. Hint: Use (6.60) to show that $|Q(x)| \leq K(1 + x)^{N/2}$ for some constant K and all x near -1 . Why does this imply that $Q(x)$ is divisible by $(1 + x)^{N/2}$? Argue that there must be some value of c so that $Q(0) = 1$.

- e. Show that $P(e^{i\omega})$ is real and nonnegative for all $\omega \in [-\pi, \pi]$. Hint: If $z = e^{i\omega}$, then $(z + z^{-1})/2 = \cos(\omega)$.
- f. Compute $Q(x)$ (with the appropriate value of c) for $N = 2, 4, 6$, and then find the corresponding $P(z)$. Compare to the results given in Examples 6.15 and 6.16.

CHAPTER 7

WAVELETS

7.1 OVERVIEW

7.1.1 Chapter Outline

In the last chapter we defined filter banks and started referring to “discrete wavelet transforms” in the finite-dimensional case. In this chapter we’ll make clear exactly what wavelets are and what they have to do with filter banks. This chapter contains a certain amount of unavoidable but elementary analysis. Our goal is to provide a basic understanding of wavelets, how they relate to filter banks, and how they can be useful. We state and provide examples concerning the essential truths about wavelets, and some rigorous proofs. But we provide only references for other more technical facts concerning wavelets, such as the existence of scaling functions, convergence of the cascade algorithm for computing scaling functions, and some of the properties of wavelets.

7.1.2 Continuous from Discrete

Suppose that by some miracle we had developed the discrete Fourier transform for sampled signals but had no notion of the underlying theory for the continuous case, that is, Fourier series. We know from Chapter 1 that any vector in \mathbb{C}^N can be constructed as a superposition of the basic discrete waveforms $\mathbf{E}_{N,k}$ defined in equation (1.22). If we plot the components of $\mathbf{E}_{N,k}$ for some fixed k and increasing

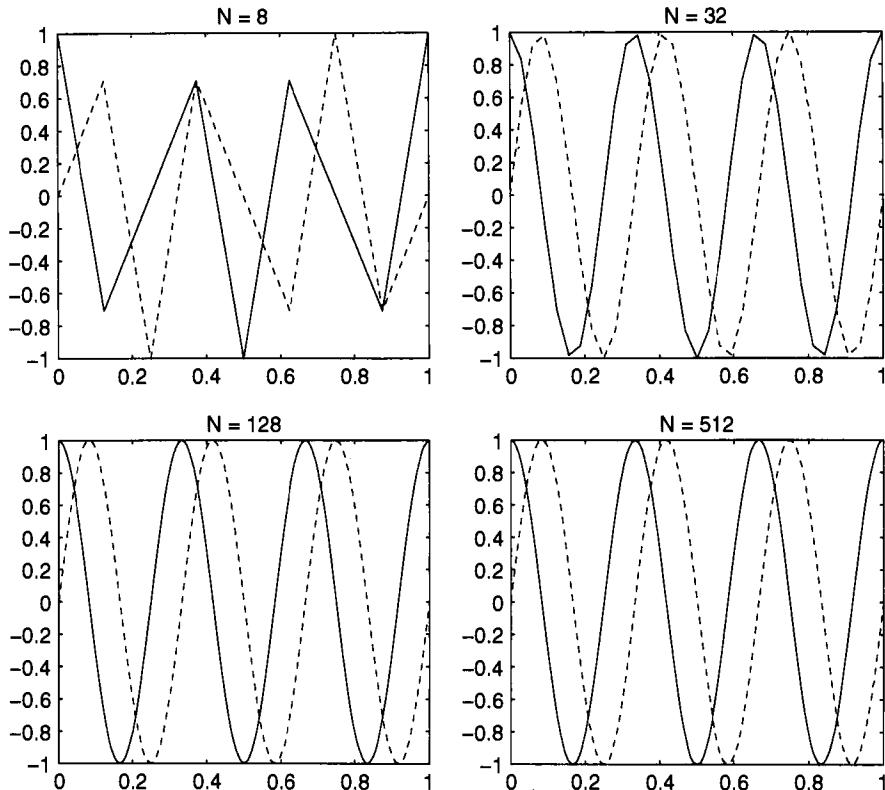


FIGURE 7.1 Real and imaginary parts of discrete waveforms $\mathbf{E}_{N,3}$ for $N = 8, 32, 128, 512$.

N , the vectors $\mathbf{E}_{N,k}$ seem to stabilize on some “mysterious” underlying function. Of course, this is none other than $e^{2\pi i k t}$ sampled at times $t = m/N$ as illustrated in Figure 7.1 for $k = 3$. Even if we had no knowledge of the basis functions $e^{2\pi i k t}$, the plots in Figure 7.1 might lead us to suspect that *some* function underlies the discrete quantity $\mathbf{E}_{N,k}$. Generally, we might posit that some continuous theory underlies our discrete computations.

It’s worth noting that the basic waveforms $\mathbf{E}_{N,k}$ are simply the columns of the DFT matrix \mathbf{F}_N^{-1} that governs the inverse discrete Fourier transform. Equivalently, the waveforms are obtained by applying the inverse discrete transform to the standard basis vectors \mathbf{e}_k in \mathbb{C}^N (recall Example 1.18).

Now the idea that we could develop the DFT with no notion of the underlying waveforms $e^{2\pi i k t}$ seems farfetched, but we’re in precisely this position with regard to wavelets! To illustrate, let’s use the multistage orthogonal discrete Haar transform (filters as in Example 6.5, periodic extension of signals) in place of the DFT to perform again the experiment above. We’ll apply the matrix \mathcal{W}_n^s governing the n -stage inverse transform to the standard basis vector \mathbf{e}_7 , for the cases $N = 8, 32, 128$,

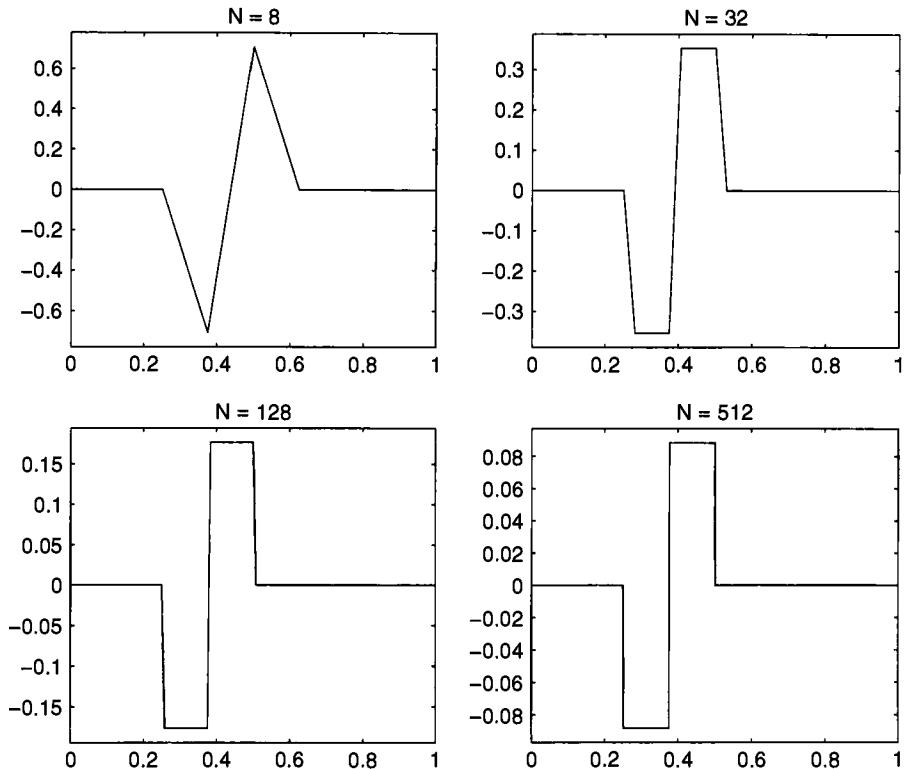


FIGURE 7.2 Vectors $\mathcal{W}_n^s(\mathbf{e}_7)$ for Haar filters with $N = 8, 32, 128, 512$.

and 512, using $n = 3, 5, 7$, and 9 stages, respectively (the maximum number of stages we can do here is $n = \log_2(N)$). The results are shown in Figure 7.2. In each case we plot the components $(\mathcal{W}_n^s(\mathbf{e}_7))_m$ versus m/N . Aside from a normalization in the vertical direction, it seems that the discrete waveforms stabilize on some underlying function.

Let's repeat this experiment with the Daubechies 4-tap orthogonal filters. With $N = 8, 32, 128$, and 512 and using $n = 1, 3, 5$, and 7 stages (the maximum number we can do for each N), we obtain the results shown in Figure 7.3. Again, apart from normalization, it appears that the discrete waveforms are converging to some underlying function. It seems that there may be a continuous theory lurking behind filter banks!

7.2 THE HAAR BASIS

We'll start by examining the simplest kind of wavelets, the *Haar functions*, which predate modern wavelet theory by about 70 years.

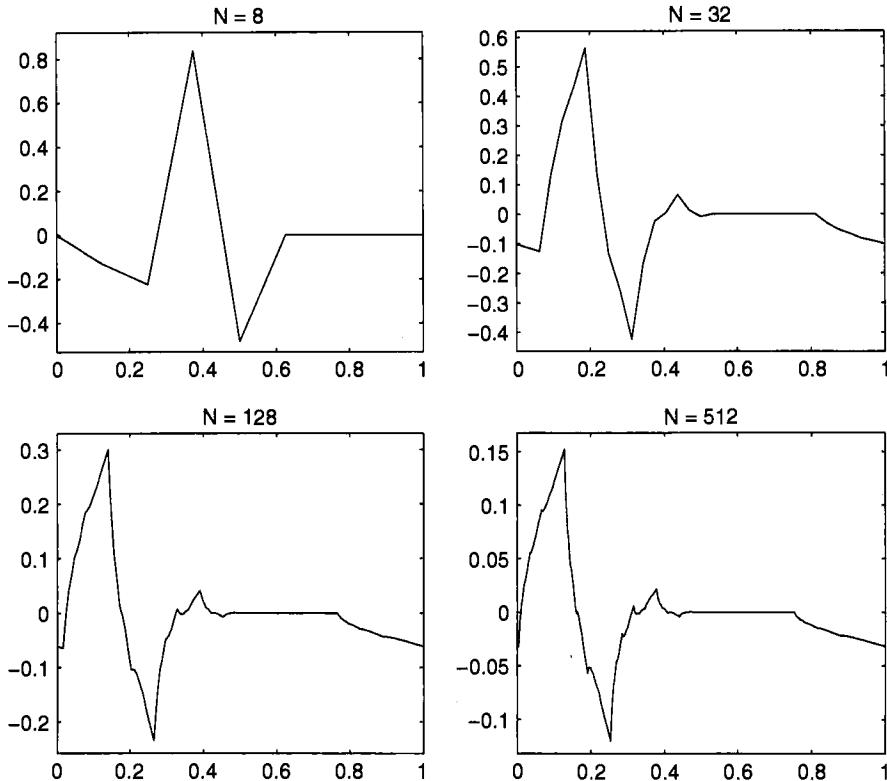


FIGURE 7.3 Vectors $\mathcal{W}_n^s(e_7)$ for Daubechies 4-tap filters with $N = 8, 32, 128, 512$.

7.2.1 Haar Functions as a Basis for $L^2(0, 1)$

Haar Function Definition and Graphs The Haar functions provide an orthogonal basis for the space $L^2(\mathbb{R})$, but for simplicity, we'll initially confine our attention to $L^2(0, 1)$. Let $\psi(t)$ be the function defined on \mathbb{R} as

$$\psi(t) = \begin{cases} 1, & 0 \leq t < \frac{1}{2}, \\ -1, & \frac{1}{2} \leq t < 1, \\ 0, & \text{else.} \end{cases} \quad (7.1)$$

The definition outside of $[0, 1]$ is for later convenience. We call ψ the *mother Haar wavelet*. In Figure 7.4 we provide a graph of $\psi(t)$. The similarity of $\psi(t)$ to that of the graphs in Figure 7.2 is no coincidence!

From $\psi(t)$ we can build an infinite set of functions $\psi_{k,n}$, the *Haar wavelets*, defined on the interval $[0, 1]$ by

$$\psi_{k,n}(t) = 2^{k/2}\psi(2^k t - n), \quad (7.2)$$

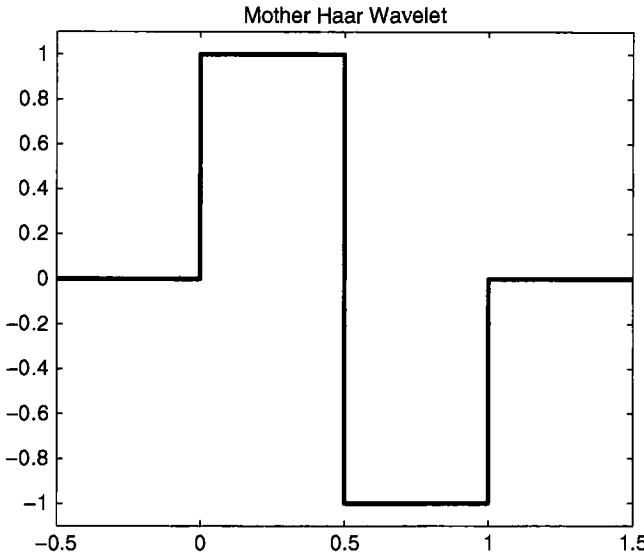


FIGURE 7.4 Mother Haar wavelet.

where $k \geq 0$ and $0 \leq n \leq 2^k - 1$. The $2^{k/2}$ in front is simply to normalize so that $\|\psi_{k,n}\|_{L^2(0,1)} = 1$, where we use the notation $\|\cdot\|_{L^2(a,b)}$ to denote the L^2 norm on (a, b) , to avoid confusion when we consider functions on multiple intervals. We'll also throw in the function

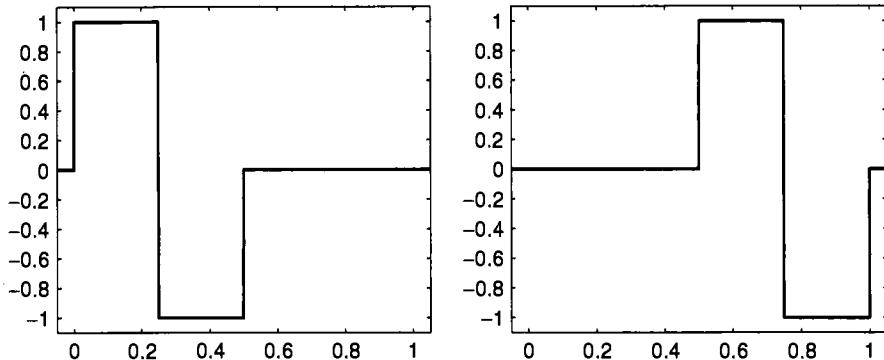
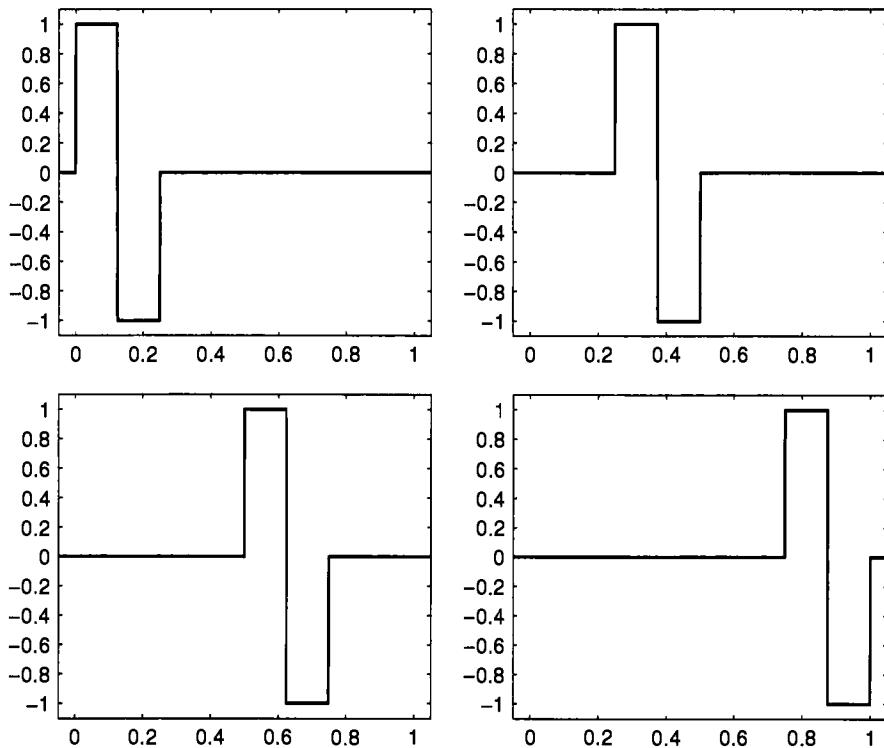
$$\phi(t) = \begin{cases} 1, & 0 \leq t < 1, \\ 0, & \text{else,} \end{cases} \quad (7.3)$$

called the *Haar scaling function*; again, we define ϕ outside $[0, 1]$ merely for later convenience. It turns out that the infinite family $\psi_{k,n}$, together with ϕ , forms an orthonormal basis for $L^2(0, 1)$.

To build up some intuition, let's graph some of these basis functions. The scaling function ϕ is easy to visualize because it's constant, and $\psi_{0,0} = \psi$ is plotted in Figure 7.4. When $k = 1$ we obtain functions $\psi_{1,0}$ and $\psi_{1,1}$, which are graphed in Figure 7.5. These are half-length versions of the mother wavelet, rescaled vertically and translated horizontally. Taking $k = 2$ gives wavelets $\psi_{2,0}, \psi_{2,1}, \psi_{2,2}, \psi_{2,3}$, illustrated in Figure 7.6. These are quarter-scale translations of the mother wavelet. In general, $\psi_{k,n}$ in equation (7.2) can be written explicitly as

$$\psi_{k,n}(t) = \begin{cases} 2^{k/2}, & n/2^k \leq t < (n + 1/2)/2^k, \\ -2^{k/2}, & (n + 1/2)/2^k \leq t < (n + 1)/2^k, \\ 0, & \text{else.} \end{cases} \quad (7.4)$$

In particular, $\psi_{k,n}$ is nonzero only on the interval $[n/2^k, (n + 1)/2^k]$, of length $1/2^k$, and for any fixed k the function $\psi_{k,n}$ is simply $\psi_{k,0}$ translated $n/2^{k+1}$ units to the right. The first index k in $\psi_{k,n}$ dictates the scale and n controls position.

**FIGURE 7.5** Scaled, translated Haar wavelets $\psi_{1,0}(t)$ and $\psi_{1,1}(t)$.**FIGURE 7.6** Scaled, translated Haar wavelets $\psi_{2,0}$, $\psi_{2,1}$, $\psi_{2,2}$, $\psi_{2,3}$.

Before proceeding, it will be helpful to define some terminology. Recall that the *closure* of a set $S \subset \mathbb{R}$ consists of all $x \in \mathbb{R}$ that can be obtained as a limit of elements in S ; the closure of S contains S itself, of course.

Definition 7.2.1 *The “support” of a function g defined on \mathbb{R} is the closure of the set on which g is nonzero. A function g is said to be “supported” in a set $A \subseteq \mathbb{R}$ if the support of g is contained in A , that is, if g is identically zero (or undefined) outside of A . If a function g is supported in a bounded interval, then g is said to have “compact support.”*

The word “compact” stems from the fact that closed, bounded subsets of \mathbb{R} are examples of *compact sets*; see [20] or any introductory text on real analysis.

The support of the Haar scaling function ϕ is the interval $[0, 1]$. The support of the Haar wavelet $\psi_{k,n}$ is the closed interval $[n/2^k, (n+1)/2^k]$.

Orthogonality Recall the notation

$$(f, g) := \int_a^b f(t)g(t) dt$$

for the inner product of two functions in $L^2(a, b)$.

Proposition 7.2.1 *The set*

$$S = \{\phi\} \cup \{\psi_{k,n} : k \geq 0, 0 \leq n \leq 2^k\}$$

is orthonormal in $L^2(0, 1)$.

Proof It's easy to see that

$$(\phi, \psi_{k,n}) = \int_0^1 \phi(t)\psi_{k,n}(t) dt = \int_0^1 \psi_{k,n}(t) dt = 0,$$

since $\phi \equiv 1$; thus ϕ is orthogonal to every $\psi_{k,n}$. It's also easy to see that if $m \neq n$, then the product $\psi_{k,m}(t)\psi_{k,n}(t) \equiv 0$, since $\psi_{k,m}$ and $\psi_{k,n}$ have disjoint support except possibly at a single point; look at Figure 7.6. As a result $(\psi_{k,m}, \psi_{k,n}) = 0$.

Now consider the product $\psi_{j,m}(t)\psi_{k,n}(t)$ where $j \neq k$, say $j < k$. One possibility is $\psi_{j,m}\psi_{k,n} \equiv 0$, if $\psi_{j,m}$ and $\psi_{k,n}$ have disjoint support, in which case orthogonality is obvious. Alternatively, we must have one of $\psi_{j,m}\psi_{k,n} = 2^{j/2}\psi_{k,n}$ (e.g., $\psi_{1,0}$ and $\psi_{2,0}$) or $\psi_{j,m}\psi_{k,n} = -2^{j/2}\psi_{k,n}$ (e.g., $\psi_{1,0}$ and $\psi_{2,1}$), because the support of $\psi_{k,n}$ is contained in either the region where $\psi_{j,m} \equiv 2^{j/2}$ or the region where $\psi_{j,m} \equiv -2^{j/2}$.

We then have

$$(\psi_{j,m}, \psi_{k,n}) = \int_0^1 \psi_{j,m}(t) \psi_{k,n}(t) dt = \pm 2^{j/2} \int_0^1 \psi_{k,n}(t) dt = 0$$

which completes the proof of Proposition 7.2.1. ■

Of course, the proposition implies that the set S is linearly independent.

Completeness in $L^2(0, 1)$ The set S in Proposition 7.2.1 is not merely orthonormal in $L^2(0, 1)$, but complete: any function $f \in L^2(0, 1)$ can be approximated to arbitrary precision in the L^2 norm by using linear combinations of elements of S . Specifically, for each $K \geq 0$ let

$$f_{K+1}(t) = c_0 \phi(t) + \sum_{k=0}^K \sum_{n=0}^{2^k - 1} c_{k,n} \psi_{k,n}(t), \quad (7.5)$$

where $c_0 = (f, \phi)/(\phi, \phi) = (f, \phi) = \int_0^1 f(t) dt$ and

$$c_{k,n} = (f, \psi_{k,n}) = 2^{k/2} \int_0^1 f(t) \psi_{k,n}(t) dt \quad (7.6)$$

as dictated by Theorem 1.10.1. Because $(\psi_{k,n}, \psi_{k,n}) = 1$, it isn't explicitly needed in the denominator for $c_{k,n}$. For convenience let's also define $f_0(t) \equiv c_0$. We can make $\|f - f_K\|_{L^2(0,1)}$ arbitrarily small by taking K sufficiently large. We'll prove this below, but let's pause to look at an example.

■ EXAMPLE 7.1

Let $f(t) = t(1-t)(2-t)$ on $(0, 1)$. The crudest possible approximation is obtained by simply using the Haar scaling function. We obtain $f_0(t) = c_0$, where $c_0 = \int_0^1 f(t) \phi(t) dt = \frac{1}{4}$, which merely approximates f by its own mean value on the interval $[0, 1]$.

If we take $K = 0$ in equation (7.5), then $\psi_{0,0}$ is included, and we obtain the approximation f_1 shown on the left in Figure 7.7. In the terminology we'll define shortly, f_1 is the “projection” of f onto the subspace spanned by ϕ and $\psi_{0,0}$ (which consists of functions that are constant on each of $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$). On the right in Figure 7.7 we have the approximation $f_2(t)$ obtained from equation (7.5) with $K = 1$, which throws $\psi_{1,0}$ and $\psi_{1,1}$ into the mix. This is the projection of f onto the subspace of functions which are constant on each interval $[0, \frac{1}{4}]$, $[\frac{1}{4}, \frac{1}{2}]$, $[\frac{1}{2}, \frac{3}{4}]$, $[\frac{3}{4}, 1]$. In Figure 7.8 we show the approximation $f_5(t)$ obtained by using all $\psi_{k,n}$ up to $k = 4$. It certainly looks like we can approximate any continuous function in this manner! In each case notice that $f_K(t)$ is constant on intervals of the form $[n/2^K, (n+1)/2^K]$, where $0 \leq n \leq 2^K - 1$.

The following proposition will be useful.

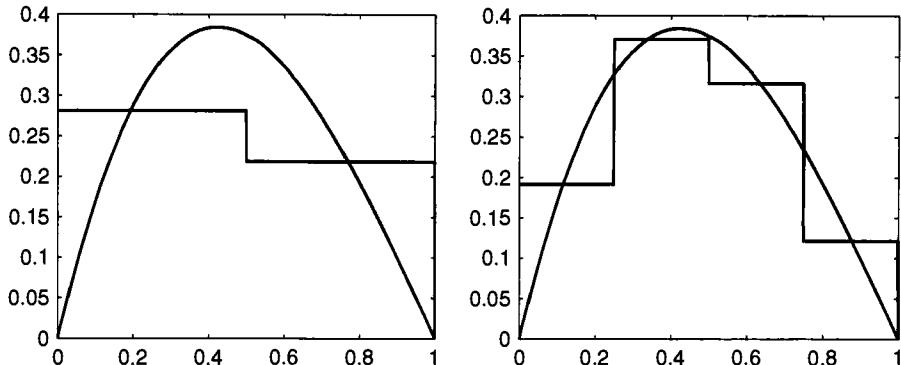


FIGURE 7.7 Haar expansions using all $\psi_{k,n}$ with $k \leq 0$ (left) and $k \leq 1$ (right).

Proposition 7.2.2 *For any $f \in L^2(0, 1)$ the (constant) value of the approximation $f_K(t)$ on any interval $[n/2^K, (n+1)/2^K]$ is the average value of f on that interval.*

Proof This is easy to prove by induction. First note that $f_0(t) \equiv c_0 = \int_0^1 f(t) dt$ is indeed the average value of f on the interval $[0, 1]$.

Suppose that the assertion holds for some fixed $K \geq 0$. Specifically, for any fixed n let $I = [n/2^K, (n+1)/2^K]$, and suppose that $f_K(t) = a$ for $t \in I$, where

$$a = 2^K \int_{n/2^K}^{(n+1)/2^K} f(t) dt$$

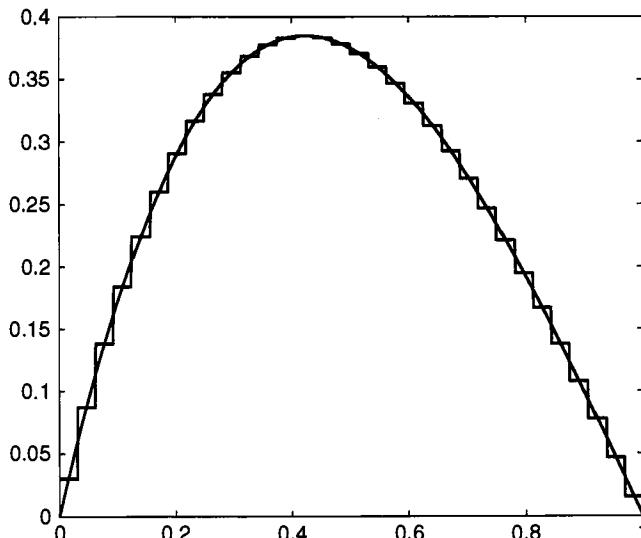


FIGURE 7.8 Haar expansion using all $\psi_{k,n}$ up to $k = 4$.

is the average value of f on I . Write $a = (a_0 + a_1)/2$, where

$$a_0 = 2^{K+1} \int_{n/2^K}^{(n+1)/2^K} f(t) dt \quad \text{and} \quad a_1 = 2^{K+1} \int_{(n+1/2)/2^K}^{(n+1)/2^K} f(t) dt$$

are the average value of f on the left and right halves of I , respectively.

On the interval I we have $f_{K+1}(t) = f_K(t) + c_{K,n}\psi_{K,n}(t)$ where

$$\begin{aligned} c_{K,n} &= \int_0^1 f(t)\psi_{K,n}(t) dt \\ &= \int_{n/2^K}^{(n+1)/2^K} f(t)\psi_{K,n}(t) dt \\ &= 2^{K/2} \int_{n/2^K}^{(n+1/2)/2^K} f(t) dt - 2^{K/2} \int_{(n+1/2)/2^K}^{(n+1)/2^K} f(t) dt \\ &= 2^{-K/2-1}(a_0 - a_1). \end{aligned}$$

The value of $f_{K+1}(t)$ on the interval $[n/2^K, (n + \frac{1}{2})/2^K]$ (the left half of I) is then

$$f_{K+1}(t) = f_K(t) + c_{K,n}\psi_{K,n}(t) = a + \frac{1}{2}(a_0 - a_1) = a_0,$$

while the value of $f_{K+1}(t)$ on $[(n + \frac{1}{2})/2^K, (n + 1)/2^K]$ (the right half of I) is

$$f_{K+1}(t) = f_K(t) + c_{K,n}\psi_{K,n}(t) = a - \frac{1}{2}(a_0 - a_1) = a_1.$$

Thus f_{K+1} is constant on any interval $[n/2^K, (n + \frac{1}{2})/2^K]$ or $[(n + \frac{1}{2})/2^K, (n + 1)/2^K]$, and equal to the average value of f on that interval. Of course, any interval $[m/2^{K+1}, (m + 1)/2^{K+1}]$ is of the form $[n/2^K, (n + \frac{1}{2})/2^K]$ or $[(n + \frac{1}{2})/2^K, (n + 1)/2^K]$, which completes the induction step and proves the proposition. ■

Remark 7.1 A simple corollary to Proposition 7.2.2 is that if f is a function that is constant on each interval $[n/2^K, (n + 1)/2^K]$, then f can be built exactly from the scaling function ϕ and a finite superposition of Haar wavelets $\psi_{k,n}$ for $k \leq K - 1$.

We'll use Proposition 7.2.2 to show the Haar wavelets form a basis for $L^2(0, 1)$. As remarked in Chapter 1, the space $L^2(0, 1)$ consists of much more than just continuous functions. Nonetheless, we'll just show that any continuous function on $[0, 1]$ can be approximated arbitrarily well in $L^2(0, 1)$ using the Haar wavelets. Indeed we can make the stronger statement that f can be approximated to any precision in the supremum norm on $[0, 1]$; that is, we can make

$$\sup_{t \in [0, 1]} |f_K(t) - f(t)|$$

arbitrarily small by taking K sufficiently large.

Theorem 7.2.1 Any continuous function f on $[0, 1]$ can be approximated to arbitrary precision in the supremum norm using equations (7.5) and (7.6).

Proof The proof requires an elementary fact from real analysis. Specifically, since f is continuous on a closed, bounded (i.e., compact) interval, f is uniformly continuous on this interval. This means that given any $\epsilon > 0$ there is some $\delta > 0$ so that $|f(t) - f(s)| < \epsilon$ whenever $|t - s| < \delta$. Fix any $\epsilon > 0$, and choose K large enough so that $1/2^K < \delta$ for the corresponding δ .

From Proposition 7.2.2 we know that on any interval of the form $[n/2^K, (n+1)/2^K)$ the function $f_K(t) \equiv a$, where a is the average value of f on this interval. The function f is continuous, so by the integral mean value theorem, we have $a = f(t^*)$ for some $t^* \in [n/2^K, (n+1)/2^K)$. For any t in this interval, then

$$|f(t) - f_K(t)| = |f(t) - a| = |f(t) - f(t^*)| < \epsilon,$$

since $|t - t^*| \leq 1/2^K < \delta$. Thus on any interval $[n/2^K, (n+1)/2^K)$ contained in $[0, 1]$ the functions f and f_K differ by no more than ϵ , and we conclude that

$$\sup_{t \in [0,1]} |f_K(t) - f(t)| \leq \epsilon,$$

which completes the proof. ■

Remark 7.2 Of course, this means that for any $\epsilon > 0$ we can obtain $|f_K(t) - f(t)| \leq \epsilon^2$ for any continuous f and suitably large K , and then we also have

$$\|f_K - f\|_{L^2(0,1)} \equiv \left(\int_0^1 (f_K(t) - f(t))^2 dt \right)^{1/2} \leq \epsilon.$$

The set S of Proposition 7.2.1 is thus complete in $L^2(0, 1)$.

7.2.2 Haar Functions as an Orthonormal Basis for $L^2(\mathbb{R})$

We can use the scaling function and wavelets to define an orthonormal basis for $L^2(\mathbb{R})$. The set

$$\{\phi(t - n) : n \in \mathbb{Z}\}$$

is orthonormal in $L^2(\mathbb{R})$. This set will become part of an orthonormal basis for $L^2(\mathbb{R})$. We'll also continue to use the functions $\psi_{k,n}(t) = 2^{k/2}\psi(2^k t - n)$ (equivalently, equation (7.4)), with $k \geq 0$, but now allow n to range over all of \mathbb{Z} . This means that all translates are allowed, up and down the real line. We still have $\|\psi_{k,n}(t)\|_{L^2(\mathbb{R})} = 1$.

Let \tilde{S} be the set

$$\tilde{S} = \{\phi(t - n) : n \in \mathbb{Z}\} \cup \{\psi_{k,n} : k \geq 0, n \in \mathbb{Z}\} \quad (7.7)$$

The same reasoning as in Proposition 7.2.1 shows that \tilde{S} is an orthonormal set in $L^2(\mathbb{R})$. Also, for each $m \in \mathbb{Z}$, some subset of \tilde{S} provides an orthonormal basis for $L^2(m, m + 1)$. For example, $\phi(t)$ together with all $\psi_{k,n}$ such that $k \geq 0$ and $0 \leq n < 2^k$ is a basis for $L^2(0, 1)$, for this set is simply the basis $\phi(t)$ and $\psi_{k,n}$ for $L^2(0, 1)$ considered above. Similarly $\phi(t - 1)$ and $\psi_{k,n}$ with $k \geq 0$ and $2^k \leq n < 2^{k+1}$ is a basis for $L^2(1, 2)$, for these are just translates one unit to the right of the basis elements for $L^2(0, 1)$. More generally, $\phi(t - m)$ together with the set of all $\psi_{k,n}$ such that $k \geq 0$ and $m2^k \leq n < (m + 1)2^{k+1}$ is a basis for $L^2(m, m + 1)$.

As a consequence, for any $\epsilon > 0$, any $m \in \mathbb{Z}$, and any function $f \in L^2(m, m + 1)$, we can obtain

$$\|f - f_m\|_{L^2(m, m+1)} < \epsilon,$$

where f_m is a linear combination of elements of \tilde{S} .

Theorem 7.2.2 *The set \tilde{S} in equation (7.7) is an orthonormal basis for $L^2(\mathbb{R})$.*

Proof We'll again prove the theorem only for continuous functions. Since any function in $L^2(\mathbb{R})$ can be approximated to arbitrary precision by continuous functions, this will suffice. Let $f \in L^2(\mathbb{R})$, so that

$$\int_{-\infty}^{\infty} f^2(t) dt < \infty.$$

Write this as

$$\sum_{m=-\infty}^{\infty} \int_m^{m+1} f^2(t) dt < \infty.$$

Thus the series $\sum_m a_m$ with positive terms $a_m = \int_m^{m+1} f^2(t) dt$ converges. For any $\epsilon > 0$ we can then choose some M large enough so that

$$\sum_{|m|>M} \int_m^{m+1} f^2(t) dt < \epsilon^2/2.$$

From Theorem 7.2.1 (and its extension to any unit interval $[m, m + 1]$ as discussed above) we know that on any given interval $[m, m + 1]$ we can approximate f with

some function f_m so that $\|f - f_m\|_{L^2(m, m+1)}^2 \leq \epsilon^2/(4M + 2)$. Define the function $\tilde{f} \in L^2(\mathbb{R})$ as

$$\tilde{f}(t) = \begin{cases} f_m(t), & t \in [m, m+1] \text{ and } |m| \leq M, \\ 0, & \text{else,} \end{cases}$$

The function $\tilde{f}(t)$ is a finite linear combination of elements of the set $\tilde{\mathcal{S}}$. Then

$$\begin{aligned} \|f - \tilde{f}\|_{L^2(\mathbb{R})}^2 &= \int_{-\infty}^{\infty} (f(t) - \tilde{f}(t))^2 dt \\ &= \sum_{|m| \leq M} \int_m^{m+1} (f(t) - \tilde{f}(t))^2 dt \\ &\quad + \sum_{|m| > M} \int_m^{m+1} (f(t) - \tilde{f}(t))^2 dt \\ &= \sum_{|m| \leq M} \|f - \tilde{f}\|_{L^2(m, m+1)}^2 + \sum_{|m| > M} \int_m^{m+1} f^2(t) dt \\ &\leq \sum_{|m| \leq M} \frac{\epsilon^2}{4M+2} + \frac{\epsilon^2}{2} \\ &\leq \frac{\epsilon^2}{2} + \frac{\epsilon^2}{2} = \epsilon^2. \end{aligned}$$

Thus for any $\epsilon > 0$ we can obtain $\|f - \tilde{f}\|_{L^2(\mathbb{R})} \leq \epsilon$, where \tilde{f} is a finite linear combination of the elements of $\tilde{\mathcal{S}}$. This proves the theorem. ■

Since $\tilde{\mathcal{S}}$ is an orthonormal basis, Theorem 1.10.1 dictates that if

$$f(t) = \sum_{m=-\infty}^{\infty} c_m \phi(t-m) + \sum_{k=0}^{\infty} \sum_{n=-\infty}^{\infty} c_{k,n} \psi_{k,n}(t), \quad (7.8)$$

then

$$c_m = (f, \phi(t-m)) = \int_m^{m+1} f(t) dt, \quad (7.9)$$

$$c_{k,n} = (f, \psi_{k,n}) = 2^{k/2} \left(\int_{n/2^k}^{(n+1)/2^k} f(t) dt - \int_{(n+1)/2^k}^{(n+1)/2^k} f(t) dt \right), \quad (7.10)$$

which exactly parallel equations (7.5) and (7.6), except that the interval of integration is not restricted to $(0, 1)$.

7.2.3 Projections and Approximations

Before continuing, it will be useful to look at the notion of “orthogonal projection” in an inner product space. The concept of projection plays a huge role in approximation theory, numerical analysis, and applied mathematics in general. To avoid a long detour we’ll develop only what we need, specific to the situation at hand. We’ll be working in the inner product space $L^2(a, b)$ ($(a, b) = \mathbb{R}$ is allowed), though the results hold for more general Hilbert spaces. And although we assume that the spaces are infinite-dimensional, everything below works in \mathbb{R}^N ; indeed it’s even easier since the sums are then finite and no convergence issues arise.

Let V be a subspace of $L^2(a, b)$ with an orthonormal basis $S = \{\cup_{k \in \mathbb{Z}} \eta_k\}$ (alternatively the basis can be finite). We assume that all linear combinations

$$g = \sum_{k \in \mathbb{Z}} a_k \eta_k, \quad (7.11)$$

where $\sum_k a_k^2 < \infty$ are elements of V , so that every square-summable sequence $\{a_k\}$ corresponds to an element of V , and vice versa (recall Theorem 1.10.2 and the remarks preceding that theorem). Recall also that the precise meaning of the infinite sum on the right in equation (7.11) is that

$$\lim_{m, n \rightarrow \infty} \left\| g - \sum_{k=-m}^n a_k \eta_k \right\| = 0. \quad (7.12)$$

as discussed in Chapter 1, for example, equation (1.42).

Definition 7.2.2 Let V be a subspace of $L^2(a, b)$ and $S = \{\cup_{k \in \mathbb{Z}} \eta_k\}$ an orthonormal basis for V as described above. If $f \in L^2(a, b)$ then the “orthogonal projection of f onto V ” is the element $P_V(f)$ of V defined by

$$P_V(f) = \sum_{k \in \mathbb{Z}} (f, \eta_k) \eta_k. \quad (7.13)$$

By Bessel’s inequality (equation (1.55), Exercise 1.34)) we have $\sum_k (f, \eta_k)^2 \leq \|f\|^2 < \infty$, so the element defined by equation (7.13) really is in V .

What’s the relation between $P_V(f)$ and f ? First, if $f \in V$, then $P_V(f) = f$, since S is a basis for V . If f is not in V , then it turns out that $P_V(f)$ is the best approximation to f that can be constructed as a superposition of the η_k . Put another way, $P_V(f)$ is the element of V that is “closest” to f , as quantified by the following proposition:

Proposition 7.2.3 Let S and $V \subseteq L^2(a, b)$ be as in Definition 7.2.2. For $f \in L^2(a, b)$ let $P_V(f)$ be the orthogonal projection of f onto V given by equation (7.13). Then $\|P_V(f) - f\| \leq \|g - f\|$ for all $g \in V$. Equality is attained only if $g = P_V(f)$.

Proof Consider a function α_n of the form

$$\alpha_n = \sum_{k=-n}^n b_k \eta_k \quad (7.14)$$

for scalars b_k ; the function α_n clearly lies in V . Let us first consider the problem of minimizing $\|\alpha_n - f\|$, or equivalently, $\|\alpha_n - f\|^2$, by choosing the b_k appropriately. This leads us to the task of minimizing

$$\begin{aligned} (\alpha_n - f, \alpha_n - f) &= \sum_{k=-n}^n b_k^2 - 2 \sum_{k=-n}^n b_k (f, \eta_k) + (f, f) \\ &= \sum_{k=-n}^n (b_k - (f, \eta_k))^2 + (f, f) - \sum_{k=-n}^n (f, \eta_k)^2 \end{aligned} \quad (7.15)$$

as a function of the b_k , where we've made use of the elementary properties of the inner product, real-valued here. It's obvious that the right side in (7.15) is minimized by taking $b_k = (f, \eta_k)$, since the last two terms don't even depend on the b_k .

Let

$$\tilde{f}_n = \sum_{k=-n}^n (f, \eta_k) \eta_k \quad (7.16)$$

denote the minimizer of $\|f - \alpha_n\|$, where α_n is of the form (7.14). Clearly, \tilde{f}_n is unique, for any other choice for the b_k on the right in (7.15) will increase $\|\alpha_n - f\|^2$. The sequence \tilde{f}_n converges to the function $P_V(f) \in V$, since

$$\|P_V(f) - \tilde{f}_n\|^2 = \sum_{|k|>n} (P_V(f), \eta_k)^2$$

converges to zero as $n \rightarrow \infty$.

To see that $\|f - P_V(f)\| \leq \|g - f\|$ for any other $g \in V$, let $g = \sum_k d_k \eta_k$ and define

$$g_n = \sum_{k=-n}^n d_k \eta_k.$$

From the discussion above we know that $\|\tilde{f}_n - f\| \leq \|g_n - f\|$ for all n . We then have

$$\|P_V(f) - f\| = \lim_{n \rightarrow \infty} \|\tilde{f}_n - f\| \leq \lim_{n \rightarrow \infty} \|g_n - f\| = \|g - f\|$$

by making use of the result of Exercise 1.35. This shows that $\|P_V(f) - f\| \leq \|g - f\|$ for any $g \in V$. The assertion that equality is attained only when $g = P_V(f)$ is left to Exercise 7.11. ■

The projection of an element of an inner product space H onto a subspace V can be defined more generally than we have done. Indeed *under suitable conditions* the projection of an element $f \in H$ onto V may be defined as the element of V that is closest to f . The phrase “suitable conditions” typically means that the subspace V should be closed (i.e., if a sequence with all elements in V has a limit in $L^2(a, b)$, then the limit is in fact in V). In the present case this property is ensured by the requirement that all functions of the form (7.11) with $\sum_k a_k^2 < \infty$ lie in V , but this may not always be true; see Exercise 7.10. Our definition of projection is sufficient for our purposes.

■ EXAMPLE 7.2

Let $H = L^2(0, 1)$, and for any fixed integer $N \geq 1$ let

$$S_N = \{\phi\} \cup \{\psi_{k,n} : 0 \leq k \leq N-1, 0 \leq n < 2^k\},$$

similar to the set S defined in Proposition 7.2.1 but truncated at $k \leq N-1$. The fact that the basis elements $\psi_{k,n}$ are doubly indexed makes no difference at all in the application of the projection formulas (we could re-index them with a single subscript). The set S_N is orthonormal in $L^2(0, 1)$. The subspace $V_N = \text{span}(S_N)$ of $L^2(0, 1)$ consists of exactly those $L^2(0, 1)$ functions that are constant on each interval of the form $[n/2^N, (n+1)/2^N]$ where $0 \leq n < 2^N$: that any $f \in V_N$ is constant on $[n/2^N, (n+1)/2^N]$ is clear. The converse, that any function constant on $[n/2^N, (n+1)/2^N]$ is in V_N (i.e., can be built from elements of S_N) follows from Remark 7.1 on page 276.

The projection of an arbitrary function $f \in L^2(0, 1)$ onto V_N is given by equations (7.5) and (7.6). The graphs in Figure 7.7 show the projection of the function $f(t) = t(t-1)(t-2)$ onto the subspaces V_1 and V_2 , while Figure 7.8 shows the projection of f onto V_5 .

In this example note that since $S_N \subset S_{N+1}$, we also have $V_N \subset V_{N+1}$. As indicated by Figures 7.7 and 7.8, for any function f the projection of f onto V_N becomes more and more detailed as N increases. Indeed, if we let \tilde{f}_N denote the projection of f onto V_N , then Remark 7.2 on page 277 shows that $\lim_{N \rightarrow \infty} \tilde{f}_N = f$. We’ll later encounter this idea—a nested sequence of subspaces that embodies more and more detail—in a more general context.

7.3 HAAR WAVELETS VERSUS THE HAAR FILTER BANK

Before continuing with our analysis of the Haar basis for $L^2(\mathbb{R})$, it may be helpful to pause and examine some parallels between the Haar decomposition of a function as expressed in equations (7.8) through (7.10) and the Haar filter bank.

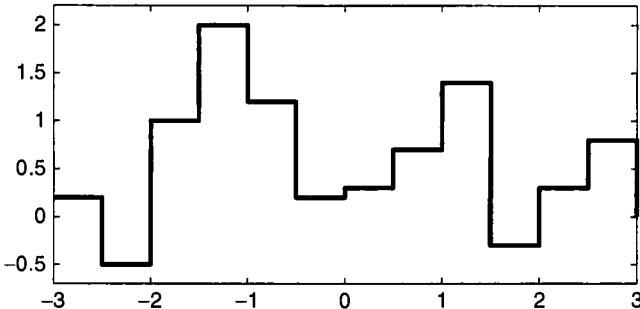


FIGURE 7.9 Piecewise constant function on half-integer intervals.

7.3.1 Single-stage Case

Functions from Sequences Let $\mathbf{x} \in L^2(\mathbb{Z})$. To illustrate parallels between the Haar filter bank and the Haar wavelets, let's use \mathbf{x} to manufacture a piecewise constant function $x(t)$ on the real line by setting

$$x(t) \equiv x_{k-1} \quad \text{for } t \in [k/2, (k+1)/2) \quad (7.17)$$

at each integer k . For example, $x(t) \equiv x_{-1}$ for $0 \leq t < \frac{1}{2}$ and $x(t) \equiv x_0$ for $\frac{1}{2} \leq t < 1$. The function $x(t)$ is in $L^2(\mathbb{R})$ (Exercise 7.6). We use x_{k-1} instead of x_k on the right in equation (7.17) solely to make formulas below prettier; it's not essential. Any $\mathbf{x} \in L^2(\mathbb{Z})$ generates such a function $x(t)$, and conversely, any function $x(t)$ that is constant on half-integer intervals corresponds to such an $\mathbf{x} \in L^2(\mathbb{Z})$. A typical example is illustrated in Figure 7.9.

Filter Bank Analysis/Synthesis We will pass \mathbf{x} through a one-stage Haar analysis filter bank with filters ℓ_a, \mathbf{h}_a , but with one minor change (again, just to make things a bit cleaner): we use the usual low-pass coefficients $(\ell_a)_0 = (\ell_a)_1 = \frac{1}{2}$, but we'll reverse the sign of \mathbf{h}_a and take $(\mathbf{h}_a)_0 = -\frac{1}{2}$, $(\mathbf{h}_a)_1 = \frac{1}{2}$. This is still a high-pass filter, and if we also negate the synthesis filter \mathbf{h}_s , then the filter bank provides perfect reconstruction. We obtain filtered, downsampled vectors

$$\mathbf{X}_\ell = D(\mathbf{x} * \ell_a) = \left(\dots, \frac{x_{-2} + x_{-3}}{2}, \frac{x_0 + x_{-1}}{2}, \frac{x_2 + x_1}{2}, \dots \right),$$

$$\mathbf{X}_h = D(\mathbf{x} * \ell_a) = \left(\dots, \frac{x_{-3} - x_{-2}}{2}, \frac{x_{-1} - x_0}{2}, \frac{x_1 - x_2}{2}, \dots \right),$$

similar to those previously computed in equations (6.3) and (6.4). In general, the components of these vectors are given by

$$(\mathbf{X}_\ell)_m = \frac{x_{2m} + x_{2m-1}}{2}, \quad (7.18)$$

$$(\mathbf{X}_h)_m = \frac{x_{2m-1} - x_{2m}}{2}. \quad (7.19)$$

To reconstruct \mathbf{x} , we first upsample. The upsampled vectors have components

$$U(\mathbf{X}_\ell)_k = (\mathbf{X}_\ell)_{k/2}, \quad U(\mathbf{X}_h)_k = (\mathbf{X}_h)_{k/2},$$

for k even and 0 for k odd. We then apply the synthesis filters $\mathbf{\ell}_s$, \mathbf{h}_s with coefficients $(\mathbf{\ell}_s)_0 = (\mathbf{\ell}_a)_{-1} = 1$ and $(\mathbf{h}_s)_0 = -1$, $(\mathbf{h}_s)_{-1} = 1$ (note that the coefficients of \mathbf{h}_s are negated). This yields

$$\begin{aligned} (\mathbf{\ell}_s * (U(\mathbf{X}_\ell)))_k &= \begin{cases} (\mathbf{X}_\ell)_{k/2}, & k \text{ even}, \\ (\mathbf{X}_\ell)_{(k+1)/2}, & k \text{ odd}, \end{cases} \\ (\mathbf{h}_s * (U(\mathbf{X}_h)))_k &= \begin{cases} -(\mathbf{X}_h)_{k/2}, & k \text{ even}, \\ (\mathbf{X}_h)_{(k+1)/2}, & k \text{ odd}. \end{cases} \end{aligned}$$

Finally, we add $\mathbf{\ell}_s * (U(\mathbf{X}_\ell)) + \mathbf{h}_s * (U(\mathbf{X}_h))$ to reconstruct \mathbf{x} . Specifically,

$$x_k = \begin{cases} (\mathbf{X}_\ell)_{k/2} - (\mathbf{X}_h)_{k/2}, & k \text{ even}, \\ (\mathbf{X}_\ell)_{(k+1)/2} + (\mathbf{X}_h)_{(k+1)/2}, & k \text{ odd}. \end{cases} \quad (7.20)$$

Haar Expansion and Filter Bank Parallels The filter bank operation above has a precise parallel in equations (7.8) through (7.10). If we expand $x(t)$ with respect to the Haar basis with equations (7.9) and (7.10), we obtain coefficients

$$c_m = (x, \phi(t - m)) = \int_{-\infty}^{\infty} x(t)\phi(t - m) dt = \frac{x_{2m} + x_{2m-1}}{2}, \quad (7.21)$$

$$\begin{aligned} c_{0,m} &= (x, \psi_{0,m}) = \int_{-\infty}^{\infty} x(t)\psi_{0,m}(t) dx = \frac{x_{2m-1} - x_{2m}}{2}, \\ c_{k,m} &= (x, \psi_{k,m}) = \int_{-\infty}^{\infty} x(t)\psi_{k,m}(t) dx = 0 \quad \text{for } k \geq 1. \end{aligned} \quad (7.22)$$

Compare the equations above with equations (7.18) and (7.19) to find

$$(\mathbf{X}_\ell)_m = c_m \text{ and } (\mathbf{X}_h)_m = c_{0,m}. \quad (7.23)$$

These sequences are identical! The inner product of $x(t)$ against integer translates of the scaling function ϕ precisely parallels the action of the low-pass filter on \mathbf{x} . The situation is illustrated in Figure 7.10. Each inner product of $x(t)$ with an integer translate of the scaling function has the effect of averaging neighboring values of $x(t)$, just like a low-pass filter. The inner product of $x(t)$ against the longest scale wavelets $\psi_{0,m}$ parallels the action of the high-pass filter, as illustrated in Figure 7.11. Each inner product of $x(t)$ with an integer translate of the wavelet has the effect of differencing neighboring values, just like a high-pass filter. Because the scaling function and wavelet are translated one unit at a time (rather than $\frac{1}{2}$), the “downsampling” is built into the process).

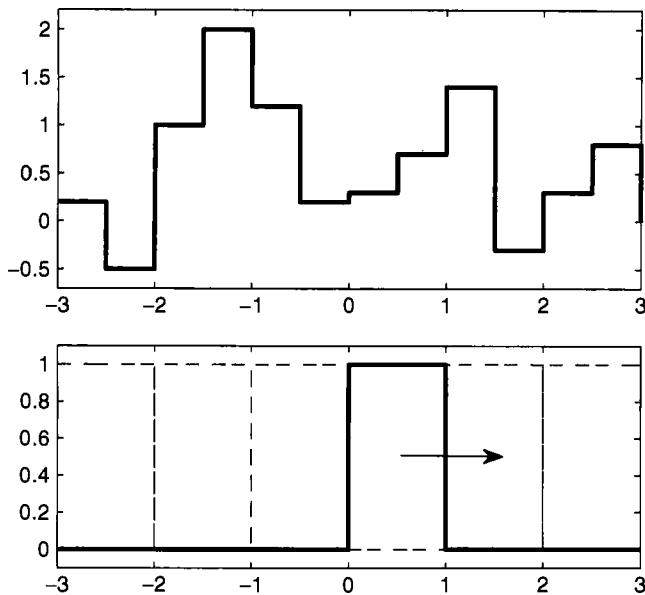


FIGURE 7.10 Action of scaling function translates on $x(t)$.

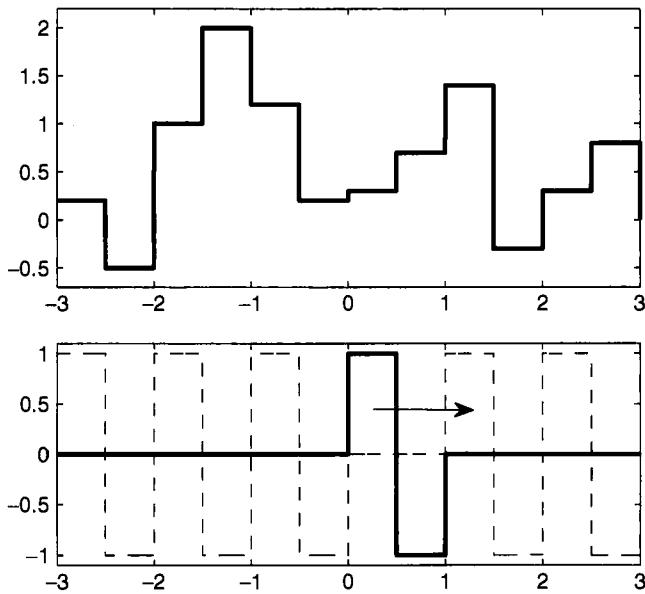


FIGURE 7.11 Action of wavelet translates on $x(t)$.

The inner product of $x(t)$ with any shorter scale wavelet $\psi_{k,m}$ with $k \geq 1$ is zero, since $x(t)$ was constructed to be constant on half-integer intervals, hence variation on such shorter scales is zero. Of course, this won't be true for more general functions. That's where a multistage transform comes into play, which we'll discuss shortly.

Now consider the process of synthesizing $x(t)$ from the c_m and $c_{0,m}$ using equation (7.8). This precisely mirrors the reconstruction of \mathbf{x} from \mathbf{X}_ℓ and \mathbf{X}_h using the synthesis filter bank. Specifically, we have from equation (7.8),

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \phi(t - n) + \sum_{n=-\infty}^{\infty} c_{0,n} \psi_{0,n}(t). \quad (7.24)$$

For any $t \in \mathbb{R}$ only two of the terms on the right in (7.24) are nonzero: namely, if $t \in [m, m+1]$, then (7.24) becomes

$$x(t) = c_m + c_{0,m} \psi_{0,m}(t), \quad (7.25)$$

since $\phi(t-m) \equiv 1$ on $[m, m+1]$. If t is in the right half of the interval $[m, m+1]$, then $(k+1)/2 \leq t < k/2 + 1$, where $k = 2m$. Here $x(t) \equiv x_k$ and $\psi_{0,m} \equiv -1$. In this case equation (7.25) becomes

$$x(t) = c_m - c_{0,m} = (\mathbf{X}_\ell)_m - (\mathbf{X}_h)_m.$$

In light of equation (7.23) this is just the statement that $x_k = (\mathbf{X}_\ell)_{k/2} - (\mathbf{X}_h)_{k/2}$, in accordance with equation (7.20).

If t is in the left half of the interval $[m, m+1]$, then $(k+1)/2 \leq t < k/2 + 1$, where $k = 2m - 1$. Here $x(t) \equiv x_k$ and $\psi_{0,m} \equiv 1$. In this case equation (7.25) becomes

$$x(t) = c_m + c_{0,m} = (\mathbf{X}_\ell)_m + (\mathbf{X}_h)_m.$$

This is just the statement that $x_k = (\mathbf{X}_\ell)_{(k+1)/2} + (\mathbf{X}_h)_{(k+1)/2}$, in accordance with equation (7.20). The synthesis of the signal $x(t)$ from its Haar basis coefficients precisely mirrors the reconstruction \mathbf{x} from the synthesis filter bank.

7.3.2 Multistage Haar Filter Bank and Multiresolution

Multistage filter banks in which we iteratively pass portions of the output back through the filter bank also have a parallel in the functional setting. It's helpful to lay a little groundwork first.

Some Subspaces and Bases For $N \geq 1$ define the set

$$S_N = \{\phi(t-n) : n \in \mathbb{Z}\} \cup \{\psi_{k,n} : 0 \leq k \leq N-1, n \in \mathbb{Z}\}$$

similar to the notation of Example 7.2 except that now we're working over the entire real line. Define the subspace

$$V_N = \text{span}(S_N) \quad (7.26)$$

in $L^2(\mathbb{R})$ for $N \geq 1$. The set S_N is an orthonormal basis for V_N . In the same vein let's also define the subspace $V_0 \subset L^2(\mathbb{R})$ consisting of functions constant on integer intervals $[k, k+1]$. The scaling function translates $\phi(t-n)$ with $n \in \mathbb{Z}$ forms an orthonormal basis for V_0 .

It is worth noting that V_N consists precisely of those functions in $L^2(\mathbb{R})$ that are constant on intervals for the form $[k/2^N, (k+1)/2^N]$ for $k \in \mathbb{Z}$. To see this, notice that any element of V_N is, on any interval $[k/2^N, (k+1)/2^N]$, a finite linear combination of elements of S_N , each of which is constant on intervals $[k/2^N, (k+1)/2^N]$. Thus elements of V_N are also constant on these intervals. Conversely, by Proposition 7.2.2, it follows that any function that is constant on intervals $[k/2^N, (k+1)/2^N]$ can be built as a linear combination of the elements of S_N .

There is another "obvious" basis for V_N . Define functions

$$\phi_{N,n}(t) = 2^{N/2} \phi(2^N t - n), \quad (7.27)$$

where $n \in \mathbb{Z}$ and ϕ is the Haar scaling function defined in equation (7.3). The function $\phi_{N,n}$ is just ϕ scaled to base width $1/2^N$ and translated $n/2^N$ units; the $2^{N/2}$ factor in front scales the L^2 norm to 1. It's easy to see that the set

$$B_N = \{\phi_{N,n} : n \in \mathbb{Z}\},$$

is also an orthonormal basis for V_N . The basis B_N is simpler than S_N , in that all the basis elements are mere translates of each other. But the basis S_N has certain merits of its own, which we detail below.

Multiresolution and Orthogonal Decomposition When we expand a function f with respect to an orthogonal basis for V_N , whether that basis is B_N , S_N , or any other orthogonal basis, we are projecting f onto V_N as per equation (7.13). The projection $f_N = P_{V_N}(f)$ is unique and is the best approximation to f available in the subspace V_N . In the present case Theorem 7.2.2 assures us that f_N converges to f in $L^2(\mathbb{R})$ as $N \rightarrow \infty$.

Of course, in practice we settle for a value of N so that f_N is "good enough" as an approximation. Suppose that we have the approximation f_N in hand but want a better approximation to f . Obviously we have to compute f_{N+1} , f_{N+2} , or higher. In the present setting there are a couple of ways to do this. We can recompute f_{N+1} "directly" by using the basis B_{N+1} , that is, forming the inner products $(f, \phi_{N+1,m})\phi_{N+1,m}$. Alternatively, we can compute f_{N+1} as a "refinement" of f_N . Specifically, we have

$$f_{N+1} = f_N + \delta_N, \quad (7.28)$$

where

$$\delta_N(t) = \sum_{m \in \mathbb{Z}} (f, \psi_{N,m}) \psi_{N,m}(t). \quad (7.29)$$

This follows from the fact that $S_{N+1} = S_N \cup \{\psi_{N,m} : m \in \mathbb{Z}\}$. Equations (7.28) and (7.29) show how to hop from the coarser approximation f_N to the finer approximation f_{N+1} ; δ_N is the required correction.

Of course, this process can be iterated. Starting with the approximation f_N , we have

$$\begin{aligned} f_N &= f_{N-1} + \delta_{N-1} \\ &= f_{N-2} + \delta_{N-2} + \delta_{N-1} \\ &= f_{N-2} + \delta_{N-3} + \delta_{N-2} + \delta_{N-1} \\ &\vdots = \vdots \\ &= f_0 + \delta_1 + \delta_2 + \cdots + \delta_{N-1}. \end{aligned} \quad (7.30)$$

Equation (7.30) is very similar to equation (6.28) from Chapter 6, with one small difference. In equation (6.28) as the index r increases the approximations α_r become coarser, while in equation (7.30) increasing indexes correspond to increasingly detailed features.

In either case, though, the point is that we have a quantitative way to relate coarser and finer approximations, and to step from one to the other using the wavelets. This is the central idea of multiresolution. In the present case we have a nested sequence of subspaces

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2(\mathbb{R}).$$

The wavelets $\psi_{N,m}$ give us a way to refine f_N to f_{N+1} via equations (7.28) and (7.29). Including the shorter scale wavelets $\psi_{N+1,m}$ in the computation allows us to refine f_{N+1} to f_{N+2} , and so on. Since

$$\lim_{N \rightarrow \infty} f_N = f,$$

we can iteratively improve the approximation to any desired degree of accuracy (as quantified by the L^2 norm).

Direct Sums For any fixed $N \in \mathbb{Z}$ let $W_N \subset L^2(\mathbb{R})$ be the subspace defined as

$$W_N = \text{span}(\{\psi_{N,m} : m \in \mathbb{Z}\}) \quad (7.31)$$

formed from the $1/2^N$ scale wavelets $\psi_{N,m}$ alone. That is, W_N consists of functions w of the form

$$w = \sum_{m=-\infty}^{\infty} a_m \psi_{N,m},$$

where $\sum_m a_m^2 < \infty$; the infinite sum for $w(t)$ is understood in the sense of equations (7.11) and (7.12). It's easy to see that $W_N \cap V_N = \{0\}$, since every element in each subspace is orthogonal all of the elements of the other subspace.

According to equation (7.28) we can express $f_{N+1} = f_N + \delta_N$, where $f_N \in V_N$ and $\delta_N \in W_N$. In fact both f_N and δ_N are unique (see Exercise 7.9); δ_N is given by equation (7.29). Since each element of V_{N+1} can be written as the sum of a unique element of V_N and a unique element of W_N , we write

$$V_{N+1} = V_N \oplus W_N.$$

We say that the subspace V_{N+1} is the “direct sum” of the subspaces V_N and W_N . Let us make the more general definition

Definition 7.3.1 *If U_0, \dots, U_n and U are subspaces of a vector space V , we write*

$$U = U_0 \oplus U_1 \oplus \cdots \oplus U_n$$

provided that for each element $u \in U$ there are unique vectors $u_k \in U_k$ such that

$$u = u_0 + u_1 + \cdots + u_n.$$

With the terminology of Definition 7.3.1, equation (7.30) shows that

$$V_N = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{N-1}.$$

The projection f_N of f onto V_N in equation (7.30) can be written as

$$f_N = f_0 + P_{W_0}(f) + \cdots + P_{W_{N-1}}(f)$$

where $P_{W_k}(f)$ denotes the projection of f onto the wavelet space W_k (and f_0 is the projection of f onto V_0). The addition of each wavelet space projection $P_{W_k}(f)$ adds a layer of increased resolution. Moreover, by Theorem 7.2.2, we have $f_N \rightarrow f$ as $N \rightarrow \infty$, so we can reasonably write

$$L^2(\mathbb{R}) = V_0 \oplus W_0 \oplus W_1 \oplus \cdots$$

In all of the above we needn't use V_0 as the base space; we can just as well write

$$V_N = V_M \oplus W_M \oplus W_{M+1} \oplus \cdots \oplus W_{N-1}$$

for any $M < N$, and so

$$L^2(\mathbb{R}) = V_M \oplus W_M \oplus W_{M+1} \oplus \dots$$

Indeed we can even take $M < 0$. In this case V_M consists of functions constant on each interval $[k/2^M, (k+1)/2^M]$, where $k \in \mathbb{Z}$, intervals of length $2^{-M} > 1$ if $M < 0$. The functions $\phi_{M,n}$ defined by equation (7.27) provide an orthonormal basis for the space V_M ; for $M < 0$ the scaling functions get wider and lower. The wavelets $\psi_{k,n}$ are also defined for any $k \in \mathbb{Z}$.

■ EXAMPLE 7.3

Look back to Example 7.1 where we compute approximations to a function $f(t) = t(t-1)(t-2)$ for $0 \leq t \leq 1$. The function f was defined on $[0, 1]$, but let's consider f as a member of $L^2(\mathbb{R})$ via zero extension. The approximation using only the Haar scaling function ϕ_0 yields the approximation

$$f_0(t) = \begin{cases} \frac{1}{4}, & 0 \leq t < 1, \\ 0, & \text{else,} \end{cases}$$

in V_0 . The inclusion of the wavelets $\psi_{0,m}$ (only the translate $m = 0$ matters here) allows us to refine $f_0(t)$ to $f_1(t)$ by adding appropriate variation on the left and right halves of the interval $[0, 1]$, as shown at the left in Figure 7.7. Inclusion of the wavelets $\psi_{1,m}$ yields $f_2(t)$ as shown at the right in Figure 7.7. The functions f_3 and f_4 aren't shown, but $f_5(t)$ is shown in Figure 7.8.

Connection to Multistage Haar Filter Banks Let's consider again the computations of Section 7.3. In particular, consider what happens if we project the function $x(t)$ defined by equation (7.17) onto the subspace V_N for various values of N . Using $N \geq 2$ is pointless, since $x(t)$ doesn't vary at this level of detail. Indeed, if $x_N(t)$ denotes the projection of $x(t)$ onto V_N , then $x_N(t) \equiv x(t)$ for all $N \geq 1$.

We computed the coefficients c_m necessary to project $x(t)$ onto V_0 (functions constant on integer intervals $[m, m+1]$) using equation (7.21), and these coefficients turned out to be simply the low-pass filtered/downsampled version of the associated vector $\mathbf{x} \in L^2(\mathbb{Z})$. The coefficients $c_{0,m}$ of the wavelets yield the adjustment necessary to obtain the projection of $x(t)$ onto V_1 ; moreover $c_{0,m}$ corresponds to the high-pass filtered/downsampled version of \mathbf{x} .

Let's instead compute the projection of $x(t)$ onto the subspace V_{-1} consisting of functions constant on intervals $[2k, 2k+2]$ of length 2. The scaling function basis consists of functions

$$\phi_{-1,m}(t) = \frac{1}{\sqrt{2}} \phi\left(\frac{t}{2} - m\right),$$

where ϕ is the Haar scaling function. These functions have base width two units; changing m to $m + 1$ translates the function 2 units to the right. An easy computation shows that we obtain coefficients

$$c_m = \int_{-\infty}^{\infty} x(t)\phi_{-1,m}(t) dt = \frac{1}{2\sqrt{2}}(x_{4m-1} + x_{4m} + x_{4m+1} + x_{4m+2}).$$

The resulting projection onto V_{-1} is given by $x_{-1}(t) = \sum_m c_m \phi_{-1,m}(t)$.

We can refine our approximation by tossing the wavelets $\psi_{-1,m}$ into the basis. The corresponding coefficients are given by

$$c_{-1,m} = \int_{-\infty}^{\infty} x(t)\psi_{-1,m}(t) dt = \frac{1}{2\sqrt{2}}(x_{4m-1} + x_{4m} - x_{4m+1} - x_{4m+2}).$$

The projection of $x(t)$ onto V_0 can then be computed as

$$x_0(t) = x_{-1}(t) + \sum_m c_{-1,m} \psi_{-1,m}(t).$$

We can further refine the approximation $x_0(t)$ by making use of the $\psi_{0,m}(t)$ wavelets. The coefficients are given by equation (7.22), and we find that

$$x_1(t) = x_0(t) + \sum_m c_{0,m}(t) \psi_{0,m}(t).$$

As noted, $x_1(t) \equiv x(t)$, so there's no point in further refinement.

We already saw in equation (7.23) that $c_{0,m} = (\mathbf{X}_h)_m$, that is, the coefficients of the shortest scale wavelets correspond to the high-pass filtered, downsampled version of \mathbf{x} . To what do c_m and $c_{-1,m}$ correspond? To the vectors $\mathbf{X}_{\ell\ell}$ and $\mathbf{X}_{\ell h}$, or minor variations thereof! In the present case (with Haar filters $(\ell_a)_0 = (\ell_a)_1 = \frac{1}{2}$ and $(\mathbf{h}_a)_0 = -\frac{1}{2}$, $(\mathbf{h}_a)_1 = \frac{1}{2}$) it's easy to check that

$$\begin{aligned} (\mathbf{X}_{\ell\ell})_m &= \frac{1}{4}(x_{4m-3} + x_{4m-2} + x_{4m-1} + x_{4m}), \\ (\mathbf{X}_{\ell h})_m &= \frac{1}{4}(x_{4m-3} + x_{4m-2} - x_{4m-1} - x_{4m}). \end{aligned}$$

In short, though not identical, both c_m and $(\mathbf{X}_{\ell\ell})_m$ consist of moving averages of four successive values, while $c_{-1,m}$ and $(\mathbf{X}_{\ell h})_m$ consist of a mix of averaged/differenced successive values. (With enough clever indexing and rescaling, we could make the corresponding quantities identical.)

A similar parallel holds for the corresponding synthesis filter bank and the synthesis of $x(t)$ from the functions $\phi_{-1,m}$, $\psi_{-1,m}$, and $\psi_{0,m}$. Moreover the parallels exists for three and higher stage banks. As with signals in $L^2(\mathbb{Z})$, we can produce coarser and coarser approximations, to any degree, for any function $f \in L^2(\mathbb{R})$ (analogous to repeatedly low-pass filtering a signal in $L^2(\mathbb{Z})$). However, in the functional setting

we can go the other way, to produce finer and finer approximations to the original function. In the discrete signal case this process is limited by the sampling rate.

7.4 ORTHOGONAL WAVELETS

7.4.1 Essential Ingredients

The essential ingredients for the multiresolution framework in the Haar wavelet case are

1. A nested chain of subspaces

$$\cdots V_{-1} \subset V_0 \subset V_1 \subset$$

all contained in $L^2(\mathbb{R})$, with the properties that

$$\bigcup_{k=-\infty}^{\infty} V_k = L^2(\mathbb{R}) \quad \text{and} \quad \bigcap_{k=-\infty}^{\infty} V_k = \{0\}. \quad (7.32)$$

The intersection property above wasn't spelled out in the Haar setting, but it's easy to see only a constant function can be an element of all the V_k in this case, and the only constant function in $L^2(\mathbb{R})$ is 0. That the union of the V_k comprises $L^2(\mathbb{R})$ is a consequence of Theorem 7.2.2.

2. A function $f(t) \in V_k$ if and only if $f(2t) \in V_{k+1}$. Also, if $f(t) \in V_0$, then $f(t-n) \in V_0$ for any $n \in \mathbb{Z}$.
3. The space V_0 possesses an orthonormal basis of the form $\phi_n(t) = \phi(t-n)$, $n \in \mathbb{Z}$. The function ϕ is called the *scaling function*.

The wavelets themselves are not essential (yet).

Our goal is to generalize this framework to other than the piecewise constant Haar setting. We begin with a definition.

Definition 7.4.1 *A set of subspaces V_k and function ϕ that satisfy properties 1 through 3 above is called a "multiresolution analysis."*

Of course, it's not at all clear that there are any multiresolution analyses other than the one constructed via the Haar functions.

Remark 7.3 It isn't absolutely essential that properties 2 and 3 involve integer translates; this is merely for convenience and convention. Also from property 2 it's easy to see that $f(t) \in V_0$ if and only if $f(2^k t) \in V_k$ for any $k \in \mathbb{Z}$. One can easily deduce that for fixed k the set $\phi(2^k t - n)$, $n \in \mathbb{Z}$ is an orthogonal basis for V_k .

7.4.2 Constructing a Multiresolution Analysis: The Dilation Equation

Suppose that we have a multiresolution analysis with scaling function $\phi \in V_0$. The integer translates $\phi(t - n)$ form an orthonormal basis for V_0 . From property 2 for a multiresolution analysis we see that the half-integer translates $\phi(2t - n)$ all lie in V_1 , are orthogonal, and in fact $\sqrt{2}\phi(2t - n)$ is an orthonormal basis for V_1 . Since $V_0 \subset V_1$ the function ϕ itself lies in V_1 , and so it must be the case that

$$\phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} c_k \phi(2t - k) \quad (7.33)$$

for some constants c_k . Equation (7.33) is called the *dilation equation*. The $\sqrt{2}$ factor out front could be absorbed into the definition of the c_k , but we leave it in order to later emphasize certain parallels to filter banks.

For the Haar scaling function we have $c_0 = c_1 = 1/\sqrt{2}$ with all other $c_k = 0$ (see Example 7.4 below). The fact that only finitely many c_k are nonzero is what makes the Haar multiresolution analysis parallel the filter bank analysis with FIR filters. Can we find other solutions to equation (7.33)—both the c_k and scaling function ϕ —in which only finitely many c_k are nonzero?

Let's see what we can deduce under the assumption that the only nonzero c_k lie in the index range $0 \leq k \leq N - 1$ (but we'll assume c_k is defined for all $k \in \mathbb{Z}$, as 0 outside this range). In this case the dilation equation becomes

$$\phi(t) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(2t - k). \quad (7.34)$$

We also normalize $\|\phi\|^2 = 1$. Compute the inner product in $(\phi, \phi) = 1$ using the expansion on the right in equation (7.34) to obtain

$$\begin{aligned} (\phi, \phi) &= \int_{\mathbb{R}} \phi^2(t) dt \\ &= 2 \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} \int_{\mathbb{R}} c_m c_k \phi(2t - m) \phi(2t - k) dt \\ &= \sum_{k=0}^{N-1} c_k^2, \end{aligned}$$

since the last integral equals 0 if $m \neq k$ and equals $\frac{1}{2}$ if $m = k$. The computation above shows that we must have

$$\sum_{k=0}^{N-1} c_k^2 = 1. \quad (7.35)$$

Now consider computing the inner product $(\phi(t), \phi(t - 1)) = 0$ by using the expansion on the right in equation (7.34). We have

$$\phi(t - 1) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(2t - k - 2),$$

and so

$$\begin{aligned} (\phi(t), \phi(t - 1)) &= \int_{\mathbb{R}} \phi(t) \phi(t - 1) dt \\ &= 2 \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} \int_{\mathbb{R}} c_m c_k \phi(2t - m) \phi(2t - k - 2) dt \\ &= \sum_{k=0}^{N-1} c_k c_{k+2} = 0, \end{aligned}$$

since the last integral equals 0 if $m \neq k + 2$ and equals $\frac{1}{2}$ if $m = k + 2$. In the last sum the upper limit can be changed to $N - 3$, since $c_{k+2} = 0$ for $k > N - 3$, but it's not essential.

Generally, if we compute the inner product $(\phi(t), \phi(t - n)) = 0$ for $0 < n < (N - 1)/2$ in this fashion, we obtain

$$\sum_{k=0}^{N-1-2n} c_k c_{k+2n} = 0. \quad (7.36)$$

The equation above is invariant under the substitution $n \rightarrow -n$, so taking $n < 0$ doesn't add anything. As it turns out, N will have to be even. So equation (7.36) embodies $N/2 - 1$ equations, since n can assume values $1, 2, \dots, (N - 2)/2$.

One more condition on the c_k can be deduced. If we integrate both sides of equation (7.34) in t over the whole real line, we obtain

$$\begin{aligned} \int_{-\infty}^{\infty} \phi(t) dt &= \sqrt{2} \sum_{k=0}^{N-1} c_k \int_{-\infty}^{\infty} \phi(2t - k) dt \\ &= \frac{\sqrt{2}}{2} \sum_{k=0}^{N-1} c_k \int_{-\infty}^{\infty} \phi(t) dt, \end{aligned}$$

since the integral of $\phi(2t - k)$ is half the integral of $\phi(t)$. If the integral of ϕ converges and is nonzero, we can conclude that

$$\sum_{k=0}^{N-1} c_k = \sqrt{2}. \quad (7.37)$$

Since N is even equations (7.35), (7.36), and (7.37) yield $1 + N/2$ equations for N unknowns for the filter coefficients, at least if the equations are independent. For $N > 2$ we have more variables than equations, so we expect many solutions.

■ EXAMPLE 7.4

Consider $N = 2$, in which case equations (7.35) and (7.37) yield

$$c_0^2 + c_1^2 = 1, \quad c_0 + c_1 = \sqrt{2},$$

while equation (7.36) yields the empty statement $0 = 0$. The only solution is $c_0 = c_1 = 1/\sqrt{2}$, which corresponds to the Haar scaling function. It's easy to see that this yields a solution to equation (7.34). Of course, even if we were given $c_0 = c_1 = 1/\sqrt{2}$, it's not clear how we would have come up with the Haar scaling function without already knowing it. In fact it's not clear that the Haar scaling function is the only possibility, even with these values of c_0 and c_1 . We'll look at how to obtain ϕ a bit later.

■ EXAMPLE 7.5

Consider $N = 4$, in which case equations (7.35) and (7.37) yield

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1, \quad c_0 + c_1 + c_2 + c_3 = \sqrt{2},$$

while equation (7.36) yields

$$c_0c_2 + c_1c_3 = 0.$$

With three equations in four unknowns we expect a free variable and a whole family of solutions. Indeed, if we choose $c_3 = t$, then we obtain two families of solutions, one of which looks like

$$c_0 = \frac{\sqrt{2}}{4} + \frac{\sqrt{2 + 8\sqrt{2}t - 16t^2}}{4}, \quad c_1 = \frac{\sqrt{2}}{2} - t,$$

$$c_2 = \frac{\sqrt{2}}{4} - \frac{\sqrt{2 + 8\sqrt{2}t - 16t^2}}{4},$$

where t is a free parameter (a computer algebra system is helpful here, or at least perseverance). The other solution family is similar, but with c_0 and c_2 interchanged. Only in the range $\sqrt{2}/4 - 1/2 \leq t \leq \sqrt{2}/4 + 1/2$ do all c_k remain real.

The choice $t = 0$ above yields the Haar filters, while $t = \sqrt{2}/2$ yields the Haar filters shifted two indexes. The choice $t = (1 - \sqrt{3})/4\sqrt{2}$ yields the Daubechies 4-tap filters from Chapter 6. Similar results are obtained for the second family of solutions.

7.4.3 Connection to Orthogonal Filters

Recall our design strategy from Chapter 6 for the low-pass analysis filter in an orthogonal filter bank. The FIR filter $\ell = (\ell_0, \ell_1, \dots, \ell_{N-1})$ with z -transform $L_a(z)$ had to satisfy $P(z) + P(-z) = 2$ (this is equation (6.49)), where $P(z) = L_a(z)L_a(z^{-1})$.

This requires that $P(z)$ should have coefficient 1 for z^0 and all even index components equal to zero. A straightforward multiplication shows that the coefficients p_m of $P(z)$ are given by

$$p_m = \sum_{k=0}^{N-1-m} \ell_k \ell_{k-m} \quad (7.38)$$

for $0 \leq m < N - 1$, and $p_{-m} = p_m$. The condition that $p_0 = 1$ is, from equation (7.38), just

$$\sum_{k=0}^{N-1} \ell_k^2 = 1. \quad (7.39)$$

Equation (7.38) shows that the condition $p_m = 0$ when m is even becomes (let $m = -2n$; the sign doesn't matter)

$$p_m = \sum_{k=0}^{N-1-2n} \ell_k \ell_{k+2n} = 0. \quad (7.40)$$

Compare the filter design equations (7.39) and (7.40) to equations (7.35) and (7.36) derived from the dilation equation—aside from the variable names, they're identical! Moreover the condition that the low-pass analysis filter satisfy $L_a(-1) = 0$ was equivalent to equation (6.50), which is exactly equation (7.37). Thus the methods we used to find appropriate low-pass filter coefficients will yield appropriate constants c_k for the dilation equation.

7.4.4 Computing the Scaling Function

It can be shown that under appropriate conditions there is a unique function ϕ that satisfies equation (7.34). We explore this in Section 7.4.5. But let's first pause and look at an algorithm for computing the scaling function, under the assumption that this function actually exists and is continuous.

We'll compute $\phi(t)$ at points of the form $t = m/2^n$ for integers m and n , the so-called *dyadic* rational numbers. One obstacle is that ϕ is defined on the entire line, so we'll have to truncate our approximation at some lower and upper bounds for t . What should these limits be?

The Haar scaling function has compact support. It seems reasonable to seek scaling functions in the general case that possess these properties, that is, the scaling functions should be identically zero outside certain lower and upper bounds. What should these bounds be? It's not hard to see that the choice $\phi(t) \equiv 0$ for $t < 0$ is always consistent with equation (7.34) no matter what the behavior of ϕ for $t \geq 0$. This doesn't prove that $\phi(t)$ must be zero for $t < 0$, but it gives us someplace to start. This property turns out to be correct; see Theorem 7.4.2 below. For an upper bound,

suppose that $\phi(t) \equiv 0$ for $t > a$ for some constant a . We can determine a as follows: the function $\phi(2t - k)$ will be zero for $2t - k > a$, that is, $t > (a + k)/2$. For the largest value of k , namely $k = N - 1$, the right side of equation (7.34) will then be identically zero if $t > (a + N - 1)/2$. Choosing a so that $a = (a + N - 1)/2$ will at least be consistent, for then both sides of (7.34) will be identically zero for $t > a$. The condition $a = (a + N - 1)/2$ yields $a = N - 1$.

We thus assume that ϕ is supported on the interval $[0, N - 1]$ and approximate ϕ only on this interval. If ϕ is continuous (which it will be), then $\phi(0) = \phi(N - 1) = 0$. From the dilation equation (7.34) with $t = 1$ we obtain

$$\phi(1) = \sqrt{2}c_0\phi(2) + \sqrt{2}c_1\phi(1).$$

When $t = 2$, we find that

$$\phi(2) = \sqrt{2}c_0\phi(4) + \sqrt{2}c_1\phi(3) + \sqrt{2}c_2\phi(2) + \sqrt{2}c_3\phi(1).$$

More generally, if $t = n \in \mathbb{Z}$, where $1 \leq n \leq N - 2$, we find that

$$\phi(n) = \sqrt{2} \sum_k c_k \phi(2n - k), \quad (7.41)$$

where the range of summation in k is such that $1 \leq 2n - k \leq N - 2$, since $\phi(2n - k)$ will be zero for integers outside this range. Equation (7.41) provides a set of $N - 2$ linear relations among the quantities $\phi(1), \dots, \phi(N - 2)$.

In fact these equations can be cast as an eigenvector problem

$$\mathbf{q} = \mathbf{A}\mathbf{q}, \quad (7.42)$$

where $\mathbf{q} = [\phi(1), \phi(2), \dots, \phi(N - 2)]^T$ is an eigenvector with eigenvalue 1 for the $N - 2 \times N - 2$ matrix \mathbf{A} with entries

$$a_{n,2n-k+1} = \sqrt{2}c_{k-1}$$

for $1 \leq n \leq N - 2$, $1 \leq k \leq N$, and the additional restriction $1 \leq 2n - k + 1 \leq N - 2$ on k . Note that we are indexing matrix columns and rows from 1 now, rather than 0 as was convenient earlier. For example, in the case $N = 8$ the matrix \mathbf{A} is 6×6 and looks like

$$\mathbf{A} = \sqrt{2} \begin{bmatrix} c_1 & c_0 & 0 & 0 & 0 & 0 \\ c_3 & c_2 & c_1 & c_0 & 0 & 0 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \\ c_7 & c_6 & c_5 & c_4 & c_3 & c_2 \\ 0 & 0 & c_7 & c_6 & c_5 & c_4 \\ 0 & 0 & 0 & 0 & c_7 & c_6 \end{bmatrix}.$$

It's not obvious that \mathbf{A} has 1 as an eigenvalue, but it does. We can use any standard method to compute the corresponding eigenvector. We thus obtain $\phi(n)$ for $1 \leq n \leq N - 2$; indeed we obtain ϕ at all integers, since $\phi(n)$ is zero for n outside this range.

We can then compute ϕ at half-integer points $n/2$ for n odd with $1 \leq n \leq 2N - 3$, for from the dilation equation (7.34) we have

$$\phi(n/2) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(n - k).$$

Since $n - k$ is an integer, $\phi(n - k)$ is known. Once we know ϕ at the half-integers, we can compute $\phi(t)$ at points of the form $t = n/4$ where n is odd, $1 \leq n \leq 4N - 5$. We find that

$$\phi(n/4) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(n/2 - k)$$

and the right side is known because $n/2 - k$ is a half-integer. In general, if we have $\phi(n/2^{r-1})$ for some r and $0 < n < 2^{r-1}(N - 1)$, we can compute $\phi(t)$ at points $t = n/2^r$ (odd n) as

$$\phi(n/2^r) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(n/2^{r-1} - k). \quad (7.43)$$

The right side will known from previous values of r .

■ EXAMPLE 7.6

Let us compute the scaling function that corresponds to the Daubechies orthogonal 4-tap filter coefficients

$$c_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}}, \quad c_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, \quad c_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}}, \quad c_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}.$$

In this case the matrix \mathbf{A} is

$$\mathbf{A} = \begin{bmatrix} c_1 & c_0 \\ c_3 & c_2 \end{bmatrix}.$$

This matrix does indeed have 1 as an eigenvalue (also $\frac{1}{2}$), with eigenvector

$$\begin{bmatrix} \phi(1) \\ \phi(2) \end{bmatrix} = \begin{bmatrix} 1 + \sqrt{3} \\ 1 - \sqrt{3} \end{bmatrix}.$$

Of course, the eigenvector can be normalized in any way we wish. We'll wait until we've computed ϕ at a number of other points, then normalize (approximately) to unit L^2 norm.

At the half-integers we find that

$$\phi\left(\frac{1}{2}\right) = \sqrt{2}c_0\phi(1) = 1 + \frac{\sqrt{3}}{2},$$

$$\phi\left(\frac{3}{2}\right) = \sqrt{2}(c_1\phi(2) + c_2\phi(1)) = 0,$$

$$\phi\left(\frac{5}{2}\right) = \sqrt{2}c_3\phi(2) = 1 - \frac{\sqrt{3}}{2}.$$

We can continue the process via equation (7.43). In Figure 7.12 we show the results on the interval $[0, 3]$ using $r = 2, 4, 6, 8$ (dyadic points $t = n/2^r$). In each case the vector $\mathbf{v} \in \mathbb{R}^{2^r(N-1)}$ that approximates ϕ is scaled so that $\|\mathbf{v}\|^2 = 2^r(N-1)$, a discrete analog to $\|\phi\|^2 = 1$. The case $r = 2$ is at the upper left, $r = 4$ the upper right, $r = 6$ the lower left, $r = 8$ the lower right.

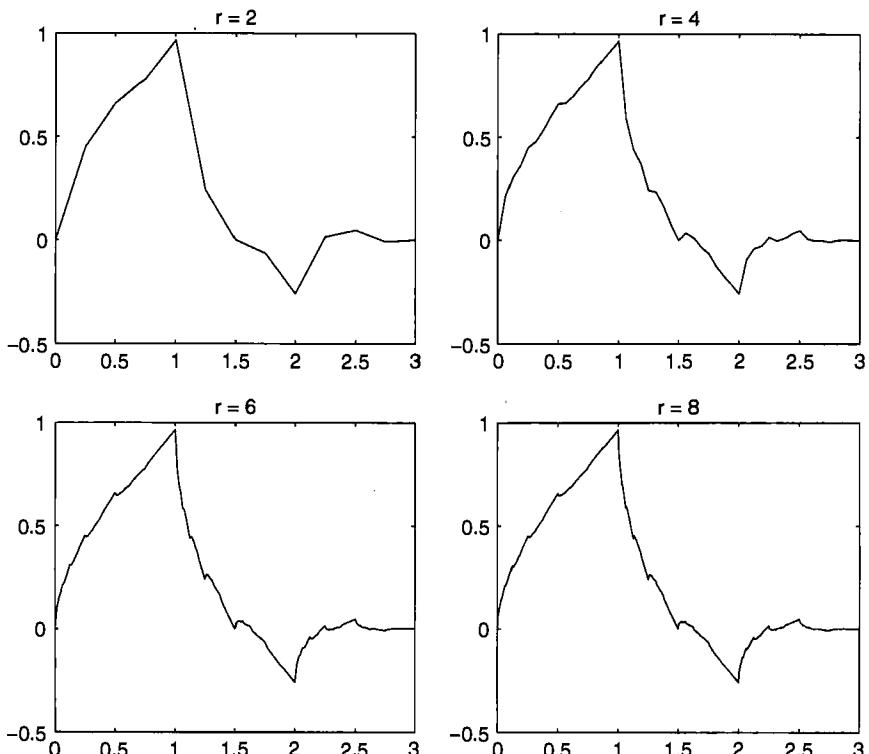


FIGURE 7.12 Approximations to Daubechies 4-tap orthogonal filter scaling function.

Remark 7.4 The preceding procedure by which we compute the scaling function $\phi(t)$ at dyadic t makes the implicit assumption that ϕ is defined at these points, indeed that $\phi(t)$ is sufficiently “nice” to have well-defined point values anywhere. This is not the case for arbitrary functions in $L^2(\mathbb{R})$ (recall the discussion of $L^2(a, b)$ in Section 1.10.4), but it is true for continuous functions. Almost all of the scaling functions and wavelets that we’ll encounter are in fact continuous. The exceptions are the Haar scaling function (which is continuous at all but two points) and one other notable exception in Section 7.5.

7.4.5 Scaling Function Existence and Properties

Fixed Point Iteration and the Cascade Algorithm Let c_0, \dots, c_{N-1} satisfy equations (7.35), (7.36), and (7.37) in the range $0 < n < (N - 1)/2$. From the work in Section 6.7.6 we know how to find such c_k .

There is rarely a closed-form expression for the scaling function, as Figure 7.12 ought to suggest; Haar and a few other simple examples are the only exceptions. In general, the scaling function must be computed, and its properties deduced, by recursive algorithms. The procedure outlined above for computing ϕ at dyadic rationals is a good example. For proving the actual existence of the scaling function the *cascade algorithm* is useful.

The cascade algorithm is an example of *fixed point iteration*. If F is a mapping that takes elements of a set V back into V , then an element $\mathbf{v}^* \in V$ is called a *fixed point* of F provided that $\mathbf{v}^* = F(\mathbf{v}^*)$. Fixed point iteration seeks such a \mathbf{v}^* by starting with an initial guess \mathbf{v}^0 , and then iterating $\mathbf{v}^{j+1} = F(\mathbf{v}^j)$ (the superscript here is an index, not a power). If we have a notion of distance on V (e.g., if V is a normed vector space), then under appropriate conditions it will be the case that \mathbf{v}^j converges to a fixed point \mathbf{v}^* ; that is, $\lim_{j \rightarrow \infty} \|\mathbf{v}^j - \mathbf{v}^*\| = 0$. We can stop iterating when $\|\mathbf{v}^{j+1} - \mathbf{v}^j\|$ is sufficiently small. The method may or may not converge, depending on the nature of F and/or the initial guess; see Exercise 7.17.

In the present case the dilation equation (7.34) can be cast as

$$\phi = F(\phi),$$

where F is the mapping from $L^2(\mathbb{R})$ to $L^2(\mathbb{R})$ defined by

$$F(\phi)(t) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi(2t - k). \quad (7.44)$$

The scaling function we seek is a fixed point for F . We begin with an initial guess ϕ^0 . A typical choice is to take ϕ^0 as the Haar scaling function. We then iteratively compute $\phi^{j+1} = F(\phi^j)$, or more explicitly,

$$\phi^{j+1}(t) = \sqrt{2} \sum_{k=0}^{N-1} c_k \phi^j(2t - k). \quad (7.45)$$

Remember, we treat the c_k as known. Of course, the process is in general hopelessly unwieldy if done symbolically. An exception is the Haar function when $\phi^1 = \phi^0$, and we're done! A few other cases can also be analyzed in closed form, but we're primarily interested in the cascade algorithm as a theoretical tool to show the existence of the scaling function.

Existence of the Scaling Function In equations (7.42) and (7.43) we outlined an algorithm to compute a function ϕ that satisfies the dilation equation (7.34) but never actually showed that ϕ exists. In developing the algorithm, we also made certain unjustified assumptions concerning the properties that ϕ possesses (e.g., that ϕ is zero outside of $[0, N - 1]$). In this section we'll examine these issues. However, we won't generally provide detailed proofs, only sketches or references.

First, let's consider the existence of the scaling function that satisfies the dilation equation (7.34). We assume that the c_k have been determined and satisfy equations (7.35), (7.36), and (7.37). As remarked, equations (7.35) and (7.36) are really the filter bank perfect reconstruction equations (7.39) and (7.40) (which themselves are equivalent to $P(z) + P(-z) = 2$, equation (6.49)). Equation (7.37) is a restatement of the low-pass condition (6.50) on the input filter ℓ_a .

With slightly different notation (and filters scaled differently, by a factor $\sqrt{2}$), the authors of [3] prove the following theorem (Theorem 5.23 in that text):

Theorem 7.4.1 *Let $L_a(z) = \sum_{k=0}^{N-1} c_k z^k$ be a polynomial that satisfies the following conditions:*

1. $L_a(1) = \sqrt{2}$
2. $|L_a(z)|^2 + |L_a(-z)|^2 = 2$ for $|z| = 1$
3. $|L_a(e^{it})| > 0$ for $|t| \leq \pi/2$

Let ϕ^0 be the Haar scaling function and the sequence ϕ^j be defined by equation (7.45). Then the sequence ϕ^j converges both pointwise and in $L^2(\mathbb{R})$ to a function ϕ that satisfies the dilation equation (7.34). Moreover ϕ satisfies $\|\phi\| = 1$ and the orthogonality condition

$$\int_{-\infty}^{\infty} \phi(t-m)\phi(t-n) dt = 0$$

for $m \neq n$.

If L_a has real coefficients then $|L_a(e^{it})| = |L_a(e^{-it})|$ (Exercise 7.15), so we can use $L_a(z) = \sum_{k=0}^{N-1} c_k z^{-k}$ in the statement of the theorem too (thus matching the conventional z -transform of the filter ℓ_a). It's also easy to show that the function ϕ so produced satisfies $\int_{\mathbb{R}} \phi(t) dt = 1$; see Exercise 7.21.

The first condition, $L_a(1) = \sqrt{2}$, is just equation (7.37), or alternatively, equation (6.50). Equation (6.50) is itself equivalent to $L_a(-1) = 0$, which quantifies the fact

that the analysis filter ℓ_a is low-pass. Thus $L_1(1) = \sqrt{2}$ will be satisfied for the filter coefficients c_k (or ℓ_k), which we obtain by the methods of Section 6.7.6.

The second condition, $|L_a(z)|^2 + |L_a(-z)|^2 = 2$ for $|z| = 1$, is also satisfied. To see this, first note that the coefficients $L_a(z)$ are real numbers, and also if $|z| = 1$, then $\bar{z} = z^{-1}$. As a consequence $L_a(z^{-1}) = \overline{L_a(z)}$. Recall also that the product filter satisfies $P(z) = L_a(z)L_a(z^{-1})$ (this is equation (6.47)). As a result for $|z| = 1$ we have

$$P(z) = L_a(z)L_a(z^{-1}) = L_a(z)\overline{L_a(z)} = |L_a(z)|^2. \quad (7.46)$$

Similarly $P(-z) = |L_a(-z)|^2$. This last equation, in conjunction with equation (7.46) and the fact that $P(z) + P(-z) = 2$ (equation (6.49)), then yields $|L_a(z)|^2 + |L_a(-z)|^2 = 2$.

The third condition doesn't automatically follow from our construction of the c_k or ℓ_k but needs to be checked on a case-by-case or family-by-family basis. Of course, all this condition really says is that $L_a(e^{it}) \neq 0$ for $-\pi/2 \leq t \leq \pi/2$, or equivalently, $P(e^{it}) \neq 0$.

■ EXAMPLE 7.7

Consider the orthogonal Haar filter coefficients $c_0 = c_1 = 1/\sqrt{2}$. In this case $L_a(z) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}z^{-1}$. In particular

$$L_a(e^{it}) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}e^{-it}.$$

It's easy to check that

$$|L_a(e^{it})| = \sqrt{1 + \cos(t)}$$

and to see that $|L_a(e^{it})| \geq 1 > 0$ for $|t| \leq \pi/2$. Thus the cascade algorithm will converge to a scaling function with the required properties. Of course, we already knew this, since $\phi^1 = \phi^0$.

■ EXAMPLE 7.8

Consider the Daubechies filter coefficients, in which case

$$L_a(z) = \frac{1}{4\sqrt{2}}((1 + \sqrt{3}) + (3 + \sqrt{3})z^{-1} + (3 - \sqrt{3})z^{-2} + (1 - \sqrt{3})z^{-3}).$$

We can plot $|L_a(e^{it})|$ for $-\pi/2 \leq t \leq \pi/2$ and easily see that $|L_a(e^{it})| > 0$. However, a bit of algebra yields

$$|L_a(e^{it})| = \left(1 + \frac{3}{2}\cos(t) - \frac{1}{2}\cos^3(t)\right)^{1/2} = (1 + \cos(t))\sqrt{1 - \cos(t)/2},$$

which makes it clear that $|L_a(t)| > 0$ for $t \in [-\pi/2, \pi/2]$. An appropriate scaling function then exists, and the cascade algorithm will converge to this function.

■ EXAMPLE 7.9

Consider the case $N = 4$ with $c_0 = c_3 = 1/\sqrt{2}$ and $c_1 = c_2 = 0$. Then

$$L_a(z) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}z^{-3}.$$

Condition 1 of Theorem 7.4.1 is satisfied. Condition 2 follows from

$$|L_a(e^{it})|^2 = 1 + \cos(3t),$$

and so $|L_a(-e^{it})|^2 = |L_a(e^{i(t+\pi)})|^2 = 1 + \cos(3(t + \pi)) = 1 - \cos(3t)$. However, the third condition is not satisfied, for $L_a(e^{i\pi/3}) = 0$. What does the cascade algorithm do in this case? The iteration of (7.44) becomes

$$\phi^{j+1}(t) = \phi^j(2t) + \phi^j(2t - 3).$$

With ϕ^0 as the Haar scaling function, it's not hard to compute that

$$\phi^1(t) = \begin{cases} 1, & t \in [0, \frac{1}{2}) \text{ or } t \in [\frac{3}{2}, 2), \\ 0, & \text{else,} \end{cases}$$

and

$$\phi^2(t) = \begin{cases} 1, & t \in [0, \frac{1}{4}), t \in [\frac{3}{4}, 1), t \in [\frac{3}{2}, \frac{7}{4}), t \in [\frac{9}{4}, \frac{5}{2}), \\ 0, & \text{else.} \end{cases}$$

A straightforward induction proof (Exercise 7.22) shows that

$$\phi^j(t) = \begin{cases} 1, & t \in \left[\frac{3k}{2^j}, \frac{3k+1}{2^j} \right), 0 \leq k < 2^j, \\ 0, & \text{else.} \end{cases}$$

That is, ϕ^j oscillates between the values 0 and 1, on intervals of length $1/2^j$, and converges to no meaningful limit in the pointwise or L^2 sense. In Figure 7.13 we show a few iterates. Although ϕ^j doesn't converge to anything in $L^2(\mathbb{R})$, there IS a solution to the dilation equation in this instance—see Exercise 7.23. Moreover the sequence ϕ^j exhibits what is called “weak L^2 convergence”; see [16] for more on weak convergence.

Example 7.9 illustrates one important fact: *not every orthogonal filter bank yields a multiresolution analysis*. Some additional condition on the low-pass filter (e.g., condition 3 of Theorem 7.4.1) is necessary.

The Support of the Scaling Function Previously we argued informally that the scaling function should be supported in the interval $[0, N - 1]$, a fact we can now prove.

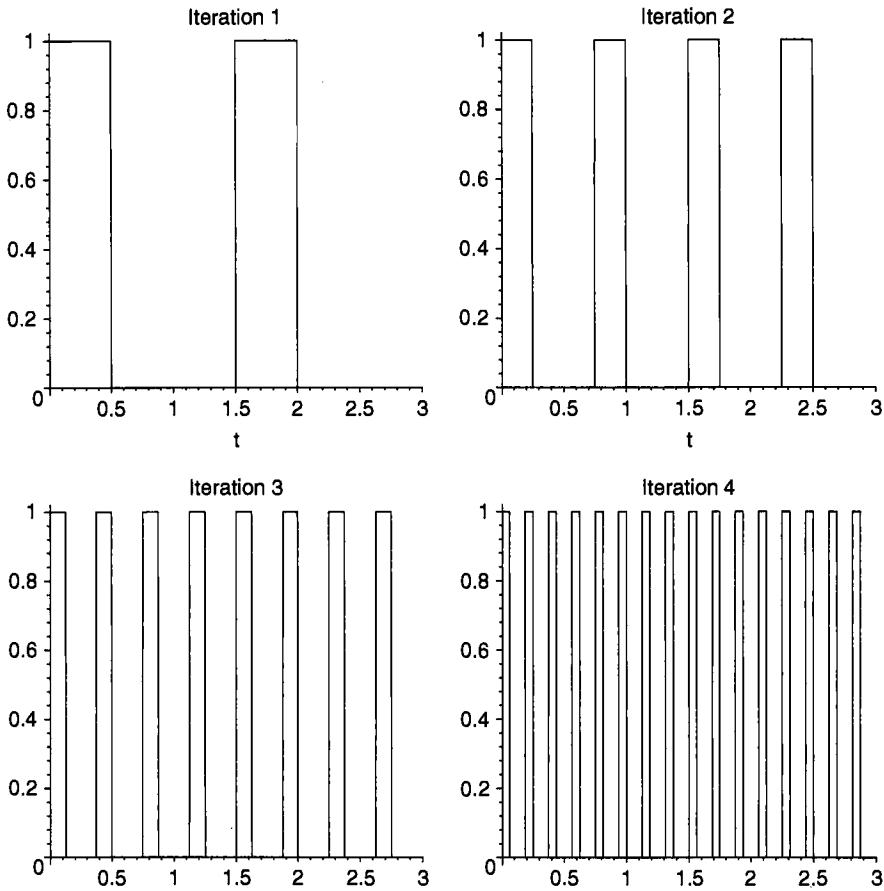


FIGURE 7.13 Iterates $\phi^1, \phi^2, \phi^3, \phi^4$ of cascade algorithm for filter $(\frac{1}{2}, 0, 0, \frac{1}{2})$.

Theorem 7.4.2 *With the assumptions of Theorem 7.4.1, the scaling function ϕ that satisfies equation (7.34) is supported in the interval $[0, N - 1]$.*

Proof First, we claim that all iterates of the cascade algorithm are supported in $[0, N - 1]$. We prove this by induction. The function ϕ^0 is indeed supported on $[0, N - 1]$. Suppose that the same holds true for ϕ^j for some fixed j . The function $\phi^j(2t - k)$ is then supported on the interval $[k/2, (N - 1)/2 + k/2]$. Since $0 \leq k \leq N - 1$, the interval $[k/2, (N - 1)/2 + k/2]$ is contained in $[0, N - 1]$. Thus the sum $\sum_{k=0}^{N-1} \phi^j(2t - k)$ is supported in $\cup_{k=0}^{N-1} [k/2, (N - 1)/2 + k/2] = [0, N - 1]$ (where we use the fact that support of a sum of functions is contained in the union of the individual supports; see Exercise 7.20). The induction hypothesis is shown, and we conclude that all iterates ϕ^j are supported in $[0, N - 1]$.

This means that $\phi^j(t) = 0$ for all t outside of $[0, N - 1]$. By Theorem 7.4.1 the iterates converge pointwise to ϕ , so that

$$\phi(t) = \lim_{j \rightarrow \infty} \phi^j(t) = 0$$

for t outside $[0, N - 1]$. Thus ϕ is supported in $[0, N - 1]$. ■

Other properties of the scaling function, such as its differentiability (which depends on N and the c_k), can also be deduced; see Chapter 7 of [24].

Back to Multiresolution A multiresolution analysis as in Definition 7.4.1 requires more than just the scaling function—it also requires the subspaces V_k with the appropriate properties. Let ϕ be the scaling function for an appropriate set of c_k , and define

$$V_k = \text{span}\{\phi(2^k t - m) : m \in \mathbb{Z}\} \subset L^2(\mathbb{R}), \quad (7.47)$$

where the span is understood in the sense of equations (7.11) and (7.12). It's easy to see that properties 2 and 3 for a multiresolution analysis hold. The nesting property $\dots \subset V_1 \subset V_0 \subset V_1 \subset \dots$ holds by virtue of the fact that ϕ satisfies the dilation equation. Only the properties of equation (7.32) need verification (which we did explicitly in the case of the Haar wavelet). These properties are demonstrated in Theorem 5.17 and Appendix A of [3] (the theorem stated in [3] requires $\int \phi(t) dt = 1$, which follows in the present case; see Exercise 7.21).

Let us summarize the results of this section.

Theorem 7.4.3 *Let coefficients $c_k, 0 \leq k \leq N - 1$ satisfy the conditions of Theorem 7.4.1 and ϕ be the solution to the dilation equation (7.34). Let the V_k be defined via equation (7.47). Then the V_k together with ϕ form a multiresolution analysis.*

7.4.6 Wavelets

For the Haar multiresolution analysis, the Haar wavelets provide a technique for conveniently refining the projection of a function onto V_k to the projection onto V_{k+1} . Since projection onto V_{k+1} embodies greater detail than projection onto V_k , this gives us a way to easily and incrementally improve resolution, to any desired degree of accuracy.

The same approach works more generally. Recall that the mother Haar wavelet in equation (7.1) can be constructed as a linear combination of scaling function dilations and translations, as $\psi(t) = \phi(2t) - \phi(2t - 1)$. Let us write this as

$$\psi(t) = \sqrt{2}d_0\phi(2t) + \sqrt{2}d_1\phi(2t - 1),$$

where $d_0 = 1/\sqrt{2}$, $d_1 = -1/\sqrt{2}$. In fact d_0 and d_1 are precisely the high-pass Haar filter coefficients h_0 and h_1 . In light of equation (6.45) of Chapter 6 this suggests that we should define the wavelet in the general case as $\psi = \sqrt{2} \sum_k d_k \phi(2t - k)$, where $d_k = (-1)^k \ell_{N-k-1}$. Since $\ell_k = c_k$ in our normalization, this yields

$$\psi(t) = \sqrt{2} \sum_{k=0}^{N-1} d_k \phi(2t - k), \quad (7.48)$$

where $d_k = (-1)^k c_{N-k-1}$. Precisely the same argument in the proof of Theorem 7.4.2 shows that the wavelet ψ is supported on the interval $[0, N - 1]$. The function $\psi(t)$, like ϕ , satisfies $\|\psi\| = 1$ (using the $L^2(\mathbb{R})$ norm); see Exercise 7.24.

Is equation (7.48) the correct definition for the wavelet? What we want is that the set

$$S_1 = \{\phi(t - m) : m \in \mathbb{Z}\} \cup \{\psi(t - m) : m \in \mathbb{Z}\} \quad (7.49)$$

provide an orthogonal basis for the space V_1 , an alternative to the basis $B_1 = \{\phi(2t - m) : m \in \mathbb{Z}\}$ of half-width scaling function translates. This is the case, which we demonstrate in the next few propositions. Note that $\psi(t - m)$ is in fact in V_1 , for it follows from equation (7.48) that $\psi(t - m)$ is a finite linear combination of translates $\phi(2t - k)$. We also have the following proposition.

Proposition 7.4.1 *The integer translates $\psi(t - m)$ and $\psi(t - n)$ are orthogonal for $m \neq n$. Also, the function $\psi(t - m)$ is orthogonal to $\phi(t - n)$ for any integer n .*

Proof Let's first show $\psi(t - m)$ is orthogonal to $\psi(t - n)$. In the equations below all sums can be taken with limits $-\infty$ to ∞ with the understanding that only finitely many terms will be nonzero in any sum (since $c_k = 0$ for $k < 0$ or $k \geq N$). We can compute the inner product $(\psi(t - m), \psi(t - n))$ as

$$\begin{aligned} (\psi(t - m), \psi(t - n)) &= \int_{-\infty}^{\infty} \psi(t - m) \psi(t - n) dt \\ &= \int_{-\infty}^{\infty} \psi(t) \psi(t + r) dt \quad (\text{substitute } r = m - n \neq 0) \\ &= 2 \int_{-\infty}^{\infty} \left(\sum_j (-1)^j c_{N-j-1} \phi(2t - j) \right) \\ &\quad \times \left(\sum_k (-1)^k c_{N-k-1} \phi(2t + 2r - k) \right) \end{aligned}$$

$$\begin{aligned}
&= 2 \sum_j \sum_k (-1)^{j+k} c_{N-j-1} c_{N-k-1} \left(\int_{-\infty}^{\infty} \phi(2t-j) \phi(2t+2r-k) dt \right) \\
&= \sum_k c_{N+2r-k-1} c_{N-k-1} \\
&= \sum_p c_{p+2r} c_p \quad (\text{let } p = N - k - 1) \\
&= 0,
\end{aligned} \tag{7.50}$$

since the last integral above is zero unless $j = 2r - k$, in which case it equals $\frac{1}{2}$ (and then $(-1)^{j+k} = (-1)^{2r} = 1$). The last line follows from equation (7.36), the “double shift” orthogonality of the c_k .

A computation similar to equation (7.50) shows that $\psi(t-m)$ is orthogonal to $\phi(t-n)$ for any n ; see Exercise 7.13. ■

Thus the translates $\psi(t-n)$ are elements of V_1 , orthogonal to each other and to the $\phi(t-k)$. But does this collection S_1 of equation (7.49) actually span V_1 ? This was easy to show explicitly in the Haar case, since each $\phi(2t-m)$ can be built as a superposition of the $\phi(t-k)$ and $\psi(t-k)$. The following result will show this in the general case, and also illustrate the connection between filter banks and multiresolution in the general case, much as equation (7.23) did for the Haar filter bank and multiresolution analysis. Keep in mind that the c_k and d_k here are nothing more than the low-pass and high-pass coefficients ℓ_k and h_k from the orthogonal filter banks of the previous chapter.

Proposition 7.4.2 *Let $\psi(t)$ be defined as in equation (7.48) with $d_k = (-1)^k c_{N-k-1}$, where c_k satisfies the conditions of Theorem 7.4.1. Let*

$$g(t) = \sum_k a_k \phi(2t+k)$$

with $\sum_k a_k^2 < \infty$ be a function in V_1 . Then

$$(g, \phi(t+m)) = \frac{1}{\sqrt{2}} (D(\mathbf{a} * \mathbf{c}))_m,$$

$$(g, \psi(t+m)) = \frac{1}{\sqrt{2}} (D(\mathbf{a} * \mathbf{d}))_m,$$

where $\mathbf{a}, \mathbf{c}, \mathbf{d}$ denote the vectors in $L^2(\mathbb{Z})$ with components a_k, c_k , and d_k , respectively, and D denotes the downsampling operator.

In short, the process of taking the inner product of g against $\phi(t+m)$ precisely parallels the low-pass filtering and downsampling of the sequence \mathbf{a} with the filter \mathbf{c} (corresponding to ℓ_a from Chapter 7). The inner products against $\psi(t+m)$ have a similar interpretation with the high-pass filter \mathbf{d} (which corresponds to \mathbf{h}_a).

Proof Let's first compute the inner product against the scaling function translates, as

$$\begin{aligned}
 (g, \phi(t + m)) &= \int_{-\infty}^{\infty} g(t) \phi(t + m) dt \\
 &= \sqrt{2} \int_{-\infty}^{\infty} \left(\sum_j a_j \phi(2t + j) \right) \left(\sum_k c_k \phi(2t + 2m - k) \right) dt \\
 &= \sqrt{2} \sum_j \sum_k a_j c_k \left(\int_{-\infty}^{\infty} \phi(2t + j) \phi(2t + 2m - k) dt \right) \\
 &= \frac{1}{\sqrt{2}} \sum_k a_{2m-k} c_k,
 \end{aligned}$$

where we've used the dilation equation (7.33) and the fact that the last integral is $\frac{1}{2}$ if $j = 2m - k$ and zero otherwise. Note that $(g, \phi(t + m))$ is exactly $(\mathbf{a} * \mathbf{c})_{2m}$, the index $2m$ component of the convolution of \mathbf{a} and \mathbf{c} (considered as element of $L^2(\mathbb{Z})$). Equivalently,

$$(g, \phi(t + m)) = \frac{1}{\sqrt{2}} (D(\mathbf{a} * \mathbf{c}))_m,$$

where D is the downsampling operator. This is the first assertion in the theorem.

The computation of the inner product $(g, \psi(t + m))$ is almost identical, with equation (7.48) in place of the dilation equation, and it yields

$$(g, \psi(t + m)) = \frac{1}{\sqrt{2}} \sum_k a_{2m-k} d_k.$$

This is equivalent to

$$(g, \psi(t + m)) = \frac{1}{\sqrt{2}} (D(\mathbf{a} * \mathbf{d}))_m,$$

where \mathbf{d} is the high-pass filter. This proves the theorem. ■

Proposition 7.4.3 *The set S_1 of equation (7.49) spans V_1 defined by equation (7.47).*

Proof First, by Proposition 7.4.1, the set S_1 is orthogonal, and indeed orthonormal, since $\|\phi\| = 1$ by design and $\|\psi\| = 1$ follows (Exercise 7.24). Thus any function $f \in L^2(\mathbb{R})$ that lies in the span of S_1 can be expressed as

$$f = \sum_k [((f, \phi(t - k))\phi(t - k) + (f, \psi(t - k))\psi(t - k)]. \quad (7.51)$$

On the other hand, if f is not in the span then the right side above will not equal f .

We will show that $\phi(2t)$ lies in the span of S_1 ; the other translates $\phi(2t - m)$ are similar. Indeed we'll show $\phi(2t)$ is a linear combination of *finitely* many elements of S_1 , and then use this to show that any function that can be built as a square-summable superposition of translates $\phi(2t - n)$ (i.e., anything in V_1) can also be built using a square-summable superposition of elements of S_1 , so S_1 spans V_1 .

Let $f(t) = \phi(2t)$ in equation (7.51). We want to show that both sides are identical, or equivalently, that the function $g(t) \in V_1$ defined by

$$g(t) = \phi(2t) - \sum_k [((\phi(2t), \phi(t - k))\phi(t - k) + (\phi(2t), \psi(t - k))\psi(t - k)] \quad (7.52)$$

is identically zero. In this case the sum in k involves only finitely many terms, since for large k the supports of the functions in the inner products do not overlap. Because $g \in V_1$, we have

$$g(t) = \sum_k a_k \phi(2t + k), \quad (7.53)$$

where $\sum_k a_k^2 < \infty$. Moreover from equation (7.52) it's easy to check that $(g, \phi(t + m)) = 0$ and $(g, \psi(t + m)) = 0$ for all $m \in \mathbb{Z}$. But, by Proposition 7.4.2, this implies that with input signal $\mathbf{a} \in L^2(\mathbb{Z})$ the output of the corresponding filter bank is identically zero, and since the filter bank is perfect reconstruction (in particular, invertible), we must conclude that $a_k = 0$ for all $k \in \mathbb{Z}$. From equation (7.53) we then have $g \equiv 0$, so $\phi(2t)$ lies in the span of S_1 . The same conclusion holds for any other translates: any particular $\phi(2t - m)$ can be built as a superposition of finitely many translates $\phi(t - k)$ and $\psi(t - k)$.

To show that any $f \in V_1$ can be built as a square-summable superposition of the elements of S_1 , let

$$f = \sum_m a_m \phi(2t - m) \quad (7.54)$$

be any element of V_1 (so $\sum_k a_k^2 < \infty$) and define

$$\tilde{f} = \sum_{m=M}^N a_m \phi(2t - m) \quad (7.55)$$

for fixed M, N . Notice that \tilde{f} is a finite linear combination of the $\phi(2t - m)$ (note \tilde{f} depends on M and N , but we don't explicitly indicate this). For any $\epsilon > 0$ we can obtain $\|f - \tilde{f}\| < \epsilon$ by choosing M and N appropriately. For notational convenience let $p_k^m = (\phi(2t - m), \phi(t - k))$ and $q_k^m = (\phi(2t - m), \psi(t - k))$ so that from above we have

$$\phi(2t - m) = \sum_k p_k^m \phi(t - k) + \sum_k q_k^m \psi(t - k)$$

(because $\phi(2t - m)$ is in the span of S_1). As remarked, the sums in k are in fact finite. For \tilde{f} of equation (7.55) we then have

$$\begin{aligned}
 \tilde{f} &= \sum_{m=M}^N a_m \phi(2t - m) \\
 &= \sum_{m=M}^N a_m \sum_k (p_k^m \phi(t - k) + q_k^m \psi(t - k)) \\
 &= \sum_k \sum_{m=M}^N (a_m p_k^m \phi(t - k) + a_m q_k^m \psi(t - k)) \\
 &= \sum_k (\tilde{p}_k \phi(t - k) + \tilde{q}_k \psi(t - k)),
 \end{aligned} \tag{7.56}$$

where

$$\tilde{p}_k = \sum_{m=M}^N a_m p_k^m, \quad \tilde{q}_k = \sum_{m=M}^N a_m q_k^m.$$

The interchange of sums in k and m is justified by the fact that both have a finite index range. Equation (7.56) shows that we can build \tilde{f} exactly from a finite linear combination of elements of S_1 . Since we can obtain $\|f - \tilde{f}\| < \epsilon$ for any positive ϵ (by taking appropriate finite values for M and N), the set S_1 spans V_1 and in fact forms an orthonormal basis. This completes the proof. ■

The function ψ defined by equation (7.48) is called the *mother wavelet* for the corresponding multiresolution analysis.

■ EXAMPLE 7.10

In Figure 7.14 is a picture of the wavelet obtained from equation (7.48) and the previously computed scaling function ϕ for the Daubechies 4-tap filters. The wavelet is approximated as follows: Suppose the scaling function ϕ is supported on $[0, N - 1]$ at dyadic rational points $t = m/2^r$, $m = 0$ to $m = 2^r(N - 1)$. Let \mathbf{q} denote the vector in $\mathbb{R}^{2^r(N-1)+1}$ with components $q_m = \phi(m/2^r)$. If we discretize equation (7.48) in the obvious way we obtain

$$w_m = \sqrt{2} \sum_{k=0}^{N-1} (-1)^k c_{N-k-1} q_{2m-k2^r},$$

where $w_m = \psi(m/R)$ is the discrete version of the wavelet function (so $\mathbf{w} \in \mathbb{R}^{2^r(N-1)+1}$). Of course, we sum only over those k such that $0 \leq 2m - k2^r \leq 2^r(N - 1)$. In this example we use $r = 7$. Compare the function in Figure 7.14 to those of Figure 7.3.

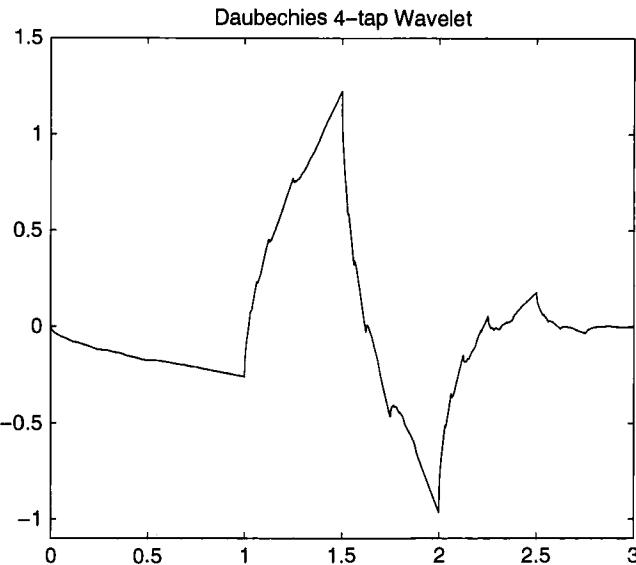


FIGURE 7.14 Mother wavelet corresponding to Daubechies 4-tap filter.

7.4.7 Wavelets and the Multiresolution Analysis

Let W_0 be the subspace of $L^2(\mathbb{R})$ spanned by the translates $\psi(t - m)$, exactly as in the Haar framework. The set $S_0 = \{\cup_{m \in \mathbb{Z}} \phi(t - m)\}$ provides an orthogonal basis for the space V_0 and $S_1 = \{\cup_{m \in \mathbb{Z}} \phi(t - m)\} \cup \{\cup_{m \in \mathbb{Z}} \psi(t - m)\}$ provides an orthogonal basis for the space V_1 . The addition of the wavelets $\psi(t - m)$ to S_0 doubles our resolution. Moreover the subspace W_0 is orthogonal to V_0 in that, if $f \in V_0$ and $g \in W_0$, then $(f, g) = 0$. As a result we have

$$V_1 = V_0 \oplus W_0.$$

Similar reasoning shows that $V_2 = V_1 \oplus W_1 = V_0 \oplus W_0 \oplus W_1$, where W_1 is the subspace spanned by the wavelets $\psi(2t - n)$ for $n \in \mathbb{Z}$. Each $\psi(2t - n)$ is orthogonal to each basis function $\phi(2t - n) \in V_1$; hence each element of W_2 is orthogonal to each element of V_1 . In particular, the function $\psi(2t - n)$ is orthogonal to all of the longer scale wavelets $\psi(t - m)$.

An iteration of the argument above shows that in general for any $N > 0$ we have

$$V_N = V_0 \oplus W_0 \oplus \cdots \oplus W_{N-1},$$

where W_k is spanned by the functions $\psi(2^k t - n)$, $n \in \mathbb{Z}$. Clearly, there's nothing special about using V_0 as the "base space." For any $M < N$ (where M and/or N may be negative) we have

$$V_N = V_M \oplus W_M \oplus \cdots \oplus W_{N-1}.$$

Let us define normalized scaling functions and wavelets

$$\phi_{k,m}(t) = 2^{k/2}\phi(2^k t - m), \quad \psi_{k,m}(t) = 2^{k/2}\psi(2^k t - m)$$

for all $k, m \in \mathbb{Z}$ (the $2^{k/2}$ in front is to obtain L^2 norm 1) and also

$$S_{M,N} = \{\phi_{M,m} : m \in \mathbb{Z}\} \cup \{\psi_{k,m} : M \leq k \leq N-1, m \in \mathbb{Z}\} \quad (7.57)$$

for $M \leq N$. An iteration of the arguments above yields the following theorem:

Theorem 7.4.4 *The set $S_{M,N}$ defined in equation (7.57) is an orthonormal basis for the subspace*

$$V_N = \text{span}(\{\phi_{N,m} : m \in \mathbb{Z}\})$$

of $L^2(\mathbb{R})$, for any $M \leq N$. Moreover we have

$$V_N = V_M \oplus W_{M+1} \oplus \cdots \oplus W_{N-1},$$

where W_k denotes the subspace of $L^2(\mathbb{R})$ with orthonormal basis $\bigcup_{m \in \mathbb{Z}} \psi_{k,m}$. Each W_k is orthogonal to V_k . For any $M \in \mathbb{Z}$,

$$L^2(\mathbb{R}) = V_M \oplus W_M \oplus W_{M+1} \oplus \cdots.$$

The last assertion follows from Theorem 7.4.3.

By Theorem 7.4.4, for any fixed $N \in \mathbb{Z}$ any function $f \in L^2(\mathbb{R})$ can be expanded in this orthonormal basis as

$$f = \sum_{m \in \mathbb{Z}} a_m \phi_{N,m} + \sum_{k \geq N} \sum_{m \in \mathbb{Z}} a_{k,m} \psi_{k,m}, \quad (7.58)$$

where $a_m = (f, \phi_{N,m})$ and $a_{k,m} = (f, \psi_{k,m})$.

As in the Haar case we can let $M \rightarrow -\infty$ in Theorem 7.4.4 to obtain

$$L^2(\mathbb{R}) = \cdots \oplus W_{M-1} \oplus W_M \oplus W_{M+1} \oplus \cdots,$$

which we interpret to mean that the set

$$S = \bigcup_{k,m \in \mathbb{Z}} \psi_{k,m}$$

of all scalings and translations of the mother wavelet forms an orthonormal basis for $L^2(\mathbb{R})$, with no need for scaling functions. See Exercise 7.7.

■ EXAMPLE 7.11

The top left panel in Figure 7.15 shows a function $f(t) \in L^2(\mathbb{R})$ supported on the interval $[0, 2]$. The function is discontinuous at $t = 2$. We show the projection of f onto the spaces V_k for various k for the Daubechies 4-tap filters. Since the support of f and the support of all the scaling functions $\phi(t - n)$ are disjoint except when $n = 0$, the projection of f onto V_0 is just a multiple of $\phi(t)$; we don't show this. The panel at the upper right shows the projection of f onto V_2 ; at the lower left is the projection onto V_4 , and at the lower right is the projection onto V_6 . Note the relative absence of any Gibbs-like phenomena away from the discontinuity at $t = 2$, since the support of the shorter scale wavelets is small and hence most don't intersect the discontinuity.

Final Remarks on Orthogonal Wavelets We should remark that there are many other types of wavelets and multiresolution analyses beside those of the Daubechies family. The scaling functions and wavelets do not need to be orthogonal nor compactly supported. The Daubechies scaling function and wavelets are

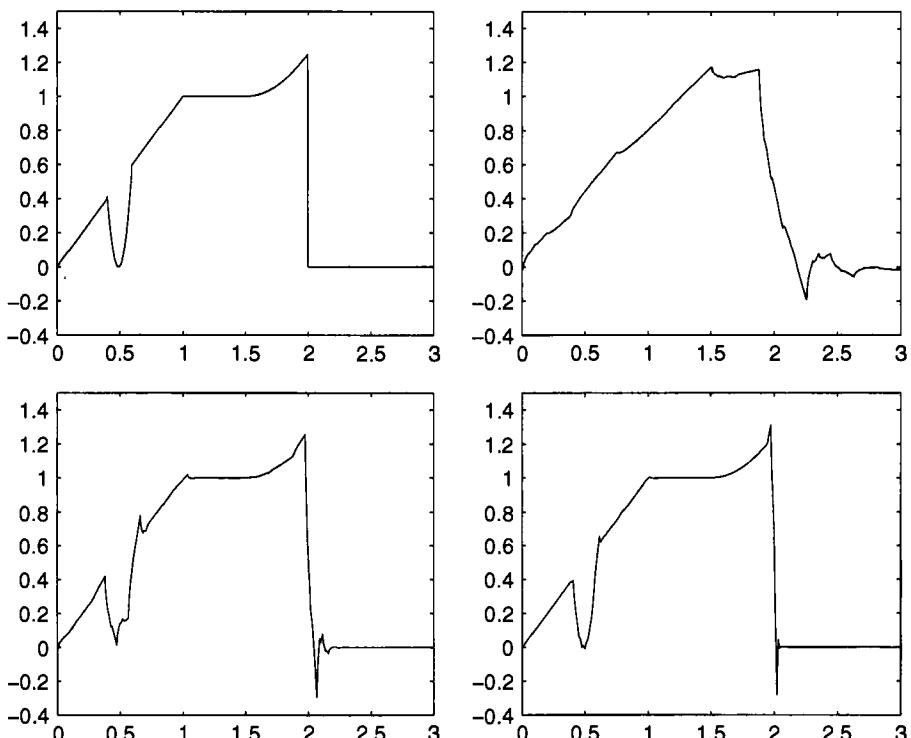


FIGURE 7.15 Function $f(t)$ (upper left) and projection onto V_2 (upper right), V_4 (lower left), and V_6 (lower right).

remarkable, however, in that they do possess compact support, yet their simple power-of-two rescalings and translates provide orthogonal bases for $L^2(\mathbb{R})$.

There's a great deal more to say about the properties possessed by various families of wavelets, such as smoothness (continuity, differentiability), ability to approximate various types of functions, and computational issues. We won't pursue these here, although the Matlab project below involves some numerical experimentation related to wavelet smoothness. See, for example, [24] for more on these issues.

7.5 BIORTHOGONAL WAVELETS

In this section we give a very brief synopsis of the essential truths for biorthogonal wavelets, without proof. The properties of biorthogonal wavelets generally mirror those for orthogonal wavelets. See [24] for more details and proofs.

7.5.1 Biorthogonal Scaling Functions

Consider a biorthogonal filter bank with analysis filters

$$\ell_a = (\ell_0, \dots, \ell_{M-1}), \quad \mathbf{h}_a = (h_0, \dots, h_{N-1}),$$

and synthesis filters ℓ_s, \mathbf{h}_s . We suppose that the filters are related via equations (6.37) and that the low-pass filters have been scaled so that their coefficients sum to one (for the orthogonal filter case we assumed the coefficients summed to $\sqrt{2}$, equation (6.50) or (7.37)). Corresponding to the analysis low-pass filter is a scaling function that satisfies

$$\phi(t) = 2 \sum_k (\ell_a)_k \phi(2t - k). \quad (7.59)$$

The factor of “2” is necessary for any hope of a solution (integrate both sides to see why; in the orthogonal case the factor was $\sqrt{2}$ because the filter coefficients summed to $\sqrt{2}$). Under suitable conditions on the coefficients of ℓ_a a solution to equation (7.59) exists, and indeed, the cascade algorithm converges to the solution. The same argument as in Theorem 7.4.2 shows that the function ϕ is supported on the interval $[0, M - 1]$. However, the translates $\phi(t - n)$ are not orthogonal to each other, although the resulting subspaces $V_k = \text{span}(\{\phi(2^k t - n) : n \in \mathbb{Z}\})$ will satisfy properties 1 and 2 for a multiresolution analysis. In particular, if $P_{V_n}(f)$ denotes the projection of $f \in L^2(\mathbb{R})$ onto V_n , then $P_{V_n}(f) \rightarrow f$ as $n \rightarrow \infty$.

Similar remarks hold true for the synthesis low-pass filter. There is a scaling function $\phi^*(t)$ that satisfies

$$\phi^*(t) = 2 \sum_k (\ell_s)_k \phi^*(2t - k) = 2 \sum_k (-1)^k (\mathbf{h}_a)_k \phi^*(2t - k). \quad (7.60)$$

Again, the integer translates $\phi^*(t - k)$ do not form an orthogonal set, but the subspaces $V_k^* = \text{span}(\{\phi^*(2^k t - n) : n \in \mathbb{Z}\})$ will satisfy properties 1 and 2 for a multiresolution analysis. By Theorem 7.4.2, the function ϕ^* is supported on the interval $[0, N - 1]$.

The scaling functions ϕ and ϕ^* are *biorthogonal* to each other, however, in that there is some integer Q such that after proper rescaling

$$\int_{\mathbb{R}} \phi(t)\phi^*(t - k) dt = \begin{cases} 1, & k = Q, \\ 0, & \text{else.} \end{cases} \quad (7.61)$$

That is, the integer translates of ϕ and ϕ^* are orthogonal to each other, except when ϕ^* is translated Q units farther to the right than ϕ . In fact, for the types of biorthogonal filters of interest here, $Q = (M - N)/2$; note that according to Exercise 6.27 the quantity $(M - N)/2$ is an integer. However, it's much easier to simply redefine ϕ or ϕ^* so that we can assume $Q = 0$; this doesn't change the multiresolution analysis—the subspaces V_k and/or V_k^* remain the same—hence we'll assume this has been done. See Exercise 7.28.

More generally, if $S_1 = \{\alpha_k\}$ and $S_2 = \{\beta_m\}$ are two bases for an inner product space V , then S_1 and S_2 are said to be *biorthogonal* if $(\alpha_k, \beta_m) = 0$ for $k \neq m$. In this case an element $v \in V$ can be expressed as

$$v = \sum_k b_k \alpha_k, \quad (7.62)$$

where $b_k = (v, \beta_k)/(\alpha_k, \beta_k)$,

7.5.2 Biorthogonal Wavelets

The wavelets are defined as in the orthogonal case. Specifically, let

$$\psi(t) = 2 \sum_k (\mathbf{h}_a)_k \phi(2t - k), \quad \psi^*(t) = 2 \sum_k (\mathbf{h}_s)_k \phi^*(2t - k). \quad (7.63)$$

Neither ψ nor ψ^* is orthogonal to its own scalings or translates, but

$$\int_{\mathbb{R}} \psi(2^j t - m) \psi^*(2^k t - n) dt = 0$$

when $j \neq k$ or $m \neq n$. Thus the scalings and translates of the wavelets are also biorthogonal to each other. If we define normalized functions

$$\phi_{j,m}(t) = 2^{j/2} \phi(2^j t - m), \quad \phi_{k,n}^*(t) = 2^{k/2} \phi^*(2^k t - n), \quad (7.64)$$

$$\psi_{j,m}(t) = 2^{j/2} \psi(2^j t - m), \quad \psi_{k,n}^*(t) = 2^{k/2} \psi^*(2^k t - n), \quad (7.65)$$

then from the scaling on ϕ and ϕ^* we have

$$\int_{\mathbb{R}} \psi_{j,m}(t) \psi_{k,n}^*(t) dt = \begin{cases} 1, & \text{if } j = k \text{ and } m = n, \\ 0, & \text{else.} \end{cases}$$

The fact that ϕ is supported on $[0, M - 1]$, ϕ^* on $[0, N - 1]$, and the reasoning of Theorem 7.4.2 shows that ψ and ψ^* are both supported on $[0, (M + N)/2 - 1]$; see Exercise 7.27.

7.5.3 Decomposition of $L^2(\mathbb{R})$

Define subspaces

$$\begin{aligned} V_k &= \text{span}(\{\phi_{k,n} : n \in \mathbb{Z}\}), & V_k^* &= \text{span}(\{\phi_{k,n}^* : n \in \mathbb{Z}\}), \\ W_k &= \text{span}(\{\psi_{k,n} : n \in \mathbb{Z}\}), & W_k^* &= \text{span}(\{\psi_{k,n}^* : n \in \mathbb{Z}\}), \end{aligned}$$

of $L^2(\mathbb{R})$. We again obtain

$$V_{k+1} = V_k \oplus W_k, \quad V_{k+1}^* = V_k^* \oplus W_k^*,$$

and indeed as by Theorem 7.4.4 we have

$$\begin{aligned} L^2(\mathbb{R}) &= V_N \oplus W_N \oplus W_{N+1} \oplus \dots, \\ L^2(\mathbb{R}) &= V_N^* \oplus W_N^* \oplus W_{N+1}^* \oplus \dots, \end{aligned}$$

for each $N \in \mathbb{Z}$. The space W_k is orthogonal to V_k^* and W_k^* is orthogonal to V_k .

Based on the decompositions above we can write an arbitrary $f \in L^2(\mathbb{R})$ as

$$f = \sum_{n \in \mathbb{Z}} a_n^* \phi_{N,n} + \sum_{k \geq N} \sum_{n \in \mathbb{Z}} a_{k,n}^* \psi_{k,n} \tag{7.66}$$

for constants $a_n^*, a_{k,n}^*$. Indeed, taking the inner product of both sides of (7.66) with $\phi_{N,m}^*$ and using orthogonality shows that

$$a_n^* = (f, \phi_{N,m}^*),$$

while taking the inner product of both sides of (7.66) with $\psi_{j,m}^*$ shows that

$$a_{k,n}^* = (f, \psi_{k,n}^*).$$

Equation (7.66) is the biorthogonal analogue of equation (7.58).

■ EXAMPLE 7.12

Consider the biorthogonal Le Gall 5/3 filters, with coefficients

$$\begin{aligned}\ell_a &= \left(\frac{1}{2}, 1, \frac{1}{2}\right), & \mathbf{h}_a &= \left(-\frac{1}{8}, -\frac{1}{4}, \frac{3}{4}, -\frac{1}{4}, -\frac{1}{8}\right), \\ \ell_s &= \left(-\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}\right), & \mathbf{h}_s &= \left(-\frac{1}{2}, 1, -\frac{1}{2}\right).\end{aligned}$$

In Figure 7.16 we show on the left the result of applying the algorithm of equations (7.42) and (7.43) for the low-pass analysis filter (rescaled to coefficients $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ which sum to one). This is the scaling function $\phi(t)$, supported on the interval $[0, 2]$, and is piecewise linear; see Exercise 7.26. The algorithm fails on ℓ_s , however, for a somewhat peculiar reason: the function $\phi^*(t)$ in this case exists as an element of $L^2(\mathbb{R})$, but ϕ^* is singular (unbounded) at each dyadic rational number! (Recall Remark 7.4 on page 300: this algorithm is predicated on the pointwise existence of the scaling function, which fails in this case—the first step, equation (7.42), never gets off the ground, even though the relevant eigenvector exists.) However, we can construct a somewhat faithful portrait of ϕ^* by implementing the cascade iteration symbolically in a computer algebra system (Maple, in this case). It's actually tractable to do 10 or 12 iterations in closed form, and obtain the piecewise defined approximation to ϕ^* shown in the right panel of Figure 7.16. For our particular computation and scaling we find that the translates satisfy the biorthogonality condition $(\phi(t), \phi^*(t - k)) = 0$ for $k \neq -1$ as asserted by equation (7.61), with $Q = (M - N)/2$.

The corresponding wavelets constructed via equation (7.63) are shown in Figure 7.17. We should note that just as the analysis and synthesis filters can be interchanged in a biorthogonal filter bank, so also can the analysis and synthesis scaling functions, or the wavelets.

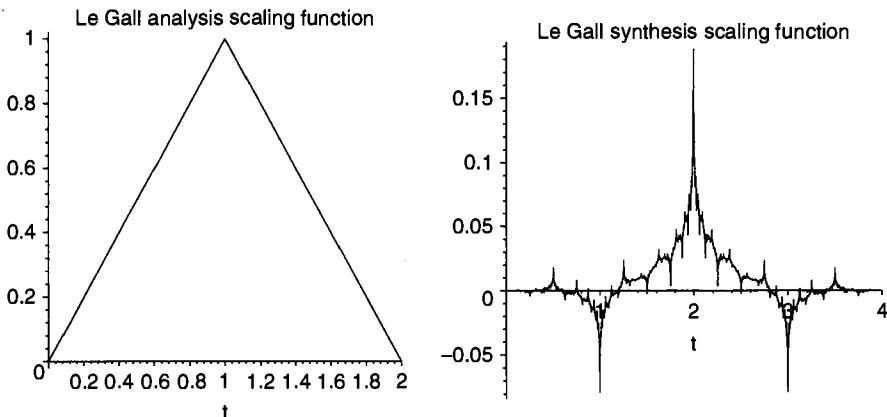


FIGURE 7.16 LeGall 5/3 analysis and synthesis scaling functions.

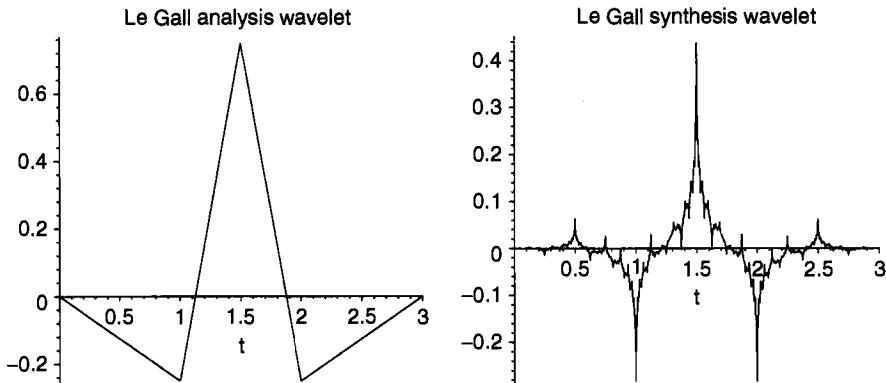


FIGURE 7.17 LeGall 5/3 analysis and synthesis wavelets.

7.6 MATLAB PROJECT

7.6.1 Orthogonal Wavelets

Below are the coefficients for the Daubechies 4-, 6-, and 8-tap orthogonal filters (6 significant figures), scaled so coefficients sum to $\sqrt{2}$.

Index	0	1	2	3
4-tap	0.482963	0.836516	0.224144	-0.129410
6-tap	0.332671	0.806892	0.459878	-0.135011
8-tap	0.230378	0.714847	0.630881	-0.027984

Index	4	5	6	7
4-tap				
6-tap	-0.085441	0.035226		
8-tap	-0.187035	0.030841	0.032883	-0.010597

1. Use the routine `dyadicortho` to compute the corresponding scaling function and wavelet for each filter. For example, for the 4-tap filter execute

```
c = [0.482963 0.836516 0.224144 -0.129410]
[phi,psi,t] = dyadicortho(c,7);
```

which will compute the scaling function and wavelet at dyadic points $t = k/2^7$, $0 \leq k \leq 2^7 \cdot 3$. You can plot the scaling function with `plot(t,phi)` and the wavelet with `plot(t,psi)`.

2. One question of interest (to mathematicians, anyway) is how smooth these functions are, that is, their continuity and differentiability. For example, the 4-tap scaling function and wavelet may look continuous, but have a fairly

“jagged” appearance—are they differentiable? Try increasing the second parameter in dyadicortho to 12 or 16 (or whatever your computer can handle). Then zoom in on the graph of ϕ at some point, for example, by plotting a small portion of ϕ , that is, execute

```
c = [0.482963 0.836516 0.224144 -0.129410]
[phi,psi,t] = dyadicortho(c,16);
plot(t(1000:1200),phi(1000:1200))
```

to zoom into the interval $[1000/2^{16}, 1200/2^{16}]$. Does the function appear differentiable?

3. One way to quantify the smoothness of a function f at a point $t = t_0$ is by considering the quantity

$$Q_f(t_0, t, \alpha) = \frac{|f(t) - f(t_0)|}{|t - t_0|^\alpha} \quad (7.67)$$

for t near t_0 and $\alpha \in [0, 1]$. The quantity Q_f may or may not be bounded for all t near t_0 , depending on f and the value of α . For example, if f is continuous at $t = t_0$, then $Q_f(t_0, t, 0)$ (the case $\alpha = 0$) will be close to zero for $t \approx t_0$, for the simple reason that $Q_f(t_0, t, 0) = |f(t) - f(t_0)|$. However, if f is merely continuous, then for $\alpha > 0$ it may be the case that $Q_f(t_0, t, \alpha)$ is unbounded near $t = t_0$ for any $\alpha > 0$. If f is in fact differentiable near $t = t_0$, then $Q_f(t_0, t, 1)$ will be bounded and indeed will approach $f'(t_0)$ as $t \rightarrow t_0$ (but the converse is not true: Q_f may stay bounded for $\alpha = 1$ but f need not be differentiable). In this case we’ll also find that $Q_f(t_0, t, \alpha)$ is bounded (in fact close to 0) for any $\alpha < 1$. In brief, the smoother f is, the larger we can take α and keep Q_f bounded for all $t \approx t_0$. The largest value of α such that $Q_f(t_0, t, \alpha)$ remains finite for all t in a neighborhood of t_0 is called the *Hölder exponent* of f at t_0 , and it provides a quantitative rating of the smoothness of f at t_0 . It can, of course, vary from one value of t_0 to another.

We can use this to study the smoothness of the scaling functions and wavelets numerically by considering a discrete analog of Q_f . Let ϕ denote the vector from part 2 above that approximates the scaling function ϕ ; we’ll assume you used a second argument 16 so that index $m = 2^{16} + 1 = 65537$ corresponds to $t_0 = 1$. Define $m=65537$ and set $\text{alpha}=0.2$ in Matlab. For any integer p the quantity

```
Qf = abs(phi(m+p)-phi(m))/(abs(p/2^16)) ^ alpha
```

is a discrete approximation to $Q_f(1.0, 1.0 + p/2^{16}, \alpha)$ with $\alpha = 0.2$. If ϕ has Hölder exponent 0.2 or larger at $t_0 = 1$, then Qf should remain bounded or even approach 0 as p approaches 0. Compute Qf for $p = 128, 64, 32, 16, 8, 4, 2, 1$ (you can use negative p too), and observe the behavior of Qf ; it should approach

zero. Then increase α (e.g., to 0.4) and repeat the computation. Try to hone in on a value of α on the razor's edge between growth and decay in Q_f . Compare your value to the published value of approximately 0.55.

Repeat the computation for another value of t_0 , such as $t_0 = \frac{15}{32}$ (corresponding to $m = 30721$). Does the Hölder exponent seem to depend on t_0 ?

4. If the function ϕ in question is actually k times continuously differentiable (but not $k + 1$), then the Hölder continuity is measured by examining

$$Q_f(t_0, t, \alpha) = \frac{|f^{(k)}(t) - f^{(k)}(t_0)|}{|t - t_0|^\alpha} \quad (7.68)$$

for t near t_0 , where $f^{(k)}$ denotes the k th derivative of f .

The Daubechies 6- and 8-tap filters yield scaling functions that are once but not twice differentiable, so $k = 1$ is appropriate in equation (7.68); that is, we should use ϕ' in place of ϕ . Construct a discrete analog of ϕ' and use the approach of part (3) above to estimate the Hölder exponent for the 6 and 8-tap scaling functions at $t_0 = 1$.

See [24] or [11] for more on the smoothness of wavelets.

7.6.2 Biorthogonal Wavelets

1. The coefficients for the Daubechies 9/7 biorthogonal filters are

Index	ℓ_a	\mathbf{h}_a	ℓ_s	\mathbf{h}_s
0	0.026749	0.091271	0.091271	-0.026749
1	-0.016864	-0.057544	0.057544	-0.016864
2	-0.078223	-0.591272	-0.591272	0.078223
3	0.266864	1.115087	-1.115087	0.266864
4	0.602949	-0.591272	-0.591272	-0.602949
5	0.266864	-0.057544	0.057544	0.266864
6	-0.078223	0.091271	0.091271	0.078223
7	-0.016864			-0.016864
8	0.026749			-0.026749

(these are scaled so the first coefficient is at position 0). Note that $(\ell_s)_k = (-1)^k (\mathbf{h}_a)_k$ and $(\mathbf{h}_s)_k = (-1)^{k+1} (\ell_a)_k$. These are the filters used in the JPEG 2000 compression standard, for lossy compression.

Set up vectors c and d in Matlab that contain the low-pass and high-pass analysis filters from the table above (the routine below will generate the synthesis filters). Compute the scaling functions and wavelets with the supplied routine `dyadicbiortho()`, as

```
[phi, phi2, psi, psi2, t, t2, t3] = dyadicbiortho(c, d, 8);
```

Here ϕ and ϕ_2 are the analysis and synthesis scaling functions ϕ and ϕ^* . The t values at which these functions are computed are returned in the arrays t

(for ϕ) and $t2$ (for ϕ^*). The wavelets ψ and ψ^* are contained in psi and psi2 , respectively. The third input argument q controls the resolution, and yields the output functions at points $t = k/2^q$ for an appropriate k range. You can thus plot ϕ , for example, with $\text{plot}(t, \text{phi})$, ϕ^* with $\text{plot}(t2, \text{phi2})$, and the wavelets with $\text{plot}(t3, \text{psi})$ and $\text{plot}(t3, \text{psi2})$.

2. Use Matlab to plot the DFT frequency response of each of the four filters, say by treating them as elements of \mathbb{R}^{256} .
3. Find another set of biorthogonal wavelet filter coefficients (e.g., from Matlab's wavelet toolbox, or find some on the internet, or use the methods of Chapter 7 to make your own). Use `dyadicbiortho()` to construct the scaling functions and wavelets.

EXERCISES

Haar Wavelets

- 7.1** Let $\phi_n(t) = \phi(t - n)$ for $n \in \mathbb{Z}$, where ϕ is the Haar scaling function defined by equation (7.3), and let $\psi_{k,n}$ denote the orthonormal Haar wavelets defined by equation (7.4). Let $f(t)$ be defined

$$f(t) = \begin{cases} t, & 0 \leq t \leq 2, \\ 0, & \text{else.} \end{cases}$$

- a. Compute the $L^2(\mathbb{R})$ inner products

$$c_n = (f, \phi_n), \quad d_{k,n} = (f, \psi_{k,n}),$$

for $k \geq 0$ explicitly in terms of k and n . Hint: For the wavelets only the range $0 \leq n \leq 2^{k+1} - 1$ is needed.

- b. Verify Parseval's identity, that

$$\int_0^2 f^2(t) dt = \sum_{n \in \mathbb{Z}} c_n^2 + \sum_{k \geq 0} \sum_{n \in \mathbb{Z}} d_{k,n}^2.$$

- 7.2** Let f and the $\psi_{k,n}$ be as in Exercise 7.1, but with no restriction on $k \in \mathbb{Z}$ (the set $\psi_{k,n}$ forms an all-wavelet orthonormal basis for $L^2(\mathbb{R})$). Compute the inner products

$$d_{k,n} = (f, \psi_{k,n}),$$

and verify that

$$\int_0^2 f^2(t) dt = \sum_{k \in \mathbb{Z}} \sum_{n \in \mathbb{Z}} d_{k,n}^2.$$

- 7.3 Justify Remark 7.1 on page 276.
- 7.4 a. Explicitly compute the projection of the function f from Exercise 7.1 onto the subspaces V_0 and V_1 of $L^2(\mathbb{R})$ defined by equation (7.26) with the Haar scaling functions/wavelets.
 b. Explicitly compute the projection of the function f from Exercise 7.1 onto the subspaces W_0 defined by equation (7.31) with the Haar scaling functions/wavelets.
 c. Verify directly that $P_{V_1}(f) = P_{V_0}(f) + P_{W_0}(f)$, where $P_V(f)$ denotes the projection of f onto the subspace V .
- 7.5 Show that the Haar wavelets $\psi_{0,n}$, $n \in \mathbb{Z}$ by themselves don't span the subspace V_1 defined by equation (7.26).
- 7.6 Show that the function $x(t)$ defined by equation (7.17) is in $L^2(\mathbb{R})$ if $\mathbf{x} \in L^2(\mathbb{Z})$.
- 7.7 Let f be a continuous function in $L^2(\mathbb{R})$. In this exercise we show that f can be approximated arbitrary well using only the wavelets $\psi_{k,n}$, with no need for the scaling function. As a consequence the wavelets $\psi_{k,n}$ form an orthonormal basis for $L^2(\mathbb{R})$.
 We'll begin with the Haar basis. Recall that for any fixed N we can expand f with respect to the orthonormal Haar basis consisting of $\phi_{N,m}$ and $\psi_{k,m}$ for $m \in \mathbb{Z}$, $k \geq N$ as

$$f = \sum_{m=-\infty}^{\infty} d_{N,m} \phi_{N,m} + \sum_{k \geq N}^{\infty} \sum_{m=-\infty}^{\infty} c_{k,m} \psi_{k,m},$$

where $d_{N,m} = (f, \phi_{N,m})$ and $c_{k,m} = (f, \psi_{k,m})$. Let \tilde{f}_N denote the expansion

$$\tilde{f}_N = \sum_{k \geq N}^{\infty} \sum_{m=-\infty}^{\infty} c_{k,m} \psi_{k,m},$$

which omits the $\phi_{N,m}$.

- a. Suppose f is supported in a bounded interval $[-a, a]$ for some constant a . Show that

$$\lim_{N \rightarrow -\infty} \|f - \tilde{f}_N\|_{L^2(\mathbb{R})} = 0.$$

Hint:

$$|d_{N,m}| = \left| \int_{-\infty}^{\infty} f(t) \phi_{N,m}(t) dt \right| \leq \sup_{[-a,a]} |\phi_{N,m}| \int_{-a}^a |f(t)| dt.$$

Use this to show $\sum_m d_{N,m}^2$ goes to zero as $N \rightarrow -\infty$.

- b. Use part (a) and the reasoning in the proof of Theorem 7.2.2 to extend to the case in which f is not compactly supported.
- c. Extend parts (a) and (b) to a general multiresolution analysis, under the assumption that the scaling function is continuous and compactly supported.

Projection

- 7.8** Let V be the subspace in \mathbb{R}^3 spanned by the orthonormal vectors $\mathbf{v}_1 = (1/\sqrt{2}, 0, 1/\sqrt{2})$ and $\mathbf{v}_2 = (0, 1, 0)$. Compute the projection of $\mathbf{w} = (1, 3, 2)$ onto V using Definition 7.2.2. Verify that this vector is indeed the closest element of V to $(1, 3, 2)$ by explicitly minimizing

$$\|\mathbf{w} - c_1\mathbf{v}_1 - c_2\mathbf{v}_2\|^2$$

as a function of c_1 and c_2 .

- 7.9** Suppose that W and \tilde{W} are subspaces of a vector space V , and that $W \cap \tilde{W} = \{\mathbf{0}\}$ (the zero vector in V).
- a. Show that if $\mathbf{w} \in W$ and $\tilde{\mathbf{w}} \in \tilde{W}$ satisfy $\mathbf{w} + \tilde{\mathbf{w}} = \mathbf{0}$, then $\mathbf{w} = \tilde{\mathbf{w}} = \mathbf{0}$.
 - b. Suppose that for some $\mathbf{v} \in V$ we have $\mathbf{v} = \mathbf{w} + \tilde{\mathbf{w}}$ for $\mathbf{w} \in W$ and $\tilde{\mathbf{w}} \in \tilde{W}$. Show that \mathbf{w} and $\tilde{\mathbf{w}}$ are unique. Hint: Use part (a).
- 7.10** Let V denote the subset of $L^2(\mathbb{R})$ consisting of those functions that are continuous.
- a. Show that V is a subspace of $L^2(\mathbb{R})$.
 - b. Let ϕ be the Haar scaling function defined in equation (7.3) (though we could use any discontinuous function for what follows). Show that there is no element of V that is “closest” to ϕ , in that

$$\|\phi - f\|_{L^2(\mathbb{R})}$$

with $f \in V$ can be made arbitrarily small but not zero. This shows that projection onto this subspace is not well-defined, at least by any definition we've given.

- 7.11** Finish the proof of Proposition 7.2.3 by showing that if $\alpha \in V$ and $\alpha \neq P_V(f)$, then $\|f - P_V(f)\| < \|\alpha - f\|$. Hint: Let \tilde{f}_n be as defined in equation (7.16). Let $\alpha \in V$ have expansion

$$\alpha = \sum_{k=-\infty}^{\infty} b_k \eta_k,$$

and let α_n be as defined by equation (7.14). If $\alpha \neq P_V(f)$, then $b_M \neq (f, \eta_M)$ for some M . Use equation (7.15) to show that $\|f - \alpha_n\|^2 - \|f - P_V(f)\|^2 \geq (b_M - (f, \eta_M))^2$ for $n \geq M$. Conclude that $\|f - P_V(f)\| < \|f - \alpha\|$.

- 7.12** Suppose that V is a subspace of an inner product space and V has orthonormal basis $\cup_{k=1}^{\infty} \eta_k$. Let $P_V(f)$ denote the projection of f onto V .

- Show that $(f - P_V(f), \phi) = 0$ for any $\phi \in V$. Hint: Just show that $(f - P_V(f), \eta_m) = 0$; a careful proof might invoke Exercise 1.37.
- Show that

$$\|f\|^2 = \|P_V(f)\|^2 + \|f - P_V(f)\|^2.$$

Orthogonal Wavelets

- 7.13** Show that the wavelet defined by equation (7.48) is orthogonal to $\phi(t - n)$ for any n . Hint: Mimic the computation of equation (7.50), and make use of the dilation equation (7.34). It should come down to showing that

$$\sum_{j=0}^{N-1} (-1)^j c_j c_{n'-j-1} = 0$$

when n' is any even integer (and take $c_k = 0$ for any k outside the range 0 to $N - 1$, of course). Show that this is always true when N is even, for any numbers c_0, \dots, c_{N-1} .

- Show that the Haar scaling function defined by equation (7.3) satisfies the dilation equation (7.34) (with $c_0 = c_1 = 1/\sqrt{2}$).
 - Show that the mother Haar wavelet defined by equation (7.1) satisfies equation (7.48).
- 7.15** Show that if $L_a(z) = \sum_k c_k z^k$ where the c_k are real, then $|L_a(e^{it})| = |L_a(e^{-it})|$ for $t \in [0, 2\pi]$.
- 7.16** Use equation (7.36) to show that N , the number of nonzero coefficients in the low-pass filter vector (c_0, \dots, c_{N-1}) , must be even. Hint: Consider a simple special case like $N = 3$ first; note that c_0 and c_{N-1} are nonzero by assumption.
- Use fixed point iteration to find a solution to the equation $x = \cos(x)$. Does the initial guess matter?
 - Use fixed point iteration to try to find a solution to the equation $x = -10e^x$ (this equation has a unique solution; plot $x + 10e^x$ for $-5 \leq x \leq 0$ to see this). What does the fixed point iteration do? Can you interpret the behavior graphically?
- 7.18** Show that the mapping F defined in equation (7.44) really does take $L^2(\mathbb{R})$ to $L^2(\mathbb{R})$.
- 7.19** Show that the approximation algorithm of Section 7.4.4 works for the Haar scaling function if we define $\phi(0) = \phi(1) = \frac{1}{2}$ (and filter coefficients $c_0 = c_1 = 1/\sqrt{2}$, and that ϕ is supported in $[0,1]$).

- 7.20** **a.** Show that if functions f_1, \dots, f_N are all defined on \mathbb{R} and f_k is supported in a set $S_k \subset \mathbb{R}$, then the sum $f_1 + \dots + f_N$ is supported in the union $\cup_k S_k$. Show that the product $f_1 \cdots f_N$ is supported in the intersection $\cap_k S_k$.
- b.** Show that if each f_k in part (a) has compact support, then the sum $f_1 + \dots + f_N$ and product $f_1 \cdots f_N$ also have compact support.
- 7.21** Show that the function $\phi(t)$ produced by the cascade algorithm with coefficients c_0, \dots, c_{N-1} satisfies

$$\int_0^{N-1} \phi(t) dt = 1.$$

Hint: The Haar scaling function ϕ^0 (the initial guess) has integral one, and make use of $L_a(1) = \sqrt{2}$.

- 7.22** Show that $\phi^j(t)$ in Example 7.9 is of the form asserted. Also verify that for any fixed j we have $(\phi^j(t), \phi^j(t-n)) = 0$ for any integers n (i.e., ϕ^j is orthogonal to its own integer translates).
- 7.23** Even though the cascade algorithm fails to converge in Example 7.9, show nonetheless that the function

$$\phi(t) = \begin{cases} 1/3, & 0 \leq t < 3, \\ 0, & \text{else,} \end{cases}$$

satisfies the dilation equation with the relevant filter coefficients and normalization $\int \phi = 1$. Does this function satisfy $(\phi(t), \phi(t-n)) = 0$ for all integers n ?

- 7.24** Show that the wavelet ψ defined by equation (7.48) satisfies $\|\psi\| = 1$ (L^2 norm), assuming that the scaling function is normalized so that $\|\phi\| = 1$.

Biorthonormal Wavelets

- 7.25** Let S_1 and S_2 be the bases

$$S_1 = \{(1, 2, 1), (0, 1, 1), (1, 1, 1)\},$$

$$S_2 = \{(0, 1, -1), (-1, 0, 1), (1, -1, 1)\},$$

for \mathbb{R}^3 . Show that S_1 and S_2 are biorthogonal with respect to the usual inner product. Use equation (7.62) to decompose the vector $(3, 1, 2)$ into a sum of the basis elements of S_1 .

- 7.26** Show that the Le Gall analysis scaling function shown on the left in Figure 7.16 satisfies the dilation equation (7.59) with low-pass coefficients $\ell_a = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$.

- 7.27** Show that if biorthogonal scaling functions ϕ and ϕ^* are supported in intervals $[0, M - 1]$ and $[0, N - 1]$, respectively, then the wavelets ψ and ψ^* defined in equation (7.63) are both supported in the interval $[0, (M + N)/2 - 1]$. *Hint:* Reason as in Theorem 7.4.2; Exercise 7.20 may be helpful.
- 7.28** The goal of this problem is to show that if ϕ and ϕ^* satisfy the relation (7.61), then $Q = (M - N)/2$. Suppose that ϕ and ϕ^* are the unique solution to equations (7.59) and (7.60) with filter coefficients $((\ell_a)_0, \dots, (\ell_a)_{M-1})$ and $((\ell_s)_0, \dots, (\ell_s)_{M-1})$, respectively. We suppose that the filters are both symmetric; that is, $(\ell_a)_k = (\ell_a)_{M-k-1}$ and $(\ell_s)_k = (\ell_s)_{N-k-1}$. Recall from Exercise 6.27 that the integers M and N are both even or both odd.
- Define a function $\tilde{\phi}(x) = \phi(M - 1 - x)$. Show that $\tilde{\phi}$ also satisfies equation (7.59) and so $\phi = \tilde{\phi}$. A similar conclusion holds for ϕ^* .
 - Use part (a) to show that if ϕ and ϕ^* satisfy equation (7.61), then $Q = (M - N)/2$.

REFERENCES

1. M. Barnsley. *Fractals Everywhere*, 2nd ed. Morgan Kaufmann, San Diego, 1993.
2. M. Barnsley. Fractal image compression, *Notices of the AMS*, **43** (1996): 657–662.
3. A. Boggess and F. Narcowich. *A First Course in Wavelets with Fourier Analysis*. Prentice Hall, Englewood Cliffs, NJ, 2001.
4. E. O. Brigham. *The Fast Fourier Transform and Its Applications*. Prentice Hall, Englewood Cliffs, NJ, 1988.
5. C. Brislawn, Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Appl. Comput. Harmonic Anal.*, **3** (1996): 337–357.
6. J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* **19** (1965): 297–301.
7. C. Brislawn. Fingerprints go digital. *Notices of the AMS*, **42** (1995): 1278–1283.
8. W. Biggs and V. E. Henson. *The DFT: An Owner's Manual for the Discrete Fourier Transform*. SIAM, Philadelphia, 1995.
9. S. Burris, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Englewood Cliffs, NJ, 1998.
10. R. Courant and D. Hilbert. *Methods of Mathematical Physics*, Vol. 1. Wiley, New York, 1989.
11. I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series In Applied Mathematics, Vol. 61. SIAM, Philadelphia, 1992.
12. Y. Fisher, ed. *Fractal Image Compression: Theory and Application*. Springer-Verlag, New York, 1995.
13. D. Gabor. Theory of communication. *J. Inst. Electr. Engr.*, London, **93** (1946): 429–457.

14. R. Gonzales, R. Woods, and S. Eddins. *Digital Image Processing Using MATLAB*. Prentice Hall, Englewood Cliffs, NJ, 2004.
15. Joint Photographic Experts Group. JPEG technical specification: Revision (DRAFT). ISO/IEC JTC1/WG8, CCIT SGVIII, August, 1990.
16. E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, New York, 1978.
17. J. McClellan, R. Schafer, and M. Yoder. *DSP First, A Multimedia Approach*. Prentice Hall, Englewood Cliffs, NJ, 1998.
18. M. Nelson and J-L. Gailly. *The Data Compression Book*. M&T Books. New York, 1996.
19. W. Pratt. *Digital Image Processing*, 2nd ed., Wiley Interscience, New York, 1991.
20. W. Rudin. *Principles of Mathematical Analysis*, 3rd ed. McGraw-Hill, New York, 1976.
21. K. Saxe. *Beginning Function Analysis*. Springer-Verlag, New York, 2002.
22. K. Sayood and M. Kauffmann. *Introduction to Data Compression*. San Francisco, 1996.
23. A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Trans. Cons. Electr.*, **46** (2000): 1103–1127.
24. G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, 1996.
25. G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, 1993.
26. M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.
27. Wavelet Digest, available at <http://www.wavelet.org/wavelet/index.html>.
28. Text resources Web page <http://www.rose-hulman.edu/mathDFT/>.

SOLUTIONS

Chapter 1

- 1.1 Color JPEG compressed images are typically 5 to 50 times smaller than they would be if stored “naively,” so the ratio of naively stored to JPEG-stored might range from a low of 0.02 to 0.2.
- 1.3 Work out $e^{2\pi ik/8} = \cos(k\pi/4) + i \sin(k\pi/4)$.
- 1.4 Use equation (1.23).
- 1.5 Apply Euler’s identity to $\cos(\omega t)$ and $\sin(\omega t)$, group the $e^{i\omega t}$ and $e^{-i\omega t}$ terms.
- 1.8 Check closure under addition.
- 1.9 Consider whether $c\mathbf{x}$ is in \mathbb{R}^n for a complex constant c .
- 1.11 Use $(p+q)^2 \leq 2p^2 + 2q^2$ to show that

$$\sum_{k=0}^{\infty} (x_k + y_k)^2 \leq 2 \left(\sum_{k=0}^{\infty} x_k^2 + \sum_{k=0}^{\infty} y_k^2 \right)$$

and also that $\mathbf{x} + \mathbf{y}$ is in $L^2(\mathbb{N})$ if \mathbf{x} and \mathbf{y} are in $L^2(\mathbb{N})$, and thus obtain closure under addition.

To show that $L^2(\mathbb{N}) \subset L^\infty(\mathbb{N})$ argue that $\sum_k x_k^2 < \infty$ implies the existence of some bound M so that $|x_k| \leq M$ for all k .

- 1.13** Use equation (1.22).
- 1.14** Use $|e^{i\omega t}| = 1$ if ω and t are real, and $e^a e^b = e^{a+b}$.
- 1.15** Use Euler's identity on each piece, e.g., $\cos(\alpha x) = (e^{i\alpha x} + e^{-i\alpha x})/2$.
- 1.18** Recall that $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\theta)$, and that vectors are orthogonal when $\mathbf{u} \cdot \mathbf{v} = 0$. The peak-to-peak distance in part (e) should be $1/\sqrt{p^2 + q^2}$.
- 1.19** Use equation (1.22). Examine $\mathbf{E}_{6,1}$ versus $\mathbf{E}_{6,5}$ and $\mathbf{E}_{6,2}$ versus $\mathbf{E}_{6,4}$.
- 1.22** Use Theorem 1.8.3 for parts (b) and (d).
- 1.23** Use Theorem 1.8.3 for parts (a) and (c), and don't forget the conjugation in the inner product.
- 1.28** For part (a) expand $\|\mathbf{u} + \mathbf{v}\|^2 + \|\mathbf{u} - \mathbf{v}\|^2$ using the inner product.
- 1.40** For part (a) the integrand for the sine coefficients is odd, while for (b) the integrand for the cosine coefficients is odd.

Chapter 2

- 2.1** Use equation (2.9).
- 2.2** The DFT is $\mathbf{F}_4 \mathbf{x} = (2, 1 - 3i, 0, 1 + 3i)^T$.
- 2.3** Use equation (2.9) and $\mathbf{F}_N^{-1} = (\mathbf{F}_N^*)/N$.
- 2.4** Use equation (2.9) and recall that the product of a matrix \mathbf{M} and the k th standard basis vector is the k th column of \mathbf{M} .
- 2.6** From the definition of the two-dimensional DFT we find DFT component $\hat{b}_{p,q}$ given by

$$\hat{b}_{p,q} = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} e^{-2\pi i(pr/m+qs/n)} \mathbf{e}_{j,k}(r, s).$$

Simplify and compute the magnitude.

- 2.11** For part (b) note that the complex argument of X_m is piecewise linear in m , except for occasional jumps where the argument exceeds $\pm\pi$.
- 2.13** Use Proposition 2.7.1.
- 2.14** Part (a) is just the definition of matrix multiplication. For part (b) start with

$$\mathbf{Z}_{k,l} = \sum_{r=0}^{m-1} \sum_{s=0}^{n-1} z_{r,s} e^{-2\pi i(kr/m+ls/n)}.$$

- 2.18** The shifted image should of course be shifted, but the DFT magnitudes should be the same.
- 2.19** Compare $\hat{a}_{k,l}$ and $\hat{a}_{m-k,n-l}$, as well as $\hat{a}_{m-k,l}$ and $\hat{a}_{k,n-l}$.

Chapter 3

- 3.1** Use equation (3.13). The DCT is $(2\sqrt{3}/3, \sqrt{2}, -2\sqrt{6}/3) \approx (1.15, 1.41, -1.63)$.
- 3.2** Use $\mathcal{C}_N^{-1} = \mathcal{C}_N^T$ and equation (3.13). The inverse DCT of $\mathbf{X} = (3, 0, 1)$ is $(\sqrt{3} + \sqrt{6}/6, \sqrt{3} - \sqrt{6}/3, \sqrt{3} + \sqrt{6}/6) \approx (2.14, 0.92, 2.14)$.
- 3.3** It should be the m th column of \mathcal{C}_N .
- 3.7** For the DCT with $d = 0.01, 0.1, 1.0, 10.0, 100.0$ we obtain percentage errors $6.4 \times 10^{-5}, 0.0053, 0.792, 30.7, 100$, respectively. In the last case the entire transform is quantized to zero.
- 3.8** Use equation (3.5.1).
- 3.9** Use equations (3.5.1) and (3.13), as well as $\mathcal{C}_N^{-1} = \mathcal{C}_N^T$.
- 3.10** Use $\mathcal{C}_N \mathcal{C}_N^T = \mathbf{I}$ and note that for any real column vector $(\mathbf{v}, \mathbf{v}) = \mathbf{v}^T \mathbf{v}$.
- 3.11** Use equation (3.10).

Chapter 4

- 4.1** Start with $(|w| + |z|)^2 \geq 0$.
- 4.2** The convolution $\mathbf{g} * \mathbf{x} = (16, 20, 7, 2)$.
- 4.3** In part (b) note that $W_m = 1$ for all m .
- 4.7** Just compare the “0th” columns of the circulant matrices!
- 4.9** Use part (iii) of Theorem 4.2.1.
- 4.11** As M gets smaller the convolution $\mathbf{f} * \mathbf{v}$ looks like a progressively less and less blurry version of \mathbf{f} .
- 4.15** The DFT $\mathbf{H} \in \mathbb{C}^N$ should have components H_k with $H_0 = 0$ and $|H_{N/2}| = 1$.
- 4.16** You should find in part (a) that $\mathbf{x} = (1, -1, 3, 5)$.
- 4.18** $A_m = (1 + e^{-2\pi im/N})/2$.
- 4.26** a. A simple ratio test shows that $\sum_{k=0}^{\infty} (e^{-k})^2 < \infty$.
 b. Use $1 + z + z^2 + \dots = 1/(1 - z)$ for $|z| < 1$ and note that $e^{-k} e^{-2\pi i kf} = (e^{-(1+2\pi if)})^k$.
- 4.32** The circular convolution in \mathbb{R}^4 is $(1, 9, -4, 9)$.

4.34 You should obtain $|H(f)| = |\sin(\pi f)|$.

4.35 Use the z -transform. We need

$$X^2(z) = 1 - 4z^{-2} + 4z^{-4}.$$

Factor.

Chapter 5

5.1 a. The DFT of \mathbf{x} is

$$\mathbf{X} = \begin{bmatrix} x_0 + x_1 + x_2 + x_3 \\ x_0 - ix_1 - x_2 + ix_3 \\ x_0 - x_1 + x_2 - x_3 \\ x_0 + ix_1 - x_2 - ix_3 \end{bmatrix}.$$

b. You should find

$$\mathbf{Y} = \begin{bmatrix} x_1 + x_2 \\ -ix_1 - x_2 \\ -x_1 + x_2 \\ ix_1 - x_2 \end{bmatrix}.$$

c. The 2-point DFT of (x_1, x_2) yields components $\tilde{X}_0 = x_1 + x_2$, $\tilde{X}_1 = x_1 - x_2$, in agreement with Theorem 5.2.1.

5.3 Mimic the proof of the Convolution Theorem 4.3.1.

5.6 In part (b) you should find $a = -t_0^2 \omega(t_0)$ and $\omega_0 = \omega(t_0) + \omega'(t_0)t_0$.

5.8 Start with

$$z_k = \int_{-1/2}^{1/2} Z(f) e^{2\pi i k f} df.$$

Then reason as in the proof of the Convolution Theorem 4.2.1, replacing sums with integrals.

Chapter 6

6.1 You should find $\mathbf{X}_\ell = (\frac{1}{2}, 0, 2)$ and $\mathbf{X}_h = (\frac{1}{2}, 2, -2)$ (in each case the element $\frac{1}{2}$ is the index zero component) and all other components zero.

6.3 You should find that both $\mathbf{x} * \ell_a$ and \mathbf{X}_ℓ have all components identical. The vector $\mathbf{x} * \mathbf{h}_a$ should have alternating components, while \mathbf{X}_h should have all identical components.

6.6 For part (c) you should find $\ell_s = \ell_a$ (both the identity filter); \mathbf{h}_s has component $(\mathbf{h}_s)_{-1} = 1$, and all other components zero.

6.8 Then synthesis filter is $\ell_s = (\frac{4}{5}, \frac{4}{5})$, all other coefficients zero.

6.14 You should find

$$\alpha_2(\mathbf{x}) = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}, \quad \delta_2(\mathbf{x}) = \begin{bmatrix} -1/2 \\ 1/2 \\ 1/2 \\ -1/2 \end{bmatrix}, \quad \delta_1(\mathbf{x}) = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}.$$

6.17 The output delay should be delay 3 (i.e., equation (6.32) should have $2z^{-3}$ on the right).

6.20 Start with

$$X(z) = \sum_{k=M_1}^{N_1} x_k z^{-k}, \quad Y(z) = \sum_{k=M_2}^{N_2} y_k z^{-k},$$

multiply, and use the Convolution Theorem.

6.22 We obtain causal filters with coefficients with coefficients $(\ell_s)_0 = \frac{1}{2}$, $(\ell_s)_1 = -1$, $(\ell_s)_2 = \frac{1}{2}$ and $(\mathbf{h}_s)_0 = -1$, $(\mathbf{h}_s)_1 = 2$, $(\mathbf{h}_s)_2 = 1$. The filter bank has delay 3.

Chapter 7

7.1 a. You should obtain $c_0 = \frac{1}{2}$, $c_1 = \frac{3}{2}$, and $c_n = 0$ for $n \neq 0, 1$. Also $d_{k,n} = -2^{-3k/2}/4$ for $0 \leq n \leq 2^{k+1} - 1$ (since the support of f and $\psi_{k,n}$ do not overlap).

b. Since $\int_0^2 f^2(t) dt = \frac{8}{3}$ the coefficients should sum to $\frac{8}{3}$ (an easy computation with geometric series).

7.4 The projection $P_{V_0}(f)(t)$ should be constant on both of $[0, 1)$ and $[1, 2)$, zero outside these intervals. Both of projection $P_{V_1}(f)(t)$ and $P_{W_0}(f)(t)$ should be constant on half-integer interval, zero outside of $[0, 2]$.

7.5 Try building $\phi(t)$ as a superposition of elements of V_1 .

7.6 Use

$$\int_{k/2}^{(k+1)/2} x^2(t) dt = \frac{1}{2} x_{k-1}^2.$$

- 7.8** We find projection $P_V(\mathbf{w}) = (3/2, 3, 3/2)$. This is the same result obtain by minimizing

$$\|\mathbf{w} - c_1 \mathbf{v}_1 - c_2 \mathbf{v}_2\|^2 = 14 - 3\sqrt{2}c_1 - 6c_2 + c_1^2 + c_2^2.$$

- 7.10** Just approximate the discontinuous scaling function with a piecewise linear function that rises linearly from the point $(-\epsilon, 0)$ to $(0, 1)$ and decreases linearly from $(1, 1)$ to $(1 + \epsilon, 0)$, and elsewhere agree with the scaling function.

- 7.15** Use $|L_a(z)|^2 = L_a(z)\overline{L_a(z)}$ and the fact that since the c_k are real $\overline{L_a(z)} = L_a(\bar{z})$.

- 7.17** a. The iterates should converge to about 0.739.
b. In this case the iterates oscillate between two values.

- 7.18** Consider

$$\int_{-\infty}^{\infty} \phi^2(2t - n) dt$$

under the change of variable $u = 2t - n$.

- 7.20** For part (a), note that if the sum $f_1(t_0) + \cdots + f_N(t_0) \neq 0$ for some t_0 , then at least one of the f_k is nonzero at t_0 . In part (b), if the product is nonzero, then all f_k are nonzero at t_0 .
- 7.23** We find that although ϕ satisfies the dilation equation we have $(\phi(t), \phi(t-1)) = \frac{2}{3}$ and $(\phi(t), \phi(t-2)) = \frac{1}{3}$. This function does not have the orthogonality properties of a scaling function.
- 7.24** Compute (ψ, ψ) using $\psi = \sqrt{2} \sum d_k \phi(2t - k)$, taking note of equation (7.35) and $d_k = (-1)^k c_{N-k-1}$.

INDEX

- ℓ^∞ , 12
- ℓ^2 , 13
- \mathbb{C}^N , 11
- \mathbb{R}^N , 11
- $C(a, b)$, 14
- $L^\infty(\mathbb{N})$, 12
- $L^\infty(\mathbb{Z})$, 13
- $L^2(\mathbb{N})$, 13, 157
- $L^2(\mathbb{Z})$, 13, 158
- $L^2(\mathbb{Z} \times \mathbb{Z})$, 170
- $L^2(a, b)$, 15
- $M_{m,n}(\mathbb{C})$, 11
- $M_{m,n}(\mathbb{R})$, 11
- Aliasing, 5, 22, 160
 - conjugate, 27
 - images, 25
- Analog-to-digital converter, 4
- Approximation, 228, 232
 - coefficients, 205
 - two-dimensions, 236
 - with Haar functions, 277
- Basic waveforms
 - discrete cosine transform, 118
 - discrete Fourier transform, 79
 - one-dimensional analog, 17
 - one-dimensional discrete, 26
 - two-dimensional analog, 20
 - two-dimensional discrete, 27
- Basis, 37
 - biorthogonal, 315
 - Hamel, 37
 - orthogonal, 39, 49
 - orthonormal, 39, 49
- Blurring, 152
- Cascade algorithm, 300
- Cauchy-Schwarz inequality, 35
- Circulant matrix, 143
- Codebook, 43
- Codewords, 43
- Color, 9
- Compact support, 273
- Complete set, 48

- Compression, 2
- JPEG, 124
- JPEG 2000, 237
- Convolution Theorem
 - $L^2(\mathbb{Z})$, 163
 - one-dimensional \mathbb{C}^N , 145
 - two-dimensional $M_{m,n}(\mathbb{C})$, 151
- Convolution
 - $L^2(\mathbb{N})$, 162
 - $L^2(\mathbb{Z})$, 161
 - circular \mathbb{C}^N , 142
- dc component, 79
- Denoise, 77, 139, 152
- Dequantization, 43
- Dequantization map, 43
- Detail, 228, 232
 - coefficients, 205
 - diagonal, 232
 - horizontal, 232
 - two-dimensions, 236
 - vertical, 232
- Digitize, 4, 6
- Dilation equation, 293
- Dimension, 38
- Discrete cosine transform
 - matrix form, 116
 - one-dimensional, 116
 - properties, 117
 - two-dimensional, 121
- Discrete Fourier transform
 - matrix form, 85
 - one-dimensional, 78
 - properties, 85
 - two-dimensional, 93
- Discrete time Fourier transform, 159
- Discrete wavelet transform, 221
 - multistage, 225
 - two-dimensional, 235
 - two-dimensional, 231
- Distortion, quantifying, 46
- Downsampling, 58, 205, 224, 239
- Edge detection, 3, 154
- Energy, 74
- Euler's identity, 17
- Extension
 - half-point, 113, 218
 - periodic, 218
- whole-point, 219
- zero padding, 218
- Fast Fourier transform, 90
- Filter bank
 - M -channel, 211
 - analysis, 205, 211
 - biorthogonal, 212
 - multistage, 214
 - orthogonal, 213, 225, 247
 - perfect reconstruction, 212
 - synthesis, 207, 211
 - two-channel, 211
- Filter design, 148
- Filter
 - acausal, 170
 - adjoint, 175
 - analysis, 205
 - antisymmetric, 262
 - causal, 170
 - Daubechies, 250
 - finite impulse response, 170
 - Haar, 202
 - halfband, 249
 - high-pass, 149, 243
 - Le Gall 5/3, 213
 - low-pass, 141, 243
 - maxflat, 250
 - product, 244, 248
 - quadrature mirror, 248, 262
 - symmetric, 170, 175, 262
 - synthesis, 206
 - taps, 149
- Filtering, 146
- Fixed point, 300
- Fixed point iteration, 300
- Fourier coefficients, 79
- Fourier series, 51
 - sine/cosine, 52
- Frequency, 19
 - natural, 19
- Frequency domain, 72
- Frequency response, 148
- Frobenius norm, 31
- Grayscale, 6
- Haar function, 270
- Hilbert space, 55

- Images
 - grayscale, 6
 - RGB, 9
- Inner product, 29
- Inpainting, 2
- Inverse discrete cosine transform
 - matrix form, 117
 - one-dimensional, 116
 - two-dimensional, 122
- Inverse discrete Fourier transform
 - matrix form, 87
 - one-dimensional, 79
 - two-dimensional, 94
- Inverse discrete time Fourier transform, 160
- Laurent polynomial, 165, 167
- Linear independence, 37
- Localization, 184
- Mask, 151
- Metric, 30
- Moving average, 139
- Multiresolution, 288, 292
 - Haar, 287
- Multiresolution analysis, 292, 305
- Natural frequency, 19
- Noise, 6, 83
 - white, 83
- Norm, 30
 - Euclidean, 31
 - Frobenius, 31
 - supremum, 33
- Normed linear space, 30
- Normed vector space, 30
- Nyquist frequency, 160
- Nyquist sampling rate, 24
- Orthogonal projection, 280
- Orthogonality, 34
- Parseval's identity, 41, 50
- Perfect reconstruction, 212
- Period, 19
- Quantization, 5, 9, 41
 - map, 42
 - interval, 42
 - matrix, 125
- Real signals, 88
- Restoration, 2
- RGB images, 9
- Root of unity, 168
- Sampling
 - images, 8
 - interval, 5
 - rate, 5
 - signals, 4
- Scalar product, 29
- Scaling function, 292
 - biorthogonal, 315
 - Haar, 271
- Shift operator, 207
- Short-time Fourier transform, 190
- Side lobes, 187
- Signal, 3
 - analog, 3
- Span, 36
- Spectral factorization, 250
- Spectrogram, 190
- Spectrum, 76
- Subband coding, 211
- Subspace, 10
- Support, 273
- Taps, 149
- Thresholding, 107
- Time domain, 72
- Upsampling, 206, 225, 239
- Vector space, 9
- Wavelength, 19
- Wavelet, 306
 - biorthogonal, 315
 - Haar, 270
 - Le Gall, 317
 - mother, 306, 310
- Window, 184
 - Bartlett, 194
 - Gaussian, 195
 - Hamming, 195
 - rectangular, 184
 - triangular, 194
- Windowed Fourier transform, 190
- Zero padding, 218
- z -transform, 164

This page intentionally left blank