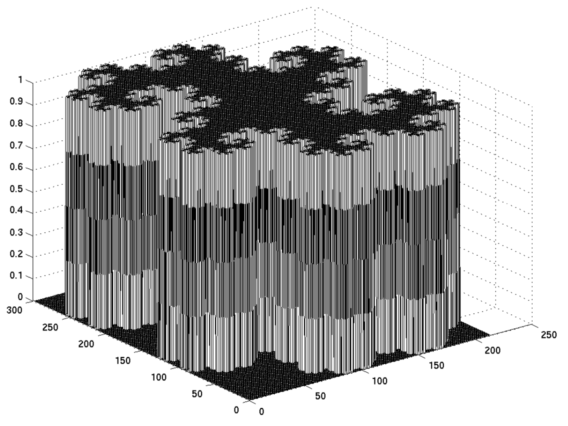
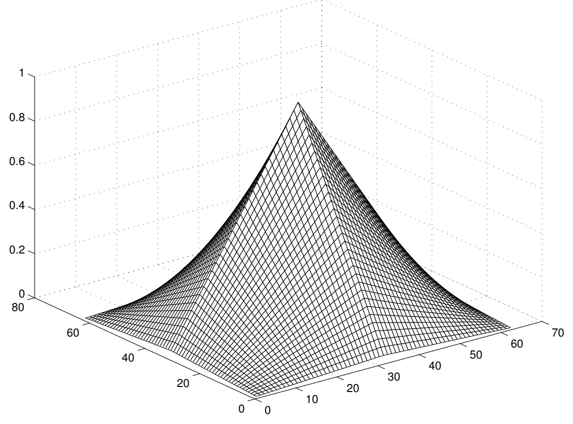
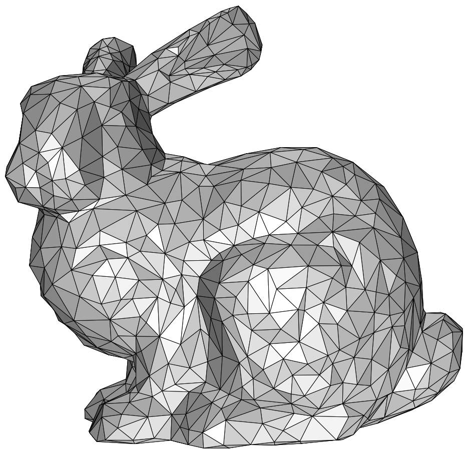


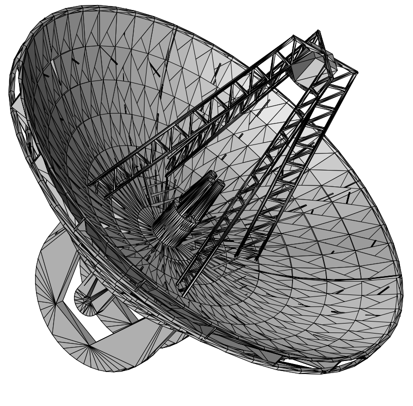


Multiresolution Signal and Geometry Processing: Filter Banks, Wavelets, and Subdivision

(Version: 2013-09-26)

Michael D. Adams





Multiresolution Signal and Geometry Processing: Filter Banks, Wavelets, and Subdivision

(Version: 2013-09-26)

Michael D. Adams

Department of Electrical and Computer Engineering
University of Victoria, Victoria, BC, Canada

University of Victoria

Published by the University of Victoria, Victoria, BC, Canada

Photography by Michael Adams

This book was typeset with LaTeX.

Library and Archives Canada Cataloguing in Publication

Adams, Michael D., 1969–, author
Multiresolution signal and geometry processing: filter banks, wavelets, and subdivision / Michael D. Adams, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada.

Includes bibliographical references and index.
Issued in print and electronic formats.
ISBN 978-1-55058-507-0 (pbk.).–ISBN 978-1-55058-508-7 (pdf)

1. Signal processing–Digital techniques–Mathematics. 2. Electric filters–Mathematics. 3. Wavelets (Mathematics). 4. Subdivision surfaces (Geometry). I. Title.

TK5102.9.A33 2013 621.382'2 C2013-906218-1
C2013-906219-X

To my students, past, present, and future

Contents

List of Tables

List of Figures

List of Listings

List of Acronyms

1D one dimensional

2D two dimensional

API application programming interface

CTFT continuous-time Fourier transform

CTFS continuous-time Fourier series

DCT discrete cosine transform

DTFS discrete-time Fourier series

DTFT discrete-time Fourier transform

FIR finite-length impulse response

FPT forward polyphase transform

GUI graphical user interface

IIR infinite-length impulse response

IPT inverse polyphase transform

MAE mean absolute error

MIMO multi-input multi-output

MRA multi-resolution approximation

MSE mean squared error

PAE peak absolute error

PR perfect reconstruction

PSNR peak-signal-to-noise ratio

SISO single-input single-output

UMD uniformly maximally decimated

License

Preface

This book provides provides a detailed introduction to multiresolution signal and geometry processing, and is primarily intended to be used as a text for graduate/undergraduate students in engineering (and other related) disciplines. The book evolved from a detailed set of lecture notes that the author prepared in order to teach graduate and undergraduate courses related to multiresolution signal and geometry processing at the University of Victoria. The initial work on these lecture notes commenced in the Fall 2003 term, and the first draft was completed during the Summer 2004 term for the teaching of ELEC 639. Over time, the lecture notes underwent many changes, eventually evolving into the book that you are now reading.

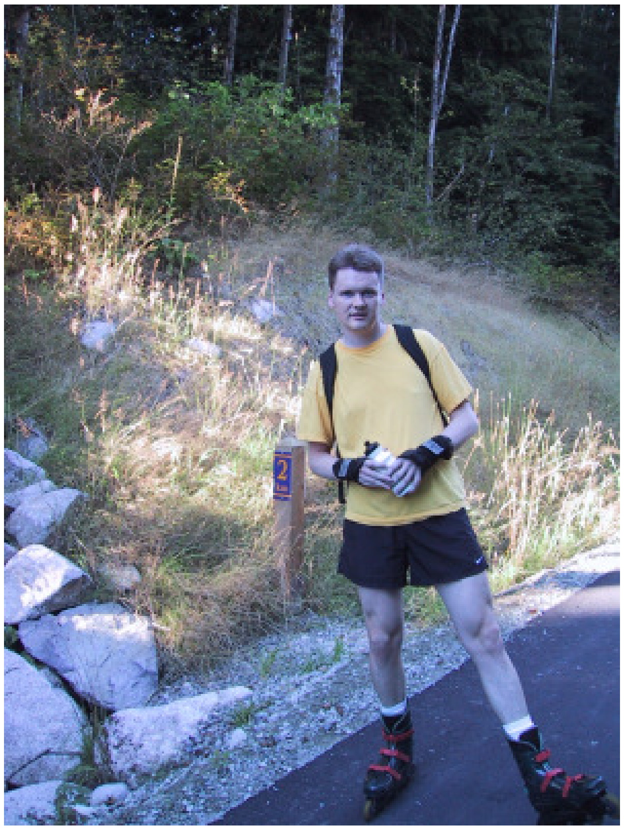
Acknowledgments

I would like to thank my colleagues Dr. Wu-Sheng Lu and Dr. Reinhard Illner for numerous technical discussions that have helped to improve the quality of these notes. Also, I would like to thank my past students for their feedback regarding earlier revisions of this book. They have helped me to eliminate numerous errors in this book that would have otherwise gone undetected. In particular, I would like to thank Ioana Sevcenco for her careful proofreading of significant portions of earlier drafts of the book. Also, I would like to express my gratitude to Dr. Leif Kobbelt for clarifying some technical details regarding $\sqrt{3}$ subdivision and Dr. Eugene Belogay for providing some additional information on his work on 3-adic wavelet systems.

I would like to thank the groups/organizations that provided some of the datasets used in the preparation of this book. In particular, several 3-D models were provided courtesy of the Stanford Computer Graphics Laboratory (e.g., Bunny and Armadillo) and NASA (e.g., telescope dish, spacesuit, and base station).

Michael Adams
Victoria, BC, Canada
September 26, 2013

About the Author



Michael Adams received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada in 1993, the M.A.Sc. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada in 1998, and the Ph.D. degree in electrical engineering from the University of British Columbia, Vancouver, BC, Canada in 2002. From 1993 to 1995, Michael was a member of technical staff at Bell-Northern Research in Ottawa, ON, Canada, where he developed real-time software for fiber-optic telecommunication systems. Since 2003, Michael has been on the faculty of the Department of Electrical and Computer Engineering at the University of Victoria, Victoria, BC, Canada, first as an Assistant Professor and currently as an Associate Professor.

Michael is the recipient of a Natural Sciences and Engineering Research Council (of Canada) Postgraduate Scholarship. He has served as a voting member of the Canadian Delegation to ISO/IEC JTC 1/SC 29 (i.e., Coding of Audio, Picture, Multimedia and Hypermedia Information), and been an active participant in the JPEG-2000 standardization effort, serving as co-editor of the JPEG-2000 Part-5 standard and principal author of one of the first JPEG-2000 implementations (i.e., JasPer). His research interests include signal processing, image/video/audio processing and coding, multiresolution signal processing (e.g., filter banks and wavelets), geometry processing, and data compression.

Part I

Preliminaries

Chapter 1

Introduction

1.1 Signal Representations

In many applications, we must deal with functions and/or sequences (i.e., signals). For this reason, it is often necessary to find useful representations for the signals of interest. For example, we often seek to express a signal x as a weighted sum of some set of functions $\{\varphi_n\}$ of the form

$$x = \sum_n \alpha_n \varphi_n$$

where the $\{\alpha_n\}$ are constants and the elements of $\{\varphi_n\}$ are called basis functions.

In practice, we always employ signal expansions that possess some structure. For example, an expansion might be chosen such that the basis functions are related to each other by elementary operations such as time-shifting, time-scaling, or modulation (i.e., frequency shifting). The use of structured expansions is motivated by practical considerations. If no structure were imposed, the resulting representations would be extremely cumbersome to employ, and lead to computationally inefficient (or intractable) algorithms.

Perhaps, one of the best known signal representations is the Fourier series for periodic signals. The Fourier series uses harmonically-related complex sinusoids as its basis functions. While this choice of basis functions does have many advantages, it does have a potential weakness. Although this representation provides an exact measure of the frequency content of a signal, it provides no time resolution. In other words, the representation does not capture at what points in time particular frequencies are present in a signal. For this reason, the Fourier series fails to provide any insight into how the behavior of a signal changes over time. Often, such insight can be highly beneficial or even essential, particularly in applications where we are interested in changes in signal behavior with time. For example, if we are required to locate transient events (such as signal singularities) in the time domain, the Fourier series is not particularly helpful.

To further illustrate the above point, consider a T-periodic function that has an impulse at the origin and is zero elsewhere on the interval $[0,T)$. If we represent this function with a Fourier series, all of the coefficients are equal in value, indicating that the function contains information at all frequencies. Unfortunately, the Fourier series coefficients do nothing to tell us of where the impulse is located in the time domain. If we have an application where we are trying to locate transient events in a signal (such as detecting singularities), the Fourier series is not helpful.

Ideally, we would like to use series expansions that simultaneously provide both perfect time and frequency resolution. Sadly, this is not possible as a consequence of the Heisenberg uncertainty principle. That is, we cannot simultaneously measure both the time and frequency content of a signal with arbitrary accuracy. Or viewed from different perspective, a function cannot have finite support in both the time and frequency domains. Therefore, we cannot find basis functions with this property, and we cannot find expansions that achieve arbitrarily high resolution in both time and frequency. Consequently, we can only trade time resolution for frequency resolution or vice versa. In most practical applications, we choose a signal representation such that each of the underlying basis functions has most of its energy concentrated in a finite interval in both the time and frequency domains.

Each of the basis functions associated with a signal representation has a particular distribution in time and frequency. In this way, we can think of the basis functions as defining a tiling of a time-frequency plane. That is, each basis function can be associated with a tile that covers the part of the plane where most of the basis function's energy is concentrated. For example, Figure 1.1 shows the tiling of the time-frequency plane used by several different signal representations.

In the case of a wavelet expansion, we choose the basis functions with a time-frequency tiling like that shown in Figure 1.1(d). The basis functions associated primarily with lower frequencies have relatively little spread in frequency and more spread in time, while the basis functions associated mainly with higher frequencies have relatively little spread in time but more spread in frequency. This leads to improved time resolution for high frequency information and lower time resolution for lower frequency information. In many applications, this tradeoff between time and frequency resolution is both natural and practically useful.

Wavelet series also have computational advantages over some of the competing signal representation methods. The calculation of wavelet series is computationally efficient. For a signal of length n, the wavelet transform is $O(n)$, whereas the discrete-time Fourier transform is $O(n \log n)$ computation.

Wavelet systems have proven to be useful in a great many applications including: the coding/compression of signals such as image, video, and audio data; singularity detection; and signal enhancement and denoising.

1.2 Historical Perspective

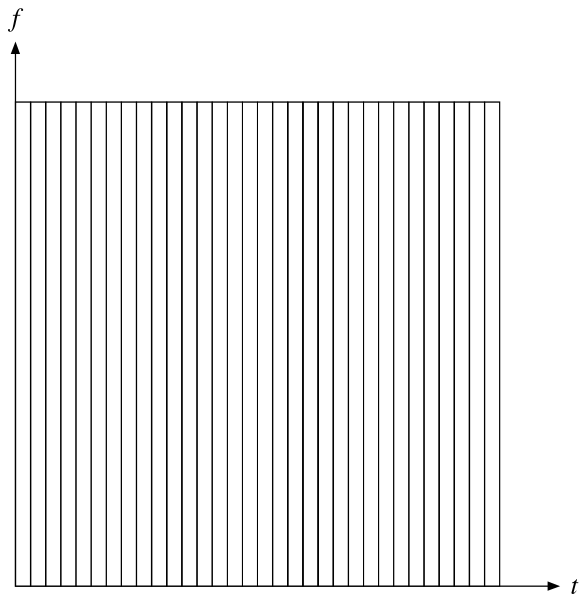
The first wavelet system was constructed by Haar [9] in 1910. The Haar wavelet system, as it is now known, uses piecewise constant functions to form an orthonormal basis for $L^2(\mathbb{R})$. Although wavelet theory has a relatively long history, it was not until the 1980s that the term "wavelet" came into use. Consequently, much of the early work on wavelets was done under the guise of other names such as Littlewood-Paley theory or Calderon-Zygmund operator theory. Although wavelet theory has intimate ties with concepts from many diverse branches of mathematics and engineering, many of these linkages were not discovered until the 1980s. Until that time, wavelet theory was really a disjoint set of ideas from many areas that lacked a clear unifying framework. In some sense, wavelet theory was only in its infancy until these linkages were first established.

In the mid-to-late 1980s, a revolution in wavelet theory occurred as a result of several important discoveries. This revolution served to draw together concepts from many different areas of mathematics and engineering resulting in a unified theory for the study of wavelet systems. In 1984, the term "wavelet" was introduced by Grossman and Morlet [8]. In 1988, a tremendous breakthrough in wavelet analysis was brought about by Daubechies. In her now classic paper [6], Daubechies introduced a family of compactly supported orthogonal wavelet systems with arbitrarily high, but fixed, regularity. The construction methods she employed were also closely related to filter banks. Daubechies' work stimulated a rapid development in wavelet theory. In 1989, Mallat [11] presented the theory of multiresolution analysis and also what later became known as the Mallat algorithm. This work provided a unifying framework for the study of wavelet systems tying together many previously disjoint ideas. In 1992, Cohen, Daubechies, and Feauveau [3] established the theory of biorthogonal wavelet systems. In the case of 2-band wavelet systems, biorthogonality has the advantage that it allows for symmetric finitely-supported basis functions. This property is not possible to have with orthogonal systems except in the trivial case of the Haar and other Haar-like transforms. The symmetry property can offer significant benefits in many applications, image compression being one example.

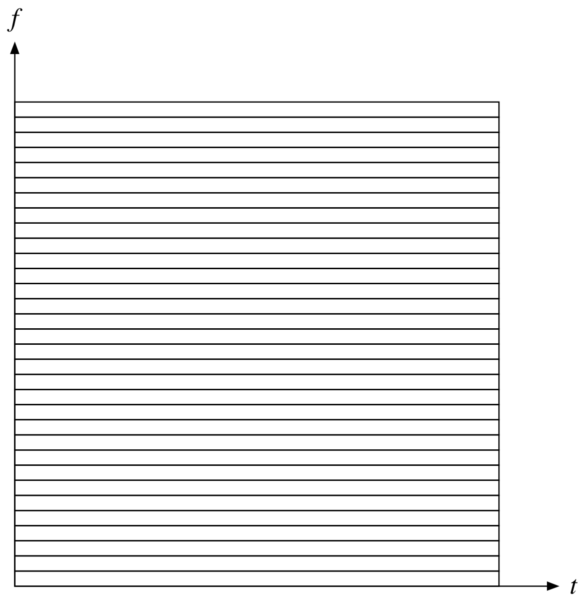
In 1995, Sweldens [16] proposed lifting, a technique for the design and implementation of wavelet systems. Subsequently, he has authored/coauthored many papers on the subject (e.g., [7], [17], [18]). Later, in 1996, Calderbank, Daubechies, Sweldens and Yeo [2] proposed a systematic lifting-based technique for constructing reversible versions of any 2-band wavelet transform. This method is of great significance in the context of lossless signal compression.

There are many important linkages between wavelet and filter bank theory. Digital filter bank methods have a somewhat shorter history than wavelet theory. Until the linkage between filter bank theory and wavelet theory was established, filter bank theory evolved independently from wavelet theory. Early contributions to (digital) filter bank theory and subband coding were made by Crochiere, Webber, and Flanagan [4], and Croisier, Esteban, and Galand [5], and others. Later, perfect reconstruction filter banks were studied in depth by many, with major contributions by Mintzer [12], Smith and Barnwell [15], Vetterli [20], and Vaidyanathan [19].

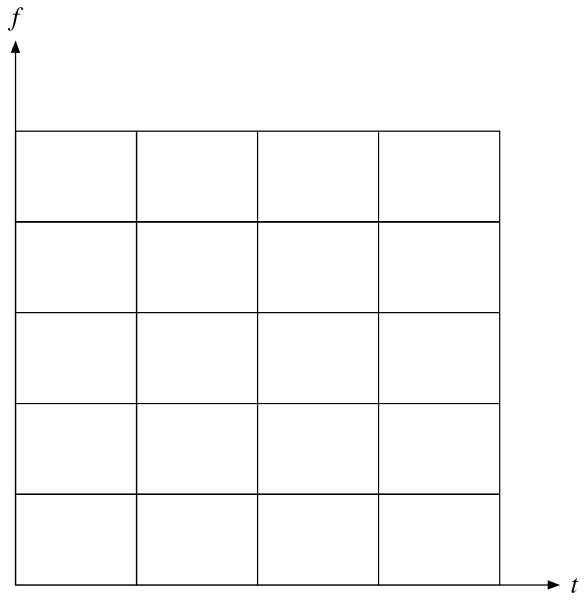
Wavelet transforms have proven to be extremely useful for image coding as many have shown (e.g., [1, 10]).



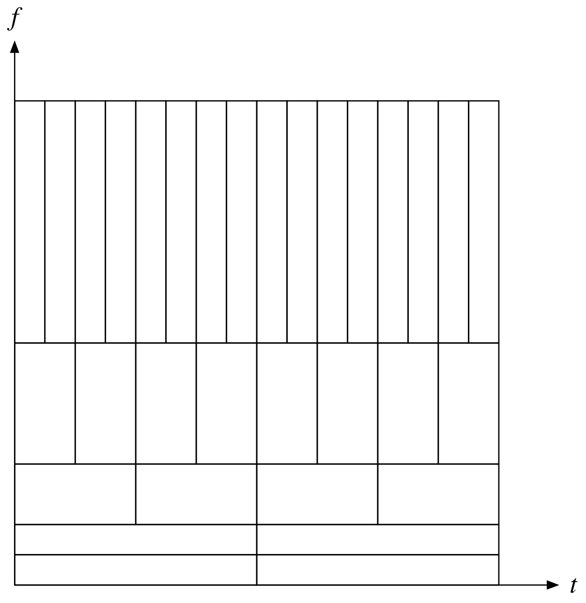
(a)



(b)



(c)



(d)

Figure 1.1: Time-frequency tilings for different signal representations. Tilings for the (a) identity basis, (b) Fourier series, (c) windowed Fourier series, and (d) wavelet series.

Consequently, many image compression schemes have adopted the use of such transforms. In 1993, Shapiro [14] introduced the notion of embedded coding with wavelets. His coding scheme, called embedded zerotree wavelet (EZW) coding, was based on a wavelet decomposition. Due to the obvious advantages of the embedding property in many applications, embedded coding quickly grew in popularity—especially as a means to build unified lossy/lossless compression systems. In 1995, Zandi et al. [21] proposed compression with reversible embedded wavelets (CREW), a reversible embedded image compression system based on some of the ideas of Shapiro. Not long after, in 1996, Said and Pearlman [13] introduced a new coding scheme known as set partitioning in hierarchical trees (SPIHT) which is similar in spirit to EZW.

1.3 Pedagogical Approach

The subject of wavelets is a difficult one to teach well, as it is inherently multidisciplinary, requiring an understanding of advanced concepts from diverse areas of mathematics (e.g., linear algebra and functional analysis) and signal processing (e.g., multirate systems). The key developments in the field have originated from a diverse group of individuals, including mathematicians, physicists, and engineers. Since an understanding of concepts from many diverse areas is required, learning about wavelets is often a challenging task. Students with a mathematics background typically lack the necessary signal processing fundamentals, while students from an engineering background usually lack the necessary mathematical foundations.

This book attempts to provide both the mathematics and signal processing background necessary to understand wavelet systems. The bias is towards engineering students, however. For this reason, more emphasis is placed on topics with which engineering students are typically less familiar, while other topics are not treated in as much detail.

1.4 Overview of the Book

This book is organized into several parts. In what follows, we describe each of these parts in order.

Part I, which is comprised of Chapters 1 and 2, provides some introductory material. In particular, Chapter 2 introduces some mathematical preliminaries that are needed in subsequent chapters. This material relates to topics such as Hilbert spaces, functional analysis, and Fourier analysis.

Part II, which is comprised of Chapters 3 and 4, primarily focuses on the study filter banks and wavelets in the one-dimensional context. In particular, Chapter 3 considers one-dimensional multirate systems and filter banks. Multirate filter banks are the computational structures used to realize wavelet transforms. Chapter 4 studies univariate wavelet systems. It presents wavelets from a mathematical perspective. Then, a link is established between wavelets and multirate filter banks.

Part III, which is comprised of Chapters 5, 6, and 7, examines filter banks and wavelets in the multidimensional setting. Chapter 5 provides some mathematical background related to multidimensional signal processing. Chapter 6 then considers filter banks in the multidimensional context. Chapter 7 examines the wavelets associated with such filter banks.

Part IV, which is comprised of Chapters 8 and 9 considers subdivision surfaces and subdivision wavelets. Chapter 8 provides some fundamentals related to geometry processing. Chapter 9 presents subdivision surfaces and subdivision wavelets.

Part V, which is comprised of Chapters 10 and 11, examines numerous applications of multiresolution signal and geometry processing. Chapter 10 considers the application of filter banks and wavelets to signal processing. This includes topics such as image compression and signal denoising. Chapter 11 covers the application of subdivision surfaces and wavelets to geometry processing.

Part VI corresponds to appendix material. This material includes some useful formulas, theorems, and tables. Some information on the Open Graphics Library (OpenGL) and Computational Geometry Algorithms Library (CGAL) is provided in addition to various source code listings.

1.5 Bibliography

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–220, April 1992.

[2] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. Technical report, Department of Mathematics, Princeton University, August 1996.

[3] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45:485–560, 1992.

[4] R. E. Crochiere, S. A. Webber, and J. L. Flanagan. Digital coding of speech in sub-bands. *Bell System Technical Journal*, 55(8):1069–1085, October 1976.

[5] A. Croisier, D. Esteban, and C. Galand. Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques. In *International Conference on Information Sciences and Systems*, pages 443–446, Patras, Greece, August 1976.

[6] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, November 1988.

[7] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4:247–269, 1998.

[8] A. Grossman and J. Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, 15(4):723–736, July 1984.

[9] A. Haar. Zur theorie der orthogonalen funktionen-systeme. *Math. Annal.*, 69:331–371, 1910.

[10] A. S. Lewis and G. Knowles. Image compression using the 2-D wavelet transform. *IEEE Trans. on Image Processing*, 1(2):244–250, April 1992.

[11] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[12] F. Mintzer. Filters for distortion-free two-band multirate filter banks. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 33(3):626–630, June 1985.

[13] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[14] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[15] M. J. T. Smith and T. P. Barnwell. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 34(3):434–441, June 1986.

[16] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In *Proc. of SPIE*, volume 2569, pages 68–79, San Diego, CA, USA, September 1995.

[17] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, 1996.

[18] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal of Mathematical Analysis*, 29(2):511–546, March 1998.

[19] P. P. Vaidyanathan. Theory and design of *m*-channel maximally decimated quadrature mirror filters with arbitrary *m*, having the perfect-reconstruction property. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 35(4):476–492, April 1987.

[20] M. Vetterli. Filter banks allowing perfect reconstruction. *Signal Processing*, 10(3):219–244, April 1986.

[21] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek. CREW: Compression with reversible embedded wavelets. In *Proc. of IEEE Data Compression Conference*, pages 212–221, Snowbird, UT, USA, March 1995.

Chapter 2

Mathematical Preliminaries

2.1 Overview

In order to study wavelet systems, we first need to establish some mathematical preliminaries. This chapter provides these mathematical foundations. We begin by introducing sets, and then progressively add more structure to sets in order to obtain metric spaces, vector spaces, normed spaces, inner product spaces, and Hilbert spaces. Some useful function and sequence spaces are also examined. As we shall see, function and sequence spaces are of critical importance in the study of wavelet systems. Later in the chapter, we turn our attention to Fourier analysis. The Fourier transform is introduced in the context of various function spaces.

2.2 Sets

One mathematical object of fundamental importance is the set. Using sets, we can construct other more sophisticated and practically useful mathematical objects. Now, we proceed to define the notion of a set.

Definition 2.1 (Set, empty set)**.** A **set** is a collection of objects called **elements**. The **empty set**, denoted $\emptyset$, is the set containing no elements.

To denote that an object x is an element of the set A, we write $x \in A$. Two sets are equal if they contain exactly the same elements.

Definition 2.2 (Subset, proper and improper subsets)**.** A set B is said to be a **subset** of a set A, denoted $B \subset A$, if every element of B is an element of A. If $B \subset A$ and $B \neq A$, then B is said to be a **proper subset** of A. (A subset that is not proper is called improper.)

Example 2.1. Let $A = \{1,2,3,4\}$ and $B = \{2,4\}$. Then, $B \subset A$. In particular, B is a proper subset of A.

At this point, we introduce some commonly used sets. The sets of natural numbers, integers, and rational numbers are denoted as $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$, respectively, and are defined as

$$\mathbb{N} = \{1,2,3,\ldots\}, \quad \mathbb{Z} = \{\ldots,-2,-1,0,1,2,\ldots\}, \quad \text{and}$$
$$\mathbb{Q} = \{x : x = p/q \text{ where } p,q \in \mathbb{Z},\ q \neq 0\}.$$

We denote the sets of real numbers and complex numbers as $\mathbb{R}$ and $\mathbb{C}$, respectively, where

$$\mathbb{C} = \{z : z = x + jy \text{ for } x,y \in \mathbb{R} \text{ and } j^2 = -1\}.$$

For convenience, we denote the nonnegative, positive, and negative integers as $\mathbb{Z}^*$, $\mathbb{Z}^+$, and $\mathbb{Z}^-$, respectively. Or more formally, we can write

$$\mathbb{Z}^* = \{x : x \in \mathbb{Z}, x \geq 0\}, \quad \mathbb{Z}^+ = \mathbb{N} = \{x : x \in \mathbb{Z}, x > 0\}, \quad \text{and} \quad \mathbb{Z}^- = \{x : x \in \mathbb{Z}, x < 0\}.$$

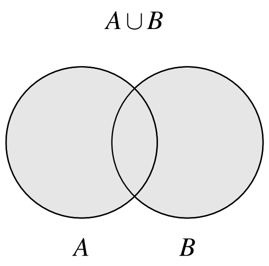


Figure 2.1: Union of sets.

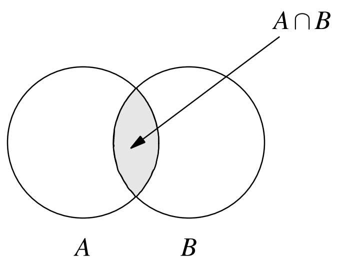


Figure 2.2: Intersection of sets.

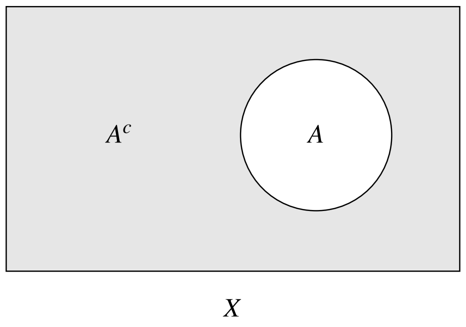


Figure 2.3: Complement of set.

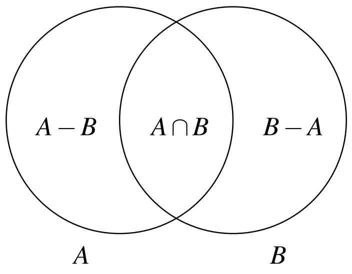


Figure 2.4: Difference of sets.

It is important to note that the sets $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$, do not include infinity. The omission of infinity from these sets is necessary in order to allow for a consistent system with respect to all of the usual arithmetic operations (i.e., addition, subtraction, multiplication, and division).

For convenience, we define the following notation for subsets of the real line $\mathbb{R}$:

$$\begin{aligned}
[a,b] &= \{x \in \mathbb{R} : a \leq x \leq b\}, \\
(a,b) &= \{x \in \mathbb{R} : a < x < b\}, \\
[a,b) &= \{x \in \mathbb{R} : a \leq x < b\}, \quad \text{and} \\
(a,b] &= \{x \in \mathbb{R} : a < x \leq b\}.
\end{aligned}$$

Several basic operations on sets are given by the definitions below.

Definition 2.3 (Union, intersection, complement, and difference of sets)**.** The **union** of two sets A and B, denoted $A \cup B$, is the set obtained by combining the elements of A and B. The **intersection** of two sets A and B, denoted $A \cap B$, is the set of elements common to both A and B. The **complement** of A in B, denoted A^c, is the set $\{x : x \in B, x \notin A\}$. The **difference** of the sets A and B, denoted $A - B$ or $A \setminus B$, is the set consisting of all elements in A that are not in B (i.e., $A - B = A \setminus B = \{x : x \in A, x \notin B\}$). (The set difference $A \setminus B$ is also referred to as the **relative complement** of B in A.)

The union, intersection, complement, and difference operations are illustrated in Figures 2.1, 2.2, 2.3, and 2.4, respectively. Two sets A and B are said to be **disjoint** if A and B have no common elements (i.e., $A \cap B = \emptyset$).

Example 2.2. Let $A = \{1,2,3\}$, $B = \{3,4,5\}$, and $C = \{1,2,3,4,5,6\}$. Then, we have $A \cap B = \{3\}$, $A \cup B = \{1,2,3,4,5\}$, $A - B = \{1,2\}$ and $B - A = \{4,5\}$. The complement of A in C is $\{4,5,6\}$. The complement of B in C is $\{1,2,6\}$.

One can show that

$$\begin{gathered}A\cup A=A,\quad A\cap A=A,\quad A\cup B=B\cup A,\quad A\cap B=B\cap A,\\ A\cup(B\cup C)=(A\cup B)\cup C\\ A\cap(B\cap C)=(A\cap B)\cap C\\ A\cup(B\cap C)=(A\cup B)\cap(A\cup C)\\ A\cap(B\cup C)=(A\cap B)\cup(A\cap C)\\ A\cap B\subset A,\quad A\cup B\supset A,\\ (A^c)^c=A,\quad X^c=\emptyset,\quad \emptyset^c=X,\\ (A\cup B)^c=A^c\cap B^c,\quad\text{and}\\ (A\cap B)^c=A^c\cup B^c.\end{gathered}$$

Definition 2.4 (Ordered n-tuple)**.** An **ordered n-tuple** $(x_1,x_2,\ldots,x_n)$ is a set of n objects, where one is designated as the first, one as the second, and so on, until the nth object is reached. When $n=2$, we have what is called an **ordered pair**. When $n=3$, we have what is called an **ordered triplet**.

Definition 2.5 (Cartesian product)**.** Let $X_1,X_2,\ldots,X_n$ be sets. Then, we define the **Cartesian product** $X_1\times X_2\times\cdots\times X_n$ as the set of all ordered n-tuples $(x_1,x_2,\ldots,x_n)$ where $x_1\in X_1, x_2\in X_2,\ldots,x_n\in X_n$.

As a matter of notation, the n-fold Cartesian product of a set S is denoted as S^n. For example, we can abbreviate $\mathbb{Z}\times\mathbb{Z}$ and $\mathbb{R}\times\mathbb{R}\times\mathbb{R}$ as $\mathbb{Z}^2$ and $\mathbb{R}^3$, respectively.

Example 2.3. Let $X_1=\{1,2\}$ and $X_2=\{3,4\}$. Then, $X_1\times X_2$ is the set containing the elements:

$$(1,3),(1,4),(2,3),(2,4).$$

Moreover, X_1^2 is the set containing the elements:

$$(1,1),(1,2),(2,1),(2,2).$$

Now, we introduce a few definitions that relate to the number of elements in a set.

Definition 2.6 (Cardinality)**.** The number of elements in a set X is said to be the **cardinality** of the set, and is denoted $\operatorname{card}X$.

Example 2.4. We have that $\operatorname{card}\{1,3,5\}=3$ and $\operatorname{card}\mathbb{Z}$ is not finite.

Definition 2.7 (Finite, countable, and uncountable sets)**.** If a set S has a one-to-one correspondence with $\{1,2,\ldots,n\}$ where n is finite, then S is said to be **finite**. If a set S has a one-to-one correspondence with the natural numbers (or equivalently, the integers), S is said to be **countably infinite**. A set that is either finite or countably infinite is said to be **countable**. A set that is not countable is referred to as **uncountable**.

Example 2.5. The set $\{1,\frac{1}{2},\frac{1}{4}\}$ is finite. The set of even integers is countable (or, more specifically, countably infinite). The set of real numbers $\mathbb{R}$ is uncountable.

Lemma 2.1. *The set $\mathbb{Q}$ (of rational numbers) is countable.*

Proof. Every nonzero rational number n has a (unique) representation of the form $n=p/q$, where $q>0$ and p and q are coprime. In the case of the rational number zero, we simply choose the representation $n=\frac{0}{1}$. We define the height h of the rational number p/q (expressed in the form above) as $h(p,q)=|p|+q$. The rational numbers can then be partitioned into classes by height. Now, we observe that each of these classes contains only finitely many elements. For example, only one rational number has height one (i.e., $\frac{0}{1}$); only two rational numbers have height two (i.e., $\frac{-1}{1}$, $\frac{1}{1}$); and only four rational numbers have height three (i.e., $\frac{-2}{1}$, $\frac{-1}{2}$, $\frac{1}{2}$, $\frac{2}{1}$). Thus, we can order the classes by height, and the elements within each class by increasing numerator. This process establishes an ordering on the set of rational numbers (i.e., $\frac{0}{1}$, $\frac{-1}{1}$, $\frac{1}{1}$, $\frac{-2}{1}$, $\frac{-1}{2}$, $\frac{1}{2}$, $\frac{2}{1}$, $\frac{-3}{1}$, and so on). Consequently, a one-to-one correspondence exists between the rational and natural numbers. Therefore, the set $\mathbb{Q}$ (of rational numbers) is countable. □

2.2.1 Maximum, Minimum, Supremum, and Infimum

A set $X \subset \mathbb{R}$ is said to be **bounded from above** if there exists $u \in \mathbb{R}$ such that $x \leq u$ for all $x \in X$. The quantity u is said to be an **upper bound** of X. Similarly, X is **bounded from below** if there exists $l \in \mathbb{R}$ such that $x \geq l$ for all $x \in X$. The quantity l is said to be a **lower bound** of X. If a set is bounded from above and below, the set is said to be **bounded**.

A real number m is said to be the **maximum** of a set $X \subset \mathbb{R}$ if $m \in X$ and m is an upper bound for X. Similarly, a real number n is said to be the **minimum** of a set $X \subset \mathbb{R}$ if $n \in X$ and n is a lower bound for X. We denote the maximum and minimum of the set X as $\max X$ and $\min X$, respectively. It is important to note that a bounded set need not have a maximum or minimum.

Example 2.6. Identify the minimum and maximum of each of the following sets:
(a) $A = \{x \in \mathbb{R} : 0 \leq x \leq 1\}$,
(b) $B = \{x \in \mathbb{R} : 0 < x < 1\}$, and
(c) $C = \{x \in \mathbb{R} : 1 \leq x\}$.

Solution. (a) For the set A, we have $\min A = 0$ and $\max A = 1$. (b) The set B has neither a minimum nor a maximum. (c) For the set C, we have $\min C = 1$ and $\max C$ does not exist. □

Let X be a nonempty set that is bounded from above, and let U denote the set of all upper bounds of X. The set U must always have a minimum, which is clearly the least upper bound of X. The least upper bound of a set X is referred to as the **supremum** of X, denoted $\sup X$. If X is not bounded from above, we define $\sup X = \infty$. If X is empty, we define $\sup X = -\infty$.

Similarly, the greatest lower bound of the set X is called the **infimum** of X, denoted $\inf X$. If X is not bounded from below, we define $\inf X = -\infty$. If X is empty, we define $\inf X = \infty$.

Example 2.7. Identify the maximum, minimum, supremum, and infimum of each of the following sets:
(a) $A_1 = \{x \in \mathbb{R} : 1 \leq x \leq 2\}$,
(b) $A_2 = \{x \in \mathbb{R} : 1 < x < 2\}$, and
(c) $A_3 = \{x \in \mathbb{R} : x < \sqrt{2}\}$.

Solution. (a) For the set A_1, we have $\min A_1 = \inf A_1 = 1$ and $\max A_1 = \sup A_1 = 2$. (b) For the set A_2, we have $\inf A_2 = 1$ and $\sup A_2 = 2$, and neither $\max A_2$ nor $\min A_2$ exist. (c) For the set A_3, we have $\inf A_3 = -\infty$, $\sup A_3 = \sqrt{2}$, and $\min A_3$ and $\max A_3$ do not exist. □

2.2.2 Characteristic Function

Definition 2.8 (Characteristic function)**.** Let A be a subset of a set X. Then, the **characteristic function** of A, denoted χ_A, is defined as

$$\chi_A(t) = \begin{cases} 1 & \text{if } t \in A \\ 0 & \text{if } t \in X \setminus A. \end{cases}$$

The characteristic function is also commonly referred to as the **indicator function**.

Example 2.8. Consider the unit-step function $u(t)$, where

$$u(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

We have that $u(t) = \chi_{[0,\infty)}(t)$.

2.2.3 Some Measure Theory Concepts

Below, we introduce a few definitions related to measure theory. We will need these definitions in later discussions.

Definition 2.9 (Measure zero)**.** A subset S of $\mathbb{R}$ is said to have **measure zero** if, for every $\varepsilon > 0$, there is a sequence $\{I_n\}_{n\in\mathbb{N}}$ of open intervals, say with $I_n = (a_n, b_n)$ and $a_n < b_n$, such that

$$S \subset \bigcup_{n\in\mathbb{N}} I_n \quad \text{and} \quad \sum_{n\in\mathbb{N}} (b_n - a_n) < \varepsilon.$$

(The first condition ensures that the intervals $\{I_n\}$ cover all of S and the second condition ensures that the interval widths $b_n - a_n$ have an arbitrarily small sum.)

Any countable subset of $\mathbb{R}$ has measure zero. For example, the sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ each have measure zero. A countable union of sets of measure zero is a set of measure zero. An uncountable set may, in some cases, have measure zero (e.g., the Cantor set).

A property of a set X is said to hold **almost everywhere** if the set of elements in X for which the property fails to hold has measure zero. The term “almost everywhere” is often abbreviated as “a.e.”.

Example 2.9. Consider $\chi_{\mathbb{Z}}(t)$ where $t \in \mathbb{R}$. Since $\chi_{\mathbb{Z}}(t)$ is zero except for $t \in \mathbb{Z}$, and $\mathbb{Z}$ is a set of measure zero, we can say that $\chi_{\mathbb{Z}}(t) = 0$ almost everywhere.

Definition 2.10 (Essential infimum and supremum)**.** The **essential infimum** of a function x defined on the set I is given by

$$\operatorname{ess\,inf}_{t\in I} x(t) = \sup\{A : x(t) \geq A \text{ a.e.}\}.$$

In other words, the essential infimum is the greatest lower bound that holds almost everywhere. The **essential supremum** of a function x defined on the set I is given by

$$\operatorname{ess\,sup}_{t\in I} x(t) = \inf\{A : x(t) \leq A \text{ a.e.}\}.$$

In other words, the essential supremum is the least upper bound that holds almost everywhere.

Example 2.10. Consider the functions f, g defined on $\mathbb{R}$ as given by

$$f(t) = \begin{cases} 2 & \text{if } t \in \mathbb{Z} \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \quad g(t) = \begin{cases} 2 & \text{if } t \in \mathbb{R} \setminus \mathbb{Z} \\ 1 & \text{otherwise.} \end{cases}$$

For these functions, we have

$$\inf_{t\in\mathbb{R}} f(t) = 1, \quad \operatorname{ess\,inf}_{t\in\mathbb{R}} f(t) = 1, \quad \sup_{t\in\mathbb{R}} f(t) = 2, \quad \operatorname{ess\,sup}_{t\in\mathbb{R}} f(t) = 1,$$
$$\inf_{t\in\mathbb{R}} g(t) = 1, \quad \operatorname{ess\,inf}_{t\in\mathbb{R}} g(t) = 2, \quad \sup_{t\in\mathbb{R}} g(t) = 2, \quad \text{and} \quad \operatorname{ess\,sup}_{t\in\mathbb{R}} g(t) = 2.$$

2.3 Integration

One fundamentally important concept in mathematics is that of the integral. Certainly, we are all well acquainted with the Riemann integral, as introduced in any basic calculus course. Unfortunately, this definition of integration has some shortcomings. For example, the space of Riemann integrable functions is not complete, when considered as a metric space. This lack of completeness is quite problematic when we need to work with function spaces. Such shortcomings motivate the introduction of the Lebesgue integral. The Lebesgue integral can be thought of as a generalization of the Riemann integral. The Lebesgue integral overcomes many of the problems associated with the Riemann integral.

2.3.1 Riemann Integral

From our early days studying calculus, we are all familiar with the Riemann integral. (Riemann is pronounced ree-mahn.)

In what follows, we will formally define the Riemann integral. Consider a bounded function f defined on a finite closed interval $a \leq t \leq b$. Since f is bounded, we have $m \leq f(t) \leq M$ for all $t \in [a,b]$ where

$$\begin{gathered} M = \sup\{f(t) : a \leq t \leq b\} \quad \text{and} \\ m = \inf\{f(t) : a \leq t \leq b\}. \end{gathered}$$

Now, we define the partition P of the interval $[a,b]$ as the finite collection of points $\{t_0, t_1, \ldots, t_n\}$ such that $a = t_0 < t_1 < \cdots < t_n = b$. Since f is bounded on $[a,b]$, it is also bounded on the ith subinterval $[t_{i-1}, t_i]$. We define

$$\begin{gathered} M_i = \sup\{f(t) : t_{i-1} \leq t \leq t_i\}, \\ m_i = \inf\{f(t) : t_{i-1} \leq t \leq t_i\}, \quad \text{and} \\ \Delta t_i = t_i - t_{i-1}. \end{gathered}$$

The graphical interpretation of the various quantities is shown in Figure 2.5. For each partition P, we can form the upper and lower sums

$$\begin{gathered} U(P,f) = \sum_{i=1}^{n} M_i \Delta t_i, \quad \text{and} \\ L(P,f) = \sum_{i=1}^{n} m_i \Delta t_i. \end{gathered}$$

It is not difficult to convince oneself that U and L satisfy the inequality

$$m(b-a) \leq L(P,f) \leq U(P,f) \leq M(b-a).$$

We now define the upper and lower integrals as

$$\begin{gathered} \overline{\int_a^b} f\,dt = \inf U(P,f) \quad \text{and} \\ \underline{\int_a^b} f\,dt = \sup L(P,f), \end{gathered}$$

where the infimum and supremum are taken over all possible partitions P of the interval $[a,b]$. One can show that the upper and lower integrals satisfy the inequality

$$\underline{\int_a^b} f\,dt \leq \overline{\int_a^b} f\,dt.$$

If the upper and lower integrals are equal, the Riemann integral of f exists and is defined as

$$\int_a^b f\,dt = \underline{\int_a^b} f\,dt = \overline{\int_a^b} f\,dt.$$

It is important to note that not all bounded functions are Riemann integrable. For example, consider the function

$$f(t) = \begin{cases} 1 & \text{if } t \in \mathbb{Q} \\ 0 & \text{otherwise.} \end{cases}$$

(This function is known as the **Dirichlet function** and is the indicator function of the rational numbers, namely $\chi_{\mathbb{Q}}$.) Clearly, f is bounded. Interestingly enough, however, the Riemann integral of this function does not exist. For any

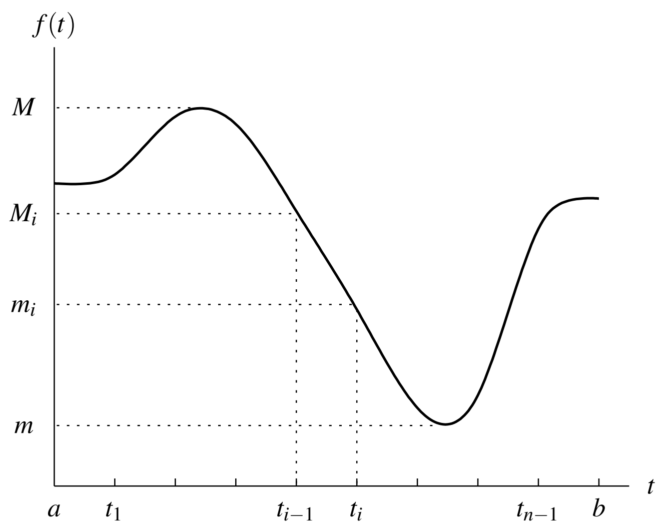


Figure 2.5: Riemann integration.

partition P of the interval $[a,b]$, we will have that each subinterval contains both rational and irrational numbers. So, we have $M_i = 1$ and $m_i = 0$ for all i. Consequently, $U(P,f) = 1(b-a) = b-a$ and $L(P,f) = 0(b-a) = 0$. This yields $\overline{\int_a^b} f dt = b-a$ and $\underline{\int_a^b} f dt = 0$. Thus, f is not Riemann integrable.

The Riemann integral has a number of important properties as stated in the theorem below. The proof of these properties can be found in many textbooks on calculus.

Theorem 2.1 (Properties of the Riemann integral)**.** *Let $R[a,b]$ denote the set of Riemann integrable functions on the interval $[a,b]$. The Riemann integral has the following properties:*

1. *If $f \in R[a,b]$ and $a \le c \le b$, then $f \in R[a,c]$, $f \in R[c,b]$, and $\int_a^b f dt = \int_a^c f dt + \int_c^b f dt$.*
2. *If $f_1, f_2 \in R$ and $\alpha_1, \alpha_2 \in \mathbb{R}$, then $f = \alpha_1 f_1 + \alpha_2 f_2 \in R$ and $\int_a^b f dt = \alpha_1 \int_a^b f_1 dt + \alpha_2 \int_a^b f_2 dt$.*
3. *If $f \in R$ and $f(t) > 0$ on $[a,b]$, then $\int_a^b f dt \ge 0$.*
4. *If $f \in R$ and $m \le f(t) \le M$ on $[a,b]$, then $m(b-a) \le \int_a^b f dt \le M(b-a)$.*
5. *If f is continuous on $[a,b]$, then there is a $\lambda \in [a,b]$ such that $\int_a^b f dt = f(\lambda)[b-a]$.*
6. *If $f, g \in R$ and $f(t) \le g(t)$, then $\int_a^b f dt \le \int_a^b g dt$.*

2.3.2 Problem with the Riemann Integral

The Riemann integral has one particularly serious shortcoming. Let $R[a,b]$ denote the set of Riemann integrable functions on the interval $[a,b]$. It is possible for a sequence $\{f_n\}$ of functions in $R[a,b]$ to converge to a function f, which may even be bounded, where $f \notin R[a,b]$. For example, consider the sequence of functions

$$f_n(t) = \lim_{m\to\infty} [\cos(n!\pi t)]^{2m} = \begin{cases} 1 & \text{if } t = \frac{k}{n!} \text{ for integer } k \\ 0 & \text{otherwise.} \end{cases}$$

That is, we have

$$f_0(t) = f_1(t) = \begin{cases} 1 & \text{if } t \in \mathbb{Z} \\ 0 & \text{otherwise,} \end{cases}$$

$$f_2(t) = \begin{cases} 1 & \text{if } 2t \in \mathbb{Z} \\ 0 & \text{otherwise,} \end{cases}$$

and generally

$$f_n(t) = \begin{cases} 1 & \text{if } n!t \in \mathbb{Z} \\ 0 & \text{otherwise.} \end{cases}$$

One can show that as $n \to \infty$, $f_n(t)$ approaches $f(t)$, where

$$f(t) = \begin{cases} 1 & \text{if } t \in \mathbb{Q} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The function f, however, is not Riemann integrable (as we saw earlier).

If a function has a sufficiently large number of discontinuities (as in the above example), its Riemann integral may fail to exist. If the number of discontinuities is sufficiently small, however, we are guaranteed that Riemann integral exists, as elucidated by the theorem below.

Theorem 2.2. *Let f be a bounded function defined on $[a,b]$. Then, f is Riemann integrable on $[a,b]$ if and only if the set of points at which f is discontinuous is a set of measure zero.*

2.3.3 Lebesgue Integral

Let C_0 denote the set of all continuous real-valued functions defined on $\mathbb{R}$ with compact support. Given a sequence $\{\phi_n\}$ in C_0 such that $\sum_n \int |\phi_n| \, dt < \infty$, one can show that $\sum_n \phi_n$ converges almost everywhere and $\sum_n \int \phi_n dt$ converges. If a function f can be expressed as $f = \sum_n \phi_n$ almost everywhere, it is reasonable to define the integral of f as $\int f dt = \sum_n \int \phi_n dt$. This idea leads to the definition of the Lebesgue integral which we formalize below.

Definition 2.11 (Lebesgue integral)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a function that satisfies $f = \sum_n \phi_n$ almost everywhere for some sequence $\{\phi_n\}$ in C_0 with $\sum_n \int |\phi_n| \, dt < \infty$. Then f is said to be **Lebesgue integrable** and the **Lebesgue integral** of f is given by

$$\int f dt = \sum_n \int \phi_n dt.$$

The Lebesgue integral has a number of important properties, some of which are given by the theorem below.

Theorem 2.3 (Properties of the Lebesgue integral)**.** *Let L denote the set of Lebesgue integrable functions (on some interval). Let $f, g \in L$ and let $\alpha, \beta \in \mathbb{R}$. Further, let $\int_R$ denote the Riemann integral. Then, we have*

1. *if $\int_R f dt$ exists, then $\int f dt$ exists and $\int_R f dt = \int f dt$;*
2. *$\alpha f \in L$ and $\int \alpha f dt = \alpha \int f dt$;*
3. *$f + g \in L$ and $\int (f+g) dt = \int f dt + \int g dt$;*
4. *$|f| \in L$ and $|\int f dt| \leq \int |f| dt$;*
5. *if $f = 0$ almost everywhere, then $\int f dt = 0$;*
6. *if $f > 0$ then $\int f dt \geq 0$.*

Let us now revisit the function f given by (2.1). Although the Riemann integral of f does not exist, the Lebesgue integral of f is, in fact, well defined. Since the measure of the set of rational numbers is zero, the function f is zero almost everywhere. Therefore, from the properties of the Lebesgue integral, we have $\int f dt = 0$.

Unless otherwise noted, all subsequent integrals appearing in this book should be assumed taken in the Lebesgue sense.

2.4 Spaces

By imposing additional structure on sets, we can obtain other useful mathematical objects called spaces. By adding different types of structure to a set, we can obtain different types of spaces. In the sections that follow, we introduce several important types of spaces, namely metric spaces, vector spaces, normed spaces, inner product spaces, and Hilbert spaces. Of particular interest are spaces with underlying sets having either functions or sequences as elements. Such spaces are called function and sequence spaces, respectively.

2.5 Metric Spaces

Now, we introduce metric spaces. Metric spaces are one of the simplest types of spaces, as they have very little structure. Before we can define a metric space, however, we must first define the notion of a metric.

Definition 2.12 (Metric)**.** A **metric** d on a set X is a real-valued function defined on $X \times X$ that satisfies the following axioms:

1. $d(x,y) \geq 0$ for all $x,y \in X$ (nonnegativity);
2. $d(x,y) = 0$ if and only if $x = y$ (strict positivity);
3. $d(x,y) = d(y,x)$ for all $x,y \in X$ (symmetry); and
4. $d(x,y) \leq d(x,z) + d(z,y)$ for all $x,y,z \in X$ (triangle inequality).

Example 2.11. Let X be the set of all real-valued continuous functions defined on the interval $[a,b]$ (i.e., $X = C[a,b]$). Define the function

$$d(x,y) = \int_a^b |x(t) - y(t)|\, dt.$$

Determine whether d is a metric on X.

Solution. We can easily confirm that d satisfies the nonnegativity, strict positivity, and symmetry properties. That is, we have that

$$\int_a^b |x(t) - y(t)|\, dt \geq 0,$$

$$\int_a^b |x(t) - y(t)|\, dt = 0 \Leftrightarrow x(t) = y(t), \quad \text{and}$$

$$\int_a^b |x(t) - y(t)|\, dt = \int_a^b |y(t) - x(t)|\, dt.$$

It now remains to be determined whether d satisfies the triangle inequality. From the definition of d, we have

$$\begin{aligned} d(x,y) &= \int_a^b |x(t) - y(t)|\, dt \\ &= \int_a^b |x(t) - z(t) + z(t) - y(t)|\, dt. \end{aligned}$$

By the Minkowski inequality for integrals (E.3), we can write

$$d(x,y) \le \int_a^b |x(t)-z(t)|\,dt + \int_a^b |z(t)-y(t)|\,dt = d(x,z)+d(z,y).$$

Thus, the triangle inequality holds. Therefore, d is a metric on X. □

Example 2.12. Let X be the set of real numbers, and define the function

$$d(x,y) = (x-y)^2.$$

Determine whether d is a metric on X.

Solution. Since the square of a real number is always nonnegative, $d(x,y) \ge 0$. Hence, the nonnegativity property holds. Clearly, we also have that $d(x,y) = (x-y)^2 = 0$ if and only if $x = y$. Thus, the strict positivity property holds. Since $d(x,y) = (x-y)^2 = (y-x)^2 = d(y,x)$, the symmetry property also holds. Now, it only remains to be determined whether the triangle inequality is satisfied. We have

$$d(x,y) = (x-y)^2 = x^2 - 2xy + y^2$$

and

$$\begin{aligned} d(x,z)+d(z,y) &= (x-z)^2 + (z-y)^2 \\ &= x^2 - 2xz + z^2 + z^2 - 2yz + y^2 \\ &= x^2 + y^2 + 2z^2 - 2xz - 2yz. \end{aligned}$$

Suppose now that $x > 0$, $y < 0$, and $z = 0$. Then, we have

$$d(x,z)+d(z,y) = x^2 + y^2$$

and $d(x,y) = x^2 - 2xy + y^2$ where $-2xy > 0$. Consequently (since $-2xy > 0$),

$$d(x,y) > d(x,z)+d(z,y).$$

Thus, the triangle inequality does not hold in general. Therefore, d is not a metric on X. □

Given the definition of a metric from above, we can now define a metric space.

Definition 2.13 (Metric space)**.** A **metric space** is a set X with a metric d, and is denoted (X,d) or simply X when the metric is understood from the context.

A metric space is a set with additional structure defined by means of a metric. This additional structure is called **topological structure**.

A metric can be thought of as a measure of distance between two members of a set. For example, consider the set X of points in the Euclidean plane (i.e., $X = \mathbb{R}^2$). Let $x, y \in X$ where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. One can show that

$$d_2(x,y) = \left[(x_1-y_1)^2 + (x_2-y_2)^2\right]^{1/2}$$

constitutes a metric for the set X. Thus, (X,d_2) is a metric space. In fact, d_2 is the familiar measure of distance in the Euclidean plane. In this context, the triangle inequality has the interpretation shown in Figure 2.6. In essence, the triangle inequality states that the shortest path between two points (with respect to the metric) is along the line segment joining them.

Different metrics can be defined over the same set. For example, let us again consider the set $X = \mathbb{R}^2$. One can show that all of the following functions are metrics on X:

$$\begin{aligned} d_1(x,y) &= |x_1-y_1| + |x_2-y_2|, \\ d_2(x,y) &= \left[(x_1-y_1)^2 + (x_2-y_2)^2\right]^{1/2}, \quad \text{and} \\ d_\infty(x,y) &= \max\{|x_1-y_1|, |x_2-y_2|\}. \end{aligned}$$

Each of these metrics defines a distinct metric space. That is, (X,d_1), (X,d_2), and (X,d_∞) are different metric spaces, although the underlying set is the same in each case.

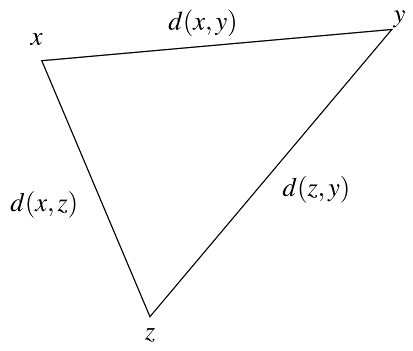


Figure 2.6: Triangle inequality for metric in the Euclidean plane.

2.5.1 Examples of Metric Spaces

Below, we give some examples of metric spaces.

Example 2.13 (Real and complex numbers $\mathbb{R}$ and $\mathbb{C}$)**.** Let the set X be either the real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$. The function

$$d(x,y) = |x-y|$$

defines a metric on X. This metric is referred to as the **usual metric** for the real/complex numbers.

Example 2.14 (Euclidean space $\mathbb{R}^n$)**.** Let X be the set of all ordered n-tuples of real numbers (i.e., $X = \mathbb{R}^n$). Then, functions (with $p \in \mathbb{R} \cup \{\infty\}$)

$$d_p(x,y) = \begin{cases} (|x_1-y_1|^p + \ldots + |x_n-y_n|^p)^{1/p} & \text{for } 1 \le p < \infty \\ \max\{|x_1-y_1|, \ldots, |x_n-y_n|\} & \text{for } p = \infty \end{cases}$$

define metrics on X, where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$.

Example 2.15 (Function space $C[a,b]$)**.** Let X be the set of all real-valued functions that are defined and continuous on the closed interval $[a,b]$. Define the function

$$d(x,y) = \max_{t \in [a,b]} |x(t) - y(t)|.$$

Then, (X,d) is a metric space.

Example 2.16 (Lebesgue space $L^p(\mathbb{R})$)**.** Let X be the set of complex-valued functions defined on $\mathbb{R}$ such that $\int_{-\infty}^{\infty} |x(t)|^p \, dt < \infty$, where $p \in \mathbb{R}$ and $1 \le p < \infty$. Define the function

$$d_p(x,y) = \left(\int_{-\infty}^{\infty} |x(t) - y(t)|^p \, dt \right)^{1/p}.$$

Then, (X, d_p) is a metric space.

Example 2.17 (Sequence space $l^p(\mathbb{N})$)**.** Let X be the set of infinite sequences of complex numbers $x = (x_1, x_2, \ldots)$ such that $\left(\sum_{n\in\mathbb{N}} |x_n|^p\right)^{1/p} < \infty$, where $p \in \mathbb{R}$ and $1 \le p < \infty$. Let $y = (y_1, y_2, \ldots)$. Then,

$$d_p(x,y) = \left(\sum_{n \in \mathbb{N}} |x_n - y_n|^p \right)^{1/p}$$

defines a metric on X. When $p = 2$, we obtain the $l^2(\mathbb{N})$ space with metric

$$d_2(x,y) = \left(\sum_{n \in \mathbb{N}} |x_n - y_n|^2 \right)^{1/2}.$$

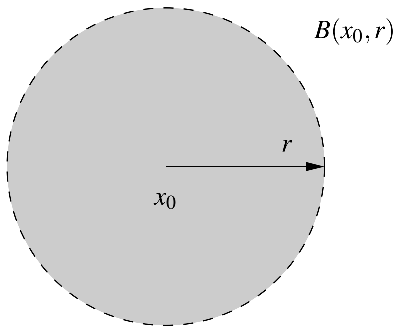


Figure 2.7: Open ball.

2.5.2 Some Topological Concepts

Since a metric space has topological structure, we can define a number of useful concepts related to this structure. We introduce several of these concepts below.

Definition 2.14 (Continuous mapping)**.** Let (X,d) and $(Y,\tilde{d})$ be metric spaces. A mapping $T : X \to Y$ is said to be **continuous at a point** $x_0 \in X$ if for every real number $\varepsilon > 0$ there exists a real number $\delta > 0$ such that

$$\tilde{d}(Tx, Tx_0) < \varepsilon \text{ for all } x \text{ satisfying } d(x,x_0) < \delta.$$

The mapping T is said to be **continuous** if it is continuous at every point in X.

Definition 2.15 (Open ball)**.** Let (X,d) be a metric space and let x_0 be an arbitrary point in (X,d). The set

$$B(x_0,r) = \{x \in X : d(x,x_0) < r\}$$

where $0 < r < \infty$ is referred to as the **open ball** with center x_0 and radius r.

Note that an open ball must have a strictly positive radius. An open ball is illustrated in Figure 2.7.

Definition 2.16 (Neighbourhood)**.** An open ball with center x_0 (i.e., $B(x_0,\delta)$) is said to be a **neighbourhood** of x_0.

Definition 2.17 (Boundary point and boundary)**.** Let (X,d) be a metric space and let S be a subset of X. A point $x \in X$ is said to be a **boundary point** of S if every neighbourhood of x contains both elements in S and elements not in S. The **boundary** of a set S, denoted $\operatorname{bdy} S$, is the set of all boundary points of S.

Example 2.18. Let $S = \{x = (x_1,x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 < 1\}$. Determine $\operatorname{bdy} S$.

Solution. We begin by observing that the set S consists of all points inside a circle of radius 1 centered at the origin, as shown in Figure 2.8(a). In other words, the set is an open ball with radius 1 centered at the origin. Therefore, $\operatorname{bdy} S = \{x = (x_1,x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 = 1\}$, as shown in Figure 2.8(b). □

Definition 2.18 (Point of adherence)**.** Let S be a subset of a metric space (X,d). A point x in (X,d) is said to be a **point of adherence** of S if each neighbourhood N of x contains a point in S.

It is important to emphasize that a point of adherence of S does not necessarily need to be in S. It also follows from the above definition that every point in S is a point of adherence of S.

Example 2.19. Let $S = \{x = (x_1,x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 < 1\}$. Then, the set of all points of adherence of S is $\{x = (x_1,x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\}$.

Definition 2.19 (Open set)**.** A set S in a metric space (X,d) is said to be **open** if S contains a neighbourhood of each one of its points (i.e., for each $x \in S$, there is a neighbourhood N with $N \subset S$).

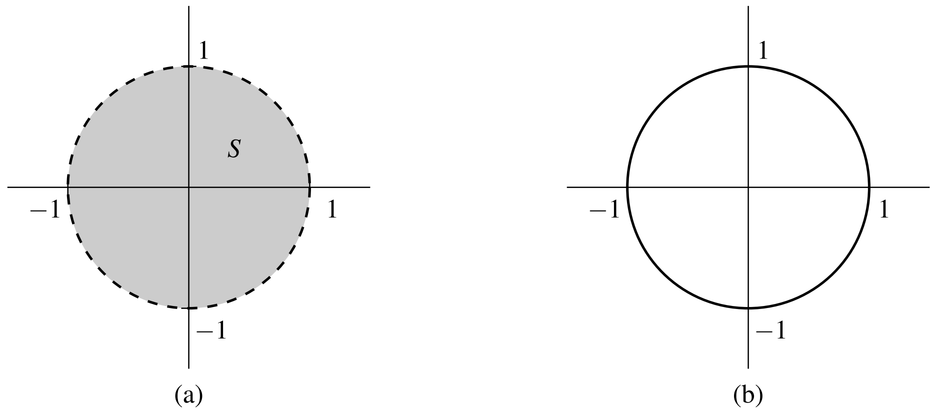


Figure 2.8: Boundary example. (a) The set S and (b) its boundary.

Informally, a subset S of a metric space (X,d) is open if S does not contain any of its boundary points (i.e., $S \cap \operatorname{bdy} S = \emptyset$). It follows from the above definition that the empty set $\emptyset$ and X itself are always open.

Example 2.20. Determine whether the subset $S = (0,1)$ of $\mathbb{R}$ is open.

Solution. The set S contains a neighbourhood of each of its points. Therefore, S is open. □

Example 2.21. Determine whether the subset $S = [2,3)$ of $\mathbb{R}$ is open.

Solution. The set S does not contain a neighbourhood of the point $2 \in S$. Therefore, S is not open. □

Definition 2.20 (Closed set)**.** Let (X,d) be a metric space. A subset $S \subset X$ is said to be **closed** if the complement of S (in X) is open.

Informally, a subset S of a metric space (X,d) is closed if S contains all of its boundary points (i.e., $\operatorname{bdy} S \subset S$).

Example 2.22. Determine whether the subset $S = [0,1]$ of $\mathbb{R}$ is closed.

Solution. The complement S^c of S in $\mathbb{R}$ is $(-\infty,0) \cup (1,\infty)$. Clearly, S^c contains a neighbourhood of each of its points. Therefore, S^c is open, implying that S is closed. □

Example 2.23. Determine whether the subset $S = [0,1)$ of $\mathbb{R}$ is closed.

Solution. The complement S^c of S in $\mathbb{R}$ is $(-\infty,0) \cup [1,\infty)$. Now, we observe that S^c does not contain a neighbourhood about its point 1. Therefore, S^c is not open, implying that S is not closed. □

Definition 2.21 (Closure)**.** Let S be a set in a metric space (X,d). The set of all points of adherence of S is said to be the **closure** of S (in X) and is denoted $\operatorname{clos} S$.

Informally, the closure of S is the union of S with its boundary points (i.e., $\operatorname{clos} S = S \cup \operatorname{bdy} S$). The closure of a set is always closed. The closure of a closed set is itself.

Example 2.24. The closure of $(0,1)$ in $\mathbb{R}$ is $[0,1]$.

Example 2.25. Let $S = [0,1) \subset \mathbb{R}$. Determine whether S is open and/or closed. Find the closure of S (in $\mathbb{R}$).

Solution. There is no neighbourhood of the point $0 \in S$ that is contained in S. Therefore, S is not open. The complement S^c of S in $\mathbb{R}$ is $(-\infty,0) \cup [1,\infty)$. There is no neighbourhood of the point $1 \in S^c$ that is contained in S^c. Therefore, S^c is not open, implying that S is not closed. The set of all points of adherence of S is given by $[0,1) \cup \{0,1\} = [0,1]$. Therefore, $\operatorname{clos} S = [0,1]$. □

Example 2.26. Let $S = \{z \in \mathbb{C} : |z| \leq 1\}$. Determine whether S is open and/or closed. Find the closure of S.

Solution. First, we consider whether S is open. Since any point in $\{z \in \mathbb{C} : |z| = 1\} \subset S$ does not have a neighbourhood contained in S, S is not open. Now, we consider whether S is closed. We have $S^c = \{z \in \mathbb{C} : |z| > 1\}$. Since all points in S^c have a neighbourhood contained in S^c, S^c is open. Therefore, S is closed. The set of all points of adherence of S is $\{z \in \mathbb{C} : |z| \le 1\}$. Thus, $\operatorname{clos} S = S$. □

Definition 2.22 (Dense set)**.** A subset S of a metric space (X,d) is said to be **dense** in X if $\operatorname{clos} S = X$.

From the above definition, it follows that the set X is always dense in (X,d). Intuitively speaking, if a subset S of (X,d) is dense in X, then for each point $x \in X$, there are points in S arbitrarily close to x. In other words, we can approximate points in X with arbitrary accuracy using points in S.

Example 2.27. The set of rational numbers $\mathbb{Q}$ is dense in $\mathbb{R}$.

Example 2.28. The set of infinitely differentiable real-valued functions defined on $I \subset \mathbb{R}$ is dense in $L^2(I)$.

Example 2.29. The set of bounded functions defined on $I \subset \mathbb{R}$ is dense in $L^p(I)$ for $1 \le p < \infty$.

Consider the subset S of the metric space (X,d). Density depends not only on the subset S but also on the "universal" set X being considered. That is, a set S may be dense in one set but not in another, as illustrated by the example below.

Example 2.30. The set of integers $\mathbb{Z}$ is dense in $\mathbb{Z}$ but not dense in $\mathbb{R}$.

2.5.3 Convergent Sequences

A metric space adds topological structure to a set by way of a metric. A metric is beneficial as it provides a measure of distance between elements in a set. Once we have a well defined measure of distance, we can begin to define other useful concepts that utilize this measure. One such notion is that of convergence as we introduce below.

Definition 2.23 (Convergent sequence, limit)**.** A sequence $\{x_n\}$ in a metric space (X,d) is said to be **convergent** if there is a point x in (X,d) with the property that for each number $\varepsilon > 0$ there is an integer N such that $d(x_n,x) < \varepsilon$ whenever $n \ge N$. The point x is called the **limit** of the sequence $\{x_n\}$, which we denote as

$$\lim_{n\to\infty} x_n = x.$$

A sequence that is not convergent is said to be **divergent**.

It is important to emphasize that, in the above definition, the limit x must be in X. We illustrate this point with the following example.

Example 2.31. Let X be the open interval $(0,1)$ on $\mathbb{R}$ with the usual (absolute value) metric. The sequence $\{\frac{1}{n}\} = \{1, \frac{1}{2}, \frac{1}{3}, \ldots\}$ is not convergent since the value that the sequence is trying to approach, namely 0, is not in X.

A sequence that is convergent in one space may not be convergent in another. That is, convergence is not a property of a sequence itself. Convergence also depends on the space in which a sequence lies. For example, the sequence $\{3, 3.1, 3.14, 3.141, 3.1415, \ldots\}$, which corresponds to increasingly better approximations of π, is convergent in $\mathbb{R}$ but not in $\mathbb{Q}$ (with the usual metrics), since $\pi \in \mathbb{R}$ but $\pi \notin \mathbb{Q}$.

Another concept related to the notion of convergence is that of a Cauchy sequence as defined below.

Definition 2.24 (Cauchy sequence)**.** A sequence $\{x_n\}$ in a metric space (X,d) is said to be **Cauchy** if for each $\varepsilon > 0$ there exists an N such that $d(x_m,x_n) < \varepsilon$ for any choice of $m,n > N$. (Note that N depends on ε.)

In less formal terms, a Cauchy sequence can be thought of as a sequence that is trying to converge. The elements in such a sequence become closer and closer in value as the sequence index increases. In this sense, the sequence is trying to approach some limit.

Example 2.32. Show that the sequence $\{x_n\} = \{\frac{1}{n}\} = \{1, \frac{1}{2}, \frac{1}{3}, \ldots\}$ (in $\mathbb{R}$ with the usual metric) is Cauchy.

Solution. Let $\varepsilon > 0$. Let $N = 2/\varepsilon$. If $n, m > N$, then

$$x_n = 1/n < 1/N = \varepsilon/2 \quad \text{and}$$
$$x_m = 1/m < 1/N = \varepsilon/2.$$

So, for all $n, m > N$, we have

$$\begin{aligned} d(x_m, x_n) &\leq d(x_m, 0) + d(0, x_n) \\ &= |1/m| + |1/n| \\ &= 1/n + 1/m \\ &= x_n + x_m \\ &< \varepsilon/2 + \varepsilon/2 \\ &= \varepsilon. \end{aligned}$$

Thus, we have that $d(x_m, x_n) < \varepsilon$ for all $m, n > N$, implying that the sequence is Cauchy. □

Example 2.33. Define the sequence $(x_n)_{n\in\mathbb{N}}$ such that x_n is the first $(n+1)$ digits in the decimal expansion of $\sqrt{2}$. That is, we have

$$x_1 = 1.4, \quad x_2 = 1.41, \quad x_3 = 1.414, \quad x_4 = 1.4142, \quad \ldots.$$

Show that $(x_n)_{n\in\mathbb{N}}$ (in $\mathbb{R}$ with the usual metric) is a Cauchy sequence.

Solution. Consider the quantity $d(x_m, x_n) = |x_m - x_n|$. If m and n are greater than N, then x_m and x_n have at least the first $(N+1)$ digits in common. Consequently, we have

$$d(x_m, x_n) < 10^{-N} \quad \text{for all } m, n > N.$$

Thus, for every $\varepsilon > 0$, we have

$$d(x_m, x_n) < \varepsilon = 10^{-N} \quad \text{for all } m, n > N,$$

where $N = -\log \varepsilon$. Therefore, the sequence $(x_n)_{n\in\mathbb{N}}$ is Cauchy. □

Example 2.34. Define the sequence

$$\{x_n\} = \operatorname{round}(na)/n$$

where a is a real constant and $\operatorname{round}(x)$ denotes the nearest integer to x. Show that the sequence $\{x_n\}$ is Cauchy.

Solution. Consider $d(x_n, a)$. We can write

$$\begin{aligned} d(x_n, a) &= \left| \frac{\operatorname{round}(na)}{n} - a \right| \\ &= \left| \frac{\operatorname{round}(na) - na}{n} \right| \\ &= \frac{|\operatorname{round}(na) - na|}{n}. \end{aligned}$$

Now we observe that $|\operatorname{round}(na) - na| \leq \frac{1}{2}$. So, we have

$$d(x_n, a) \leq \tfrac{1}{2n}.$$

Using the triangle inequality, we can write

$$\begin{aligned} d(x_m,x_n) &\leq d(x_m,a)+d(a,x_n) \\ &= \tfrac{1}{2m}+\tfrac{1}{2n}. \end{aligned}$$

Consider the preceding inequality for $m,n > N$ where $N = 1/\varepsilon$ (and $\varepsilon > 0$). Observing that

$$\tfrac{1}{2m} < \tfrac{1}{2N} = \varepsilon/2 \quad \text{and} \quad \tfrac{1}{2n} < \tfrac{1}{2N} = \varepsilon/2,$$

we can conclude

$$d(x_m,x_n) < \varepsilon/2+\varepsilon/2 = \varepsilon.$$

Thus, the sequence is Cauchy. □

As one might suspect, there is an important relationship between the convergent and Cauchy properties of sequences, as stated by the theorem below.

Theorem 2.4 (Convergent sequence)**.** *Every convergent sequence in a metric space is a Cauchy sequence.*

Proof. If $\lim_{n\to\infty} x_n = x$, then for each $\varepsilon > 0$ there exists an N such that

$$d(x_n,x) < \varepsilon/2 \quad \text{for all } n > N.$$

So, by the triangle inequality we have (for $m,n > N$)

$$d(x_m,x_n) \leq d(x_m,x)+d(x,x_n) < \frac{\varepsilon}{2}+\frac{\varepsilon}{2} = \varepsilon.$$

Thus, we have that $d(x_m,x_n) < \varepsilon$ for all $m,n > N$. Therefore, the sequence $\{x_n\}$ is Cauchy. □

The converse of the above theorem is not necessarily true. That is, not every Cauchy sequence is a convergent sequence. Essentially, a Cauchy sequence can fail to converge due to a "hole" in the metric space in which the sequence lies. Some metric spaces are such that they do not have this problem. We give such spaces a special name as defined below.

Definition 2.25 (Completeness)**.** A metric space (X,d) is said to be **complete** if each Cauchy sequence in (X,d) is a convergent sequence in (X,d) (i.e., has a limit which is an element of X). A metric space that is not complete is said to be **incomplete**.

To use the analogy from above, a complete metric space has no holes. For this reason, a Cauchy sequence cannot fail to converge in such a space.

By virtue of the above definition, in a complete metric space, the Cauchy and convergent properties are equivalent. This equivalence has many important implications. For example, when working with complete metric spaces, we can show that a sequence is convergent simply by showing that it is Cauchy. In practice, this is usually much easier to do than proving convergence directly. To show that a sequence is Cauchy, we need only consider the sequence and the relevant metric. We need not worry about the limit of the sequence, if it even exists in the first place. In this way, we can show that a sequence in a complete metric space is convergent without ever having to find its limit (which may be difficult to do).

Example 2.35 (Real numbers $\mathbb{R}$)**.** The real line $\mathbb{R}$ with the usual (absolute value) metric is a complete metric space. Any closed interval $[a,b]$ on $\mathbb{R}$ with the usual metric is a complete metric space. Any open interval (a,b) on $\mathbb{R}$ with the usual metric is an incomplete metric space.

Example 2.36 (Complex numbers $\mathbb{C}$)**.** The complex plane $\mathbb{C}$ with the usual (magnitude) metric is a complete metric space.

Example 2.37 (Rational numbers $\mathbb{Q}$)**.** The metric space X consisting of the set of rational numbers (i.e., $X = \mathbb{Q}$) with the usual absolute value metric (i.e., $d(x,y) = |x-y|$) is not complete. Consider, for example, the sequence $\{3, 3.1, 3.14, 3.141, 3.1415, \ldots\}$. This sequence is a Cauchy sequence, but it is not convergent since π is not a rational number.

Example 2.38 (Integers $\mathbb{Z}$)**.** Show that the set X of all integers (i.e., $X = \mathbb{Z}$) with the metric d defined by $d(x,y) = |x-y|$ is a complete metric space.

Solution. Consider an arbitrary Cauchy sequence $\{x_n\}$ in X. For each $\varepsilon > 0$, there exists N such that

$$d(x_m, x_n) < \varepsilon \quad \text{for all } m,n > N.$$

Consider the choice of $\varepsilon = 1$. There must exist N such that

$$d(x_m, x_n) = |x_m - x_n| < 1 \quad \text{for all } m,n > N.$$

This implies, however, that $x_m = x_n$ for all $m,n > N$. Thus, $\{x_n\}$ is of the form

$$(x_1, x_2, \ldots, x_N, x_{N+1}, x_{N+1}, x_{N+1}, \ldots).$$

Clearly, such a sequence converges to a limit in X since $x_{N+1} \in X$. Since $\{x_n\}$ was chosen as an arbitrary Cauchy sequence, this shows that every Cauchy sequence is convergent. Hence, the metric space (X,d) is complete. □

Example 2.39 (Continuous functions)**.** Let X be the set of all continuous real-valued functions on $A = [-1,1]$ and define the metric on X as

$$d(x,y) = \int_{-1}^{1} [x(t) - y(t)]^2 dt.$$

Show that the metric space (X,d) is not complete.

Solution. Consider the sequence $\{f_n\}$ given by

$$f_n(t) = \begin{cases} 0 & \text{for } t < -\frac{1}{n} \\ \frac{1}{2}nt + \frac{1}{2} & \text{for } -\frac{1}{n} \le t \le \frac{1}{n} \\ 1 & \text{for } t > \frac{1}{n}. \end{cases}$$

A plot of the corresponding function is shown in Figure 2.9. We have

$$\begin{aligned} d(f_m, f_n) &= \begin{cases} \int_{-1/n}^{-1/m} (\frac{1}{2}nt + \frac{1}{2})^2 dt + \int_{-1/m}^{1/m} [(\frac{1}{2}mt + \frac{1}{2}) - (\frac{1}{2}nt + \frac{1}{2})]^2 dt + \int_{1/m}^{1/n} [1 - (\frac{1}{2}nt + \frac{1}{2})]^2 dt & \text{for } m > n \\ \int_{-1/m}^{-1/n} (\frac{1}{2}mt + \frac{1}{2})^2 dt + \int_{-1/n}^{1/n} [(\frac{1}{2}mt + \frac{1}{2}) - (\frac{1}{2}nt + \frac{1}{2})]^2 dt + \int_{1/n}^{1/m} [(\frac{1}{2}mt + \frac{1}{2}) - (1)]^2 dt & \text{for } m < n \end{cases} \\ &= \begin{cases} \frac{(n-m)^2}{6m^2 n} & \text{for } m > n \\ \frac{(m-n)^2}{6n^2 m} & \text{for } m < n. \end{cases} \end{aligned}$$

Thus, as $m,n \to \infty$, we have $d(f_m, f_n)$ goes to zero. Therefore, the sequence is Cauchy. The limit function $f(t)$ is given by

$$f(t) = \begin{cases} 0 & \text{for } t < 0 \\ \frac{1}{2} & \text{for } t = 0 \\ 1 & \text{for } t > 0. \end{cases}$$

Since $f(t)$ is not continuous, it is not in X. So, we have found a Cauchy sequence in X that does not have a limit in X. Therefore, the metric space X is not complete. □

We can characterize closed sets using the notion of a limit as specified by the theorem below.

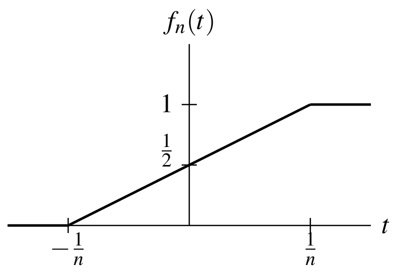


Figure 2.9: Function.

Theorem 2.5. *Let S be a nonempty subset of a metric space (X,d). Then, $x \in \operatorname{clos} S$ if and only if there is a sequence (x_n) in S such that $\lim_{n\to\infty} x_n = x$.*

The above theorem is helpful as it provides some insight into what elements are added to a set by the closure operation. The above theorem can also be used to show the result below.

Theorem 2.6 (Closed set theorem)**.** *A set S in a metric space (X,d) is a closed set if and only if every convergent sequence $\{x_n\}$ with $\{x_n\} \subset S$ has its limit in S.*

Theorem 2.7 (Continuity and convergence)**.** *Let X and Y be metric spaces. The mapping $T : X \to Y$ is continuous if and only if*

$$T\left(\lim_{n\to\infty} x_n\right) = \lim_{n\to\infty} T(x_n)$$

for every convergent sequence $(x_n)_{n\in\mathbb{N}}$ in X. That is, a mapping T is continuous if and only if it preserves convergent sequences.

The above theorem essentially says that a continuous mapping preserves limits. So, the order of a limit and continuous mapping can be interchanged.

2.6 Vector Spaces

In the case of metric spaces, we impose topological structure on a set. By imposing a different kind of structure on a set, we can obtain an object known as a vector space. We formally define this object below.

Definition 2.26 (Vector space)**.** A **vector space** (or linear space) over a scalar field F (such as $\mathbb{R}$ or $\mathbb{C}$) is a nonempty set V, together with two algebraic operations,

1. a mapping $(x,y) \mapsto x+y$ from $V \times V$ into V called **vector addition** and
2. a mapping $(a,x) \mapsto ax$ from $F \times V$ into V called **scalar multiplication**,

for which the following conditions hold:

1. for all $x,y \in V$, $x+y \in V$ (closure under vector addition);
2. for all $x \in V$ and all $a \in F$, $ax \in V$ (closure under scalar multiplication);
3. for all $x,y \in V$, $x+y = y+x$ (commutativity of vector addition);
4. for all $x,y,z \in V$, $(x+y)+z = x+(y+z)$ (associativity of vector addition);
5. for all $x \in V$ and all $a,b \in F$, $(ab)x = a(bx)$ (associativity of scalar multiplication);
6. for all $x \in V$ and all $a,b \in F$, $(a+b)x = ax+bx$ (distributivity of scalar sums);

7. for all $x, y \in V$ and all $a \in F$, $a(x+y) = ax + ay$ (distributivity of vector sums);

8. there exists $0 \in V$ such that $x + 0 = x$ for all $x \in V$ (additive identity);

9. for all $x \in V$, there exists a $(-x) \in V$ such that $x + (-x) = 0$ (additive inverse); and

10. for all $x \in V$, $1x = x$, where 1 denotes the multiplicative identity of the field F (scalar multiplication identity).

We denote the above vector space $(V, F, +, \cdot)$ or simply V when the other parameters are clear from the context.

We use the symbol 0 to denote the zero scalar as well as the zero vector. This should not normally be a cause of confusion, however, as the intended meaning should be clear from the context.

A vector space over the field of real numbers is called a **real vector space**. A vector space over the field of complex numbers is called a **complex vector space**.

A vector space is a set with additional structure defined by means of vector addition and scalar multiplication operations. This additional structure is called **algebraic structure**. Note that a vector space lacks any topological structure (such as that imposed by a metric).

2.6.1 Examples of Vector Spaces

Below, we give some examples of vector spaces.

Example 2.40 (Euclidean space $\mathbb{R}^n$)**.** The real Euclidean vector space $\mathbb{R}^n$ is the set V of all ordered n-tuples of real numbers together with the scalar field $F = \mathbb{R}$ for which vector addition and scalar multiplication are defined, respectively, as

$$\begin{aligned} x + y &= (x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n) \text{ and} \\ ax &= (ax_1, ax_2, \cdots, ax_n), \end{aligned}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are in V and $a \in F$.

Example 2.41 (Space $\mathbb{C}^n$)**.** In an analogous way to the previous example, we can also define the space $\mathbb{C}^n$. This space is the set V of all ordered n-tuples of complex numbers and the scalar field $F = \mathbb{C}$ with vector addition and scalar multiplication defined as in the previous example.

Example 2.42 (Function space $C[a,b]$)**.** Let V be the set of all continuous real-valued functions defined on $[a,b]$. Then, the vector space $C[a,b]$ is the set V and scalar field $\mathbb{R}$ with vector addition and scalar multiplication defined, respectively, as

$$\begin{aligned} (x+y)(t) &= x(t) + y(t) \text{ and} \\ (ax)(t) &= ax(t), \end{aligned}$$

where $x, y \in V$ and $a \in \mathbb{R}$. Since the scalar field is $\mathbb{R}$, this is an example of a real vector space.

Example 2.43 (Lebesgue space $L^2(\mathbb{R})$)**.** Let V be the set of all measurable complex-valued functions $x(t)$ defined on $\mathbb{R}$ such that $\int_{-\infty}^{\infty} |x(t)|^2 \, dt < \infty$. Then, the complex vector space $L^2(\mathbb{R})$ is the set V and scalar field $F = \mathbb{C}$ with vector addition and scalar multiplication defined, respectively, as

$$\begin{aligned} (x+y)(t) &= x(t) + y(t) \text{ and} \\ (ax)(t) &= ax(t), \end{aligned}$$

where $x, y \in V$ and $a \in F$. In a similar fashion, we can also define a real vector space $L^2(\mathbb{R})$. In this case, the elements of V are real-valued functions (instead of complex-valued functions) and the field F is $\mathbb{R}$ (instead of $\mathbb{C}$).

Example 2.44 (Sequence space $l^2(\mathbb{Z})$)**.** Let V be the set of all complex sequences $x = (\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$ such that $\sum_{n\in\mathbb{Z}} |x_n|^2 < \infty$. Then, the complex vector space $l^2(\mathbb{Z})$ is the set V and scalar field $F = \mathbb{C}$ with vector addition and scalar multiplication defined, respectively, as

$$\begin{aligned} x + y &= (\ldots, x_{-2} + y_{-2}, x_{-1} + y_{-1}, x_0 + y_0, x_1 + y_1, x_2 + y_2, \ldots) \text{ and} \\ ax &= (\ldots, ax_{-2}, ax_{-1}, ax_0, ax_1, ax_2, \ldots), \end{aligned}$$

where $x = (\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$ and $y = (\ldots, y_{-2}, y_{-1}, y_0, y_1, y_2, \ldots)$ are in V and $a \in F$. In a similar way, we can also define a real vector space $l^2(\mathbb{Z})$. In this case, the elements of V are real sequences (instead of complex sequences) and the field F is $\mathbb{R}$ (instead of $\mathbb{C}$).

2.6.2 Subspaces

A nonempty subset S of a vector space V over F is said to be a **vector subspace** of V if the following conditions hold:

1. $x + y \in S$ for all $x, y \in S$ (closure under vector addition); and
2. $ax \in S$ for all $x \in S$ and all $a \in F$ (closure under scalar multiplication).

One can readily confirm that a subspace of a vector space is itself a vector space. A subspace S of the vector space V is said to be **proper** if $S \neq V$ and **improper** if $S = V$.

Example 2.45. Let V be the vector space of all polynomials over $\mathbb{R}$ with the usual vector addition and scalar multiplication operations. The set S of constant functions is a subset of V. One can easily confirm that S is closed under vector addition and scalar multiplication. Therefore, S is a subspace of V.

2.6.3 Linear Transformations

Definition 2.27 (Linear transformation)**.** A transformation T of a vector space V into a vector space W, where V and W have the same scalar field F, is said to be a **linear transformation** if

1. for all $x \in V$ and all $a \in F$, $T(ax) = aT(x)$ (homogeneity); and
2. for all $x, y \in V$, $T(x + y) = T(x) + T(y)$ (additivity).

Definition 2.28 (Null space and range space)**.** The **null space** of a linear transformation $T : V \to W$, denoted $N(T)$, is the subset of V given by

$$N(T) = \{x \in V : Tx = 0\}$$

(i.e., the set of all vectors mapped to the zero vector under the transformation T). The **range space** of a linear transformation $T : V \to W$, denoted $R(T)$, is defined as

$$R(T) = \{y = Tx : x \in V\}$$

(i.e., the set of vectors produced by applying T to each of the elements of V).

Theorem 2.8. *A transformation T of V into W, where V and W are vector spaces over the same scalar field F, is linear if and only if*

$$T(a_1x_1 + a_2x_2 + \ldots + a_nx_n) = a_1T(x_1) + a_2T(x_2) + \ldots + a_nT(x_n)$$

for all $x_1, x_2, \ldots, x_n \in V$, all $a_1, a_2, \ldots, a_n \in F$, and all finite n.

The condition in the above theorem is referred to as the principle of **superposition**.

Definition 2.29 (Projection)**.** A linear transformation P of a vector space V into itself is said to be a **projection** if $P^2 = P$ (i.e., P is **idempotent**).

2.6.4 Linear Independence

Definition 2.30 (Collinear vectors)**.** If x and y are nonzero vectors and $x = ay$ for some scalar a, then x and y are said to be **collinear**.

Example 2.46. Consider the vectors $x = (1,4,-3)$ and $y = (-2,-8,6)$ in $\mathbb{R}^3$. Determine whether x and y are collinear.

Solution. By inspection, we have $y = -2x$. Therefore, x and y are collinear. □

Definition 2.31 (Linear combination)**.** Let $\{x_1, x_2, \ldots, x_n\}$ be a finite set of elements in a vector space V. A vector $x \in V$ is said to be a **linear combination** of vectors $x_1, x_2, \ldots, x_n$ if there exist scalars $a_1, a_2, \ldots, a_n$ such that

$$x = a_1x_1 + a_2x_2 + \ldots + a_nx_n.$$

With regards to the above definition, it is important to emphasize that a linear combination has only a *finite* number of terms. This is true regardless of whether the vector space in question is finite- or infinite-dimensional. Furthermore, it is not simply by coincidence that we define a linear combination in this way. The fundamental issue here is that, in the context of a vector space, an infinite series is not a meaningful construct. Any time that we deal with an infinite series, we must address the issue of convergence. Since a vector space does not have any topological structure (like that imposed by a metric), we cannot measure closeness of vectors, and consequently, we have no way to define convergence for an infinite series.

Example 2.47. Any element of $\mathbb{R}^3$ is a linear combination of the vectors $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$.

Example 2.48. Let V be the space of all polynomials of degree n. Then, any vector $x \in V$ is a linear combination of the monomials $1, t, t^2, \ldots, t^n$.

Example 2.49. Let V be the vector space $\mathbb{R}^4$. Determine whether the vector $x \in V$ is a linear combination of elements in the set $S \subset V$, where $x = (2,-5,-2,9)$ and $S = \{(-1,2,1,1),(3,-1,-2,4),(-5,-1,3,2)\}$.

Solution. We must determine if there exist scalars a_1, a_2, a_3 such that

$$a_1(-1,2,1,1) + a_2(3,-1,-2,4) + a_3(-5,-1,3,2) = (2,-5,-2,9).$$

Thus, we must solve the overdetermined system of equations

$$\begin{bmatrix} -1 & 3 & -5 \\ 2 & -1 & -1 \\ 1 & -2 & 3 \\ 1 & 4 & 2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -5 \\ -2 \\ 9 \end{bmatrix}.$$

This system of equations has the solution $a_1 = -1$, $a_2 = 2$, and $a_3 = 1$. Therefore, x is a linear combination of elements of S. In particular, we have

$$x = -(-1,2,1,1) + 2(3,-1,-2,4) + (-5,-1,3,2).$$

□

Example 2.50. Let V be the vector space $l^2(\mathbb{N})$. Define $x \in V$ and $S \subset V$ as $x = \{1/n^2\}$ and $S = \{\varphi_1, \varphi_2, \ldots\}$, where $\varphi_n = (\delta[n-1], \delta[n-2], \ldots)$. Determine whether x is a linear combination of elements in S.

Solution. Obviously, we can express x as

$$\begin{aligned} x &= \sum_{n=1}^{\infty} \tfrac{1}{n^2}\varphi_n \\ &= \varphi_1 + \tfrac{1}{4}\varphi_2 + \tfrac{1}{9}\varphi_3 + \ldots \\ &= (1,0,0,\ldots) + \tfrac{1}{4}(0,1,0,0,\ldots) + \tfrac{1}{9}(0,0,1,0,\ldots) + \ldots. \end{aligned}$$

This representation of x, however, does not correspond to a linear combination, as a linear combination must be comprised of a finite number of terms. Therefore, x is not a linear combination of elements in S. □

Definition 2.32 (Span)**.** Let A be a set in a vector space. For any nonempty subset A of a vector space V, the set of all (finite) linear combinations of vectors in A is called the **span** of A, denoted $\operatorname{span} A$.

One can show that $\operatorname{span} A$ is a subspace of V. We refer to this subspace as the space spanned by A.

Example 2.51. Let A be the subset of $\mathbb{R}^3$ given by $A = \{(1,0,0),(0,1,0)\}$. The span of A is the xy-plane.

Definition 2.33 (Linear independence)**.** A (nonempty) finite set of elements $\{x_1, x_2, \ldots, x_n\}$ in a vector space V is said to be **linearly independent** if the only set of scalars $\{a_1, a_2, \ldots, a_n\}$ satisfying the equation

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$$

is the trivial solution with $a_1 = a_2 = \ldots = a_n = 0$. An infinite set A of vectors in V is said to be linearly independent if every (nonempty) finite subset of A is linearly independent. A set that is not linearly independent is called **linearly dependent**.

It follows from the definition of linear independence that any set containing the zero vector is linearly dependent.

Example 2.52. Let V be the vector space $C[0,1]$ (as defined in Example 2.42), and let A be the infinite set $\{1, t, t^2, \ldots\}$. Determine whether A is linearly independent.

Solution. We must determine whether every finite subset of A is linearly independent. Let S be an arbitrary finite subset of A given by $S = \{t^{k_0}, t^{k_1}, \ldots, t^{k_n}\}$ where $k_0 < k_1 < \ldots < k_n$. Now, we need to consider the solution of the equation

$$a_0t^{k_0} + a_1t^{k_1} + \ldots + a_nt^{k_n} \equiv 0 \text{ for all } t \in [0,1],$$

where $a_0, a_1, \ldots, a_n$ are scalars. Since the left-hand side of the above equation is an k_nth order polynomial, it can be zero for at most k_n distinct values of t. Thus, the only way this polynomial can be zero for all $t \in [0,1]$ is if $a_0 = a_1 = \ldots = a_n = 0$. Therefore, S is linearly independent. Since S was chosen as an arbitrary finite subset of A, this shows that A is also linearly independent. $\square$

2.6.5 Bases and Dimensionality

Definition 2.34 (Hamel basis)**.** A set A in a vector space V is said to be a **Hamel basis** of V if A is linearly independent and $\operatorname{span} A = V$.

The Hamel basis is a generalization of the familiar concept of a coordinate system. One can show that all Hamel bases of a vector space have the same cardinal number.

Example 2.53. The set $\{(1,0,0),(0,1,0),(0,0,1)\}$ is a Hamel basis for $\mathbb{R}^3$.

Example 2.54. Let V be the vector space of polynomials of degree less than or equal to two including the zero polynomial. The set $\{1, t, t^2\}$ is a Hamel basis of V. Another Hamel basis of V is the set $\{2, t-t^2, t\}$.

Definition 2.35 (Dimension)**.** The cardinal number of any Hamel basis of a vector space V is said to be the **dimension** of V, denoted $\dim V$.

The dimension of V is simply the number of linearly independent vectors required to span V. If the dimension of V is finite, we say that V is **finite dimensional**. Otherwise, we say that V is **infinite dimensional**. Of most interest to us are infinite-dimensional spaces (e.g., function and sequence spaces).

Example 2.55. The vector space $\{0\}$ has dimension zero.

Example 2.56. The vector space $\mathbb{R}^n$ has dimension n.

The dimension of a vector space depends on its field of scalars as illustrated by the following example.

Example 2.57. Over the field of complex numbers, the vector space of complex numbers has dimension 1. (A basis is $\{1\}$.) Over the field of real numbers, the vector space of complex numbers has dimension 2. (A basis is $\{1,i\}$.)

Example 2.58. Let V be the set of all polynomials of order less than or equal to n. Each element of V can be expressed as a linear combination of elements from the set $A = \{1,t,t^2,\ldots,t^n\}$. It can be shown that A is linearly independent. Thus, $\dim V = \operatorname{card} A = n+1$.

Example 2.59. Let V be the real vector space comprised of sequences of real numbers $x = (x_1,x_2,\ldots)$ such that $\sum_{n\in\mathbb{N}} |x_n|^2 < \infty$. Let $A = \{e_1,e_2,\ldots\}$, where e_n is the sequence $e_n = (\delta[n-1],\delta[n-2],\ldots)$. That is,

$$e_1 = (1,0,0,\ldots),\ e_2 = (0,1,0,0,\ldots),\ e_3 = (0,0,1,0,0,\ldots),\ \ldots.$$

It is easy to convince oneself that A is linearly independent. So one might be tempted to conclude that A is a Hamel basis for V. This, however, is not the case. Consider a sequence $y \in V$ with an infinite number of nonzero entries. One such sequence is $y = (e^{-n})$. This sequence has an expansion of the form $y = \sum_{n\in\mathbb{N}} a_n e_n$. This sum does not constitute a linear combination, since a linear combination must (by definition) have a finite number of terms. Thus, we have that $y \in V$, but $y \notin \operatorname{span} A$. So, $\operatorname{span} A \neq V$ and, consequently, A is not a Hamel basis for V. (Aside: One can show, however, that $\operatorname{span} A$ is the vector subspace of V comprised of all sequences that are nonzero on only a finite number of entries.)

2.6.6 Inner Sum and Direct Sum

Definition 2.36 (Disjoint subspaces)**.** Two vector subspaces V and W of the same dimensionality are said to be **disjoint** if $V \cap W = \{0\}$ (i.e., the only common vector between V and W is the zero vector).

Note that the disjoint qualifier applied to a vector space has a different meaning than in the context of sets (i.e., unlike disjoint sets, disjoint spaces have a common element, namely the zero vector).

Example 2.60. Consider the vector space $V = \mathbb{R}^3$. Let $A = \operatorname{span}\{(1,0,0)\}$ and $B = \operatorname{span}\{(0,1,0)\}$. Then, A and B are mutually disjoint subspaces of V.

Definition 2.37 (Inner sum)**.** If V and W are subspaces of the vector space U, then the **inner sum** of V and W, denoted $V+W$, is the space consisting of all points $x = v+w$ where $v \in V$ and $w \in W$.

Note that the inner sum of two subspaces is not generally the same as their union (i.e., $V+W \neq V \cup W$). For example, consider the subspaces V and W of $\mathbb{R}^3$ given by $V = \operatorname{span}\{(1,0,0)\}$ and $W = \operatorname{span}\{(0,1,0)\}$. We have that the vector $(1,1,0)$ is in $V+W$ but not $V \cup W$.

Let V and W be subspaces of the vector space U. If $U = V+W$ and V and W are disjoint, W is called the **algebraic complement** of V in U. (Similarly, V is the algebraic complement of W in U.)

Theorem 2.9 (Existence of algebraic complement)**.** *Let V be a vector subspace of a vector space U. Then, there exists another subspace W such that V and W are disjoint, and $U = V+W$ (i.e., W is the algebraic complement of V in U).*

Proof. See [10, p. 199, Theorem 4.10.3]. □

Lemma 2.2 (Uniqueness of decomposition using algebraic complements)**.** *Let V and W be subspaces of a vector space U. Then for each $x \in V+W$, there is a unique $v \in V$ and a unique $w \in W$ such that $x = v+w$ if and only if V and W are disjoint.*

Example 2.61. Let V be the vector space of all real-valued functions defined on $\mathbb{R}$. Let W_1 be the subspace of V consisting of all even functions. Let W_2 be the subspace of V consisting of all odd functions. Then, we have $V = W_1 + W_2$. Furthermore, W_1 and W_2 are disjoint. So, W_2 is the algebraic complement of W_1 in V.

Definition 2.38 (Direct sum)**.** The **direct sum** of the vector spaces V and W, denoted $V \oplus W$, is a vector space U with the underlying set $V \times W$. Thus, a point in $V \oplus W$ is an ordered pair (v,w) with $v \in V$ and $w \in W$. Vector addition is defined componentwise $(v_1,w_1)+(v_2,w_2) = (v_1+v_2, w_1+w_2)$. Scalar multiplication is defined as $a(v,w) = (av,aw)$.

Example 2.62. Consider the vector space $U = (GF(2))^3$ whose underlying set is comprised of all ordered triplets of elements from $GF(2)$. (The field $GF(2)$ consists of two elements 0 and 1. Addition is defined as the boolean-exclusive-or operation, while multiplication is defined as the boolean-and operation.) One can show that

$$V = \{(0,0,0),(1,0,0)\} \quad \text{and} \quad W = \{(0,0,0),(0,1,0)\}$$

are subspaces of U. Find $V+W$ and $V \oplus W$.

Solution. The elements of $V+W$ are formed by adding each element in V with each element in W. So, we have

$$\begin{gathered}(0,0,0)+(0,0,0) = (0,0,0),\\ (0,0,0)+(0,1,0) = (0,1,0),\\ (1,0,0)+(0,0,0) = (1,0,0), \quad \text{and}\\ (1,0,0)+(0,1,0) = (1,1,0).\end{gathered}$$

Thus, we have

$$V+W = \{(0,0,0),(0,1,0),(1,0,0),(1,1,0)\}.$$

From the definition of $V \oplus W$, we can write

$$V \oplus W = \{(0,0,0,0,0,0),(0,0,0,0,1,0),(1,0,0,0,0,0),(1,0,0,0,1,0)\}.$$

□

Clearly, $V+W$ and $V \oplus W$ are different spaces. If V and W are disjoint, however, one can show that $V+W$ and $V \oplus W$ have the same mathematical structure. That is, they are different representations of the same thing. This type of equivalence is known as an **isomorphism**. For this reason, the direct sum is often used in its isomorphic form to mean $V \oplus W = V+W$ where V and W are disjoint. In the remainder of this book, we will only be concerned with the isomorphic form of the direct sum. In passing, we note that some authors denote the inner sum of orthogonal (and disjoint) subspaces V and W as $V \oplus W$.

Example 2.63. Let V be the vector space consisting of all real-valued sequences defined on $\mathbb{Z}$ with the usual vector addition and scalar multiplication operations. Let W_1 denote the subspace of V formed by taking all of the even sequences in V. Let W_2 denote the subspace of V formed by taking all of the odd sequences in V. We have that $V = W_1 + W_2$. Since W_1 and W_2 are disjoint, we also have $V = W_1 \oplus W_2$. (In this case, we are using the isomorphic form of the direct sum.) We also have that W_2 is the algebraic complement of W_1 in V.

Example 2.64. Let V be the vector space of all polynomials defined on $\mathbb{R}$ of degree less than or equal to two (including the zero polynomial) with the usual vector addition and scalar multiplication operations. Let W_0 be the subspace of V consisting of all constant functions. Let W_1 be the subspace of V consisting of all polynomials of the form $f(t) = at$, where $a \in \mathbb{R}$. Let W_2 be the subspace of V consisting of all polynomials of the form $f(t) = at^2$, where $a \in \mathbb{R}$. One can easily see that $V = W_0 + W_1 + W_2$. Moreover, since W_0, W_1, and W_2 are mutually disjoint, we also can write $V = W_0 \oplus W_1 \oplus W_2$. (Here, we are using the isomorphic form of the direct sum.)

2.7 Normed Spaces

Metric spaces have topological structure but not algebraic structure, while vector spaces have algebraic structure but not topological structure. We can, however, define a new type of space called a normed space that has both of these types of structure. Before we can define a normed space, we must first introduce the notion of a norm.

Definition 2.39 (Norm). A function $x \mapsto \|x\|$ from a vector space V into $\mathbb{R}$ is called a **norm** on V if the following conditions are satisfied:

1. for all $x \in V$, $\|x\| \geq 0$ (nonnegativity);

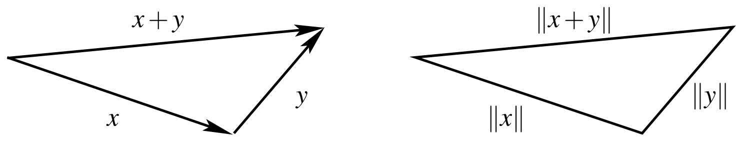


Figure 2.10: Triangle inequality for norm in the Euclidean plane.

2. $\|x\| = 0$ if and only if $x = 0$ (strict positivity);

3. for all $x \in V$ and all $a \in F$, $\|ax\| = |a| \, \|x\|$ (homogeneity); and

4. for all $x, y \in V$, $\|x+y\| \le \|x\| + \|y\|$ (triangle inequality).

A norm can be thought of as a measure of the length of a vector. The triangle inequality property is illustrated for the case of the Euclidean plane in Figure 2.10.

In passing, we note that the triangle inequality in the above definition (i.e., Definition 2.39) implies that

$$\text{for all } x, y \in V, \;\; |\|y\| - \|x\|| \le \|y - x\|. \tag{2.2}$$

This relationship can be used to prove an important property of the norm, which is given by the theorem below.

Theorem 2.10 (Continuity of the norm)**.** *The norm $\|\cdot\|$, which is a mapping from the vector space V into $\mathbb{R}$, is continuous.*

Proof. To prove that $\|\cdot\|$ is continuous, we must show that, for every real number $\varepsilon > 0$, there exists a real number $\delta > 0$ such that if $\|x - x_0\| < \delta$ then $|\|x\| - \|x_0\|| < \varepsilon$. Let $\varepsilon > 0$ and choose $\delta = \varepsilon$. Thus, we have

$$\|x - x_0\| < \varepsilon$$

and we can also write from (2.2) that

$$|\|x\| - \|x_0\|| \le \|x - x_0\|.$$

Combining the preceding two inequalities, we obtain

$$|\|x\| - \|x_0\|| \le \|x - x_0\| < \varepsilon$$

which implies

$$|\|x\| - \|x_0\|| < \varepsilon.$$

Therefore, the norm is continuous. □

The above theorem is extremely useful. Since a norm is a continuous mapping, the order of limits and norms can be interchanged.

Example 2.65. Consider the vector space $V = \mathbb{R}^2$. Define

$$\|x\| = \left[x_1^2 + x_2^2\right]^{1/2},$$

where $x = (x_1, x_2)$. Show that $\|\cdot\|$ is a norm on V.

Solution. Since the square of a real number is always nonnegative, and the square root of the sum of nonnegative numbers is nonnegative, we have $\|x\| \geq 0$ for all $x \in V$. Furthermore, $\|x\| = 0$ if and only if $x = 0$. Thus, the nonnegativity and strict positivity conditions are satisfied. Consider the homogeneity property. We can write

$$\begin{aligned} \|ax\| &= \|a(x_1, x_2)\| \\ &= \|(ax_1, ax_2)\| \\ &= \left[(ax_1)^2 + (ax_2)^2\right]^{1/2} \\ &= \left[a^2(x_1^2 + x_2^2)\right]^{1/2} \\ &= |a| \left[x_1^2 + x_2^2\right]^{1/2} \\ &= |a| \, \|x\| . \end{aligned}$$

So, the homogeneity condition is satisfied. Using the Minkowski inequality for sums (E.2), we can write

$$\begin{aligned} \|x+y\| &= \|(x_1 + y_1, x_2 + y_2)\| \\ &= \left[(x_1+y_1)^2 + (x_2+y_2)^2\right]^{1/2} \\ &\leq \left[x_1^2 + x_2^2\right]^{1/2} + \left[y_1^2 + y_2^2\right]^{1/2} \\ &= \|x\| + \|y\| . \end{aligned}$$

Thus, the triangle inequality holds. Hence, $\|\cdot\|$ is a norm on V. □

Example 2.66. Consider the vector space $V = \mathbb{R}^2$. Define

$$\|x\| = |x_1|^2 + |x_2|^2 ,$$

where $x = (x_1, x_2)$. Show that $\|\cdot\|$ is not a norm on V.

Solution. First, we have that

$$|x_1|^2 + |x_2|^2 \geq 0 \quad \text{and}$$
$$|x_1|^2 + |x_2|^2 = 0 \Leftrightarrow x_1 = x_2 = 0.$$

Thus, $\|\cdot\|$ satisfies the nonnegativity and strict positivity conditions. Consider the homogeneity condition. We can write

$$\begin{aligned} \|ax\| &= |ax_1|^2 + |ax_2|^2 \\ &= |a|^2 |x_1|^2 + |a|^2 |x_2|^2 \\ &= |a|^2 (|x_1|^2 + |x_2|^2) \\ &= |a|^2 \|x\| . \end{aligned}$$

Therefore, $\|\cdot\|$ does not satisfy the homogeneity condition. Consider the triangle inequality. Suppose that $x = (1,0)$ and $y = (1,0)$. We have

$$\|x\| = |1|^2 + |0|^2 = 1,$$
$$\|y\| = |1|^2 + |0|^2 = 1, \quad \text{and}$$
$$\|x+y\| = |2|^2 + |0|^2 = 2^2 = 4.$$

So, we have $\|x+y\| > \|x\| + \|y\|$. Hence, the triangle inequality does not generally hold. Since $\|\cdot\|$ satisfies neither the homogeneity nor triangle inequality condition, $\|\cdot\|$ is not a norm on V. □

Keeping the definition of a norm (from above) in mind, we are now in a position to define a normed space.

Definition 2.40 (Normed space)**.** A vector space V with a norm $\|\cdot\|$ defined on V is called a **normed space**, and is denoted $(V, \|\cdot\|)$ or simply V when the norm is implied from the context.

The norm on V defines a metric d on V which is given by

$$d(x,y) = \|x-y\|$$

and is called the **induced metric** (i.e., the metric induced by the norm). Thus, a normed space is also a metric space with the above induced metric. Thus, a normed space has both topological and algebraic structure.

Theorem 2.11 (Induced metric)**.** *If $\|\cdot\|$ defines a norm on the normed space V, then a metric d on V is given by*

$$d(x,y) = \|x-y\|.$$

Proof. The nonnegativity property of d can be deduced from the nonnegativity property of $\|\cdot\|$. That is, we have

$$d(x,y) = \|x-y\| \geq 0.$$

The strict positivity property of d can be deduced from the strict positivity property of $\|\cdot\|$. That is, we have

$$d(x,y) = \|x-y\| = 0 \Leftrightarrow x-y=0 \Leftrightarrow x=y.$$

The symmetry property of d can be deduced using the homogeneity property of $\|\cdot\|$. In particular, we have

$$d(x,y) = \|x-y\| = \|-(x-y)\| = \|y-x\| = d(y,x).$$

Lastly, the triangle inequality property of d can be deduced using the triangle inequality property of $\|\cdot\|$ as follows:

$$\begin{aligned} d(x,y) &= \|x-y\| \\ &= \|(x-z)-(y-z)\| \\ &\leq \|x-z\| + \|-(y-z)\| \\ &= \|x-z\| + \|z-y\| \\ &= d(x,z)+d(z,y). \end{aligned}$$

Thus, d satisfies the triangle inequality. □

Many norms can be defined on a given vector space V. Different norms on the same vector space V yield different normed spaces. Each norm defines a new normed space and different metric on V.

Some vector spaces are traditionally equipped with standard norms. If no norm is explicitly specified, it is understood that the standard norm is to be employed.

Definition 2.41 (Banach space)**.** A **Banach space** is a complete normed space (complete in the metric induced by the norm).

2.7.1 Infinite Series

As discussed previously, in a purely vector space context, an infinite series is not a meaningful construct—the difficulty being that vector spaces lack the topological structure necessary to define the convergence of such a series. The situation is different, however, in the case of normed spaces. Such spaces have a well defined notion of convergence which they inherit from their underlying metric space.

Consider an infinite series of the form

$$\sum_{k\in\mathbb{N}} x_k. \tag{2.3}$$

If $(x_k)_{k\in\mathbb{N}}$ is a sequence in a normed space V, we can associate $(x_k)_{k\in\mathbb{N}}$ with a sequence $(s_n)_{n\in\mathbb{N}}$ of partial sums given by

$$s_n = \sum_{k=1}^{n} x_k.$$

If the sequence $(s_n)_{n\in\mathbb{N}}$ is convergent with limit S, then the infinite series in (2.3) is said to converge to S. In other words, we say that the sum of the above infinite series is S if $\|s_n - S\| \to 0$ as $n \to \infty$. Thus, an infinite series (in a normed space) can be defined in a meaningful way.

2.7.2 Examples of Normed Spaces

Below, we give some examples of normed spaces.

Example 2.67 (Euclidean space $\mathbb{R}^n$)**.** We define

$$\|x\|_p = \begin{cases} \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p} & \text{for } 1 \le p < \infty \\ \max\{|x_1|, \ldots, |x_n|\} & \text{for } p = \infty. \end{cases} \tag{2.4}$$

One can easily show that, for $1 \le p \le \infty$, $\|x\|_p$ is a norm on $\mathbb{R}^n$. (The triangle inequality for $\|x\|_p$ follows directly from the Minkowski inequality for sums (E.2).) The metric induced by the norm is given by

$$d_p(x,y) = \|x-y\|_p = \begin{cases} \left(\sum_{k=1}^{n} |x_k - y_k|^p\right)^{1/p} & \text{for } 1 \le p < \infty \\ \max\{|x_1 - y_1|, \ldots, |x_n - y_n|\} & \text{for } p = \infty. \end{cases}$$

Furthermore, one can show that $(\mathbb{R}^n, \|\cdot\|_p)$ is a Banach space for $1 \le p \le \infty$.

Example 2.68 (Space $\mathbb{C}^n$)**.** One can show that $\|x\|_p$ from Example 2.67 also defines a norm on the space $\mathbb{C}^n$. Furthermore, one can show that $(\mathbb{C}^n, \|\cdot\|_p)$ is a Banach space for $1 \le p \le \infty$.

Example 2.69 (Lebesgue space $L^p(\mathbb{R})$)**.** Consider, for $p \in [1,\infty)$, the Lebesgue space $L^p(\mathbb{R})$ consisting of all scalar-valued measurable functions $x(t)$ defined on $\mathbb{R}$ such that $(\int_{-\infty}^{\infty} |x(t)|^p\,dt)^{1/p} < \infty$. One can show that

$$\|x\|_p = \left(\int_{-\infty}^{\infty} |x(t)|^p\,dt\right)^{1/p}$$

is a norm on $L^p(\mathbb{R})$. This norm induces the metric

$$d_p(x,y) = \|x-y\|_p = \left(\int_{-\infty}^{\infty} |x(t)-y(t)|^p\,dt\right)^{1/p}.$$

It is easily verified that $\|x\|_p$ is a norm. The triangle inequality for $\|x\|_p$ follows directly from the Minkowski inequality for integrals (E.3). Thus, $(L^p, \|\cdot\|_p)$ is a normed space. Moreover, one can further show that this is a Banach space.

Example 2.70 (Space $l^p(\mathbb{Z})$)**.** Consider, for $p \in [1,\infty)$, the vector space $l^p(\mathbb{Z})$ of all sequences $x = (\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$ of scalars such that $\sum_{n\in\mathbb{Z}} |x_n|^p < \infty$. One can show that

$$\|x\|_p = \left(\sum_{n\in\mathbb{Z}} |x_n|^p\right)^{1/p}$$

is a norm on $l^p(\mathbb{Z})$. This norm induces the metric

$$d_p(x,y) = \|x-y\|_p = \left(\sum_{n\in\mathbb{Z}} |x_n - y_n|^p \right)^{1/p}.$$

We can easily show that $\|x\|_p$ is a norm on l^p. The triangle inequality for $\|x\|_p$ follows from the Minkowski inequality for infinite sums (E.2). The space l^p with the above induced metric is complete. Therefore, the space $(l^p, \|\cdot\|_p)$ is a Banach space.

2.7.3 Schauder Bases

In the case of infinite-dimensional Banach spaces, different concepts of independence exist. We are already familiar with the notion of linear independence. Another type of independence is often useful as given by the definition below.

Definition 2.42 (ω-independence)**.** Let $\{f_k\}_{k\in I}$ be a sequence in the normed space V. We say that $\{f_k\}_{k\in I}$ is **ω-independent** if whenever the series $\sum_{k\in I} a_k f_k$ is convergent and equal to zero for some scalar coefficients $\{a_k\}_{k\in I}$, then necessarily $a_k = 0$ for all $k \in I$.

The definition of ω-independence above sounds somewhat similar to that of linear independence. So, one might wonder if any relationship exists between these two types of independence. In fact, such a relationship does exist as given by the lemma below.

Lemma 2.3. *Let $\{f_k\}_{k\in I}$ be a sequence in the normed space V. If $\{f_k\}_{k\in I}$ is ω-independent, then $\{f_k\}_{k\in I}$ is linearly independent.*

It is worth noting that the converse of the above result is not true. That is, linear independence does not imply ω-independence. In this sense, ω-independence is a stronger form of independence than linear independence.

Usually, when dealing with infinite-dimensional spaces, we are most interested in ω-independence. For this reason, some authors use the term "independence" or "linear independence" to mean "ω-independence" in this context. Unfortunately, this can sometimes lead to confusion (especially for students).

Another concept related to independence is that of a minimal set as defined below.

Definition 2.43 (Minimal set)**.** A sequence $\{e_n\}_{n\in I}$ in a normed space is said to be **minimal** if, for each $k \in I$, $e_k \notin \operatorname{clos}\operatorname{span}\{e_n\}_{n\in I\setminus\{k\}}$ (i.e., no vector is in the closed linear span of the others).

One can show that a minimal set is ω-independent (and therefore also linearly independent).

Using the concept of an infinite series, we can introduce another type of basis as defined below.

Definition 2.44 (Schauder basis)**.** Let $E = \{e_n\}_{n\in I}$ be a subset of a normed space V. If, for each $x \in V$, there is a unique set of scalars $\{a_n\}_{n\in I}$ such that

$$x = \sum_{n\in I} a_n e_n \tag{2.5}$$

where the series converges in norm to x, then E is said to be a **Schauder basis** of V.

One can show that a Schauder basis is ω-independent. If every permutation of a Schauder basis $\{e_n\}$ is also a basis, then the basis is said to be **unconditional**. In other words, if the basis is unconditional, the convergence of the series in (2.5) does not depend on the order in which terms are summed. Note that the unconditional property is not obtained trivially. That is, many bases do not have this property. In fact, some spaces do not even possess an unconditional basis (e.g., L^1). For obvious practical reasons, however, we are usually only interested in unconditional bases in engineering.

Example 2.71. The space $l^p(\mathbb{N})$ has a Schauder basis (e_n), where $e_n = (\delta_{n,k})$. That is, $e_1 = (1,0,0,\ldots)$, $e_2 = (0,1,0,0,\ldots)$, $e_3 = (0,0,1,0,0,\ldots)$, and so on.

2.8 Inner Product Spaces

In vector spaces, we can add vectors and multiply vectors by scalars, and in normed spaces we can also measure the length of vectors. Even in a normed space, however, we lack the notion of the angle between vectors and perpendicularity (i.e., orthogonality). This leads to the definition of inner product spaces. Such spaces have a well defined notion of the angle between vectors.

Before we can define an inner product space, we must first introduce the concept of an inner product.

Definition 2.45 (Inner product)**.** An **inner product** on a vector space V over a field F is a mapping $\langle \cdot, \cdot \rangle$ of $V \times V$ into F with the following properties:

1. $\langle x,x \rangle \geq 0$ for all $x \in V$ (nonnegativity);
2. for all $x \in V$, $\langle x,x \rangle = 0$ if and only if $x = 0$ (strict positivity);
3. $\langle x,y \rangle^* = \langle y,x \rangle$ for all $x,y \in V$ (conjugate symmetry);
4. $\langle ax,y \rangle = a \langle x,y \rangle$ for all $x,y \in V$ and all $a \in F$ (homogeneity); and
5. $\langle x+y,z \rangle = \langle x,z \rangle + \langle y,z \rangle$ for all $x,y,z \in V$ (additivity).

From the above definition, it follows that an inner product on a real vector space is a real-valued function, and an inner product on a complex vector space is a complex-valued function. In the case that V is a real vector space, the conjugate symmetry condition simply becomes

$$\langle x,y \rangle = \langle y,x \rangle .$$

In the mathematics literature, the homogeneity condition for the inner product is most commonly defined as above. In other words, the inner product is defined to be linear in the first operand, and is therefore conjugate linear in the second operand (i.e., $\langle x,ay \rangle = a^* \langle x,y \rangle$). Examples of works using this convention include [1, 4, 6, 8, 9]. Some authors, however, define the homogeneity condition as

$$\langle x,ay \rangle = a \langle x,y \rangle .$$

Examples of work that use this convention include [2, 11, 12]. In this case, the inner product is defined to be linear in the second operand and is therefore conjugate linear in the first operand (i.e., $\langle ax,y \rangle = a^* \langle x,y \rangle$). In other words, what some authors call $\langle x,y \rangle$, other authors call $\langle y,x \rangle$. Of course, this distinction is unimportant in the case of inner products on real vector spaces. When complex vector spaces are involved, however, one must be careful to distinguish between these two different conventions.

Example 2.72. Consider the vector space $V = \mathbb{C}^2$ (i.e., the space associated with ordered pairs of complex numbers). Define

$$\langle x,y \rangle = x_1 y_1^* + x_2 y_2^*$$

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. Show that $\langle \cdot, \cdot \rangle$ is an inner product on V.

Solution. We can write

$$\begin{aligned} \langle x,x \rangle &= x_1 x_1^* + x_2 x_2^* \\ &= |x_1|^2 + |x_2|^2 \\ &\geq 0. \end{aligned}$$

Thus, we can see that the nonnegativity and strict positivity properties hold. Next, we can write

$$\begin{aligned} \langle x,y \rangle &= x_1 y_1^* + x_2 y_2^* \\ &= (y_1 x_1^* + y_2 x_2^*)^* \\ &= \langle y,x \rangle^* . \end{aligned}$$

Therefore, the conjugate symmetry property holds. We have

$$\begin{aligned}\langle ax,y\rangle &= (ax_1)y_1^* + (ax_2)y_2^* \\ &= a(x_1y_1^* + x_2y_2^*) \\ &= a\langle x,y\rangle .\end{aligned}$$

Thus, the homogeneity property holds. Let $z = (z_1, z_2)$. We can write

$$\begin{aligned}\langle x+y,z\rangle &= (x_1+y_1)z_1^* + (x_2+y_2)z_2^* \\ &= x_1z_1^* + y_1z_1^* + x_2z_2^* + y_2z_2^* \\ &= x_1z_1^* + x_2z_2^* + y_1z_1^* + y_2z_2^* \\ &= \langle x,z\rangle + \langle y,z\rangle .\end{aligned}$$

So, the additivity property holds. Therefore, $\langle\cdot,\cdot\rangle$ is an inner product on V. □

Example 2.73. Consider the vector space $V = \mathbb{R}^2$ (i.e., the space associated with ordered pairs of real numbers). Define

$$\langle x,y\rangle = x_1^2y_1^2 + x_2^2y_2^2.$$

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. Determine whether $\langle\cdot,\cdot\rangle$ is an inner product on V.

Solution. We have

$$\begin{aligned}\langle x,x\rangle &= x_1^2x_1^2 + x_2^2x_2^2 \\ &= x_1^4 + x_2^4.\end{aligned}$$

From this, we can see that the nonnegativity and strict positivity properties hold. We can write

$$\begin{aligned}\langle x,y\rangle &= x_1^2y_1^2 + x_2^2y_2^2 \\ &= y_1^2x_1^2 + y_2^2x_2^2 \\ &= \langle y,x\rangle .\end{aligned}$$

Thus, the conjugate symmetry property holds. We have

$$\begin{aligned}\langle ax,y\rangle &= (ax_1)^2y_1^2 + (ax_2)^2y_2^2 \\ &= a^2x_1^2y_1^2 + a^2x_2^2y_2^2 \\ &= a^2(x_1^2y_1^2 + x_2^2y_2^2) \\ &= a^2\langle x,y\rangle .\end{aligned}$$

Thus, the homogeneity property does not hold. We have

$$\begin{aligned}\langle x+y,z\rangle &= (x_1+y_1)^2z_1^2 + (x_2+y_2)^2z_2^2 \\ &= (x_1^2+2x_1y_1+y_1^2)z_1^2 + (x_2^2+2x_2y_2+y_2^2)z_2^2 \\ &= x_1^2z_1^2 + y_1^2z_1^2 + x_2^2z_2^2 + y_2^2z_2^2 + 2x_1y_1z_1^2 + 2x_2y_2z_2^2 \\ &= \langle x,z\rangle + \langle y,z\rangle + 2x_1y_1z_1^2 + 2x_2y_2z_2^2.\end{aligned}$$

Thus, the additivity property does not hold. Since neither the homogeneity nor additivity property holds, $\langle\cdot,\cdot\rangle$ is not an inner product on V. □

With the earlier definition of an inner product in mind, we are now in a position to define an inner product space.

Definition 2.46 (Inner product space)**.** A vector space V with an inner product defined on V is called an **inner product space**, and is denoted $(V, \langle \cdot, \cdot \rangle)$ or simply V when the inner product is implied from the context.

A finite-dimensional real inner product space is referred to as a **Euclidean space**.

As it turns out, an inner product can always be used to generate a norm as elucidated by the theorem below.

Theorem 2.12 (Induced norm)**.** *If $\langle \cdot, \cdot \rangle$ defines an inner product on the inner product space V, then a norm $\|\cdot\|$ on V is given by*

$$\|x\| = \langle x, x \rangle^{1/2}.$$

This norm is called the ***induced norm*** *(i.e., the norm induced by the inner product).*

Proof. From the nonnegativity of the inner product, we have $\|x\| = \langle x, x \rangle^{1/2} \geq 0$. From the strict positivity property of the inner product, we have $\|x\| = \langle x, x \rangle^{1/2} = 0$ if and only if $x = 0$. From the homogeneity and conjugate symmetry properties of the inner product, we can write

$$\begin{aligned}
\|ax\| &= \langle ax, ax \rangle^{1/2} \\
&= \left[|a|^2 \langle x, x \rangle\right]^{1/2} \\
&= |a| \langle x, x \rangle^{1/2} \\
&= |a| \|x\|.
\end{aligned}$$

Now, we determine whether the triangle inequality is satisfied. We can write

$$\begin{aligned}
\|x+y\|^2 &= \langle x+y, x+y \rangle \\
&= \|x\|^2 + 2\operatorname{Re}\langle x, y \rangle + \|y\|^2.
\end{aligned}$$

Using the Schwarz inequality (yet to be introduced), we can write

$$\begin{aligned}
2\operatorname{Re}\langle x, y \rangle &\leq 2\left|\langle x, y \rangle\right| \\
&\leq 2\left(\|x\|^2 \|y\|^2\right)^{1/2} \\
&= 2\|x\| \|y\|.
\end{aligned}$$

Combining the above results, we have

$$\|x+y\|^2 \leq \|x\|^2 + 2\|x\| \|y\| + \|y\|^2 = (\|x\| + \|y\|)^2$$

which implies that $\|x+y\| \leq \|x\| + \|y\|$. Therefore, $\|\cdot\|$ satisfies all of the properties of a norm. □

Since an inner product on V induces a norm on V, an inner product space is also a normed space. Furthermore, since a norm on V induces a metric on V, an inner product space is also a metric space. (Recall that a norm $\|\cdot\|$ induces the metric d given by $d(x, y) = \|x-y\| = \langle x-y, x-y \rangle^{1/2}$.) Thus, inner product spaces are both normed spaces and metric spaces.

An inner product space is a metric space, a normed space, and a vector space. As such it has both topological and algebraic structure. Furthermore, an inner product space has additional structure defined by means of its inner product. This structure is called **geometric structure**.

Often, we are interested in complete metric spaces. Inner product spaces that are associated with complete metric spaces are given a special name as identified below.

Definition 2.47 (Hilbert space)**.** A **Hilbert space** is a complete inner product space (complete in the metric induced by the inner product).

2.8.1 Examples of Inner Product Spaces

Below, we give some examples of inner product spaces.

Example 2.74 (Euclidean space $\mathbb{R}^n$)**.** The Euclidean space $\mathbb{R}^n$ is a Hilbert space with the inner product defined as

$$\langle x,y\rangle = x_1y_1 + x_2y_2 + \ldots + x_ny_n$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. This inner product induces the norm

$$\|x\| = \langle x,x\rangle^{1/2} = \left(x_1^2 + x_2^2 + \ldots + x_n^2\right)^{1/2}$$

and metric

$$d(x,y) = \|x-y\| = \langle x-y, x-y\rangle^{1/2} = \left[(x_1-y_1)^2 + (x_2-y_2)^2 + \ldots + (x_n-y_n)^2\right]^{1/2}.$$

Note that the above inner product is nothing more than the familiar dot product from Euclidean geometry (i.e., $\langle x,y\rangle = x\cdot y$).

Example 2.75 (Space $\mathbb{C}^n$)**.** The space $\mathbb{C}^n$ is a Hilbert space with the inner product given by

$$\langle x,y\rangle = x_1y_1^* + x_2y_2^* + \ldots + x_ny_n^*$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. This inner product induces the norm

$$\|x\| = \langle x,x\rangle^{1/2} = (x_1x_1^* + x_2x_2^* + \ldots + x_nx_n^*)^{1/2} = (|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2)^{1/2}$$

and metric

$$d(x,y) = \langle x-y, x-y\rangle^{1/2} = \left(|x_1-y_1|^2 + |x_2-y_2|^2 + \ldots + |x_n-y_n|^2\right)^{1/2}.$$

Example 2.76 (Lebesgue space $L^2[a,b]$)**.** The complex Lebesgue space $L^2[a,b]$ is a Hilbert space with the inner product

$$\langle x,y\rangle = \int_a^b x(t)y^*(t)dt$$

and (induced) norm

$$\|x\| = \langle x,x\rangle^{1/2} = \left(\int_a^b |x(t)|^2\,dt\right)^{1/2}.$$

Note that the above integrals must be taken in the Lebesgue sense.

Example 2.77 (Lebesgue space $L^2(\mathbb{R})$)**.** The complex Lebesgue space $L^2(\mathbb{R})$ is a Hilbert space with the inner product

$$\langle x,y\rangle = \int_{-\infty}^{\infty} x(t)y^*(t)dt$$

and (induced) norm

$$\|x\| = \langle x,x\rangle^{1/2} = \left(\int_{-\infty}^{\infty} |x(t)|^2\,dt\right)^{1/2}.$$

Again, the above integrals must be taken in the Lebesgue sense.

Example 2.78 (Sequence space $l^2(\mathbb{Z})$)**.** The space $l^2(\mathbb{Z})$ is a Hilbert space with the inner product defined as

$$\langle x,y\rangle = \sum_{k\in\mathbb{Z}} x_k y_k^*$$

where $x = (\ldots,x_{-2},x_{-1},x_0,x_1,x_2,\ldots)$ and $y = (\ldots,y_{-2},y_{-1},y_0,y_1,y_2,\ldots)$. The norm induced by this inner product is given by

$$\|x\| = \langle x,x\rangle^{1/2} = \left(\sum_{k\in\mathbb{Z}} |x_k|^2\right)^{1/2}.$$

In passing, we note that the Lebesgue spaces L^p and l^p with $p \neq 2$ are not inner product spaces. To prove this, one can show that the norm on each of these spaces does not satisfy the parallelogram law. This implies that the norm cannot be induced from an inner product. (A complete proof for the l^p case can be found in [7, p. 133].)

As we will see later, the Hilbert spaces $L^2(\mathbb{R})$ and $l^2(\mathbb{Z})$ play a crucial role in the study of wavelet systems.

2.8.2 Properties of Inner Product Spaces

Often we must deal with subspaces of Hilbert spaces. For this reason, we would like to know under what circumstances a subspace of a Hilbert space is itself a Hilbert space. The answer to this question is, in part, given by the theorem below.

Theorem 2.13. *A closed subspace of a Hilbert space is itself a Hilbert space.*

Proof. This follows from the fact that a closed vector subspace of a Banach space is itself a Banach space. See [5, p. 20, Theorem 1.5.3] for a proof. □

2.8.3 Relationships Involving Norms and Inner Products

In this section, we consider some important relationships involving norms and inner products in inner product spaces.

Since an inner product on an inner product space V induces a norm on V, one might wonder if inner product can be determined (unambiguously) from the induced norm. This is, in fact, the case. For an inner product space, we can determine the inner product from the induced norm through the use of the theorem below.

Theorem 2.14 (Polarization identity)**.** *If V is a complex inner product space, then*

$$\langle x,y\rangle = \tfrac{1}{4}\sum_{n=0}^{3} j^n \|x+j^n y\|^2 = \tfrac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2 + j\|x+jy\|^2 - j\|x-jy\|^2\right)$$

for all $x,y \in V$. If V is a real inner product space, then

$$\langle x,y\rangle = \tfrac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2\right)$$

for all $x,y \in V$. The above relationship is referred to as the ***polarization identity****.*

Proof. We have

$$\begin{aligned}\|x\pm y\|^2 &= \langle x\pm y, x\pm y\rangle\\ &= \langle x,x\rangle + \langle x,\pm y\rangle + \langle \pm y,x\rangle + \langle \pm y,\pm y\rangle\\ &= \|x\|^2 \pm \langle x,y\rangle \pm \langle y,x\rangle + \|y\|^2\\ &= \|x\|^2 \pm 2\operatorname{Re}\langle x,y\rangle + \|y\|^2.\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\|x \pm jy\|^2 &= \langle x \pm jy, x \pm jy \rangle \\
&= \langle x,x \rangle + \langle x, \pm jy \rangle + \langle \pm jy, x \rangle + \langle \pm jy, \pm jy \rangle \\
&= \|x\|^2 \mp j \langle x,y \rangle \pm j \langle y,x \rangle + \|y\|^2 \\
&= \|x\|^2 \mp j \langle x,y \rangle \pm j \langle x,y \rangle^* + \|y\|^2 \\
&= \|x\|^2 \mp j(\langle x,y \rangle - \langle x,y \rangle^*) + \|y\|^2 \\
&= \|x\|^2 \mp j[2j \operatorname{Im} \langle x,y \rangle] + \|y\|^2 \\
&= \|x\|^2 \pm 2 \operatorname{Im} \langle x,y \rangle + \|y\|^2 .
\end{aligned}$$

Using the above two identities, we can write

$$\begin{aligned}
\tfrac{1}{4} \sum_{n=0}^{3} j^n \|x + j^n y\|^2 &= \tfrac{1}{4} \left(\|x+y\|^2 + j\|x+jy\|^2 - \|x-y\|^2 - j\|x-jy\|^2 \right) \\
&= \tfrac{1}{4} \Big(\|x\|^2 + 2\operatorname{Re}\langle x,y \rangle + \|y\|^2 - (\|x\|^2 - 2\operatorname{Re}\langle x,y \rangle + \|y\|^2) \\
&\qquad + j(\|x\|^2 + 2\operatorname{Im}\langle x,y \rangle + \|y\|^2) - j(\|x\|^2 - 2\operatorname{Im}\langle x,y \rangle + \|y\|^2) \Big) \\
&= \tfrac{1}{4} (4\operatorname{Re}\langle x,y \rangle + 4j\operatorname{Im}\langle x,y \rangle) \\
&= \operatorname{Re}\langle x,y \rangle + j\operatorname{Im}\langle x,y \rangle \\
&= \langle x,y \rangle .
\end{aligned}$$

□

A norm induced from an inner product will always possess certain properties. One such property is given by the theorem below.

Theorem 2.15 (Parallelogram law)**.** *If V is an inner product space (complex or real), then*

$$\|x+y\|^2 + \|x-y\|^2 = 2\left(\|x\|^2 + \|y\|^2 \right) \tag{2.6}$$

*for all $x, y \in V$. This relationship is known as the **parallelogram law**.*

Proof. We begin by observing that

$$\|x \pm y\|^2 = \|x\|^2 \pm 2\operatorname{Re}\langle x,y \rangle + \|y\|^2 .$$

Using this identity, we can write

$$\begin{aligned}
\|x+y\|^2 + \|x-y\|^2 &= \|x\|^2 + 2\operatorname{Re}\langle x,y \rangle + \|y\|^2 + \|x\|^2 - 2\operatorname{Re}\langle x,y \rangle + \|y\|^2 \\
&= 2\|x\|^2 + 2\|y\|^2 \\
&= 2\left(\|x\|^2 + \|y\|^2 \right) .
\end{aligned}$$

□

The parallelogram law simply states that the sum of the squares of the lengths of the four sides of a parallelogram equals the sum of the squares of the lengths of the two diagonals. In the case that the parallelogram is a rectangle, the parallelogram law reduces to the Pythagorean theorem (which is introduced later in Theorem 2.18). The converse of the above theorem (i.e., Theorem 2.15) is also true. That is, if V is a normed space and its norm satisfies (2.6), then there is a unique inner product defined on V that generates the norm.

Another important relationship between the inner product and induced norm is given by the theorem below.

Theorem 2.16 (Schwarz inequality)**.** *If V is an inner product space (complex or real), then*

$$|\langle x, y\rangle| \leq \|x\| \, \|y\|$$

for all $x, y \in V$, with equality holding if and only if x and y are linearly dependent. Moreover, $\langle x, y\rangle = \|x\| \, \|y\|$ if and only if x is a nonnegative multiple of y.

Proof. The identity is easily confirmed when x or y is zero. In what follows, suppose that $y \neq 0$. We have

$$\begin{aligned} 0 \leq \|x - ay\|^2 &= \langle x - ay, x - ay\rangle \\ &= \langle x, x\rangle + \langle x, -ay\rangle + \langle -ay, x\rangle + \langle -ay, -ay\rangle \\ &= \langle x, x\rangle - a^* \langle x, y\rangle - a \langle y, x\rangle + |a|^2 \langle y, y\rangle \\ &= \|x\|^2 - a^* \langle x, y\rangle - a \langle y, x\rangle + |a|^2 \|y\|^2 . \end{aligned}$$

Since this relationship must hold for any a, choose $a = \langle x, y\rangle / \|y\|^2$ so that $a^* = \langle y, x\rangle / \|y\|^2$. Making this substitution for a, we obtain

$$0 \leq \|x\|^2 - \tfrac{\langle y,x\rangle}{\|y\|^2} \langle x, y\rangle - \tfrac{\langle x,y\rangle}{\|y\|^2} \langle y, x\rangle + \left|\tfrac{\langle x,y\rangle}{\|y\|^2}\right|^2 \|y\|^2$$

which we can rewrite as

$$0 \leq \|x\|^2 - \frac{|\langle x, y\rangle|^2}{\|y\|^2} - \frac{|\langle x, y\rangle|^2}{\|y\|^2} + \frac{|\langle x, y\rangle|^2}{\|y\|^4} \|y\|^2 .$$

Observing that the last two terms on the right-hand side of the inequality cancel, we have

$$0 \leq \|x\|^2 - \frac{|\langle x, y\rangle|^2}{\|y\|^2}.$$

Multiplying both sides by $\|y\|^2$ and rearranging, we obtain

$$|\langle x, y\rangle|^2 \leq \|x\|^2 \|y\|^2 .$$

Taking the square root of both sides yields

$$|\langle x, y\rangle| \leq \|x\| \, \|y\| .$$

For equality to hold, $x - ay$ must have norm of zero. This is equivalent to $x - ay = 0$ or $x = ay$ which shows linear dependence. □

From the Schwarz inequality, it follows that, for any two vectors x and y in an inner product space,

$$-1 \leq \frac{\langle x, y\rangle}{\|x\| \, \|y\|} \leq 1.$$

This leads us to define the angle $\theta_{x,y}$ between two vectors x and y as

$$\cos \theta_{x,y} = \frac{\langle x, y\rangle}{\|x\| \, \|y\|}.$$

This definition agrees with the usual notion of the angle between two vectors in Euclidean spaces such as $\mathbb{R}^2$ or $\mathbb{R}^3$. Furthermore, this definition is more generally applicable to any inner product space.

Lastly, from our earlier study of normed spaces, we know that the (induced) norm must also satisfy the triangle inequality. That is, we have that $\|x + y\| \leq \|x\| + \|y\|$ with equality holding if and only if x is a nonnegative multiple of y.

The Schwarz inequality can be used to prove the important property of the inner product stated in the theorem below.

Theorem 2.17 (Continuity of the inner product)**.** *The inner product on a vector space V over a field F, which is a mapping from $V \times V$ into F, is continuous.*

Proof. See [7, p. 138, Lemma 3.2-2]. □

The preceding theorem is quite important as it implies that the order of limits and inner products can be interchanged.

2.8.4 Orthogonality

Since, for an inner product space, we can define angles between vectors, we can also define a corresponding notion of orthogonality as given below.

Definition 2.48 (Orthogonal vectors)**.** Two vectors x and y in an inner product space V are said to be **orthogonal**, denoted $x \perp y$, if

$$\langle x, y \rangle = 0.$$

By virtue of the above definition, the zero vector is orthogonal to every other vector. For a subset A of an inner product space V and a vector $x \in V$, if $x \perp y$ for all $y \in A$, we say that x is orthogonal to A, denoted $x \perp A$. For two subsets A and B of V, if $x \perp y$ for all $x \in A$ and all $y \in B$, we say that A is orthogonal to B, denoted $A \perp B$. Two subspaces U and W are said to be orthogonal if $U \perp W$.

Example 2.79. In the inner product space $V = \mathbb{R}^2$, the vectors $x = (1,2)$ and $y = (-6,3)$ are orthogonal. To verify this, we compute

$$\langle x, y \rangle = (1)(-6) + (2)(3) = 0.$$

Example 2.80. Consider the inner product space $V = L^2[0,1]$. Define

$$x(t) = \cos 2\pi t \quad \text{and} \quad y(t) = \cos 4\pi t.$$

Show that $x \perp y$.

Solution. We compute the inner product of x and y as

$$\begin{aligned}\langle x, y \rangle &= \int_0^1 (\cos 2\pi t)(\cos 4\pi t)dt \\ &= \int_0^1 (\tfrac{1}{2}\cos 6\pi t + \tfrac{1}{2}\cos 2\pi t)dt \\ &= \left[\tfrac{1}{12\pi}\sin 6\pi t + \tfrac{1}{4\pi}\sin 2\pi t\right]\Big|_0^1 \\ &= (0+0) - (0+0) \\ &= 0.\end{aligned}$$

Thus, we have that $x \perp y$. □

Example 2.81. Let A and B denote subsets of $\mathbb{R}^3$, where $A = \{(1,0,-1),(2,1,1)\}$ and $B = \{(1,-3,1),(2,-6,2)\}$. Show that $A \perp B$.

Solution. We have

$$\begin{gathered}\langle (1,0,-1),(1,-3,1) \rangle = 0, \\ \langle (1,0,-1),(2,-6,2) \rangle = 0, \\ \langle (2,1,1),(1,-3,1) \rangle = 0, \quad \text{and} \\ \langle (2,1,1),(2,-6,2) \rangle = 0.\end{gathered}$$

Thus, $x \perp y$ for all $x \in A$ and all $y \in B$. Therefore, $A \perp B$. □

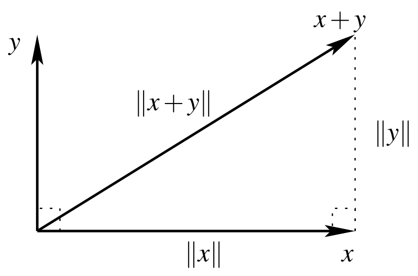


Figure 2.11: Pythagorean relation in Euclidean plane.

Orthogonal vectors satisfy a number of useful identities, one of which is the following:

Theorem 2.18 (Pythagorean theorem)**.** *If two vectors x and y in an inner product space are orthogonal, then*

$$\|x+y\|^2 = \|x\|^2 + \|y\|^2.$$

Proof. Since $x \perp y$, we have that $\langle x,y\rangle = \langle y,x\rangle = 0$. So, we can write

$$\begin{aligned}\|x+y\|^2 &= \langle x+y, x+y\rangle \\ &= \langle x,x\rangle + \langle x,y\rangle + \langle y,x\rangle + \langle y,y\rangle \\ &= \|x\|^2 + \|y\|^2.\end{aligned}$$

□

The geometric interpretation of the Pythagorean theorem in the case of the Euclidean plane is shown in Figure 2.11.

2.8.5 Orthogonal Projection

Often we would like to approximate a vector in some inner product space V with a vector in a subspace of V. Usually we wish to do this in such a way as to minimize the error with respect to the norm. This leads to the notion of an orthogonal projection.

Before we can define the orthogonal projection, we first must introduce the concept of an orthogonal complement.

Definition 2.49 (Orthogonal complement)**.** Let W be a nonempty subset of an inner product space V. The set of all elements of V orthogonal to W, denoted $W^\perp$, is called the **orthogonal complement** of W (in V). That is, we define

$$W^\perp = \{v \in V : v \perp W\}.$$

Example 2.82. Suppose S is the subset of the inner product space $\mathbb{R}^3$ given by $S = \{(0,0,1)\}$. Then, $S^\perp$ is the xy-plane.

Example 2.83. Consider the inner product space $V = l^2(\mathbb{N})$. Let $S = \{e_1, e_2, e_3\} \subset V$, where

$$e_1 = (1,0,0,\ldots), \quad e_2 = (0,1,0,0,\ldots), \quad \text{and} \quad e_3 = (0,0,1,0,0,\ldots).$$

The orthogonal complement $S^\perp$ of S in V is the set of all sequences in $l^2(\mathbb{N})$ of the form $(0,0,0,a_4,a_5,a_6,\ldots)$.

Definition 2.50 (Orthogonal projection operator)**.** A projection P on an inner product space is said to be **orthogonal** if its range and null spaces are orthogonal (i.e., $R(P) \perp N(P)$).

Suppose that we have a subspace W of an inner product space V and an arbitrary vector $x \in V$. One might wonder whether a unique closest point to x in W exists. To help us to answer this question, we first introduce the theorem below.

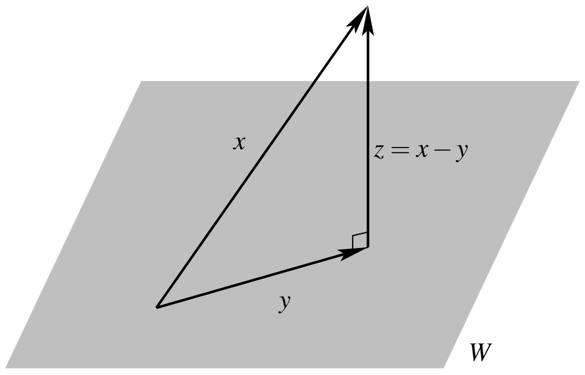


Figure 2.12: Orthogonal projection.

Theorem 2.19 (Projection theorem)**.** *If W is a closed subspace of a Hilbert space V, then every element $x \in V$ has a unique decomposition of the form $x = y + z$ where $y \in W$ and $z \in W^{\perp}$.*

As a matter of terminology, we refer to the vector y in the preceding theorem as the **orthogonal projection** of x onto W. For the case of the inner product space $\mathbb{R}^3$, the orthogonal projection has the geometric interpretation illustrated in Figure 2.12. The projection theorem is fundamentally important as it guarantees the existence of an orthogonal projection in any Hilbert space. It is important to note that the above theorem does not hold for inner product spaces in general. Completeness is required. Following from the result of Theorem 2.19, we have the important result stated below.

Corollary 2.1. *Let W be a closed subspace of a Hilbert space V, and let $x_0 \in V$. Further, let P be the orthogonal projection of V onto W. There exists a unique vector $y_0 \in W$ that is closest to x_0 as given by $y_0 = Px_0$. That is, $\|x_0 - Px_0\| = \inf\{\|x_0 - y\| : y \in W\}$.*

Proof. We begin by observing that

$$x_0 - Px_0 \in W^{\perp} \quad \text{and} \quad Px_0 - y \in W.$$

So, we have

$$x_0 - Px_0 \perp Px_0 - y \quad \text{for all } y \in W.$$

Consequently, using the Pythagorean theorem, we can write

$$\|(x_0 - Px_0) + (Px_0 - y)\|^2 = \|x_0 - Px_0\|^2 + \|Px_0 - y\|^2$$

or equivalently

$$\|x_0 - y\|^2 = \|x_0 - Px_0\|^2 + \|Px_0 - y\|^2.$$

From this last equation, we can see that

$$\|x_0 - y\|^2 \geq \|x_0 - Px_0\|^2$$

with equality holding only if $y = Px_0$. Thus, the vector $y \in W$ which minimizes $\|x_0 - y\|$ is $y = y_0 = Px_0$. $\square$

From the above result, we see that given a subspace W of an inner product space V, the unique closest vector to any vector $x_0 \in V$ is the orthogonal projection of x_0 onto W (assuming that such a projection exists). Thus, the orthogonal projection is quite useful when we must approximate vectors in one space by vectors in a corresponding subspace.

2.8.6 Orthonormal Bases

The most elementary type of basis is the Hamel basis, as introduced earlier in the context of vector spaces. Recall that a Hamel basis is a purely algebraic concept. In the case of finite-dimensional spaces, Hamel bases are all we need to consider (as other types of bases are simply special cases of Hamel bases). In the case of infinite-dimensional spaces, however, the situation is quite different. Every Hamel basis of an infinite-dimensional space is uncountable. Moreover, we can only prove that such a basis exists without being able to describe its elements. This is obviously quite problematic, since most practical applications require the use of highly structured and easy to describe bases. The above shortcoming motivates the use of another type of basis, called an orthonormal basis.

To begin, we introduce the notion of orthogonal and orthonormal sets.

Definition 2.51 (Orthogonal and orthonormal sets)**.** A set of points $\{x_n\}_{n\in I}$ in an inner product space V is said to be **orthogonal** if $x_n \perp x_m$ for $m,n \in I$, $m \neq n$. A set of points $\{x_n\}$ in an inner product space V is said to be **orthonormal** if

$$\langle x_n, x_m \rangle = \delta_{n,m}$$

for all $m,n \in I$. In other words, an orthonormal set is an orthogonal set whose elements have a norm of one.

An orthogonal set of points may contain the zero vector, since the zero vector is orthogonal to every vector.

Example 2.84. Consider the inner product space $L^2[-\pi,\pi]$. Let $S = \{f_n\}_{n\in\mathbb{N}}$, where

$$f_n(t) = \tfrac{1}{\sqrt{\pi}} \cos nt.$$

Show that S is an orthonormal set.

Solution. First, let us consider $\langle f_m, f_n\rangle$ for $m \neq n$. Using the identity $\cos A \cos B = \frac{1}{2}[\cos(A+B) + \cos(A-B)]$, we can write

$$\begin{aligned}
\langle f_m, f_n \rangle &= \int_{-\pi}^{\pi} (\tfrac{1}{\sqrt{\pi}} \cos mt)(\tfrac{1}{\sqrt{\pi}} \cos nt)dt \\
&= \tfrac{1}{\pi} \int_{-\pi}^{\pi} (\cos mt)(\cos nt)dt \\
&= \tfrac{1}{2\pi} \int_{-\pi}^{\pi} [\cos([m+n]t) + \cos([m-n]t)]dt \\
&= \tfrac{1}{2\pi} \left[\tfrac{1}{m+n}\sin([m+n]t) + \tfrac{1}{m-n}\sin([m-n]t)\right]\Big|_{-\pi}^{\pi} \\
&= \tfrac{1}{2\pi}[(0+0)-(0+0)] \\
&= 0.
\end{aligned}$$

Second, let us consider $\langle f_m, f_n\rangle$ for $m = n$. Using the identity $\cos^2 A = \frac{1}{2} + \frac{1}{2}\cos 2A$, we can write

$$\begin{aligned}
\langle f_n, f_n \rangle &= \int_{-\pi}^{\pi} \left(\tfrac{1}{\sqrt{\pi}} \cos nt\right)^2 dt \\
&= \tfrac{1}{2\pi} \int_{-\pi}^{\pi} (1 + \cos 2nt)\, dt \\
&= \left[t + \tfrac{1}{2n} \sin 2nt\right]\Big|_{-\pi}^{\pi} \\
&= \tfrac{1}{2\pi}[\pi - (-\pi)] \\
&= 1.
\end{aligned}$$

Thus, we have that $\langle f_n, f_m\rangle = \delta_{n,m}$, implying that S is orthonormal. □

An important relationship exists between the orthonormality and ω-independence properties as given by the theorem below.

Theorem 2.20 (ω-independence of an orthonormal set)**.** *Let $\{x_n\}$ be an orthonormal set in an inner product space. Then, $\{x_n\}$ is ω-independent (and is therefore also linearly independent).*

Proof. This theorem is left for the reader to prove as an exercise problem. □

There exist orthonormal sets in inner product spaces with sufficiently many elements that every element in the space can be represented by (possibly) infinite linear combinations of elements in the orthonormal set. We refer to such a set as an orthonormal basis. In the case of orthonormal bases, infinite sums (i.e., infinite linear combinations) are permitted. In this context, an infinite sum is defined in the same manner as in the case of normed spaces (i.e., in terms of limits of sequences of partial sums). The advantage of orthonormal bases is that, even in the infinite-dimensional case, the bases can be highly structured and easy to describe. More formally, we define an orthonormal basis as specified below.

Definition 2.52 (Orthonormal basis)**.** An orthonormal set E in an inner product space V is said to be an **orthonormal basis** if $\operatorname{span} E$ is dense in V (i.e., $V = \operatorname{clos}\operatorname{span} E$).

It follows form the above definition that an orthonormal set E is an orthonormal basis if and only if $\operatorname{clos}\operatorname{span} E = V$.

Unless the space in question is finite-dimensional, an orthonormal basis is not a basis in the algebraic sense. That is, an orthonormal basis is a Hamel basis only in the case of finite-dimensional spaces.

Example 2.85. The inner product space $\mathbb{R}^n$ has many orthonormal bases including the canonical one $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, where

$$\varphi_1 = (1,0,\ldots,0),\ \ \varphi_2 = (0,1,0,\ldots,0),\ \ \ldots,\ \ \varphi_n = (0,\ldots,0,1).$$

Example 2.86. The inner product space $l^2(\mathbb{N})$ has the orthonormal basis $\{\varphi_n\}_{n\in\mathbb{N}}$ where φ_n is a sequence with the nth element equal to one and all other elements equal to zero. That is, we have

$$\varphi_1 = (1,0,0,\ldots),\ \ \varphi_2 = (0,1,0,0,\ldots),\ \ \varphi_3 = (0,0,1,0,0,\ldots),\ \ \ldots.$$

Example 2.87. The inner product space $L^2[-\pi,\pi]$ has the orthonormal basis $\{\varphi_n\}_{n\in\mathbb{Z}}$, where

$$\varphi_n(t) = \tfrac{1}{\sqrt{2\pi}} e^{jnt}, \quad n = 0, \pm 1, \pm 2, \ldots.$$

We can easily confirm that $\{\varphi_n\}_{n\in\mathbb{Z}}$ constitutes an orthonormal set. For $m \neq n$, we have

$$\begin{aligned}
\langle \varphi_m, \varphi_n \rangle &= \int_{-\pi}^{\pi} \varphi_m(t)\varphi_n^*(t)dt \\
&= \tfrac{1}{2\pi} \int_{-\pi}^{\pi} e^{j(m-n)t} dt \\
&= \tfrac{1}{2\pi} \left[\tfrac{1}{j(m-n)} e^{j(m-n)t} \right] \Big|_{-\pi}^{\pi} \\
&= \frac{e^{j\pi(m-n)} - e^{-j\pi(m-n)}}{j2\pi(m-n)} \\
&= \frac{(-1)^{m-n} - (-1)^{m-n}}{j2\pi(m-n)} \\
&= 0.
\end{aligned}$$

For $m = n$, we have

$$\begin{aligned}\langle \varphi_n, \varphi_n \rangle &= \int_{-\pi}^{\pi} \varphi_n(t)\varphi_n^*(t)dt \\ &= \int_{-\pi}^{\pi} |\varphi_n(t)|^2 dt \\ &= \int_{-\pi}^{\pi} \left|\tfrac{1}{\sqrt{2\pi}} e^{jnt}\right|^2 dt \\ &= \int_{-\pi}^{\pi} \tfrac{1}{2\pi} dt \\ &= 1.\end{aligned}$$

Thus, $\{\varphi_n\}_{n\in\mathbb{Z}}$ is clearly an orthonormal set. The proof that this set generates the entire space (i.e., is an orthonormal basis) is not easy, and is omitted here.

A Hamel basis is a purely algebraic concept. Orthonormal bases are more useful than Hamel bases when dealing with infinite-dimensional spaces. As illustrated by the previous example, the space $L^2[-\pi,\pi]$ has a countable orthonormal basis consisting of simple functions (i.e., harmonically related complex sinusoids). One can show, however, that every Hamel basis of $L^2[-\pi,\pi]$ is uncountable.

Since orthonormal bases are often quite useful, one might wonder under what conditions such a basis exists. One useful result in this regard is given by the theorem below.

Theorem 2.21 (Existence of orthonormal basis)**.** *Every Hilbert space has an orthonormal basis.*

The above result is yet another reason why we like to work with Hilbert spaces. Hilbert spaces always have an orthonormal basis. It is worth noting that the same statement cannot be made about inner product spaces in general. Completeness is essential in order to guarantee the existence of an orthonormal basis.

A Hilbert space is said to be **separable** if it has a countable orthonormal basis. In the remainder of this book, we will only concern ourselves with separable Hilbert spaces.

Theorem 2.22. *Let $E = \{\varphi_n\}_{n\in I}$ be an orthonormal basis of an inner product space V. Then, each $x \in V$ has a unique representation of the form*

$$x = \sum_{n\in I} a_n \varphi_n$$

where

$$a_n = \langle x, \varphi_n \rangle .$$

(Note: The index set I may be infinite. That is, the above sum may be an infinite linear combination.)

The above theorem is particularly useful as it provides a means for computing the coefficients of an orthonormal expansion.

The theorem below provides necessary and sufficient conditions for an orthonormal set to constitute an orthonormal basis.

Theorem 2.23. *An orthonormal set $E = \{\varphi_n\}$ in a Hilbert space V is an orthonormal basis of V if and only if the only vector $x \in V$ that is orthogonal to every element of E is the zero vector.*

Some other useful results related to orthonormal sets and bases are given below.

Theorem 2.24 (Bessel inequality)**.** *Let $S = \{x_n\}_{n\in I}$ be an orthonormal set (where the index set I may be infinite) in an inner product space V. Then, for all $x \in V$,*

$$\sum_n |\langle x, x_n \rangle|^2 \le \|x\|^2$$

*with equality holding if and only if $x \in \operatorname{clos}\operatorname{span} S$. This relationship is known as the **Bessel inequality**.*

Proof. We can write

$$\begin{aligned}\left\| x - \sum_k \langle x, x_k \rangle x_k \right\|^2 &= \left\langle x - \sum_k \langle x, x_k \rangle x_k, x - \sum_l \langle x, x_l \rangle x_l \right\rangle \\ &= \langle x, x \rangle - \sum_k \langle x, x_k \rangle \langle x_k, x \rangle - \sum_l \langle x, x_l \rangle^* \langle x, x_l \rangle + \sum_k \sum_l \langle x, x_k \rangle \langle x, x_l \rangle^* \langle x_k, x_l \rangle \\ &= \|x\|^2 - \sum_k \langle x, x_k \rangle \langle x, x_k \rangle^* - \sum_l \langle x, x_l \rangle^* \langle x, x_l \rangle + \sum_k \langle x, x_k \rangle \langle x, x_k \rangle^* \\ &= \|x\|^2 - \sum_k |\langle x, x_k \rangle|^2 - \sum_l |\langle x, x_l \rangle|^2 + \sum_k |\langle x, x_k \rangle|^2 \\ &= \|x\|^2 - \sum_k |\langle x, x_k \rangle|^2 .\end{aligned}$$

(We used the fact that $\langle x_k, x_l \rangle = \delta_{k,l}$ in the simplification above.) Since the left-hand side of the above equation must be nonnegative, we have

$$0 \le \|x\|^2 - \sum_k |\langle x, x_k \rangle|^2 \quad \Rightarrow \quad \|x\|^2 \ge \sum_k |\langle x, x_k \rangle|^2 .$$

If $x \in \operatorname{clos} \operatorname{span} S$, then $x - \sum_k \langle x, x_k \rangle x_k = 0$, and it follows from above that the equality part of the inequality holds. □

Theorem 2.25 (Parseval identity and Plancherel formula)**.** *Let $\{x_n\}$ be an orthonormal basis of a Hilbert space V. Then, for all $x, y \in V$,*

$$\langle x, y \rangle = \sum_n \langle x, x_n \rangle \langle y, x_n \rangle^*$$

and

$$\|x\|^2 = \sum_n |\langle x, x_n \rangle|^2 .$$

These two relationships are known as the ***Parseval identity*** *and* ***Plancherel formula****, respectively.*

Proof. Let $x = \sum_k \langle x, x_k \rangle x_k$ and $y = \sum_l \langle y, x_l \rangle x_l$. Using the fact that $\langle x_k, x_l \rangle = \delta_{k,l}$, we can write

$$\begin{aligned}\langle x, y \rangle &= \left\langle \sum_k \langle x, x_k \rangle x_k, \sum_l \langle y, x_l \rangle x_l \right\rangle \\ &= \sum_k \sum_l \langle x, x_k \rangle \langle y, x_l \rangle^* \langle x_k, x_l \rangle \\ &= \sum_k \langle x, x_k \rangle \langle y, x_k \rangle^* .\end{aligned}$$

If we let $x = y$, we obtain

$$\langle x, x \rangle = \|x\|^2 = \sum_k \langle x, x_k \rangle \langle x, x_k \rangle^* = \sum_k |\langle x, x_k \rangle|^2 .$$

□

Previously, we introduced the orthogonal projection. We did not explain at that point how one can compute an orthogonal projection. Now, we are in a position to address this matter. An orthogonal projection can be computed by using an orthonormal basis as specified by the theorem below.

Theorem 2.26. *Let W be a closed subspace of a Hilbert space V, and let $E = \{\phi_n\}_{n \in I}$ be an orthonormal basis for W. Further, let P denote the orthogonal projection of V onto W. Then, P is given by*

$$Px = \sum_{n \in I} \langle x, \phi_n \rangle \phi_n,$$

where $x \in V$.

Example 2.88. Consider the Hilbert space $V = \mathbb{R}^3$. Let W denote the subspace of V with the orthonormal basis $E = \{(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}), (-\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})\}$. Find the closest point in W to the point $(1,2,1) \in V$.

Solution. The closest point in W to a point $x \in V$ is given by the orthogonal projection of x onto W. Let y denote this closest point. We have

$$\begin{aligned}
y &= \sum_{n=1}^{2} \langle x, \varphi_n \rangle \varphi_n \\
&= \left\langle (1,2,1), (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) \right\rangle (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) + \left\langle (1,2,1), (-\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) \right\rangle (-\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) \\
&= \sqrt{2}(\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) + 0 \\
&= (1,0,1).
\end{aligned}$$

□

2.8.7 Gram-Schmidt Orthogonalization

For a finite-dimensional inner product space, given a set of linearly independent vectors, we can find an orthonormal basis with the same span. To do this, we use the algorithm described below.

Theorem 2.27 (Gram-Schmidt orthogonalization)**.** *Let V be an inner product space and $S = \{x_1, x_2, \ldots, x_n\}$ be a linearly independent subset of V. Let $E = \{e_1, e_2, \ldots, e_n\}$ be another set of vectors generated by the following iterative process:*

$$e_k = \frac{1}{\|v_k\|} v_k$$

where

$$v_k = \begin{cases} x_k - \sum_{l=1}^{k-1} \langle x_k, e_l \rangle e_l & \text{for } k = 2, 3, \ldots, n \\ x_1 & \text{for } k = 1. \end{cases}$$

Then E is an orthonormal set of vectors such that $\operatorname{span}(E) = \operatorname{span}(S)$. *(The kth step in the orthonormalization process is illustrated graphically in Figure 2.13.)*

From the above theorem, we can see that any finite-dimensional Hilbert space must have an orthonormal basis. The theorem provides a constructive algorithm for obtaining such a basis.

Example 2.89. Consider the inner product space $V = \mathbb{R}^4$. Let S be the linearly independent subset of V given by $S = \{(1,0,1,0), (1,1,1,1), (0,1,2,-1)\}$. Use the Gram-Schmidt process to find an orthonormal basis for $\operatorname{span} S$.

Solution. FIRST ITERATION. We have

$$\begin{gathered}
v_1 = x_1 = (1,0,1,0), \\
\|v_1\| = \sqrt{1^2 + 1^2} = \sqrt{2}, \quad \text{and} \\
e_1 = \frac{v_1}{\|v_1\|} = \tfrac{1}{\sqrt{2}}(1,0,1,0) = (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0).
\end{gathered}$$

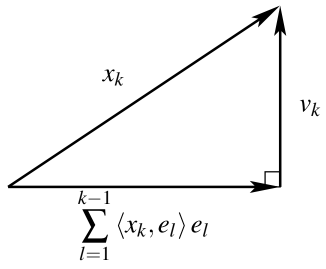


Figure 2.13: Gram-Schmidt orthogonalization process (*k*th step).

SECOND ITERATION. We have

$$\begin{aligned} v_2 &= x_2 - \langle x_2, e_1 \rangle e_1 \\ &= (1,1,1,1) - \left\langle (1,1,1,1), (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) \right\rangle (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) \\ &= (1,1,1,1) - \sqrt{2}(\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) \\ &= (1,1,1,1) - (1,0,1,0) \\ &= (0,1,0,1), \end{aligned}$$

$$\|v_2\| = \sqrt{1^2 + 1^2} = \sqrt{2}, \quad \text{and}$$

$$\begin{aligned} e_2 &= \frac{v_2}{\|v_2\|} \\ &= \tfrac{1}{\sqrt{2}}(0,1,0,1) \\ &= (0, \tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}). \end{aligned}$$

THIRD ITERATION. We have

$$\begin{aligned} v_3 &= x_3 - \langle x_3, e_1 \rangle e_1 - \langle x_3, e_2 \rangle e_2 \\ &= (0,1,2,-1) - \left\langle (0,1,2,-1), (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) \right\rangle (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) - \left\langle (0,1,2,-1), (0, \tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) \right\rangle (0, \tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}) \\ &= (0,1,2,-1) - \sqrt{2}(\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0) - 0 \\ &= (0,1,2,-1) - (1,0,1,0) \\ &= (-1,1,1,-1), \end{aligned}$$

$$\|v_3\| = \sqrt{(-1)^2 + 1^2 + 1^2 + (-1)^2} = \sqrt{4} = 2, \quad \text{and}$$

$$\begin{aligned} e_3 &= \frac{v_3}{\|v_3\|} \\ &= \tfrac{1}{2}(-1,1,1,-1) \\ &= (-\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, -\tfrac{1}{2}). \end{aligned}$$

Thus, we have that $\{e_1, e_2, e_3\}$ are an orthonormal basis, where

$$e_1 = (\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}, 0), \quad e_2 = (0, \tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}), \quad \text{and} \quad e_3 = (-\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, -\tfrac{1}{2}).$$

□

Example 2.90. Consider the inner product space $V = L^2[-1,1]$. Let S be the linearly independent subset of V given by $S = \{1,t,t^2\}$. Let $W = \operatorname{span} S$. That is, W is the subspace of V consisting of polynomials of degree less than or equal to two (including the zero polynomial). Use the Gram-Schmidt process to find an orthonormal basis for W.

Solution. We use the Gram-Schmidt process. FIRST ITERATION. We have

$$\begin{gathered} v_1 = 1, \\ \|v_1\|^2 = \langle v_1, v_1\rangle = \int_{-1}^{1} dt = 2, \quad \text{and} \\ e_1 = \tfrac{1}{\sqrt{2}}. \end{gathered}$$

SECOND ITERATION. We have

$$\begin{aligned} v_2 &= x_2 - \langle x_2, e_1\rangle e_1 \\ &= t - \left\langle t, \tfrac{1}{\sqrt{2}} \right\rangle \tfrac{1}{\sqrt{2}} \\ &= t - \tfrac{1}{\sqrt{2}} \int_{-1}^{1} t(\tfrac{1}{\sqrt{2}}) dt \\ &= t - \tfrac{1}{2} \left[\tfrac{1}{2}t^2\right]\Big|_{-1}^{1} \\ &= t - \tfrac{1}{2}[\tfrac{1}{2} - \tfrac{1}{2}] \\ &= t, \end{aligned}$$

$$\begin{aligned} \|v_2\|^2 &= \langle v_2, v_2\rangle \\ &= \int_{-1}^{1} t^2 dt \\ &= \left[\tfrac{1}{3}t^3\right]\Big|_{-1}^{1} \\ &= \tfrac{1}{3} - (-\tfrac{1}{3}) \\ &= \tfrac{2}{3}, \quad \text{and} \end{aligned}$$

$$e_2 = \frac{v_2}{\|v_2\|} = \frac{t}{\sqrt{\frac{2}{3}}} = \sqrt{\tfrac{3}{2}}t.$$

THIRD ITERATION. We have

$$\begin{aligned} v_3 &= x_3 - \langle x_3, e_1\rangle e_1 - \langle x_3, e_2\rangle e_2 \\ &= t^2 - \left\langle t^2, \tfrac{1}{\sqrt{2}} \right\rangle \tfrac{1}{\sqrt{2}} - \left\langle t^2, \sqrt{\tfrac{3}{2}}t \right\rangle \sqrt{\tfrac{3}{2}}t \\ &= t^2 - \tfrac{1}{\sqrt{2}} \int_{-1}^{1} \tfrac{1}{\sqrt{2}} t^2 dt - \sqrt{\tfrac{3}{2}}t \int_{-1}^{1} \sqrt{\tfrac{3}{2}} t^3 dt \\ &= t^2 - \tfrac{1}{\sqrt{2}} \left[\tfrac{1}{\sqrt{2}} \tfrac{t^3}{3}\right]\Big|_{-1}^{1} - \sqrt{\tfrac{3}{2}}t \left[\sqrt{\tfrac{3}{2}} \tfrac{t^4}{4}\right]\Big|_{-1}^{1} \\ &= t^2 - \tfrac{1}{2} \left[\tfrac{1}{3}t^3\right]\Big|_{-1}^{1} - \tfrac{3}{2}t \left[\tfrac{1}{4}t^4\right]\Big|_{-1}^{1} \\ &= t^2 - \tfrac{1}{2}(\tfrac{1}{3} - (-\tfrac{1}{3}) - \tfrac{3}{2}t(\tfrac{1}{4} - \tfrac{1}{4})) \\ &= t^2 - \tfrac{1}{3}, \end{aligned}$$

$$\begin{aligned}
\|v_3\|^2 &= \langle v_3, v_3\rangle \\
&= \int_{-1}^{1} (t^2 - \tfrac{1}{3})^2 dt \\
&= \int_{-1}^{1} (t^4 - \tfrac{2}{3}t^2 + \tfrac{1}{9}) dt \\
&= \left[\tfrac{1}{5}t^5 - \tfrac{2}{9}t^3 + \tfrac{1}{9}t\right]\Big|_{-1}^{1} \\
&= (\tfrac{1}{5} - \tfrac{2}{9} + \tfrac{1}{9}) - (-\tfrac{1}{5} + \tfrac{2}{9} - \tfrac{1}{9}) \\
&= \tfrac{8}{45}, \quad \text{and}
\end{aligned}$$

$$\begin{aligned}
e_3 &= \frac{v_3}{\|v_3\|} \\
&= \frac{t^2 - \frac{1}{3}}{\sqrt{\frac{8}{45}}} \\
&= \sqrt{\tfrac{45}{8}}t^2 - \sqrt{\tfrac{45}{8}}(\tfrac{1}{3}) \\
&= \sqrt{\tfrac{45}{8}}t^2 - \sqrt{\tfrac{5}{8}}.
\end{aligned}$$

Thus, an orthonormal basis for W is given by

$$y_1(t) = \tfrac{1}{\sqrt{2}}, \quad y_2(t) = \sqrt{\tfrac{3}{2}}t, \quad \text{and} \quad y_3(t) = \sqrt{\tfrac{45}{8}}t^2 - \sqrt{\tfrac{5}{8}}.$$

(As an aside, we note that the functions y_1, y_2, y_3 are, in fact, the first three **Legendre polynomials**.) □

2.8.8 Riesz Bases

Although an orthonormal basis is often convenient to employ, we can instead choose to use a basis that is not orthonormal (or even orthogonal). Sometimes orthonormality places too many constraints on the choice of basis vectors. By dropping the orthonormality constraint, we gain additional freedom in the choice of basis vectors. This leads us to another type of basis known as a Riesz basis.

We begin by introducing the notion of biorthogonality.

Definition 2.53 (Biorthogonal and biorthonormal sets)**.** Two sets $\{x_n\}_{n\in I}$ and $\{y_n\}_{n\in I}$ are said to be **biorthogonal** if $x_m \perp y_n$ for $m, n \in I$ and $m \neq n$. If, in addition, we have $\langle x_n, y_n\rangle = 1$ for all $n \in I$, then the sets are said to be **biorthonormal**.

It follows from the above definition that an orthogonal set is biorthogonal with itself, and similarly, an orthonormal set is biorthonormal with itself.

Example 2.91. For the space $V = \mathbb{R}^3$, consider the sets $X = \{x_1, x_2\} \subset V$ and $Y = \{y_1, y_2\} \subset V$ where

$$\begin{gathered}
x_1 = (1, 0, -1), \quad x_2 = (2, 2, 2), \\
y_1 = (0, 1, -1), \quad \text{and} \quad y_2 = (1, -1, 1).
\end{gathered}$$

Determine whether the sets X and Y are biorthogonal or biorthonormal.

Solution. Computing the various inner products, we obtain

$$\begin{gathered}
\langle x_1, y_2\rangle = \langle (1,0,-1), (1,-1,1)\rangle = 0 \\
\langle x_2, y_1\rangle = \langle (2,2,2), (0,1,-1)\rangle = 0 \\
\langle x_1, y_1\rangle = \langle (1,0,-1), (0,1,-1)\rangle = 1 \quad \text{and} \\
\langle x_2, y_2\rangle = \langle (2,2,2), (1,-1,1)\rangle = 2.
\end{gathered}$$

Since $\langle x_m, y_n\rangle = 0$ for $m \neq n$, X and Y are biorthogonal. As $\langle x_2, y_2\rangle \neq 1$, X and Y are not biorthonormal. □

Example 2.92. For the space $V = \mathbb{R}^3$, consider the sets $X = \{x_1, x_2, x_3\} \subset V$ and $Y = \{y_1, y_2, y_3\} \subset V$ where

$$x_1 = (1,0,-1), \quad x_2 = (1,1,1), \quad x_3 = (0,1,1),$$
$$y_1 = (0,1,-1), \quad y_2 = (1,-1,1), \quad \text{and} \quad y_3 = (-1,2,-1).$$

Determine whether the sets X and Y are biorthogonal and/or biorthonormal.

Solution. Computing the various inner products, we have

$$\begin{aligned}
\langle x_1, y_2\rangle &= \langle (1,0,-1),(1,-1,1)\rangle = 0\\
\langle x_1, y_3\rangle &= \langle (1,0,-1),(-1,2,-1)\rangle = 0\\
\langle x_2, y_1\rangle &= \langle (1,1,1),(0,1,-1)\rangle = 0\\
\langle x_2, y_3\rangle &= \langle (1,1,1),(-1,2,-1)\rangle = 0\\
\langle x_3, y_1\rangle &= \langle (0,1,1),(0,1,-1)\rangle = 0\\
\langle x_3, y_2\rangle &= \langle (0,1,1),(1,-1,1)\rangle = 0\\
\langle x_1, y_1\rangle &= \langle (1,0,-1),(0,1,-1)\rangle = 1,\\
\langle x_2, y_2\rangle &= \langle (1,1,1),(1,-1,1)\rangle = 1, \quad \text{and}\\
\langle x_3, y_3\rangle &= \langle (0,1,1),(-1,2,-1)\rangle = 1.
\end{aligned}$$

Since $\langle x_m, y_n\rangle = 0$ for $m \neq n$, X and Y are biorthogonal. As $\langle x_m, y_n\rangle = \delta_{m,n}$, X and Y are also biorthonormal. □

For a finite-dimensional inner product space V of dimension n, any linearly independent set E of n vectors satisfies $\operatorname{span} E = V$. In the case of an infinite-dimensional space, however, the situation is different. For an infinite-dimensional space V, an infinite set E of linearly independent vectors need not satisfy $\operatorname{clos}\operatorname{span} E = V$. Of course, one issue here is that a proper subspace of an infinite-dimensional space can itself be infinite dimensional.

Other subtleties can also arise in the case of infinite-dimensional spaces. For example, in the case of finite-dimensional spaces, the only constraint that we need to place on basis vectors is that they be linearly independent. In the case of infinite-dimensional spaces, however, the situation is somewhat more complicated. In the infinite-dimensional case, linear independence (or even ω-independence or minimality) alone is not enough to ensure a well behaved basis. This point is illustrated by the examples below.

Example 2.93. Let $\{e_n\}_{n\in\mathbb{N}}$ be an orthonormal basis of a Hilbert space, and define the sequence $\{v_n\}_{n\in\mathbb{N}}$ as

$$v_n = \tfrac{1}{n} e_n.$$

One can show that $\{v_n\}_{n\in\mathbb{N}}$ is still a Schauder basis and is therefore ω-independent. (The ω-independence of $\{v_n\}_{n\in\mathbb{N}}$ is considered further in Problem 2.69.) Suppose now that we choose

$$x = \alpha e_k$$

for some $k \in \mathbb{N}$ and $\alpha \in \mathbb{R}$. This implies that

$$x = \alpha e_k = \alpha k v_k.$$

Consider the representation of x in terms of the basis $\{v_n\}_{n\in\mathbb{N}}$. We trivially have

$$x = \sum_{n\in\mathbb{N}} a_n v_n$$

where

$$a_n = \begin{cases} \alpha k & \text{for } n = k \\ 0 & \text{otherwise.} \end{cases}$$

(In other words, only one of the $\{a_n\}$ is nonzero, namely a_k.) Taking the norms of the vector x and coefficient sequence $\{a_n\}_{n\in\mathbb{N}}$, we obtain

$$\|x\| = \|\alpha e_k\| = |\alpha|\,\|e_k\| = |\alpha| \quad \text{and}$$
$$\|a\|_{l^2}^2 = \sum_{n\in\mathbb{N}} |a_n|^2 = \alpha^2 k^2 \quad \Rightarrow \quad \|a\|_{l^2} = |\alpha|\,k.$$

From the preceding equations, we can see that, as $k \to \infty$, the norm of the coefficient sequence $\{a_n\}_{n\in\mathbb{N}}$ grows without bound, while the norm of the vector x remains fixed at $|\alpha|$. Thus, the basis $\{v_n\}_{n\in\mathbb{N}}$ is very poorly behaved in the sense that a vector of reasonable size (as measured by the norm) cannot necessarily be represented by a coefficient sequence of reasonable size (as measured by the l^2 norm). Furthermore, as $k \to \infty$, the magnitude of the expansion coefficient $a_k = \alpha k$ grows without bound, in spite of the fact that the vector x has a bounded norm.

Example 2.94. Let $E = \{e_0, e_1, e_2, \ldots\}$ be an orthonormal set in a Hilbert space V. From E, we construct the linearly independent set $X = \{x_n\}$ as

$$x_n = \left(\cos\frac{1}{n}\right) e_0 + \left(\sin\frac{1}{n}\right) e_n, \quad n \in \mathbb{N}.$$

(The set X can also be shown to be ω-independent. See Problem 2.71.) Since $e_0 = \lim_{n\to\infty} x_n$, it follows that $e_0 \in \operatorname{clos}\operatorname{span} X$. Now, suppose that e_0 can be expressed in the form

$$e_0 = \sum_{n\in\mathbb{N}} a_n x_n. \tag{2.7}$$

From the definition of x_n, we can rewrite this as

$$e_0 = \sum_{n\in\mathbb{N}} a_n \left(\left(\cos\frac{1}{n}\right) e_0 + \left(\sin\frac{1}{n}\right) e_n\right).$$

Manipulating this further, we obtain

$$e_0 = \left(\sum_{n\in\mathbb{N}} a_n \left(\cos\frac{1}{n}\right)\right) e_0 + \sum_{n\in\mathbb{N}} a_n \left(\sin\frac{1}{n}\right) e_n.$$

Since $e_n \perp e_0$ for $n \in \mathbb{N}$, we can see that

$$\sum_{n\in\mathbb{N}} a_n \left(\cos\frac{1}{n}\right) = 1 \quad \text{and} \tag{2.8a}$$
$$\sum_{n\in\mathbb{N}} a_n \left(\sin\frac{1}{n}\right) e_n = 0. \tag{2.8b}$$

Since $\{e_n\}_{n\in\mathbb{N}}$ is orthonormal, we can deduce by taking the norm of both sides of (2.8b) that

$$\sum_{n\in\mathbb{N}} |a_n|^2 \sin^2\frac{1}{n} = 0$$

which implies

$$a_n = 0 \quad \text{for } n \in \mathbb{N}.$$

In turn, this implies from (2.7) that $e_0 = 0$. Thus, we have a contradiction, as we know that $e_0 \neq 0$. Therefore, our supposition that e_0 can be represented in the form of (2.7) was incorrect.

The significance of the above example is that it shows that one must be careful in the selection of basis vectors for an infinite-dimensional space. We can encounter problems if the only constraint placed on the basis vectors is that they be linearly independent. In the above example, we have a vector in the closed linear span of a linearly independent set that cannot be represented in terms of an infinite linear combination of elements from the set. The problem in the above example is that $\|x_n - e_0\| \to 0$ as $n \to \infty$. So, although X is linearly independent, we have that, for large n, all of the x_n cluster around e_0. In order to address problems like that above, we place additional constraints on the basis vectors. This leads to the notion of a Riesz basis as defined below.

Definition 2.54 (Riesz basis)**.** A sequence $\{e_n\}_{n \in I}$ is said to be a **Riesz basis** of a Hilbert space V if, for every $x \in V$, there exists a unique scalar sequence $a = \{a_n\}_{n \in I}$ in $l^2(I)$ such that

$$x = \sum_{n \in I} a_n e_n, \tag{2.9}$$

and there exist two strictly positive real constants A and B (which are independent of x) such that

$$A \sum_{n \in I} |a_n|^2 \leq \|x\|^2 \leq B \sum_{n \in I} |a_n|^2 \tag{2.10}$$

(i.e., $A \|a\|_{l^2}^2 \leq \|x\|^2 \leq B \|a\|_{l^2}^2$). Equivalently, we can rewrite (2.10) as $\frac{1}{B} \|x\|^2 \leq \|a\|_{l^2}^2 \leq \frac{1}{A} \|x\|^2$. We sometimes refer to (2.10) as the **Riesz condition**. The constants A and B are referred to as the lower and upper **Riesz bounds**, respectively.

One can show that a Riesz basis is unconditional.

LINEAR INDEPENDENCE. By considering the case of $x = 0$ in (2.9), it trivially follows from (2.10) that $\{e_n\}_{n \in I}$ is ω-independent and, therefore (by Lemma 2.3), linearly independent as well. In particular, suppose that $\{e_n\}$ is not ω-independent. Then, we have that $0 = \sum_{n \in I} a_n e_n$ for some $\{a_n\}$ having at least one nonzero element. From the Riesz condition, we have that $A \sum_{n \in I} |a_n|^2 \leq 0$. The left-hand side of this inequality, however, must be strictly positive, since at least one of the $\{a_n\}$ is nonzero. Thus, we have a contradiction. Therefore, our initial supposition must have been incorrect, and we conclude that $\{e_n\}$ must be ω-independent.

MINIMALITY. A Riesz basis $\{e_n\}_{n \in I}$ is minimal. Suppose that $\{e_n\}_{n \in I}$ is not minimal. Then, one of the basis vectors, say e_M, lies in the closed linear span of $\{e_n\}_{n \in I \setminus \{M\}}$. That is, for every $\varepsilon > 0$, there must exist N and $\{c_n\}_{n \in \{1,2,\ldots,N\} \setminus \{M\}}$ satisfying

$$\left\| e_M - \sum_{n \in \{1,2,\ldots,N\} \setminus \{M\}} c_n e_n \right\| < \varepsilon.$$

Rearranging the left-hand side, we can write

$$\left\| \sum_{n \in \{1,2,\ldots,N\} \cup \{M\}} c'_n e_n \right\| < \varepsilon, \quad \text{where} \quad c'_n = \begin{cases} -c_n & \text{for } n \neq M \\ 1 & \text{for } n = M. \end{cases}$$

From the Riesz condition (2.10), the left-hand side of the preceding inequality is bounded from below. In particular, we have

$$\begin{gathered} \left(A \sum_{n \in \{1,2,\ldots,N\} \cup \{M\}} \left|c'_n\right|^2 \right)^{1/2} < \varepsilon \quad \Rightarrow \\ \sqrt{A} \left(1 + \sum_{n \in \{1,2,\ldots,N\} \setminus \{M\}} \left|c'_n\right|^2 \right)^{1/2} < \varepsilon \quad \Rightarrow \\ \sqrt{A} < \varepsilon. \end{gathered}$$

Since $A > 0$, this inequality cannot hold for every $\varepsilon < 0$. Therefore, we have a contradiction, and $\{e_n\}_{n \in I}$ must be minimal.

DISTANCE BETWEEN VECTORS. The distance between any two basis vectors in $\{e_n\}$ is bounded. To show this, let

$$a_n = \begin{cases} 1 & \text{for } n = k \\ -1 & \text{for } n = m \\ 0 & \text{otherwise.} \end{cases}$$

From the Riesz condition, we have

$$\begin{aligned} & A\,\|a\|_{l^2}^2 \le \|e_k - e_m\|^2 \le B\,\|a\|_{l^2}^2 \\ \Leftrightarrow\quad & 2A \le \|e_k - e_m\|^2 \le 2B \\ \Leftrightarrow\quad & \sqrt{2A} \le \|e_k - e_m\| \le \sqrt{2B}. \end{aligned}$$

Therefore, the distance between basis vectors is at least $\sqrt{2A}$ and at most $\sqrt{2B}$.

NORM OF VECTORS. The norm of each of the basis vectors in $\{e_n\}$ is bounded. To show this, let

$$a_n = \begin{cases} 1 & \text{for } n = m \\ 0 & \text{otherwise.} \end{cases}$$

From the Riesz condition, we then have

$$\begin{aligned} & A\,\|a\|_{l^2}^2 \le \|e_m\|^2 \le B\,\|a\|_{l^2}^2 \\ \Leftrightarrow\quad & A \le \|e_m\|^2 \le B \\ \Leftrightarrow\quad & \sqrt{A} \le \|e_m\| \le \sqrt{B}. \end{aligned}$$

Therefore, the norm of a basis vector is at least $\sqrt{A}$ and at most $\sqrt{B}$.

STABILITY OF BASIS. One consequence of the Riesz condition is that a small difference in norm of vectors implies a small difference in the norm of expansion coefficient sequence. A small perturbation in a vector (as measured by the norm) will always result in a small perturbation in its corresponding expansion coefficient sequence (as measured by the l^2 norm) and vice versa. In other words, the transformation from the vector x to the sequence $\{a_n\}$ is bounded so that a small change in x cannot result in an unbounded change in $\{a_n\}$. (This follows from $A\,\|a\|_{l^2}^2 \le \|x\|^2 \Leftrightarrow \|a\|_{l^2}^2 \le A^{-1}\,\|x\|^2$.) Similarly, the inverse transformation from the sequence $\{a_n\}$ to the vector x is bounded so that a small change in $\{a_n\}$ cannot result in an unbounded change in x. (This follows from $\|x\|^2 \le B\,\|a\|_{l^2}^2$.)

One can view the Riesz condition as a form of partial energy equivalence. Although the l^2 norm of the coefficient sequence is not necessarily equal to the norm of the vector x being represented (as in the case of an orthonormal basis), it is bounded by the norm of x. Furthermore, one can see that an orthonormal basis is a special case of a Riesz basis. In the case of an orthonormal basis, we have (by the Plancherel formula) that

$$\|x\|^2 = \sum_{n\in I} |\langle x, e_n\rangle|^2.$$

This, however, is equivalent to the Riesz condition with $A = B = 1$. In passing, we note that an orthonormal basis can be generated from a Riesz basis by applying the Gram-Schmidt orthonormalization procedure.

Given a particular Riesz basis of an inner product space, there is another related Riesz basis that is frequently of interest. This second basis is known as the dual basis and is formally defined below.

Theorem 2.28 (Existence of dual Riesz basis)**.** *Let $\{e_n\}_{n\in I}$ be a Riesz basis of a Hilbert space V with lower and upper Riesz bounds A and B, respectively. Then, there exists another Riesz basis $\{\tilde{e}_n\}_{n\in I}$ of V with lower and upper Riesz bounds $\frac{1}{B}$ and $\frac{1}{A}$, respectively, such that for all $x \in V$,*

$$x = \sum_{n\in I} \langle x, \tilde{e}_n\rangle\, e_n = \sum_{n\in I} \langle x, e_n\rangle\, \tilde{e}_n.$$

We call $\{\tilde{e}_n\}_{n\in I}$ the ***dual Riesz basis*** *of $\{e_n\}_{n\in I}$.*

It follows from the definition of dual bases that a Riesz basis and its dual are biorthonormal. In practical terms, a dual basis is important because it provides a means for calculating expansion coefficients.

For more information on Riesz bases beyond what is presented herein, the reader is referred to [3].

Example 2.95 (Riesz bases for the Euclidean plane)**.** Consider the Hilbert space $V = \mathbb{R}^2$ (i.e., the Euclidean plane). We have the basis $E = \{e_1, e_2\}$ of V, and we would like to represent the vector $x \in V$ in terms of an expansion of the form

$$x = a_1 e_1 + a_2 e_2$$

where $a_1, a_2 \in \mathbb{R}$. That is, we have the scenario depicted in Figure 2.14(a). Since $\{e_1, e_2\}$ is not orthonormal, we cannot simply compute the expansion coefficients a_1, a_2 through the orthogonal projection of x onto each vector of the basis $\{e_1, e_2\}$. Instead, we must compute the oblique projections as shown in Figure 2.14(a). That is, we need to compute the lengths $\{c_1, c_2\}$. Then, we have

$$a_k = \frac{c_k}{\|e_k\|}.$$

The necessary oblique projections, however, can be calculated through the orthogonal projection of x onto the vectors of the dual basis $\{\tilde{e}_1, \tilde{e}_2\}$. Since $\{e_1, e_2\}$ and $\{\tilde{e}_1, \tilde{e}_2\}$ are biorthonormal, we have that

$$e_1 \perp \tilde{e}_2, \quad e_2 \perp \tilde{e}_1, \quad \text{and} \quad \|\tilde{e}_k\| = \frac{1}{\|e_k\| \cos\theta_k},$$

where θ_k denotes the angle between the vectors e_k and $\tilde{e}_k$. (Recall that the angle θ between two vectors u and v is given by $\cos\theta = \frac{\langle u, v\rangle}{\|u\|\|v\|}$.) This leads to the geometry shown in Figure 2.14(b).

Consider the computation of $\{c_k\}$. From simple geometry, we have

$$a_k = \frac{c_k}{\|e_k\|}, \quad b_k = \frac{\langle x, \tilde{e}_k\rangle}{\|\tilde{e}_k\|}, \quad \text{and} \quad \cos\theta_k = \frac{b_k}{c_k}.$$

Combining these equations, we obtain

$$\begin{aligned} a_k &= \frac{c_k}{\|e_k\|} \\ &= \frac{b_k}{\|e_k\| \cos\theta_k} \\ &= \frac{\langle x, \tilde{e}_k\rangle}{\|e_k\| \, \|\tilde{e}_k\| \cos\theta_k} \\ &= \langle x, \tilde{e}_k\rangle . \end{aligned}$$

Thus, the expansion coefficients a_1, a_2 are calculated in terms of inner products with the dual basis vectors $\tilde{e}_1, \tilde{e}_2$.

In the case of finite-dimensional vector spaces, we can determine the dual of a Riesz basis in a straightforward manner. This is illustrated by the example below.

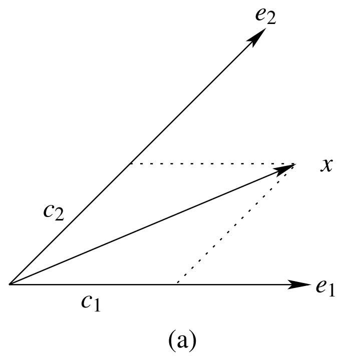
Example 2.96. Let $E = \{e_1, e_2\}$ and $\tilde{E} = \{\tilde{e}_1, \tilde{e}_2\}$ denote dual Riesz bases of the Hilbert space $V = \mathbb{R}^2$. If

$$e_1 = (1, 1) \quad \text{and} \quad e_2 = (1, 2),$$

find $\tilde{E}$.

Proof. Since E is a Riesz basis, we know that each $x = (x_1, x_2) \in V$ has a representation of the form

$$x = \sum_{n=1}^{2} a_n e_n,$$



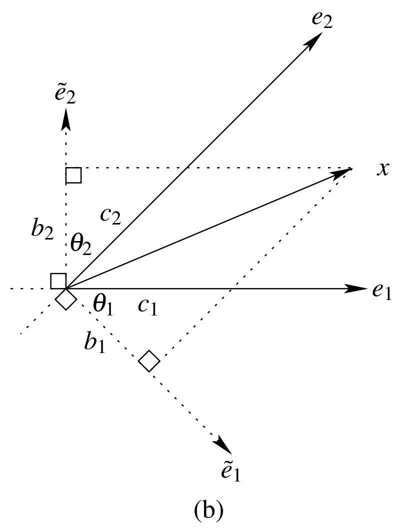


Figure 2.14: Dual bases in the Euclidean plane.

where $a_n = \langle x, \tilde{e}_n \rangle$. Substituting the given values for e_1 and e_2, we have

$$(x_1, x_2) = a_1(1,1) + a_2(1,2) = (a_1 + a_2, a_1 + 2a_2).$$

Rewriting this equation in matrix form, we obtain

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

Solving for a_1 and a_2 in terms of x_1 and x_2, we obtain

$$\begin{aligned} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \end{aligned}$$

The preceding matrix equation, however, is equivalent to

$$\begin{aligned} a_1 &= \langle x, (2,-1) \rangle \\ a_2 &= \langle x, (-1,1) \rangle. \end{aligned}$$

From these equations, we can trivially deduce that

$$\tilde{e}_1 = (2,-1) \quad \text{and} \quad \tilde{e}_2 = (-1,1).$$

□

Example 2.97. Let $E = \{e_1, e_2, e_3\}$ and $\tilde{E} = \{\tilde{e}_1, \tilde{e}_2, \tilde{e}_3\}$ be dual Riesz bases of the Hilbert space $V = \mathbb{R}^3$, where

$$e_1 = (6,3,2), \ e_2 = (-3,-1,-1), \ e_3 = (2,2,1), \ \tilde{e}_1 = (1,1,-4), \ \tilde{e}_2 = (1,2,-6), \ \text{and} \ \tilde{e}_3 = (-1,0,3).$$

Find the expansion of the vector $x = (1,3,-2)$ in terms of elements of E.

Solution. We seek to find an expansion of the form

$$x = \sum_{n=1}^{3} a_n e_n \tag{2.11}$$

where $a_n = \langle x, \tilde{e}_n \rangle$. Computing the expansion coefficients, we obtain

$$\begin{aligned} a_1 &= \langle x, \tilde{e}_1 \rangle \\ &= \langle (1,3,-2), (1,1,-4) \rangle \\ &= 1+3+8 \\ &= 12 \end{aligned}$$

$$\begin{aligned} a_2 &= \langle x, \tilde{e}_2 \rangle \\ &= \langle (1,3,-2), (1,2,-6) \rangle \\ &= 1+6+12 \\ &= 19 \end{aligned}$$

$$\begin{aligned} a_3 &= \langle x, \tilde{e}_3 \rangle \\ &= \langle (1,3,-2), (-1,0,3) \rangle \\ &= -1-6 \\ &= -7 \end{aligned}$$

Finally, substituting the above values for a_1, a_2, a_3 into (2.11), we obtain

$$x = 12e_1 + 19e_2 - 7e_3.$$

One can readily confirm that the right-hand side of the preceding equation, in fact, sums to $x = (1,3,-2)$. □

2.9 l^p Spaces

One very useful class of normed spaces is the l^p spaces. These spaces contain sequences in their underlying sets. Let $p \in [1,\infty) \cup \{\infty\}$. The underlying set is comprised of all sequences $(x_n)_{n \in I}$ such that

$$\begin{cases} \sum_{n \in I} |x_n|^p < \infty & p \in [1,\infty) \\ \sup_{n \in I} |x_n| < \infty & p = \infty. \end{cases}$$

(Note that the cases of $p = 1$, $p = 2$, and $p = \infty$ correspond to the spaces of absolutely-summable, square-summable, and bounded sequences, respectively.) Vector addition and scalar multiplication are defined in the straightforward way. The norm employed is given by

$$\|x\|_p = \begin{cases} \left[\sum_{n \in I} |x_n|^p\right]^{1/p} & p \in [1,\infty) \\ \sup_{n \in I} |x_n| & p = \infty. \end{cases}$$

One can show that the l^p spaces are Banach spaces.

In the case that $p = 2$, we obtain an inner product space l^2, where the inner product is given by

$$\langle x, y \rangle = \sum_{n \in I} x_n y_n^*.$$

Moreover, since the l^p spaces are complete, l^2 is also a Hilbert space.

One might wonder if any relationship exists between the different members of the l^p class of spaces. To gain some insight in this regard, we consider the relationships between absolute summability, square summability, and boundedness of sequences.

Proposition 2.1. *If a sequence $\{a_k\}_{k\in I}$ is absolutely summable, then it is also square summable.*

Proof. Suppose that $\{a_k\}_{k\in I}$ is absolutely summable. Then, we have

$$\sum_{k\in I} |a_k| < \infty$$

which implies (by squaring both sides)

$$\left(\sum_{k\in I} |a_k|\right)^2 < \infty. \tag{2.12}$$

Let us further consider the quantity $(\sum_{k\in I} |a_k|)^2$. We can write

$$\begin{aligned}\left(\sum_{k\in I} |a_k|\right)^2 &= \left(\sum_{k\in I} |a_k|\right)\left(\sum_{l\in I} |a_l|\right)\\ &= \sum_{k\in I}\sum_{l\in I} |a_k|\,|a_l|\\ &= \sum_{k\in I}\sum_{\substack{l\in I\\ l\neq k}} |a_k|\,|a_l| + \sum_{k\in I} |a_k|^2\\ &\geq \sum_{k\in I} |a_k|^2.\end{aligned}$$

Combining this inequality with (2.12), we have

$$\sum_{k\in I} |a_k|^2 \leq \left(\sum_{k\in I} |a_k|\right)^2 < \infty.$$

which implies

$$\sum_{k\in I} |a_k|^2 < \infty.$$

Thus, $\{a_k\}_{k\in I}$ is square summable. □

Proposition 2.2. *If a sequence $\{a_k\}_{k\in I}$ is either absolutely summable or square summable, then it is bounded.*

Proof. Suppose that $\{a_k\}$ is not bounded. Then, there exists some $n\in I$ for which a_n is infinite. This being the case, both of the sums $\sum_{k\in I} |a_k|$ and $\sum_{k\in I} |a_k|^2$ are infinite. Hence, $\{a_k\}$ is neither absolutely nor square summable. Consequently, $\{a_k\}$ can only be absolutely or square summable if it is bounded. □

Thus, from above, we have that 1) absolute summability implies square summability (i.e., $l^1(I)\subset l^2(I)$), and 2) absolute or square summability implies boundedness (i.e., $l^1(I)\subset l^\infty(I)$ and $l^2(I)\subset l^\infty(I)$). In other words, we have the following interesting relationship:

$$l^1(I)\subset l^2(I)\subset l^\infty(I).$$

In other words, the spaces are nested.

2.10 L^p Spaces

Another very useful class of normed spaces is the Lebesgue spaces $L^p(I)$. These spaces contain functions as elements of their underlying sets. Let $p \in [1,\infty) \cup \{\infty\}$. The underlying set is comprised of all measurable functions $x(t)$ defined on I such that

$$\begin{cases} \int_I |x(t)|^p \, dt < \infty & p \in [1,\infty) \\ \operatorname{ess\,sup}_{t \in I} |x(t)| < \infty & p = \infty. \end{cases}$$

(Note that the cases of $p = 1$, $p = 2$, and $p = \infty$ correspond to the spaces of absolutely-integrable, square-integrable, and essentially-bounded functions.) Vector addition and scalar multiplication are defined in the straightforward way. The norm employed is given by

$$\|x\|_p = \begin{cases} \left[\int_I |x(t)|^p \, dt\right]^{1/p} & p \in [1,\infty) \\ \operatorname{ess\,sup}_{t \in I} |x(t)| & p = \infty. \end{cases}$$

One can show that the L^p spaces are Banach spaces.

In the case that $p = 2$, we obtain an inner product space L^2, where the inner product is given by

$$\langle x, y \rangle = \int_I x(t) y^*(t) dt.$$

Moreover, since the L^p spaces are complete, L^2 is also a Hilbert space.

In the case of the L^p spaces, we need to consider a subtlety in the definition of the equality of functions. Recall that a norm must be such that $\|x\| = 0$ if and only if $x = 0$. Now, consider the functions $x_1, x_2 \in L^p(\mathbb{R})$, given by

$$x_1(t) \equiv 0 \quad \text{and} \quad x_2(t) = \begin{cases} 1 & \text{for } t = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, in a pointwise sense, x_1 and x_2 are not equal. Now consider $\|x_1 - x_2\|_{L^p}$. Since x_1 and x_2 differ only on a set of measure zero, $x_1 - x_2$ is zero almost everywhere. From the properties of Lebesgue integral, we know that

$$\|x_1 - x_2\|_{L^p} = \left[\int_{-\infty}^{\infty} |x_1 - x_2|^p \, dt\right]^{1/p} = 0.$$

Thus, we have

$$\|x_1 - x_2\|_{L^p} = 0.$$

From the strict positivity property of a norm, however, this implies that $x_1 = x_2$. So, we can see that equality of functions in L^p spaces cannot be defined as pointwise equality. In fact, we have that two functions are considered equal if they differ only on set of measure zero. Thus, L^p spaces should be viewed as being comprised of equivalence classes of functions, where each equivalence class consists of a set of functions that only differ on a set of measure zero.

As a consequence of the above definition of equality, we can see that functions in L^p spaces are not well defined at any given specific point. When we speak of a function having a particular value at a certain point, we are actually referring to the properties of a single representative function from an equivalence class. For example, when we say that an element of an L^p space is continuous, what we actually mean, in more precise terms, is that a member of its equivalence class is continuous.

Now, we consider the relationship between the various L^p spaces. In the case of l^p spaces, we observed a nesting relationship between spaces. In the case of the L^p spaces, the situation is quite different. For example, for a function f, we can show that $f \in L^1(\mathbb{R})$ does not necessarily imply $f \in L^2(\mathbb{R})$ and vice versa. That is, absolute integrability does not imply square integrability, nor does square integrability imply absolute integrability. There exist functions that are absolutely integrable but not square integrable and vice versa. This point is illustrated by the examples below.

Example 2.98. Let $f(t) = (1+|t|)^{-3/4}$. Show that $f \in L^2(\mathbb{R})$ but $f \notin L^1(\mathbb{R})$.

Solution. First, we consider whether $f \in L^1(\mathbb{R})$. We have

$$\begin{aligned}
\int_{-\infty}^{\infty} |f(t)|\, dt &= \int_{-\infty}^{\infty} f(t) dt \\
&= 2\int_{0}^{\infty} f(t) dt \\
&= 2\int_{0}^{\infty} (1+t)^{-3/4} dt \\
&= 2\left[4(1+t)^{1/4}\right]\Big|_{0}^{\infty} \\
&= \infty.
\end{aligned}$$

Since the above integral does not exist, $f \notin L^1(\mathbb{R})$. Now, we consider whether $f \in L^2(\mathbb{R})$. We have

$$\begin{aligned}
\int_{-\infty}^{\infty} |f(t)|^2\, dt &= \int_{-\infty}^{\infty} f^2(t) dt \\
&= 2\int_{0}^{\infty} f^2(t) dt \\
&= 2\int_{0}^{\infty} (1+t)^{-3/2} dt \\
&= 2\left[-\frac{2}{\sqrt{1+t}}\right]\Big|_{0}^{\infty} \\
&= 2\left[0-(-2)\right] \\
&= 4.
\end{aligned}$$

Therefore, $f \in L^2(\mathbb{R})$. Thus, we have shown that $f \notin L^1(\mathbb{R})$ and $f \in L^2(\mathbb{R})$. □

Example 2.99. Let $f(t) = \chi_{[-1,1]}(t)/\sqrt{|t|}$. Show that $f \in L^1(\mathbb{R})$ but $f \notin L^2(\mathbb{R})$.

Solution. First, we consider whether $f \in L^1(\mathbb{R})$. We have

$$\begin{aligned}
\int_{-\infty}^{\infty} |f(t)|\, dt &= \int_{-\infty}^{\infty} f(t) dt \\
&= 2\int_{0}^{\infty} f(t) dt \\
&= 2\int_{0}^{1} t^{-1/2} dt \\
&= 2\left[2t^{1/2}\right]\Big|_{0}^{1} \\
&= 2(2-0) \\
&= 4.
\end{aligned}$$

Therefore, $f \in L^1(\mathbb{R})$. Now, we consider whether $f \in L^2(\mathbb{R})$. We have

$$\begin{aligned}
\int_{-\infty}^{\infty} |f(t)|^2 dt &= \int_{-\infty}^{\infty} f^2(t) dt \\
&= 2\int_{0}^{\infty} f^2(t) dt \\
&= 2\int_{0}^{1} f^2(t) dt \\
&= 2\int_{0}^{1} t^{-1} dt \\
&= 2\left[\ln t\right]\Big|_0^1 \\
&= 2(0-(-\infty)) \\
&= \infty.
\end{aligned}$$

Since the preceding integral does not converge, $f \notin L^2(\mathbb{R})$. Thus, $f \in L^1(\mathbb{R})$ and $f \notin L^2(\mathbb{R})$. □

From the above examples, we can see that $f \in L^1(\mathbb{R})$ does not imply $f \in L^2(\mathbb{R})$. Nor does $f \in L^2(\mathbb{R})$ imply $f \in L^1(\mathbb{R})$. That is, $L^1(\mathbb{R})$ and $L^2(\mathbb{R})$ are not nested spaces.

Lastly, we observe a few other useful properties of the L^p spaces. Let $C_0^\infty(\mathbb{R})$ denote the set of all infinitely differentiable real-valued functions with compact support. One can show that $C_0^\infty(\mathbb{R})$ is dense in $L^2(\mathbb{R})$. Also, one can show that $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ is dense in $L^2(\mathbb{R})$. Lastly, if $f \in L^2(\mathbb{R})$ and f is compactly supported, then $f \in L^1(\mathbb{R})$.

2.11 Miscellany

An expression of the form $\sum_{n=n_0}^{n_1} a_n z^{-n}$ (where n_0 and n_1 are finite) is called a **Laurent polynomial**. That is, a Laurent polynomial is a polynomial in both z and z^{-1}.

For two integers a and b, we say that a divides b, abbreviated $a \mid b$, if there exists another integer k such that $b = ka$ (i.e., b is an integer multiple of a). For example, we have that $2 \mid 4$ and $3 \mid 27$ while $7 \nmid 13$.

A function $f : \mathbb{R} \to \mathbb{C}$ is said to be **infinitely differentiable** if its derivatives of all orders exist and are continuous. A function $f : \mathbb{R} \to \mathbb{C}$ is said to be ***n*-times continuously differentiable** if its first n derivatives exist and are continuous.

Definition 2.55 (Support)**.** The **support** of a function f, denoted $\operatorname{supp} f$, is the closure of the set

$$\{t : f(t) \neq 0\}.$$

In other words, the support of a function is the smallest closed set that contains all of the points where the function is nonzero.

Example 2.100. Let $x(t) = \chi_{[0,1)}(t)$ and $y(t) = \chi_{[0,1)}(t) + \chi_{[2,3)}(t)$. Then, $\operatorname{supp} x = [0,1]$ and $\operatorname{supp} y = [0,1] \cup [2,3]$.

A function f defined on $\mathbb{R}$ is said to have **compact support** if $\operatorname{supp} f \subset [a,b]$ for $a,b \in \mathbb{R}$. That is, f has compact support if it is only nonzero on some finite interval.

Example 2.101. Let $x_1(t) = \chi_{[0,1)}(t)$ and $x_2(t) = \sin t$. Then, x_1 has compact support (with $\operatorname{supp} x_1 \subset [0,1]$), and x_2 does not have compact support.

Definition 2.56 (Moment)**.** The kth moment of a sequence x defined on $\mathbb{Z}$ is given by

$$m_k = \sum_{n \in \mathbb{Z}} n^k x[n].$$

The kth moment of a function x defined on $\mathbb{R}$ is given by

$$m_k = \int_{-\infty}^{\infty} t^k x(t) dt.$$

A function or sequence f is said to have p **vanishing moments** if the first p moments of the function vanish (i.e., are zero).

The transpose, conjugate transpose, and transposed inverse of a matrix $\boldsymbol{A}$ are denoted, respectively, as $\boldsymbol{A}^T$, $\boldsymbol{A}^\dagger$, and $\boldsymbol{A}^{-T}$. The symbols $\boldsymbol{I}_n$ and $\boldsymbol{J}_n$ denote the $n \times n$ identity and anti-identity matrices, respectively, where the subscript n may be omitted when clear from the context. The (k,l)th **minor** of the $n \times n$ matrix $\boldsymbol{A}$ is the determinant of the $(n-1) \times (n-1)$ matrix formed by removing the kth row and lth column from $\boldsymbol{A}$.

Definition 2.57 (Schur product)**.** The **Schur product** of two $m \times n$ matrices $\boldsymbol{A} = (a_{k,l})$ and $\boldsymbol{B} = (b_{k,l})$, denoted $\boldsymbol{A} \circ \boldsymbol{B}$, is defined as the $m \times n$ matrix $\boldsymbol{A} \circ \boldsymbol{B} = (c_{k,l})$, where $c_{k,l} = a_{k,l} b_{k,l}$. (That is, the Schur product is a matrix containing the scalar products between corresponding elements in two matrices.)

A square matrix $\boldsymbol{A}$ is said to be **unitary** if $\boldsymbol{A}^{-1} = \boldsymbol{A}^\dagger$.

For a polynomial matrix $\boldsymbol{A}(z)$, the notation $\boldsymbol{A}_*(z)$ denotes the matrix obtained by conjugating the polynomial coefficients without conjugating z.

The **paraconjugate** of a matrix $\boldsymbol{A}(z)$ is $\boldsymbol{A}_*^T(z^{-1})$.

If a square matrix $\boldsymbol{A}(z)$ is such that

$$\boldsymbol{A}(z)\boldsymbol{A}_*^T(z^{-1}) = \boldsymbol{I}$$

then $\boldsymbol{A}$ is said to be **paraunitary** (i.e., the matrix times its paraconjugate is the identity matrix). A paraunitary matrix is unitary on the unit circle (i.e., for $|z| = 1$).

The **adjugate** of the $n \times n$ matrix $\boldsymbol{A}$, denoted $\operatorname{Adj}\boldsymbol{A}$, is the $n \times n$ matrix whose (k,l)th entry is given by $(-1)^{k+l}$ $M_{l,k}$ where $M_{l,k}$ is the (l,k)th minor of $\boldsymbol{A}$.

The inverse of a square matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A}^{-1} = \frac{1}{\det\boldsymbol{A}} \operatorname{Adj}\boldsymbol{A}.$$

Now, we define a few special types of matrices that are often quite useful.

Definition 2.58 (Circulant matrix)**.** An $N \times N$ matrix of the form

$$\begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_N \\ a_N & a_1 & a_2 & \cdots & a_{N-1} \\ a_{N-1} & a_N & a_1 & \cdots & a_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & a_4 & \cdots & a_1 \end{bmatrix}$$

is said to be **circulant** (or **right circulant**). An $N \times N$ matrix of the form

$$\begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_N \\ a_2 & a_3 & a_4 & \cdots & a_1 \\ a_3 & a_4 & a_5 & \cdots & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_N & a_1 & a_2 & \cdots & a_{N-1} \end{bmatrix}$$

is said to be **left circulant**.

Definition 2.59 (Toeplitz matrix)**.** An $n \times n$ matrix with each diagonal having entries of equal value is said to be **Toeplitz**. In other words, such a matrix has the form

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{N-2} \\ a_{-2} & a_{-1} & a_0 & \cdots & a_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{-(N-1)} & a_{-(N-2)} & a_{-(N-3)} & \cdots & a_0 \end{bmatrix}.$$

Definition 2.60 (Vandermonde matrix)**.** An $n \times n$ matrix in which the kth column vector is of the form $\begin{bmatrix} a_k^0 & a_k^1 & \cdots & a_k^{n-1} \end{bmatrix}^T$ for $k = 0, 1, \ldots, n-1$ is said to be a **Vandermonde matrix**. In other words, such a matrix has the form

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_0^2 & a_1^2 & a_2^2 & \cdots & a_{n-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0^{n-1} & a_1^{n-1} & a_2^{n-1} & \cdots & a_{n-1}^{n-1} \end{bmatrix}.$$

The determinant of a Vandermonde matrix $\boldsymbol{A}$ is given by

$$\det \boldsymbol{A} = \prod_{0 \le k < l \le n-1} (a_l - a_k).$$

If the columns of the matrix are distinct, the matrix is nonsingular. The DFT matrix is an example of a Vandermonde matrix.

We denote the Mth root of unity as $W_M = e^{-j2\pi/M}$. We denote the $M \times M$ DFT matrix as $\boldsymbol{W}_M$. That is,

$$\boldsymbol{W}_M = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W & \cdots & W^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{M-1} & \cdots & W^{(M-1)^2} \end{bmatrix}. \tag{2.13}$$

2.12 Fourier Analysis

In what follows, we introduce convolution and the Fourier transform.

2.12.1 Convolution

One operation of great importance is the convolution as defined below.

Definition 2.61 (Convolution)**.** Let $f, g \in L^1(\mathbb{R})$. The **convolution** of f and g, denoted $f * g$, is defined as

$$f * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

and $f * g \in L^1(\mathbb{R})$.

The convolution operation has a number of important properties. Some of these properties are given below.

Theorem 2.29 (Properties of convolution)**.** *If $f, g, h \in L^1(\mathbb{R})$, then*

$$\begin{gathered} f * g = g * f, \\ (f * g) * h = f * (g * h), \quad \textit{and} \\ f * (g + h) = f * g + f * h \end{gathered}$$

(i.e., convolution is commutative, associative, and distributive).

Theorem 2.30 (Convolution)**.** *If $f, g \in L^1(\mathbb{R})$, then $f * g \in L^1(\mathbb{R})$ and*

$$\int_{-\infty}^{\infty} |f * g(t)|\, dt \le \left(\int_{-\infty}^{\infty} |f(t)|\, dt \right) \left(\int_{-\infty}^{\infty} |g(t)|\, dt \right).$$

If $f, g \in L^2(\mathbb{R})$, then $f * g \in C_0$ and

$$\sup_{t \in \mathbb{R}} |f * g(t)| \le \|f\|_2 \|g\|_2 .$$

Proof.

$$\begin{aligned}
\int_{-\infty}^{\infty} |f * g(t)|\,dt &= \int_{-\infty}^{\infty} \left| \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \right| dt \\
&\leq \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(\tau)|\,|g(t-\tau)|\,d\tau dt \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(\tau)|\,|g(t-\tau)|\,dt d\tau \quad \text{by Fubini's theorem} \\
&= \int_{-\infty}^{\infty} |f(\tau)|\,d\tau \int_{-\infty}^{\infty} |g(t-\tau)|\,dt \\
&= \int_{-\infty}^{\infty} |f(\tau)|\,d\tau \int_{-\infty}^{\infty} |g(t)|\,dt.
\end{aligned}$$

□

2.12.2 Fourier Transform in $L^1(\mathbb{R})$

Often we are interested in studying the frequency-domain properties of functions. In this regard, the Fourier transform is a useful tool. The Fourier transform is defined as follows.

Definition 2.62 (Fourier transform)**.** For a function $f \in L^1(\mathbb{R})$, the **Fourier transform** of f, denoted $\hat{f}$ or $\mathcal{F}\{f\}$, is the complex-valued function defined on $\mathbb{R}$ given by

$$(\mathcal{F}f)(\omega) = \hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt. \tag{2.14}$$

The inverse of the Fourier transform is computed as given by the theorem below.

Theorem 2.31 (Inverse Fourier transform)**.** *Suppose that a function f and its Fourier transform $\hat{f}$ are both in $L^1(\mathbb{R})$. Then, the inverse Fourier transform of $\hat{f}$ is given by*

$$(\mathcal{F}^{-1}\hat{f})(t) = f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)e^{j\omega t}d\omega.$$

Note that, in the above theorem, it is necessary that both f and $\hat{f}$ be in $L^1(\mathbb{R})$. That is, it is not sufficient simply for f to be in $L^1(\mathbb{R})$. This is due to the fact that there exist functions in $L^1(\mathbb{R})$ having Fourier transforms that are not in $L^1(\mathbb{R})$. For example, $\chi_{[-1/2,1/2]}(\cdot)$ is in $L^1(\mathbb{R})$, but its Fourier transform $\operatorname{sinc}(\frac{1}{2}\cdot)$ is not in $L^1(\mathbb{R})$.

The Fourier transform has a number of important properties. Some of these properties are given in the theorems below.

Theorem 2.32 (Linearity of Fourier transform)**.** *If $f,g \in L^1(\mathbb{R})$ and $c \in \mathbb{C}$, then*

$$[f+g]\widehat{\ } = \hat{f} + \hat{g} \quad \text{and} \quad [cf]\widehat{\ } = c\hat{f}.$$

Theorem 2.33 (Fourier transform properties)**.** *If $f \in L^1(\mathbb{R})$, then*

1. *$[f(\cdot - t_0)]\widehat{\ }(\omega) = e^{-j\omega t_0}\hat{f}(\omega)$ (translation);*
2. *$[e^{j\omega_0 \cdot} f(\cdot)]\widehat{\ }(\omega) = \hat{f}(\omega - \omega_0)$ (modulation);*
3. *$[f(a\cdot)]\widehat{\ }(\omega) = \frac{1}{|a|}\hat{f}(\omega/a)$ (scaling); and*
4. *$[f^*(\cdot)]\widehat{\ }(\omega) = \hat{f}^*(-\omega)$ (conjugation).*

Theorem 2.34 (Riemann-Lebesgue lemma)**.** *If $f \in L^1(\mathbb{R})$, then $\hat{f}$ is continuous on $\mathbb{R}$ and*

$$\lim_{|\omega|\to\infty} \hat{f}(\omega) = 0.$$

*(This is known as the **Riemann-Lebesgue lemma**.)*

Theorem 2.35 (Convolution theorem)**.** *If $f, g \in L^1(\mathbb{R})$, then*

$$[f * g]\hat{}(\omega) = \hat{f}(\omega)\hat{g}(\omega).$$

Theorem 2.36 (Parseval relation)**.** *If $f, g \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, then*

$$\int_{-\infty}^{\infty} f(t)g^*(t)dt = \tfrac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)\hat{g}^*(\omega)d\omega.$$

Theorem 2.37 (Differentiation)**.** *Let f be a continuous n-times piecewise differentiable function such that $f, f^{(1)}, \ldots, f^{(n)} \in L^1(\mathbb{R})$, and*

$$\lim_{|t|\to\infty} f^{(n)}(t) = 0 \quad \text{for } n = 0, 1, \ldots, n-1.$$

Then,

$$\widehat{f^{(n)}} = (j\omega)^n \hat{f}.$$

Theorem 2.38 (Moments)**.** *The kth moment μ_k of a function x is given by*

$$\mu_k = j^k \hat{x}^{(k)}(0),$$

where $\hat{x}^{(k)}$ denotes the kth order derivative of $\hat{x}$.

Proof. From the definition of the Fourier transform, we have

$$\hat{x}(\omega) = \int_{\mathbb{R}} x(t)e^{-j\omega t}dt$$

Differentiating both sides of this equation k times with respect to ω, we obtain

$$\begin{aligned}\hat{x}^{(k)}(\omega) &= \left(\tfrac{d}{d\omega}\right)^k \int_{\mathbb{R}} x(t)e^{-j\omega t}dt \\ &= \int_{\mathbb{R}} x(t)\left(\tfrac{d}{d\omega}\right)^k [e^{-j\omega t}]dt \\ &= \int_{\mathbb{R}} x(t)(-jt)^k e^{-j\omega t}dt \\ &= (-j)^k \int_{\mathbb{R}} t^k x(t)e^{-j\omega t}dt.\end{aligned}$$

Substituting $\omega = 0$ into the preceding equation, we have

$$\begin{aligned}\hat{x}^{(k)}(0) &= (-j)^k \int_{\mathbb{R}} t^k x(t)dt \\ &= (-j)^k \mu_k.\end{aligned}$$

Rearranging, we obtain

$$\mu_k = j^k \hat{x}^{(k)}(0).$$

□

2.12.3 Fourier Transform in $L^2(\mathbb{R})$

Often we need to work with the Hilbert space $L^2(\mathbb{R})$. Therefore, we might wish to employ the Fourier transform in this context. Unfortunately, since $L^2(\mathbb{R})$ is not completely contained in $L^1(\mathbb{R})$, some functions in $L^2(\mathbb{R})$ are not absolutely integrable. As a consequence, the definition of the Fourier transform for $L^1(\mathbb{R})$ functions, as given by (2.14), cannot be used directly in the context of $L^2(\mathbb{R})$. For any function $f \in L^2(\mathbb{R}) \setminus L^1(\mathbb{R})$, the integral in (2.14) is not well defined. So, we must develop an alternative definition of the Fourier transform that can be used in the case of $L^2(\mathbb{R})$.

Recall from earlier that $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ is dense in $L^2(\mathbb{R})$. This implies that any function in $L^2(\mathbb{R})$ can be represented with arbitrary accuracy using functions from $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$. Suppose that we have a function $f \in L^2(\mathbb{R})$. Such a function can always be expressed as the limit of a sequence $\{f_n\}_{n\in\mathbb{N}}$ of functions in $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$. That is, we can express f as

$$f = \lim_{n\to\infty} f_n.$$

Clearly, each f_n is absolutely integrable since $f_n \in L^1(\mathbb{R}) \cap L^2(\mathbb{R}) \subset L^1(\mathbb{R})$. Since f_n is absolutely integrable, the Fourier transform $\hat{f}_n$ of f_n exists. So, it would seem natural to define the Fourier transform of the $L^2(\mathbb{R})$ function f as the limit of the sequence $\{\hat{f}_n\}_{n\in\mathbb{N}}$ of Fourier transforms. In fact, this is precisely what we do. The above line of reasoning leads us to the definition of the Fourier transform in $L^2(\mathbb{R})$ as given below.

Definition 2.63 (Fourier transform in $L^2(\mathbb{R})$)**.** Let $f \in L^2(\mathbb{R})$ and let $\{f_n\}_{n\in\mathbb{N}}$ be a sequence of functions in $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ such that $\lim_{n\to\infty} f_n = f$ where the convergence is with respect to the L^2 norm. Then, we define the Fourier transform $\hat{f}$ of f as

$$\hat{f} = \lim_{n\to\infty} \hat{f}_n,$$

where the convergence is with respect to the norm in $L^2(\mathbb{R})$.

We can calculate the Fourier transform of a $L^2(\mathbb{R})$ function as given by the theorem below.

Theorem 2.39 (Fourier transform in $L^2(\mathbb{R})$)**.** *Let $f \in L^2(\mathbb{R})$. Then*

$$\hat{f}(\omega) = \lim_{n\to\infty} \int_{-n}^{n} f(t) e^{-j\omega t} dt$$

where the convergence is with respect to the norm in $L^2(\mathbb{R})$. (Essentially, we have that $\hat{f}(\omega) = \lim_{n\to\infty} \int_{-\infty}^{\infty} f_n(t) e^{-j\omega t} dt$, where $f_n(t) = f(t)\operatorname{rect}(\frac{t}{2n})$, $n \in \mathbb{N}$, and $f_n \in L^1(\mathbb{R})$.)

One can show that the inverse Fourier transform can be calculated as given by the theorem below.

Theorem 2.40 (Inverse Fourier transform in $L^2(\mathbb{R})$)**.** *Let $f \in L^2(\mathbb{R})$. Then,*

$$f(t) = \lim_{n\to\infty} \frac{1}{2\pi} \int_{-n}^{n} \hat{f}(\omega) e^{j\omega t} d\omega,$$

where the convergence is with respect to the norm in $L^2(\mathbb{R})$.

Now, we consider a few important properties of the Fourier transform in the context of $L^2(\mathbb{R})$. The first of these is given by the theorem below.

Theorem 2.41 (Parseval relation)**.** *If $f, g \in L^2(\mathbb{R})$, then*

$$\int_{-\infty}^{\infty} f(t) g^*(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) \hat{g}^*(\omega) d\omega$$

or equivalently

$$\langle f, g \rangle = \tfrac{1}{2\pi} \langle \hat{f}, \hat{g} \rangle .$$

The above theorem is quite significant. It shows that the Fourier transform preserves inner products (up to a scale factor). In other words, the angles between vectors are preserved by the Fourier transform.

Now we consider another fundamentally important result concerning the Fourier transform as given below.

Theorem 2.42 (Plancherel)**.** *For each $f \in L^2(\mathbb{R})$, $\hat{f} \in L^2(\mathbb{R})$ and*

$$\|f\|_2^2 = \tfrac{1}{2\pi} \left\|\hat{f}\right\|_2^2.$$

It follows from the above theorem that $\hat{f} \in L^2(\mathbb{R})$ if and only if $f \in L^2(\mathbb{R})$. The significance of the above theorem is twofold. First, it shows that the Fourier transform preserves the norm of functions (up to a scale factor). Secondly, it shows that the Fourier transform is a mapping of $L^2(\mathbb{R})$ onto itself.

Lastly, we note that the convolution, linearity, translation, modulation, scaling, conjugation, and differentiation properties also hold for the $L^2(\mathbb{R})$ definition of the Fourier transform.

2.12.4 Time and Frequency Resolution

As it turns out, there is a fundamental limit as to how much a function can be simultaneously concentrated in both time and frequency. More formally, this result can be stated in terms of the theorem below.

Theorem 2.43 (Heisenberg uncertainty)**.** *The temporal variance σ_t and the frequency variance σ_ω of $f \in L^2(\mathbb{R})$ satisfy*

$$\sigma_t^2 \sigma_\omega^2 \geq \tfrac{1}{4}$$

and equality holds if and only if f is of the form

$$f(t) = ae^{j\lambda t - b(t-u)^2}$$

where $u, \lambda \in \mathbb{R}$, $a, b \in \mathbb{C}$, and

$$\begin{aligned}
u &= \tfrac{1}{\|f\|^2} \int_{-\infty}^{\infty} t\,|f(t)|^2\,dt \\
\lambda &= \tfrac{1}{2\pi\|f\|^2} \int_{-\infty}^{\infty} \omega \left|\hat{f}(\omega)\right|^2 d\omega, \\
\sigma_t^2 &= \tfrac{1}{\|f\|^2} \int_{-\infty}^{\infty} (t-u)^2\,|f(t)|^2\,dt, \quad \text{and} \\
\sigma_\omega^2 &= \tfrac{1}{2\pi\|f\|^2} \int_{-\infty}^{\infty} (\omega-\lambda)^2 \left|\hat{f}(\omega)\right|^2 d\omega.
\end{aligned}$$

2.12.5 Continuous-Time Fourier Series

Theorem 2.44 (Continuous-time Fourier series)**.** *A T-periodic function f defined on $\mathbb{R}$ can be represented in the form*

$$f(t) = \sum_{k\in\mathbb{Z}} a_k e^{jk\omega_0 t},$$

where $\omega_0 = \frac{2\pi}{T}$,

$$a_k = \tfrac{1}{T} \int_T f(t) e^{-jk\omega_0 t}\,dt,$$

and $\int_T$ denotes integration over an interval of length T. Such a representation is known as a Fourier series.

2.12.6 Discrete-Time Fourier Series

Theorem 2.45 (Discrete-time Fourier series)**.** *An N-periodic sequence f defined on $\mathbb{Z}$ can be represented in the form*

$$f[n] = \sum_{k=0}^{N-1} a_k e^{jk\omega_0 n}$$

where $\omega_0 = \frac{2\pi}{N}$ and

$$a_k = \tfrac{1}{N} \sum_{n=0}^{N-1} f[n] e^{-jk\omega_0 n}.$$

Such a representation is known as a Fourier series.

2.12.7 Discrete-Time Fourier Transform

Definition 2.64 (Discrete-time Fourier transform)**.** The (discrete-time) Fourier transform $\hat{f}$ of a sequence f is defined as

$$\hat{f}(\omega) = \sum_{n\in\mathbb{Z}} f[n] e^{-j\omega n}.$$

Theorem 2.46 (Inverse discrete-time Fourier transform)**.** *The inverse (discrete-time) Fourier transform f of $\hat{f}$ is given by*

$$f[n] = \tfrac{1}{2\pi} \int_{2\pi} \hat{f}(\omega) e^{j\omega n} d\omega.$$

Theorem 2.47 (Properties of discrete-time Fourier transform)**.** *The discrete-time Fourier transform has the following properties:*

1. $(f[\cdot - n_0])^{\wedge}(\omega) = e^{-j\omega n_0}\hat{f}(\omega)$ *(translation);*
2. $(e^{j\omega_0 \cdot} f[\cdot])^{\wedge}(\omega) = \hat{f}(\omega - \omega_0)$ *(modulation); and*
3. $(f^*[\cdot])^{\wedge}(\omega) = \hat{f}^*(-\omega)$ *(conjugation).*

Theorem 2.48 (Moments)**.** *The kth moment m_k of a sequence x is given by*

$$m_k = j^k \hat{x}^{(k)}(0),$$

where $\hat{x}^{(k)}$ denotes the kth order derivative of $\hat{x}$.

Proof. From the definition of the (discrete-time) Fourier transform, we have

$$\hat{x}(\omega) = \sum_{n\in\mathbb{Z}} c[n] e^{-jn\omega}.$$

Differentiating both sides of the equation k times, we obtain

$$\begin{aligned}\hat{x}^{(k)}(\omega) &= \sum_{n\in\mathbb{Z}} (-jn)^k c[n] e^{-jn\omega} \\ &= (-j)^k \sum_{n\in\mathbb{Z}} n^k c[n] e^{-jn\omega}.\end{aligned}$$

Evaluating the above equation at $\omega = 0$ and using the definition of the kth moment of c, we can write

$$\begin{aligned}\hat{x}^{(k)}(0) &= (-j)^k \sum_{n\in\mathbb{Z}} n^k c[n] \\ &= (-j)^k m_k.\end{aligned}$$

Rearranging the preceding equation, we have

$$m_k = (\tfrac{1}{-j})^k \hat{x}^{(k)}(0) = j^k \hat{x}^{(k)}(0).$$

□

2.12.8 $\mathcal{Z}$ Transform

Definition 2.65 ($\mathcal{Z}$ transform)**.** The $\mathcal{Z}$ **transform** of a sequence x defined on $\mathbb{Z}$ is denoted as $\mathcal{Z}x$ or $X(z)$ and given by

$$X(z) = \sum_{n \in \mathbb{Z}} x[n] z^{-n}.$$

Theorem 2.49 (Inverse $\mathcal{Z}$ transform)**.** *The **inverse** $\mathcal{Z}$ **transform** is given by*

$$x[n] = \frac{1}{2\pi j} \oint X(z) z^{n-1} dz,$$

where $\oint$ denotes an integration in a counterclockwise direction around a closed path in the complex plane.

2.12.9 Miscellany

Theorem 2.50 (Poisson summation formula)**.** *If $f \in L^2(\mathbb{R})$ and f is once differentiable, then*

$$\sum_{n \in \mathbb{Z}} f(t+n) = \sum_{n \in \mathbb{Z}} \hat{f}(2\pi n) e^{j2\pi nt}.$$

*This relationship is known as the **Poisson summation formula**. In the special case that $t = 0$, we have*

$$\sum_{n \in \mathbb{Z}} f(n) = \sum_{n \in \mathbb{Z}} \hat{f}(2\pi n).$$

Proof. Let $g(t) = \sum_{n \in \mathbb{Z}} f(t+n)$. We observe that g is periodic with period one. So, we can express g in terms of a Fourier series

$$g(t) = \sum_{n \in \mathbb{Z}} \hat{f}(2\pi n) e^{j2\pi nt}.$$

Combining this expression for g with the one above shows the desired result. □

Theorem 2.51 (Shannon sampling theorem)**.** *Let $f \in L^2(\mathbb{R})$. If $\operatorname{supp} \hat{f} \subset [-\pi/T, \pi/T]$, then*

$$f(t) = \sum_{n \in \mathbb{Z}} f(nT) h_T(t - nT)$$

where

$$h_T(t) = \operatorname{sinc}(\pi t/T).$$

Proof. See [8, p. 44, Theorem 3.1]. □

2.13 Problems

2.1 Determine the infimum, supremum, minimum, and maximum of the following subsets of $\mathbb{R}$:
(a) $A = [-1, 0]$;
(b) $A = (0, \infty)$;
(c) $A = (-\infty, 4]$;
(d) $A = (0, 3)$; and
(e) $A = (-2, 5] \cup [7, 8]$.

2.2 Let S denote a subset of $\mathbb{R}$. In each of the cases below, determine whether S is open and/or closed, and if S is not closed find the closure of S in $\mathbb{R}$.
(a) $S = [-1, 1]$;
(b) $S = (0, 2]$;
(c) $S = (3, 4)$;
(d) $S = \emptyset$;
(e) $S = (-\infty, 2) \cup (3, \infty)$; and
(f) $S = [3, 4) \cup (4, 5)$.

2.3 Show that if $\{x_n\}$ is a sequence such that $d(x_{n+1}, x_n) < Cr^n$ for $C, r \in \mathbb{R}$ and $0 \leq r < 1$, then $\{x_n\}$ is a Cauchy sequence.

2.4 Consider the set X of continuous real-valued functions on $[0, 1]$ with metric

$$d(x,y) = \int_0^1 |x(t) - y(t)| \, dt.$$

Let the sequence $\{x_n\}_{n \in \mathbb{N}}$ be defined by

$$x_n(t) = \begin{cases} n & \text{if } 0 \leq t \leq n^{-2} \\ t^{-1/2} & \text{if } n^{-2} \leq t \leq 1. \end{cases}$$

(a) Show that $\{x_n\}$ is Cauchy.
(b) Show that $\{x_n\}$ is not convergent.

2.5 Let $X = \mathbb{R}^2$ and let

$$d(x,y) = \left[|x_1 - y_1|^{1/2} + |x_2 - y_2|^{1/2} \right]^2.$$

Determine whether (X, d) is a metric space.

2.6 Let (X, d) be a metric space. Using the triangle inequality, show that:
(a) for all $x, y, z \in X$, $|d(x,z) - d(y,z)| \leq d(x,y)$; and
(b) for all $w, x, y, z \in X$, $|d(x,y) - d(z,w)| \leq d(x,z) + d(y,w)$.

2.7 Let X be any nonempty set. Define

$$d(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise.} \end{cases}$$

(a) Show that d is a metric on X.
(b) Show that (X, d) is a complete metric space.

2.8 Let X be the set of all ordered n-tuples $x = (x_1, x_2, \ldots, x_n)$ of real numbers and $d(x,y) = \max_i |x_i - y_i|$ where $y = (y_1, y_2, \ldots, y_n)$. Show that (X, d) is complete.

2.9 Define the metric $d_p(x,y)$ on $\mathbb{R}^2$ by

$$d_p(x,y) = \begin{cases} \left(|x_1 - y_1|^p + |x_2 - y_2|^p\right)^{1/p} & \text{for } 1 \leq p < \infty \\ \max\{|x_1 - y_1|, |x_2 - y_2|\} & \text{for } p = \infty. \end{cases}$$

Show that, for all $x, y \in \mathbb{R}^2$,

$$\begin{gathered} d_\infty(x,y) \leq d_p(x,y) \leq d_1(x,y), \\ d_p(x,y) \leq 2 d_\infty(x,y), \text{ and} \\ d_1(x,y) \leq 2 d_p(x,y). \end{gathered}$$

[Hint: Let $\lambda \in [0, 1]$ and define $f(p) = \lambda^p + (1 - \lambda)^p$. Show that $f(p) \leq 1$ for $1 \leq p < \infty$ by computing $\frac{df}{dp}$.]

2.10 Show that the strict positivity, symmetry, and triangle inequality properties of a metric imply the nonnegativity property.

2.11 Let d be a metric on X. Determine for which values of the constant k each of the following functions is a metric on X:
(a) kd;
(b) $d+k$.

2.12 Identify the closure of
(a) the integers on $\mathbb{R}$;
(b) the rational numbers on $\mathbb{R}$;
(c) the complex numbers with rational real and imaginary parts in $\mathbb{C}$.

2.13 Show that if $d(x,y)$ is a metric on X, then each of the following is also a metric on X:
(a) $d'(x,y) = \frac{d(x,y)}{1+d(x,y)}$;
(b) $d'(x,y) = \min(1, d(x,y))$.

2.14 Let $X = (0,1) \subset \mathbb{R}$ with the usual metric d (i.e., $d(x,y) = |x-y|$). Show that (X,d) is not complete.

2.15 Let V be the set of all infinite sequences of real numbers $x = (x_1, x_2, \ldots)$. Show that $(V, \mathbb{R}, +, \cdot)$ satisfies the axioms of a vector space (where vector addition and scalar multiplication are defined in the usual way for sequences).

2.16 Show that the functions $f_n(t) = e^{nt}$ for $n = 0, 1, 2, \ldots$ form a linearly independent set. [Hint: Consider $a_1 e^{m_1 t} + \cdots + a_n e^{m_n t} \equiv 0$ and differentiate $(n-1)$ times.]

2.17 Show that the set of all $m \times n$ matrices can be viewed as a vector space.

2.18 Show that the set of all even functions defined on the real line can be viewed as a vector space.

2.19 Let V and W be disjoint subspaces of the vector space U. Show that for all $x \in V \setminus \{0\}$ and all $y \in W \setminus \{0\}$, $\{x, y\}$ is linearly independent.

2.20 If V and W are subspaces of a vector space U, show that $V \cap W$ is a subspace of U, but $V \cup W$ need not be one.

2.21 If W is any nonempty subset of a vector space V, show that $\operatorname{span} W$ is a subspace of V.

2.22 For each of the cases given below, determine whether $S \subset \mathbb{R}^3$ constitutes a subspace of $\mathbb{R}^3$:
(a) $S = \{x = (x_1, x_2, x_3) : x_1 = x_3, x_2 = 0\}$;
(b) $S = \{x = (x_1, x_2, x_3) : x_1, x_2, x_3 \geq 0\}$; and
(c) $S = \{x = (x_1, x_2, x_3) : 2x_1 + x_2 - 5x_3 = 1\}$.

2.23 If V and W are vector spaces, show that $V \oplus W$ is a vector space.

2.24 Show that if V and W are subspaces of a vector space, then $V + W$ is a subspace.

2.25 Let V be an n-dimensional vector space with the basis $E = \{e_1, e_2, \ldots, e_n\}$. Show that the representation of any $x \in V$ as a linear combination of vectors in E is unique.

2.26 Let V be the vector space $\mathbb{R}^4$. Let S be the subset of V given by

$$S = \{x = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4 : x_1 + x_2 + x_3 + x_4 = c\}.$$

Determine for what values of c (if any) the subset S is a vector subspace of V.

2.27 Let V be the vector space comprised of all real-valued sequences. (a) Show that the subset S_1 of V comprised of all sequences with a finite number of nonzero entries is a vector subspace of V. (b) Show that the subset S_2 of V comprised of all sequences with an infinite number of nonzero entries is not a vector subspace of V.

2.28 Let V be the vector space $L^2[0,2\pi]$ and let A be the set of all functions $x_n(t) = e^{jnt}$, $n = 0,1,2,\ldots$. Show that A is linearly independent. [Hint: Consider $a_1 e^{jn_1 t} + \cdots + a_m e^{jn_m t} \equiv 0$ and differentiate $(m-1)$ times.]

2.29 Let V be an N-dimensional vector space. Show that every set of $N+1$ vectors is linearly dependent.

2.30 Let V be the real vector space comprised of all functions of the form $x(t) = \alpha\cos(2t+\theta)$ where $\alpha, \theta \in \mathbb{R}$. Show that $E = \{\cos 2t, \sin 2t\}$ is a basis of V.

2.31 Let X be the normed space of the real numbers with the usual norm (i.e., $\|x\| = |x|$). Show that if $\{x_n\}$ and $\{y_n\}$ are Cauchy sequences in X, then $\{v_n\} = \{x_n + y_n\}$ is a Cauchy sequence.

2.32 Show that the sum of two Cauchy sequences (in a normed space) is a Cauchy sequence.

2.33 Let V be the vector space of all ordered pairs $x = (x_1, x_2)$ of real numbers. Show that norms on V are defined by each of the following:
(a) $\|x\|_1 = |x_1| + |x_2|$;
(b) $\|x\|_2 = (x_1^2 + x_2^2)^{1/2}$; and
(c) $\|x\|_\infty = \max\{|x_1|, |x_2|\}$.

2.34 Show that, if $(V, \|\cdot\|)$ is a normed vector space, then

$$|\|x\| - \|y\|| \leq \|x - y\|$$

for all $x, y \in V$.

2.35 Consider the vector space $V = L^p[0,1]$ of all functions with

$$\|x\|_p = \int_0^1 |x(t)|^p \, dt < \infty$$

where $0 < p < 1$. (a) Show that $\|x\|_p$ is not a norm on V. (b) Show that $d_p(x,y) = \|x-y\|_p$ is a metric on V. [Hint: If $0 \leq a \leq 1$, then $a \leq a^p \leq 1$.]

2.36 Show that in a normed space V the following inequality holds for all $x, y \in V$:

$$2\|x\|^2 - 4\|x\|\|y\| + 2\|y\|^2 \leq \|x+y\|^2 + \|x-y\|^2 \leq 2\|x\|^2 + 4\|x\|\|y\| + 2\|y\|^2.$$

2.37 Show that for any inner product space V

$$\|z-x\|^2 + \|z-y\|^2 = \tfrac{1}{2}\|x-y\|^2 + 2\left\|z - \tfrac{1}{2}(x+y)\right\|^2$$

for all $x, y, z \in V$. This relationship is known as **Apollonius' identity**.

2.38 Show that for any inner product space V

$$2(\langle x,y\rangle + \langle y,x\rangle) = \|x+y\|^2 - \|x-y\|^2$$

for all $x, y \in V$.

2.39 If $\langle\cdot,\cdot\rangle_1$ and $\langle\cdot,\cdot\rangle_2$ are two inner products on a vector space V, show that $\langle\cdot,\cdot\rangle = \langle\cdot,\cdot\rangle_1 + \langle\cdot,\cdot\rangle_2$ is also an inner product on V.

2.40 Show that, in an inner product space V, if $x \neq y$ and $\|x\| = \|y\| = 1$, then $\|x+y\| \leq 2$, where $x, y \in V$ (i.e., the norm in an inner product space is strictly convex).

2.41 Let V be an inner product space. Show that, for any $x, y \in V$,

$$\|x \pm y\|^2 = \|x\|^2 \pm 2\operatorname{Re}\langle x,y\rangle + \|y\|^2 \quad \text{and}$$
$$\|x \pm jy\|^2 = \|x\|^2 \pm 2\operatorname{Im}\langle x,y\rangle + \|y\|^2 .$$

2.42 Show that, for any inner product space V, if $\sum_{n\in\mathbb{N}} x_n = x$ (where $x \in V$), then $\sum_{n\in\mathbb{N}} \langle x_n, y\rangle = \langle x, y\rangle$ for any $y \in V$.

2.43 For any real inner product space V, show that

$$\|x-y\| + \|y-z\| = \|x-z\|, \quad x,y,z \in V,$$

if and only if $y = ax + (1-a)z$ for some $a \in [0,1]$. [Hint: Use the Schwarz inequality.]

2.44 Let $\{e_k\}_{k\in\mathbb{N}}$ be an orthonormal basis of an inner product space V. Consider the expansion of any arbitrary $x \in V$ given by

$$x = \sum_{k\in\mathbb{N}} a_k e_k$$

where $a_k = \langle x, e_k\rangle$. Show that the number n of a_k such that $|a_k| > 1/m$ must satisfy $n < m^2 \|x\|^2$.

2.45 For the given subset S of the inner product space V in each of the cases below, use the Gram-Schmidt process to find an orthonormal basis for $\operatorname{span} S$:
(a) $V = \mathbb{R}^4$, $S = \{(1,-1,-1,1), (1,-2,1,2), (1,1,-3,1)\}$; and
(b) $V = L^2[0,1]$, $S = \{1, t, t^2\}$.

2.46 Consider the inner product space $V = C[-1,1]$ with inner product $\langle x,y\rangle = \int_{-1}^{1} x(t)y(t)dt$. Let W_e and W_o denote the subspaces of V consisting of the even and odd functions, respectively. Show that $W_e^\perp = W_o$. (Note: Here, the notation $C[-1,1]$ denotes the space of continuous real-valued functions on $[-1,1]$. The inner product is given by $\langle x,y\rangle = \int_{-1}^{1} x(t)y(t)dt$.)

2.47 Show that, in any inner product space V,

$$\langle x, ay+bz\rangle = a^* \langle x,y\rangle + b^* \langle x,z\rangle$$

for all $x,y,z \in V$ and all $a,b \in \mathbb{C}$.

2.48 Let $\{e_n\}_{n\in\mathbb{Z}}$ be an orthonormal set in a Hilbert space. Show that the vectors

$$x = \sum_{n\in\mathbb{Z}} a_n e_n \quad \text{and} \quad y = \sum_{n\in\mathbb{Z}} (-1)^n a_{p-n}^* e_n$$

are orthogonal if p is odd.

2.49 Find the orthogonal projection of the vector x in the Hilbert space V onto the subspace with orthonormal basis E, for each of the following:
(a) $V = \mathbb{R}^4$, $E = \left\{ \left(\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}\right), \left(\frac{2}{\sqrt{21}}, \frac{4}{\sqrt{21}}, 0, \frac{1}{\sqrt{21}}\right), \left(-\frac{3}{\sqrt{18}}, \frac{1}{\sqrt{18}}, -\frac{2}{\sqrt{18}}, \frac{2}{\sqrt{18}}\right) \right\}$, $x = (1,0,-1,1)$;
(b) $V = L^2[0,1]$, $E = \{\sqrt{2}\cos 2\pi t, \sqrt{2}\sin 2\pi t\}$, $x(t) = t+1$;
(c) $V = \mathbb{R}^4$, $E = \left\{ \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right), \left(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) \right\}$, $x = (-2,4,6,0)$;
(d) $V = \mathbb{R}^4$, $E = \left\{ \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) \left(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) \left(\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}\right) \right\}$, $x = (0,1,0,1)$; and
(e) V is the space of real-valued polynomial functions defined on $[-1,1]$, $E = \{\frac{1}{\sqrt{2}}, \sqrt{\frac{3}{2}}t\}$, $x = t^2+1$.

2.50 If x and y are vectors in the inner product space V, compute the inner product $\langle x, y\rangle$ and indicate if $x \perp y$, for each of the following:
(a) $V = L^2[0,1]$, $x(t) = t^2 - 1$, $y(t) = t+1$;
(b) $V = L^2(\mathbb{R})$, $x(t) = e^{-|t|}$, $y(t) = e^{-2|t|}$;
(c) $V = \mathbb{R}^3$, $x = (1,3,1)$, $y = (2,-3,7)$;
(d) $V = L^2[-1,1]$, $x(t) = t$, $y(t) = t^2 - 2$;
(e) $V = l^2(\mathbb{Z})$, $\{x_n\} = 3^{-n}u[n]$, $\{y_n\} = 2^{-n}u[n]$, where $u[n]$ denotes the unit-step sequence; and
(f) $V = L^2[-1,1]$, $x(t) = t^2 - 1$, $y(t) = t+1$.

2.51 Show that a countable orthonormal set $E = \{e_n\}_{n\in I}$ in an inner product space is linearly independent.

2.52 Show that (a) if V and W are orthogonal subspaces, then V and W are disjoint; and (b) the converse is not necessarily true.

2.53 Implement the Gram-Schmidt algorithm in a programming language of your choice. Then, use your program to find an orthonormal basis for the space spanned by each of the following linearly independent sets:
(a) $A = \{(1,0,1,-2),(0,0,-1,1),(1,0,2,-1)\}$; and
(b) $A = \{(-2,-1,0,1,2),(2,-2,-1,0,1),(1,2,-2,-1,0),(0,1,2,-2,-1)\}$.

2.54 Consider the space of functions defined on $[0,\infty)$ with the inner product

$$\langle x, y\rangle = \int_{\mathbb{R}} f(t) g^*(t) e^{-t^2} dt.$$

Apply the Gram-Schmidt process to the sequence $\{f_n\}_{n\in\mathbb{Z}^*}$ of functions, where

$$f_n(t) = t^n.$$

The resulting polynomials are known as the **Laguerre polynomials**.

2.55 Let $X = \{x_n\}$ denote a subset of the inner product space V. For each of the following, determine whether the set X is orthogonal/orthonormal:
(a) $V = \mathbb{R}^3$, $X = \{(3,-1,-2),(2,0,3),(-3,-13,2)\}$;
(b) $V = L^2[0,1]$, $X = \{1,t,t^2\}$;
(c) $V = L^2[-1,1]$, $X = \{1,t,\frac{3}{2}t^2 - \frac{1}{2}\}$.

2.56 Let V denote an inner product space. Let $E = \{e_n\}_{n\in I}$ and $\tilde{E} = \{\tilde{e}_n\}_{n\in I}$ be countable subsets of V. If E and $\tilde{E}$ are biorthogonal, show that E is linearly independent and $\tilde{E}$ is also linearly independent.

2.57 Let E and $\tilde{E}$ denote biorthogonal bases of an inner product space V. In each of the following cases, compute the expansion of the vector x in terms of the elements of E.
(a) $V = \mathbb{R}^2$, $E = \{(1,1),(1,2)\}$, and $\tilde{E} = \{(2,-1),(-1,1)\}$, and $x = (1,-2)$.
(b) $V = \mathbb{R}^4$, $E = \{(\frac{1}{2},-1,\frac{1}{2},-\frac{1}{2}),(0,\frac{1}{2},\frac{1}{2},-\frac{1}{2}),(-\frac{1}{2},0,-\frac{1}{2},1),(\frac{1}{2},0,-\frac{1}{2},\frac{1}{2})\}$,
$\tilde{E} = \{(\frac{1}{2},1,0,\frac{3}{2}),(-\frac{1}{2},1,0,\frac{1}{2}),(\frac{3}{2},3,2,\frac{1}{2}),(1,2,2,1)\}$, and $x = (-8,2,4,0)$.

2.58 If $\{e_n\}$ is an orthonormal sequence in a Hilbert space V and $\{a_n\}$ is a sequence in l^2, show that there exists $x \in V$ such that

$$\langle x, e_n\rangle = a_n \quad \text{and} \quad \|\{a_n\}\|_{l^2} = \|x\|.$$

2.59 Determine all possible values of $a \in \mathbb{R}$ for which the function $f(t) = t^a$ belongs to:
(a) $L^2[0,1]$;
(b) $L^2[1,\infty)$;
(c) $L^2[0,\infty)$

2.60 Let A and B denote subsets of an inner product space V. For each of the cases given below, determine whether A and B are biorthonormal and/or biorthogonal.
(a) $V = \mathbb{R}^4$, $A = \{(1,0,-1,2),(0,1,2,-2),(4,-4,-7,10)\}$, $B = \{(-3,0,4,4),(0,-1,2,1),(5,2,-7,-6)\}$;
(b) $V = \mathbb{R}^2$, $A = \{(3,4),(4,3)\}$, $B = \{(-3,4),(4,-3)\}$;
(c) $V = L^2[-\pi,\pi]$, $A = \{\sin t, \cos t\}$, $B = \{t, t^2\}$.
[**Note:** In part (c), a table of integrals can be used to evaluate integrals like $\int_{-\pi}^{\pi} t^2 \sin t dt$. Integration by parts is not required.]

2.61 Let W denote a subspace of the inner product space V, where W has the orthonormal basis E. In each of the cases below, find the expansion of $x \in W$ in terms of the elements of E.
(a) $V = \mathbb{R}^3$, $E = \{(\frac{1}{\sqrt{2}},0,\frac{1}{\sqrt{2}}),(-\frac{1}{\sqrt{3}},-\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}}),(\frac{1}{\sqrt{6}},-\frac{2}{\sqrt{6}},-\frac{1}{\sqrt{6}})\}$, $x = (\sqrt{6},-\sqrt{6},0)$.
(b) $V = L^2[-1,1]$, $E = \{e_n\}_{n\in\mathbb{Z}^*}$, $x(t) = |t|$, and

$$e_n(t) = \begin{cases} 1/\sqrt{2} & n = 0 \\ \cos \pi n t & \text{otherwise.} \end{cases}$$

[Note: In part (b), a table of integrals can be used to avoid integration by parts.]

2.62 Let W be a subspace of the inner product space V. Let E and $\tilde{E}$ be dual Riesz bases of W. For each of the cases below, express $x \in W$ in terms of the elements of E and in terms of the elements of $\tilde{E}$.
(a) $V = \mathbb{R}^2$, $E = \{e_1 = (\frac{3}{4},\frac{5}{4}), e_2 = (\frac{5}{4},\frac{3}{4})\}$, $\tilde{E} = \{\tilde{e}_1 = (-\frac{3}{4},\frac{5}{4}), \tilde{e}_2 = (\frac{5}{4},-\frac{3}{4})\}$, $x = (1,-2)$.
(b) $V = L^2(\mathbb{R})$, $E = \{e_n(t) = \frac{1}{2}\chi_{[n,n+1)}\}_{n\in\mathbb{Z}}$, $\tilde{E} = \{\tilde{e}_n(t) = 2\chi_{[n,n+1)}\}_{n\in\mathbb{Z}}$, $x = \chi_{[1,3)}$.
(c) V is the space of polynomial functions defined on $[-1,1]$, $E = \{t^2 - t, t\}$, $\tilde{E} = \{\frac{5}{2}t^2, \frac{5}{2}t^2 + \frac{3}{2}t\}$, $x(t) = t^2 - 4t$.
(d) V is the space of polynomial functions defined on $[-1,1]$, $E = \{\frac{45}{8}t^2 - \frac{15}{8}, \frac{3}{2}t, -\frac{15}{8}t^2 + \frac{9}{8}\}$, $\tilde{E} = \{t^2, t, 1\}$, $x(t) = 3t + 3$.

2.63 Prove that in any Hilbert space V, for $x, y \in V$, $\|x\| = \sup_{\|y\|=1} |\langle x, y\rangle|$.

2.64 Prove that any finite-dimensional inner product space is a Hilbert space.

2.65 Let $f, g \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$.
(a) Show that

$$\langle f, g\rangle = \tfrac{1}{2\pi}\left\langle \hat{f}, \hat{g}\right\rangle.$$

(b) Show that $\|f\|^2 = \frac{1}{2\pi}\left\|\hat{f}\right\|^2$.

2.66 Show that a countable orthogonal set $E = \{e_n\}_{n\in I}$ is linearly independent if and only if the set does not contain the zero vector.

2.67 Let $E = \{e_n\}_{n\in I}$ and $\tilde{E} = \{\tilde{e}_n\}_{n\in I}$ each be a basis of the N-dimensional (where N is finite) Hilbert space V such that every vector $x \in V$ can be expressed as

$$x = \sum_{n\in I} \langle x, \tilde{e}_n\rangle e_n.$$

Show that E and $\tilde{E}$ are biorthonormal.

2.68 For each of the cases below, show that the sequence $\{x_n\}$ is Cauchy.
(a) $\{x_n\}_{n\in\mathbb{N}}$ where $x_n = 1/2^n$;

2.69 Let $\{e_n\}_{n\in\mathbb{N}}$ be an orthonormal basis of a Hilbert space, and define the sequence $\{f_n\}_{n\in\mathbb{N}}$ as $f_n = \frac{1}{n}e_n$. Show that $\{f_n\}_{n\in\mathbb{N}}$ is (a) ω-independent and (b) minimal.

2.70 Let $\{e_n\}_{n\in\mathbb{N}}$ be an orthonormal basis of a Hilbert space, and define the sequence $\{f_n\}_{n\in\mathbb{N}}$ as $f_n = \frac{1}{n}e_n + e_1$. Show that $\{f_n\}_{n\in\mathbb{N}}$ is (a) ω-independent and (b) not minimal.

2.71 Let $\{e_n\}_{n\in\{0,1,2,\ldots\}}$ be an orthonormal basis of a Hilbert space, and define the sequence $\{f_n\}_{n\in\mathbb{N}}$ as

$$f_n = \left(\cos\tfrac{1}{n}\right)e_0 + \left(\sin\tfrac{1}{n}\right)e_n.$$

Show that $\{f_n\}_{n\in\mathbb{N}}$ is ω-independent

2.72 Show that a countable orthonormal set $E = \{e_n\}_{n\in I}$ in an inner product space is ω-independent.

2.73 Let h be a symmetric/antisymmetric sequence defined on $\mathbb{Z}$. Such a sequence is of the form $h[n] = sh[2c-n]$, where $c \in \frac{1}{2}\mathbb{Z}$ is the center of symmetry and $s \in \{-1,1\}$. Show that $\hat{h}$ can be expressed as

$$\hat{h}(\omega) = \begin{cases} e^{-jc\omega}\sum_{n\in\mathbb{Z}} h[n]\cos([n-c]\omega) & s=1 \\ -je^{-jc\omega}\sum_{n\in\mathbb{Z}} h[n]\sin([n-c]\omega) & s=-1. \end{cases}$$

2.74 Let h be a symmetric/antisymmetric sequence defined on $\mathbb{Z}$. Such a sequence is of the form $h[n] = sh[2c-n]$, where $c \in \frac{1}{2}\mathbb{Z}$ is the center of symmetry and $s \in \{-1,1\}$. Let m_k denote the kth moment of h.
(a) Suppose that $s=1$. Show that, if $m_k = 0$ for $k \in \{0,1,\ldots,2N\}$, then $m_{2N+1} = 0$.
(b) Suppose that $s=-1$. Show that, if $m_k = 0$ for $k \in \{1,2,\ldots,2N+1\}$, then $m_{2N+2} = 0$.

2.75 Evaluate each of the following expressions:
(a) $\{0,\frac{1}{3},\frac{1}{2},1,\sqrt{2},\pi,42\}\setminus\mathbb{Q}$;
(b) $[(\{0,1,2\}\cup\{3,4\})\cap\{1,3\}]\setminus\{1\}$;
(c) $[0,1)\cup\{1\}\cup(1,2]$.

2.76 In each of the cases below, for the set X, indicate whether X is finite, countably infinite, or uncountable.
(a) $X = \{0,2,4,6,8,\ldots\}\cap\{0,1,2,3\}$;
(b) $X = [0,1]$;
(c) $X = \mathbb{R}\setminus\mathbb{Q}$;
(d) $X = \{\infty\}$.

2.77 Explain the difference between a set and sequence. Give an example of a situation where a set might be more appropriate to use than a sequence and vice versa.

2.78 Write an expression for each of the following:
(a) the function f translated by 5 and then dilated by 3 (where the dilation of a function g by a is defined as $g(a\cdot)$;
(b) the sum of the function f translated by 2 and the function g;
(c) the product of the functions f and g;
(d) the value at t of the function f translated by b convolved with g;
(e) the sequence consisting of all integer translations of the function f.

2.79 In each of the cases below, for the subset S of the metric space X, find the boundary of S (in X).
(a) $X = \mathbb{C}$ and $S = \{z\in\mathbb{C} : \operatorname{Re}^2 z + \operatorname{Im}^2 z < 4\}$.

2.80 In each of the cases below, for the subset S of the metric space X, determine if S is open and/or closed, and if S is not closed, find $\operatorname{clos} S$. Also, determine if S is dense in X.
(a) $X = \mathbb{C}$ and $S = \{z\in\mathbb{C} : \operatorname{Re}^2 z + \operatorname{Im}^2 z < 4\}\cup\{2\}$;
(b) $X = \mathbb{R}$ and $S = \{0,1\}$.

2.81 Consider the transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ defined by $T(x_1,x_2) = (x_1,0)$. Show that T is both (a) linear and (b) idempotent (i.e., $T^2 = T$), and therefore T is a projection operator.

2.82 Consider the projection operator $T : \mathbb{R}^2 \to \mathbb{R}^2$ defined by $T(x_1, x_2) = (x_1, 0)$ (i.e., a projection from $\mathbb{R}^2$ onto the x axis). Find (a) $N(T)$ (i.e., the null space of T) and (b) $R(T)$ (i.e., the range space of T). (c) Determine whether T is an orthogonal projection.

2.83 Let $(x_n)_{n \in \mathbb{N}}$ denote a sequence of vectors in a vector space. Explain why, in a purely vector space context, an expression of the form $\sum_{n \in \mathbb{N}} x_n$ is meaningless.

2.84 Determine for what values of α the sequence (x_1, x_2, x_3) of vectors in $\mathbb{R}^3$ is linearly independent, where $x_1 = (1, 0, 1)$, $x_2 = (0, 2, 1)$, and $x_3 = (\alpha, 1, 1)$.

2.85 In each of the cases below, for the vector x in the normed space V, find $\|x\|$.
(a) $V = l^2(\mathbb{Z})$ and $x = (2^{-n/2}\chi_{Z^*}[n])_{n \in \mathbb{Z}}$ (where Z^* denotes the set of nonnegative integers);
(b) $V = L^2(\mathbb{R})$ and $x(t) = e^{-t}\chi_{[0,\infty)}(t)$.

2.86 In each of the cases below, for the vectors x, y in the inner product space V, find the angle θ between x and y.
(a) $V = L^2(\mathbb{R})$, $x = \chi_{[0,1]}$, $y = \chi_{[-1/2,1/2]}$;
(b) $V = l^2(\mathbb{Z})$, $x = \left(\frac{3}{5}\delta[n] + \frac{4}{5}\delta[n-1]\right)_{n \in \mathbb{Z}}$, $y = \left(\frac{1}{\sqrt{2}}\delta[n] - \frac{3}{\sqrt{2}}\delta[n-1]\right)_{n \in \mathbb{Z}}$.

2.14 Bibliography

[1] A. Boggess and F. J. Narcowich. *A First Course in Wavelets with Fourier Analysis*. Prentice Hall, Upper Saddle River, NJ, USA, 2001.

[2] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Upper Saddle River, NJ, USA, 1998.

[3] O. Christensen. *An Introduction to Frames and Riesz Bases*. Birkhauser, Boston, MA, USA, 2003.

[4] C. K. Chui. *An Introduction to Wavelets*. Academic Press, San Diego, CA, USA, 1992.

[5] L. Debnath and P. Mikusinski. *Introduction to Hilbert Spaces with Applications*. Academic Press, New York, NY, USA, 2nd edition, 1999.

[6] J. C. Goswami and A. K. Chan. *Fundamentals of Wavelets: Theory, Algorithms and Applications*. John Wiley & Sons, New York, NY, USA, 1999.

[7] E. Kreyszig. *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.

[8] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, USA, 2nd edition, 1999.

[9] A. Mertins. *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. John Wiley & Sons, Toronto, 2000.

[10] A. W. Naylor and G. R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag, New York, NY, USA, 1982.

[11] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[12] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

Part II

One-Dimensional Filter Banks and Univariate Wavelets

Chapter 3

One-Dimensional Multirate Filter Banks

3.1 Introduction

Multirate filter banks play an important role in the study of wavelet systems. In particular, the class of systems known as uniformly maximally-decimated (UMD) filter banks is especially significant in this regard. Wavelet transforms can be constructed through the design of UMD filter banks and implemented very efficiently in terms of these structures.

In order to study UMD filter banks, we must first understand the fundamentals of multirate systems. In what follows, we begin by introducing some basic multirate system concepts, and then use these concepts to establish a general framework for the study of UMD filter banks. The link between UMD filter banks and wavelet systems will be established later in Chapter 4.

3.2 Multirate Systems

Depending on the number of different sampling rates that it employs, a discrete-time system can be classified as being either unirate or multirate. A system that processes signals at a single sampling rate is said to be **unirate**. Most of us are all too familiar with unirate systems as they are traditionally studied in any introductory digital signal processing course. In contrast, a system that processes signals at more than one sampling rate is said to be **multirate**.

Multirate systems are extremely useful for many signal processing applications. Often, a multirate system can be used to perform a task more easily or efficiently than is possible with a unirate system. Other times, a signal processing task inherently requires the use of multiple sampling rates. In such cases, a multirate system must be used.

Since multirate systems employ more than one sampling rate, such systems need a means for changing, or converting between, rates. This is accomplished through processes known as upsampling and downsampling, which we introduce next.

3.2.1 Downsampling

One of the basic operations in multirate systems is that of decreasing the sampling rate. This operation is known as downsampling and is performed by a processing element known as a downsampler[1] (or compressor).

Definition 3.1 (Downsampling)**.** The M**-fold downsampler**, shown in Figure 3.1, takes an input sequence $x[n]$ and produces the output sequence

$$y[n] = (\downarrow M)x[n] = x[Mn] \tag{3.1}$$

where M is an integer. The constant M is referred to as the **downsampling factor**.

[1]The term "decimator" is also sometimes employed as a synonym for "downsampler". Unfortunately, "decimator" can also be used to mean "a downsampler in cascade with an anti-aliasing filter", as in the case of sampling rate conversion applications. Thus, in order to avoid confusion, the author prefers to use the terms "downsampler" and "compressor" instead of "decimator".

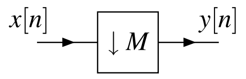


Figure 3.1: M-fold downsampler.

In simple terms, the downsampling operation keeps every Mth sample and discards the others. From (3.1), it follows that downsampling is a linear time-varying operation.

Example 3.1. Suppose that we have the sequence $x = (x_n)_{n\in\mathbb{Z}}$. That is,

$$x = (\ldots, x_{-6}, x_{-5}, x_{-4}, x_{-3}, x_{-2}, x_{-1}, \boxed{x_0}, x_1, x_2, x_3, x_4, x_5, x_6, \ldots).$$

Then, we have

$$(\downarrow 2)x = (\ldots, x_{-6}, x_{-4}, x_{-2}, \boxed{x_0}, x_2, x_4, x_6, \ldots) \quad \text{and} \quad (\downarrow 3)x = (\ldots, x_{-6}, x_{-3}, \boxed{x_0}, x_3, x_6, \ldots).$$

(The boxed elements above correspond to the elements with index 0.)

Often, we like to work with the $\mathcal{Z}$-domain representation of sequences. The relationship between the input and output of the downsampler in the $\mathcal{Z}$ domain is given by the following theorem.

Theorem 3.1 (Downsampling in $\mathcal{Z}$ domain)**.** *Suppose that $y[n] = (\downarrow M)x[n]$. Let $X(z)$ and $Y(z)$ denote the $\mathcal{Z}$ transforms of $x[n]$ and $y[n]$, respectively. Then, $Y(z)$, which we abbreviate as $(\downarrow M)X(z)$, is given by*

$$Y(z) = (\downarrow M)X(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}e^{-j2\pi k/M}). \tag{3.2}$$

Proof. The $\mathcal{Z}$ transform of $y[n]$ can be written as

$$\begin{aligned} Y(z) &= \sum_{n\in\mathbb{Z}} y[n]z^{-n} \\ &= \sum_{n\in\mathbb{Z}} x[Mn]z^{-n}. \end{aligned}$$

Now, we define the sequence

$$v[n] = \begin{cases} x[n] & \text{if } M \mid n \\ 0 & \text{otherwise.} \end{cases}$$

Using this definition (which implies that $v[Mn] = x[Mn]$ for $n \in \mathbb{Z}$), we have

$$Y(z) = \sum_{n\in\mathbb{Z}} v[Mn]z^{-n}.$$

Now, we employ a change of variable. Let $n' = Mn$ so that $n = n'/M$. Applying the change of variable and dropping the primes, we obtain

$$Y(z) = \sum_{\substack{n\in\mathbb{Z} \\ M\mid n}} v[n]z^{-n/M}.$$

Since $v[n]$ is zero when $M \nmid n$, the constraint on the summation index that $M \mid n$ can be removed to yield

$$\begin{aligned} Y(z) &= \sum_{n\in\mathbb{Z}} v[n]z^{-n/M} \\ &= V(z^{1/M}). \end{aligned} \tag{3.3}$$

To complete the proof, we express $V(z)$ in terms of $X(z)$. We observe that $v[n]$ can be expressed as

$$v[n] = c[n]x[n]$$

where

$$c[n] = \begin{cases} 1 & \text{if } M \mid n \\ 0 & \text{otherwise.} \end{cases}$$

The M-periodic sequence $c[n]$ has the Fourier series representation

$$c[n] = \tfrac{1}{M} \sum_{k=0}^{M-1} e^{j2\pi kn/M}.$$

Thus, we can compute the $\mathcal{Z}$ transform of $v[n]$ as follows:

$$\begin{aligned} V(z) &= \mathcal{Z}\{c[n]x[n]\} \\ &= \sum_{n\in\mathbb{Z}} \left(\tfrac{1}{M} \sum_{k=0}^{M-1} e^{j2\pi kn/M} x[n] \right) z^{-n} \\ &= \tfrac{1}{M} \sum_{n\in\mathbb{Z}} \sum_{k=0}^{M-1} e^{j2\pi kn/M} x[n] z^{-n} \\ &= \tfrac{1}{M} \sum_{k=0}^{M-1} \sum_{n\in\mathbb{Z}} x[n] \left(ze^{-j2\pi k/M} \right)^{-n} \\ &= \tfrac{1}{M} \sum_{k=0}^{M-1} X(ze^{-j2\pi k/M}). \end{aligned} \tag{3.4}$$

Combining (3.3) and (3.4), we have

$$\begin{aligned} Y(z) &= V(z^{1/M}) \\ &= \tfrac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} e^{-j2\pi k/M}). \end{aligned}$$

□

Assuming that the sampling period before and after downsampling is normalized to one, this leads directly to the frequency-domain relation

$$Y(e^{j\omega}) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(e^{j(\omega - 2\pi k)/M} \right). \tag{3.5}$$

From (3.5), we can see that downsampling has a very simple interpretation in the frequency domain. The spectrum of the downsampled signal is merely the average of M shifted versions of the original input spectrum. Due to our convention of normalizing the sampling period after downsampling to one, the spectrum is also stretched. It is important to understand, however, that this spectrum stretching effect is only a consequence of the sampling period renormalization and is not caused by downsampling itself. This matter is further elucidated by the example below.

Example 3.2. Consider an M-fold downsampler with input x and output y. By convention, the sampling rate associated with the input x is assumed to be $T = 1$. Thus, the spectrum of x is given by $X(e^{j\omega T}) = X(e^{j\omega})$. We have that $y[n] = (\downarrow M)x[n]$. So, the sampling period associated with the output y is $T' = MT = M$ (assuming that we do not artificially renormalize the sampling period to one after downsampling). Thus, the spectrum of y is given by $Y(e^{j\omega T'}) = Y(e^{j\omega M})$, where

$$\begin{aligned}
Y(e^{j\omega M}) &= \left.\left[\frac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}e^{-j2\pi k/M})\right]\right|_{z=e^{j\omega M}} \\
&= \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(\omega M-2\pi k)/M}) \\
&= \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(\omega-2\pi k/M)}).
\end{aligned}$$

Clearly, in the above expression, we can see that the spectrum of y does not contain dilated versions of the spectrum of x. The stretching phenomenon is only a result of the renormalization (to one) of the sampling rate after downsampling.

As is evident from equation (3.5), downsampling can result in multiple baseband frequencies in the input signal being mapped to a single frequency in the output signal. This phenomenon is known as **aliasing**. If aliasing occurs, it is not possible to recover the original signal from its downsampled version. Aliasing can be avoided if $x[n]$ is a lowpass signal bandlimited to $|\omega| < \pi/M$. This is equivalent to saying that the Nyquist criterion must not be violated if aliasing is to be avoided.

To illustrate the frequency-domain effects of downsampling, let us consider two examples. In the first example, the input signal is chosen to be sufficiently bandlimited that aliasing will not occur. In the second example, the input signal is selected to result in aliasing.

Example 3.3. In the first case, suppose we take a signal with the spectrum shown in Figure 3.2(a) and apply it to the input of a two-fold downsampler. The spectrum of the downsampled signal is formed by the scaled sum of the two shifted versions of the original spectrum shown in Figure 3.2(b). Due to the renormalization of the sampling period, these spectra also appear stretched. The resulting spectrum is shown in Figure 3.2(c). Because the two spectra in Figure 3.2(b) do not overlap, there is no aliasing. The shape of the original spectrum is clearly discernible in the final output spectrum.

Example 3.4. In the second case, suppose we take a signal with the spectrum shown in Figure 3.3(a) and apply it to the input of a two-fold downsampler. The spectrum of the downsampled signal is formed by the scaled sum of the two shifted versions of the original spectrum shown in Figure 3.3(b). Again, these spectra appear stretched due to the renormalization of the sampling period. The resulting spectrum is shown in Figure 3.3(c). Because in Figure 3.3(b) the two spectra overlap, aliasing occurs. Consequently, it is not possible to recover the original signal from its downsampled version. This is also evident from the spectrum of the downsampled signal shown in Figure 3.3(c). Due to aliasing, the spectrum has been distorted and no longer resembles that of the original input.

3.2.2 Upsampling

Another basic operation in multirate systems is that of increasing the sampling rate. This operation is called upsampling and is performed by a processing element known as an upsampler[2] (or expander).

Definition 3.2 (Upsampling)**.** The M**-fold upsampler**, depicted in Figure 3.4, takes an input sequence $x[n]$ and produces the output sequence

$$y[n] = (\uparrow M)x[n] = \begin{cases} x[n/M] & \text{if } M \mid n \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where M is an integer. The constant M is referred to as the **upsampling factor**.

In simple terms, upsampling results in the insertion of $M-1$ zeros between the samples of the original signal. From equation (3.6), it follows that upsampling is a linear time-varying operation.

[2] The term "interpolator" is also sometimes employed as a synonym for "upsampler". Unfortunately, "interpolator" has many other different and conflicting meanings. For this reason, the author favors the use of the terms "upsampler" and "expander" instead of "interpolator".

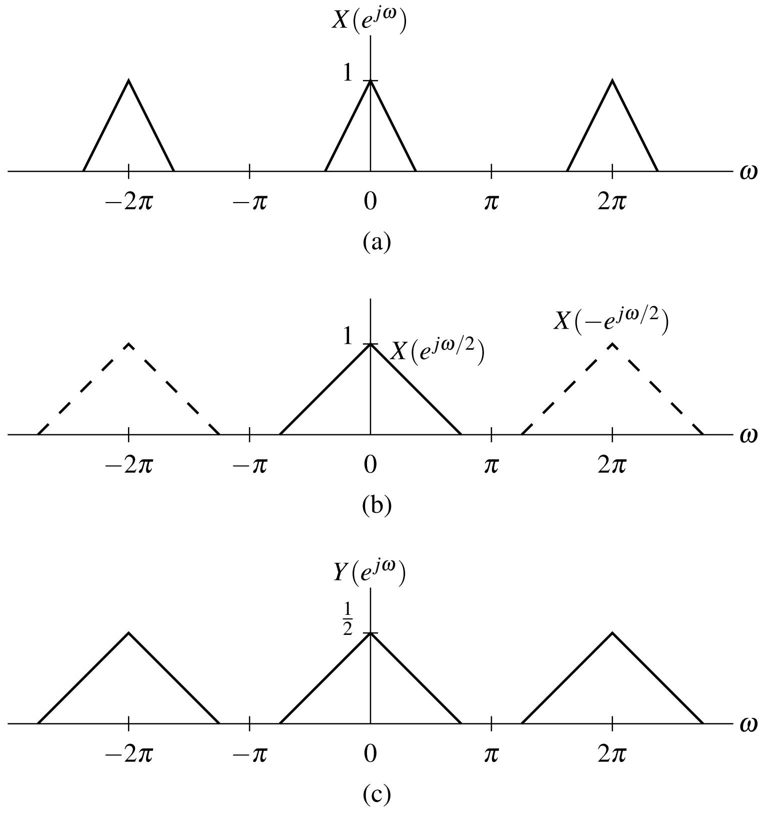


Figure 3.2: Effects of two-fold downsampling in the frequency domain (no aliasing case). (a) Spectrum of the input signal. (b) The two stretched and shifted versions of the original input spectrum used to form the spectrum of the downsampled signal. (c) Spectrum of the downsampled signal.

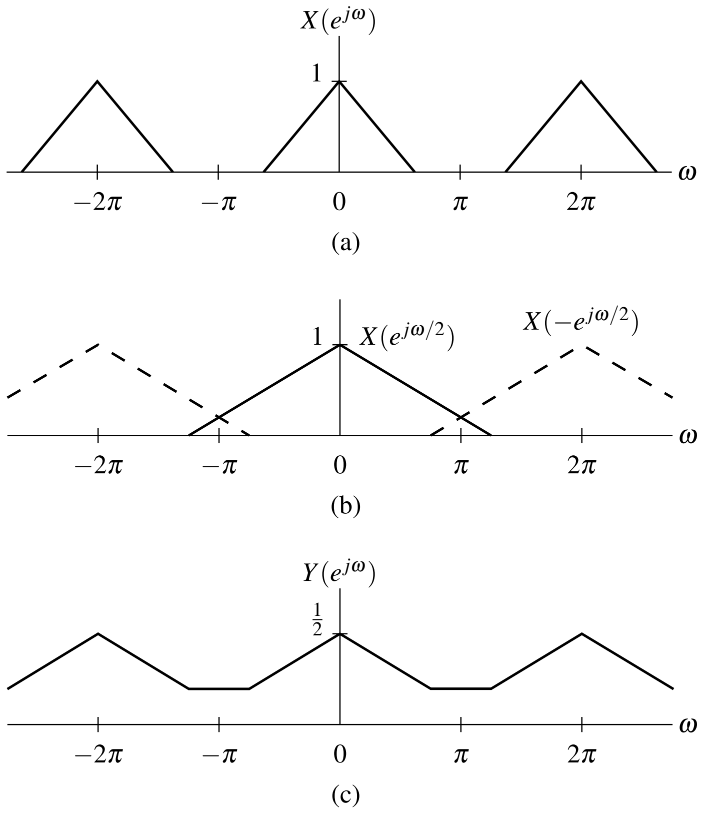


Figure 3.3: Effects of two-fold downsampling in the frequency domain (aliasing case). (a) Spectrum of the input signal. (b) The two stretched and shifted versions of the original input spectrum used to form the spectrum of the downsampled signal. (c) Spectrum of the downsampled signal.

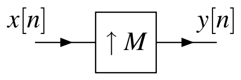


Figure 3.4: M-fold upsampler.

Example 3.5. Suppose that we have the sequence $x = (x_n)_{n\in\mathbb{Z}}$. That is,

$$x = (\ldots, x_{-3}, x_{-2}, x_{-1}, \boxed{x_0}, x_1, x_2, x_3, \ldots).$$

Then, we have

$$(\uparrow 2)x = (\ldots, 0, x_{-3}, 0, x_{-2}, 0, x_{-1}, 0, \boxed{x_0}, 0, x_1, 0, x_2, 0, x_3, 0, \ldots) \quad \text{and}$$
$$(\uparrow 3)x = (\ldots, 0, 0, x_{-3}, 0, 0, x_{-2}, 0, 0, x_{-1}, 0, 0, \boxed{x_0}, 0, 0, x_1, 0, 0, x_2, 0, 0, \ldots).$$

(The boxed elements above correspond to the elements with index 0.)

Often, we like to work with the $\mathcal{Z}$-domain representation of sequences. The relationship between the input and output of the upsampler in the $\mathcal{Z}$ domain is given by the theorem below.

Theorem 3.2 (Upsampling relation in the $\mathcal{Z}$ domain)**.** *Suppose that we have two sequences $x[n]$ and $y[n]$ such that $y[n] = (\uparrow M)x[n]$. Let $X(z)$ and $Y(z)$ denote the $\mathcal{Z}$ transforms of $x[n]$ and $y[n]$, respectively. Then, $Y(z)$, which we abbreviate as $(\uparrow M)X(z)$, is given by*

$$Y(z) = (\uparrow M)X(z) = X(z^M). \tag{3.7}$$

Proof. The $\mathcal{Z}$ transform of $y[n]$ can be written as

$$Y(z) = \sum_{n\in\mathbb{Z}} y[n]z^{-n}.$$

Since $y[n]$ is zero for $M \nmid n$, we can discard all of the terms where $M \nmid n$ to obtain

$$Y(z) = \sum_{\substack{n\in\mathbb{Z} \\ M|n}} y[n]z^{-n}.$$

Now, we employ a change of variable. Let $n' = n/M$ so that $n = Mn'$. Applying the change of variable and dropping the primes, we have

$$Y(z) = \sum_{n\in\mathbb{Z}} y[Mn]z^{-Mn}.$$

Simplifying further, we obtain

$$\begin{aligned} Y(z) &= \sum_{n\in\mathbb{Z}} x[n]z^{-Mn} \\ &= \sum_{n\in\mathbb{Z}} x[n](z^M)^{-n} \\ &= X(z^M). \end{aligned}$$

□

Assuming that the sampling period before and after upsampling is normalized to one, this directly yields the frequency-domain relation

$$Y(e^{j\omega}) = X(e^{j\omega M}). \tag{3.8}$$

From (3.8), we can see that upsampling has a very simple interpretation in the frequency domain. The upsampling process simply serves to move the location of the sampling frequency on the frequency axis. Due to our convention of normalizing the sampling period after upsampling to one, the spectrum is also compressed. It is important to understand, however, that this compression effect is only a consequence of the sampling period renormalization and is not caused by upsampling itself. This matter is further elucidated by the example below.

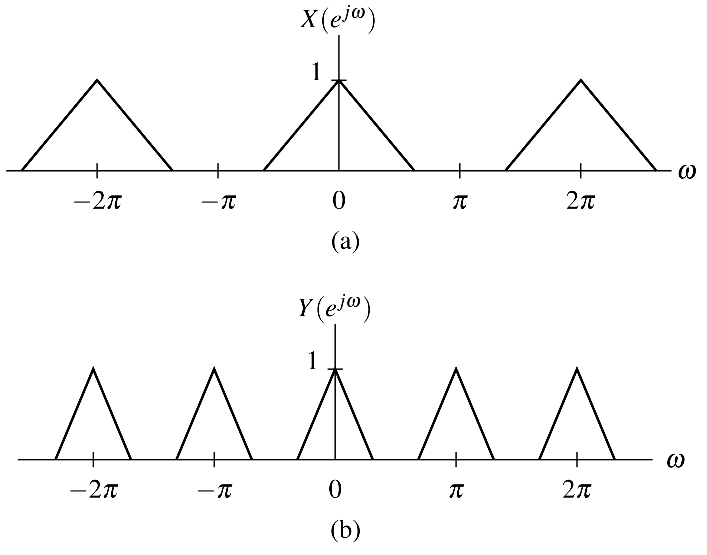


Figure 3.5: Effects of two-fold upsampling in the frequency domain. (a) Spectrum of the input signal. (b) Spectrum of the upsampled signal.

Example 3.6. Consider an M-fold upsampler with input x and output y. By convention, the sampling rate associated with the input x is assumed to be $T = 1$. Thus, the spectrum of x is given by $X(e^{j\omega T}) = X(e^{j\omega})$. We have that $y[n] = (\uparrow M)x[n]$. So, the sampling period associated with the output y is $T' = T/M = 1/M$ (assuming that we do not artificially renormalize the sampling period to one after upsampling). Thus, the spectrum of y is given by $Y(e^{j\omega T'}) = Y(e^{j\omega/M})$, where

$$\begin{aligned} Y(e^{j\omega/M}) &= \left[X(z^M)\right]\big|_{z=e^{j\omega/M}} \\ &= X(e^{j\omega M/M}) \\ &= X(e^{j\omega}). \end{aligned}$$

Clearly, in the above expression, we can see that the spectrum of y does not contain dilated versions of the spectrum of x. The dilation phenomenon is only a result of the renormalization (to one) of the sampling rate after upsampling.

Since the shape of the spectrum is not altered by upsampling, there is no information loss and the original signal can always be recovered from its upsampled version. Upsampling, however, does result in the creation of multiple copies of the original baseband spectrum. This phenomenon is called **imaging**. These copies of the baseband spectrum are referred to as **images**.

Example 3.7. To illustrate the frequency-domain effects of upsampling, let us consider an example. Suppose we take a signal with the spectrum shown in Figure 3.5(a) and apply it to the input of a two-fold upsampler. The spectrum of the upsampled signal is simply that shown in Figure 3.5(b). Due to the sampling period renormalization, the spectrum appears compressed. Although multiple copies of the original baseband spectrum appear, each copy has the same shape as the original input spectrum. As a result, the original signal can always be recovered from the upsampled signal with the appropriate use of filtering.

3.2.3 More Comments on Downsampling/Upsampling

Sometimes when manipulating $\mathcal{Z}$-transform expressions involving upsampling and downsampling, we need to exercise special care. One such situation is illustrated by way of the example below.

Example 3.8. Consider the function

$$Y(z) = \sum_{k=0}^{M-1} X(z^{1/M} W_M^k),$$

where $W_M = e^{-j2\pi/M}$ and $M \in \mathbb{N}$. Often, we encounter expressions of the form of $Y(z)$ when analyzing systems containing downsamplers. Suppose now that we want to evaluate the expression $Y(z^M)$. Clearly, we have (by substitution) that

$$Y(z^M) = \sum_{k=0}^{M-1} X([z^M]^{1/M} W_M^k). \tag{3.9}$$

At this point, we need to be careful how we choose to simplify the right-hand side of the preceding equation. Our initial instinct may be to assert that $(z^M)^{1/M} = z$, but this is not necessarily so. In particular, for any $z \in \mathbb{C}$ and any $M \in \mathbb{N}$, while it is true that

$$(z^{1/M})^M = z,$$

it is not necessarily true that

$$(z^M)^{1/M} = z.$$

The latter identity is only true if z is the principal Mth root of z^M. For example, let $z = -1$ and $M = 2$. In this case, we have $(z^M)^{1/M} = [(-1)^2]^{1/2} = 1^{1/2} = 1 \neq z$. For this reason, we need to consider another means for simplifying (3.9). To begin, we observe that every complex number v has M distinct Mth roots, namely

$$r, rW_M, rW_M^2, \ldots, rW_M^{M-1},$$

where r is the principal Mth root of v. Therefore, in general, we have that $(z^M)^{1/M}$ must be of the form zW_M^l where $l \in \mathbb{Z}$. With this in mind, we can rewrite (3.9) (by substituting zW_M^l for $(z^M)^{1/M}$) as

$$\begin{aligned} Y(z^M) &= \sum_{k=0}^{M-1} X(zW_M^l W_M^k) \\ &= \sum_{k=0}^{M-1} X(zW_M^{l+k}). \end{aligned}$$

Now, we employ a change of variable. Let $k' = l + k$ so that $k = k' - l$. Applying the change of variable and dropping the primes, we obtain

$$Y(z^M) = \sum_{k=l}^{l+M-1} X(zW_M^k).$$

Since $W_M^p = W_M^{p+M}$ for all $p \in \mathbb{Z}$ (i.e., successive integer powers of W_M form an M-periodic sequence) and the summation index k takes on M consecutive integer values (i.e., we are summing over one period of this M-periodic sequence), we can subtract l from the lower and upper summation limits without changing the resulting sum. (The terms are simply permuted but their sum remains the same.) So, finally, we can conclude that

$$Y(z^M) = \sum_{k=0}^{M-1} X(zW_M^k). \tag{3.10}$$

Perhaps, it is worth noting that, had we incorrectly substituted $(z^M)^{1/M} = z$ directly in (3.9), we would have obtained the same final expression for $Y(z^M)$. This, however, would have only been due to luck, and not due to our solution having been properly justified.

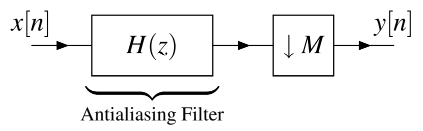


Figure 3.6: An M-fold decimator.

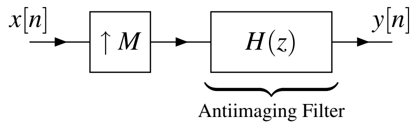


Figure 3.7: An M-fold interpolator.

In passing, we note that the following generalization of the above result also holds:

$$\text{if } Y(z) = \sum_{k=0}^{M-1} X(z^{1/M} W_M^k), \text{ then } Y(z^L) = \sum_{k=0}^{M-1} X(z^{L/M} W_M^k), \tag{3.11}$$

where $L, M \in \mathbb{N}$. (The preceding result can be shown to hold in a similar fashion as (3.10) by observing that $(z^L)^{1/M}$ is of the form $(z^{1/M})^L W_M^\ell = z^{L/M} W_M^\ell$ for some $\ell \in \mathbb{Z}$.) The relationships (3.10) and (3.11) are sometimes used later in order to simplify certain expressions.

3.2.4 Decimation and Interpolation

As we have seen above, we can change the sampling rate by an integer factor M by using only a downsampler or upsampler as appropriate. In practice, however, we typically employ more than just a downsampler or upsampler in order to perform such a rate change.

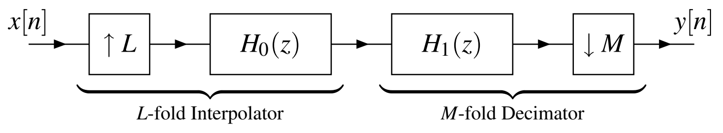
First, let us consider decreasing the sampling rate by an integer factor M. Although we could do this using only an M-fold downsampler, there is a potential problem with such an approach, namely that downsampling can result in aliasing. Since aliasing is often undesirable, we would like to ensure that it does not occur. To accomplish this, we simply perform a lowpass filtering operation prior to downsampling. In this way, we can ensure that the signal to be downsampled is sufficiently bandlimited to avoid aliasing. As a matter of terminology, the lowpass filter employed in this context is called an **antialiasing filter**. The above approach leads to what is known as an **M-fold decimator** (i.e., an antialiasing filter followed by an M-fold downsampler), as shown in Figure 3.6. The passband gain of the antialiasing filter is one. (It is left as an exercise for the reader to show that this is the correct value to use for the passband gain. See Problem 3.44.)

Now, let us consider increasing the sampling rate by an integer factor M. Although this could be done using only an M-fold upsampler, there is a potential problem with such an approach, namely that upsampling results in imaging. In many situations, however, imaging is undesirable. Consequently, we usually perform a lowpass filtering operation after upsampling to eliminate images of the original baseband spectrum. The lowpass filter used in this context is called an **antiimaging filter**. The above approach leads to what is known as an **M-fold interpolator** (i.e., an upsampler followed by an antiimaging filter), as shown in Figure 3.7. The passband gain of the antiimaging filter is M. (It is left an exercise for the reader to show that this the correct value to use for the passband gain. See Problem 3.44.)

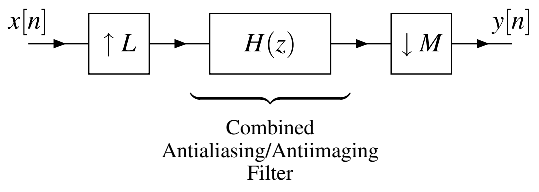
In passing, we note that, although decimators and interpolators are often very useful, in practice, they would not be implemented directly using the computational structures shown in Figures 3.6 and 3.7. This is because these structures are very computationally inefficient. In particular, in these structures, filtering is performed on the side of the downsampler/upsampler with the higher sampling rate, leading to very poor computational efficiency. We will later consider more computationally efficient ways of implementing decimators and interpolators.

3.2.5 Rational Sampling Rate Changes

Although we have seen how to change the sampling rate by an integer factor, in some situations, we may need to change the sampling rate by a noninteger factor. Suppose, for example, that we want to change sampling rate by a rational factor L/M. We can accomplish this by using an L-fold interpolator in cascade with an M-fold decimator, as shown Figure 3.8(a). This system can be simplified by combining the antiimaging and antialiasing filters into a single filter (i.e., $H(z) = H_0(z)H_1(z)$), as shown in Figure 3.8(b).



(a)



(b)

Figure 3.8: Rational sampling rate converter. (a) Before and (b) after combining antiimaging and antialiasing filters.

Although the system shown in Figure 3.8(b) is conceptually simple, it is very computationally inefficient. In particular, the system does not exploit the fact that: 1) most of the terms in the convolutional sum are zero (due to the insertion of zeros by the upsampler); and 2) most of the convolution result is discarded (due to the downsampler). We will later examine ways in which to perform this sampling rate conversion in a more computationally efficient manner.

3.2.6 Cascaded Upsampling and Downsampling Identities

In multirate systems, we often encounter cascaded downsampling operations or cascaded upsampling operations. Thus, we would like to consider the effect of cascading such operations. In this regard, one can easily show that the below theorem holds.

Theorem 3.3 (Cascaded downsampling/upsampling)**.** *The downsampling and upsampling operators have the following properties:*

$$(\downarrow M)(\downarrow L) = \downarrow LM = \downarrow ML \quad \textit{and}$$
$$(\uparrow M)(\uparrow L) = \uparrow LM = \uparrow ML.$$

In other words, we have the identities shown in Figures 3.9(a) and (b).

Proof. First, we show that $(\downarrow M)(\downarrow L) = (\downarrow ML)$. Consider $y = (\downarrow M)(\downarrow L)x$. Let $v = (\downarrow L)x$ so that $y = (\downarrow M)v$. We have

$$v[n] = x[Ln] \quad \text{and} \quad y[n] = v[Mn].$$

Combining the preceding two equations, we obtain

$$y[n] = x[LMn],$$

which is equivalent to $y = (\downarrow LM)x$.

Now, we show that $(\uparrow M)(\uparrow L) = (\uparrow ML)$. Consider $y = (\uparrow M)(\uparrow L)x$. Let $v = (\uparrow L)x$ so that $y = (\uparrow M)v$. We have

$$v[n] = \begin{cases} x[n/L] & \text{if } L|n \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad y[n] = \begin{cases} v[n/M] & \text{if } M|n \\ 0 & \text{otherwise.} \end{cases}$$

Combining the preceding two equations, we obtain

$$y[n] = \begin{cases} x[\frac{n}{ML}] & \text{if } LM|n \\ 0 & \text{otherwise,} \end{cases}$$

which is equivalent to $y = (\uparrow LM)x$. □

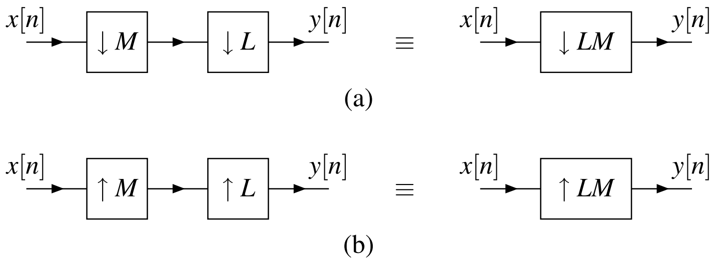


Figure 3.9: Identities for (a) cascaded downsampling and (b) cascaded upsampling operations.

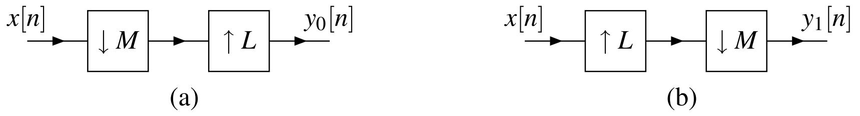


Figure 3.10: Cascade interconnection of upsamplers and downsamplers.

3.2.7 Commutativity of Upsampling and Downsampling

In multirate systems, we often encounter upsamplers and downsamplers connected in series with one another. That is, we frequently see structures like those shown in Figures 3.10(a) and (b). Consequently, one might wonder if there is any relationship between the input-output behavior of these two systems. For example, one might wonder if these systems are equivalent (i.e., if upsampling and downsampling commute). The answer to this question is given by the theorem below.

Theorem 3.4 (Commutativity of upsampling and downsampling)**.** *The L-fold upsampling and M-fold downsampling operators commute if and only if L and M are coprime. That is,*

$$(\uparrow L)(\downarrow M) = (\downarrow M)(\uparrow L)$$

if and only if L and M are coprime. In other words, we have the identity shown in Figure 3.11.

Proof. Consider the system shown in Figure 3.10(a). We have

$$\begin{aligned}
Y_0(z) &= (\uparrow L)\left[(\downarrow M)X(z)\right] \\
&= (\uparrow L)\tfrac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}W_M^k) \\
&= \left[\tfrac{1}{M}\sum_{k=0}^{M-1} X(\lambda^{1/M}W_M^k)\right]\Bigg|_{\lambda=z^L} \\
&= \tfrac{1}{M}\sum_{k=0}^{M-1} X(z^{L/M}W_M^k).
\end{aligned} \tag{3.12}$$

(In the preceding simplification, we used relationship (3.11).) Consider the system shown in Figure 3.10(b). We have

$$\begin{aligned}
Y_1(z) &= (\downarrow M)\left[(\uparrow L)X(z)\right] \\
&= (\downarrow M)\left[X(z^L)\right] \\
&= \tfrac{1}{M}\sum_{k=0}^{M-1} X([z^{1/M}W_M^k]^L) \\
&= \tfrac{1}{M}\sum_{k=0}^{M-1} X(z^{L/M}W_M^{kL}).
\end{aligned} \tag{3.13}$$

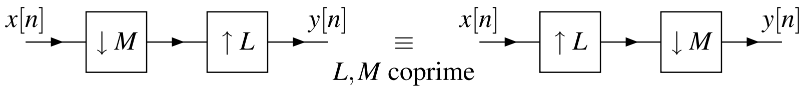


Figure 3.11: Commutativity of upsampling and downsampling.

Comparing (3.12) and (3.13), we can see that $Y_0(z) = Y_1(z)$ if and only if the sets $\{W_M^k\}_{k=0}^{M-1}$ and $\{W_M^{kL}\}_{k=0}^{M-1}$ are equivalent. These sets are equivalent if and only if L and M are coprime. □

The above relationship involving upsampling and downsampling has both theoretical and practical utility. It can sometimes be used to simplify expressions involving upsampling and downsampling operations, and also may be used to obtain more desirable implementations of multirate systems in some situations.

3.2.8 Noble Identities

Often a downsampler or upsampler appears in cascade with a filter. Although it is not always possible to interchange the order of upsampling/downsampling and filtering without changing system behavior, it is sometimes possible to find an equivalent system with the order of these operations reversed, through the use of two very important relationships called the noble identities.

Theorem 3.5 (Noble identities)**.** *For any two sequences with $\mathcal{Z}$ transforms $X(z)$ and $F(z)$, the following identities hold:*

$$F(z)\left[(\downarrow M)X(z)\right] = (\downarrow M)\left[F(z^M)X(z)\right] \quad \textit{and}$$
$$(\uparrow M)\left[F(z)X(z)\right] = F(z^M)\left[(\uparrow M)X(z)\right].$$

*These relationships are known as the first and second **noble identities**, respectively. These identities are illustrated in Figure 3.12.*

Proof. To begin, we prove the first noble identity. Consider the system shown in Figure 3.13(a). We have

$$\begin{aligned} Y_0(z) &= F(z)\left[(\downarrow M)X(z)\right] \\ &= F(z)\left[\tfrac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}W^k)\right] \\ &= \tfrac{1}{M}F(z)\sum_{k=0}^{M-1} X(z^{1/M}W^k). \end{aligned}$$

Consider the system shown in Figure 3.13(b). We have

$$\begin{aligned} Y_1(z) &= (\downarrow M)\left[F(z^M)X(z)\right] \\ &= \tfrac{1}{M}\sum_{k=0}^{M-1}\left[X(\lambda)F(\lambda^M)\right]\big|_{\lambda=z^{1/M}W^k} \\ &= \tfrac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}W^k)F(zW^{kM}) \\ &= \tfrac{1}{M}F(z)\sum_{k=0}^{M-1} X(z^{1/M}W^k). \end{aligned}$$

(Note that $W^{kM} = (e^{-j2\pi k/M})^M = 1$.) So, $Y_0(z) = Y_1(z)$. Thus, the first noble identity holds.

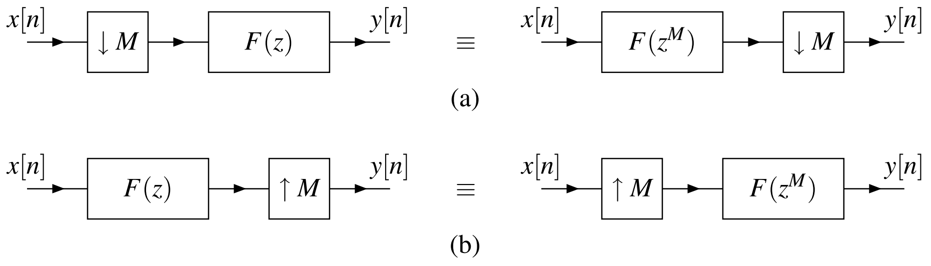


Figure 3.12: The noble identities. The (a) first and (b) second noble identities.

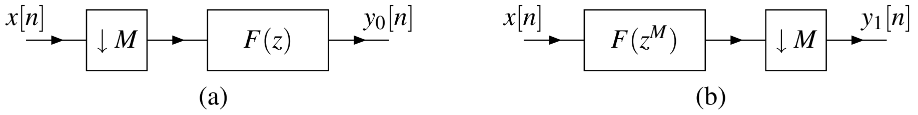


Figure 3.13: Systems for first noble identity.

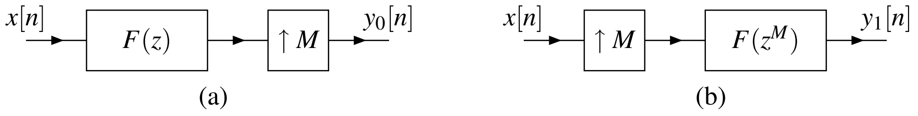


Figure 3.14: Systems for second noble identity.

Now, we prove the second noble identity. Consider the system shown in Figure 3.14(a). We have

$$\begin{aligned}
Y_0(z) &= (\uparrow M)\,[F(z)X(z)] \\
&= F(\lambda)X(\lambda)|_{\lambda=z^M} \\
&= F(z^M)X(z^M).
\end{aligned}$$

Consider the system shown in Figure 3.14(b). We have

$$\begin{aligned}
Y_1(z) &= F(z^M)\,[(\uparrow M)X(z)] \\
&= F(z^M)\,[X(\lambda)]|_{\lambda=z^M} \\
&= F(z^M)X(z^M).
\end{aligned}$$

So, $Y_0(z) = Y_1(z)$. Thus, the second noble identity holds. □

The first noble identity allows us to replace a filtering operation on one side of a downsampler with an equivalent filtering operation on the other side of the downsampler. This identity is illustrated in Figure 3.12(a), where the transfer function $F(z)$ is a rational polynomial expression. The second noble identity allows us to replace a filtering operation on one side of an upsampler with an equivalent filtering operation on the other side of the upsampler. This identity is shown in Figure 3.12(b). It is important to emphasize that, in order for these identities to hold, $F(z)$ must be a rational polynomial expression.

In addition to their theoretical utility, the noble identities are of great practical significance. For performance reasons, it is usually desirable to perform filtering operations on the side of an upsampler or downsampler with the lower sampling rate. Using the noble identities, we can move filtering operations across upsamplers and downsamplers in order to achieve improved computational efficiency.

3.2.9 Polyphase Representation of Signals and Filters

One concept of fundamental importance in the study of multirate systems is that of a polyphase representation of a signal.

Definition 3.3 (Polyphase representation)**.** The **polyphase representation** of the signal $x[n]$, with respect to an integer M and a set of integers $\{m_k\}_{k=0}^{M-1}$, is defined as

$$x[n] = \sum_{k=0}^{M-1} ((\uparrow M)x_k)[n+m_k], \tag{3.14}$$

where

$$x_k[n] = (\downarrow M)(x[n-m_k]) = x[Mn-m_k]$$

and the set $\{m_k\}_{k=0}^{M-1}$ is chosen such that

$$\operatorname{mod}(m_k, M) \neq \operatorname{mod}(m_l, M) \text{ whenever } k \neq l. \tag{3.15}$$

As a matter of terminology, we refer to M as a **sampling factor**, the elements of the set $\{m_k\}_{k=0}^{M-1}$ as **coset offsets**, and $x_0[n], x_1[n], \ldots, x_{M-1}[n]$ as **polyphase components**.

Example 3.9. Consider the sequence $x = (a_n)_{n\in\mathbb{Z}}$. That is, we have

$$x = (\ldots, a_{-4}, a_{-3}, a_{-2}, a_{-1}, \boxed{a_0}, a_1, a_2, a_3, a_4, \ldots)$$

where the boxed element is the one with index zero. Let us consider a polyphase representation of x of the form

$$x = \sum_{k=0}^{M-1} ((\uparrow M)x_k)[n+m_k], \tag{3.16}$$

where $M = 2$, $m_0 = 0$, and $m_1 = 1$. The polyphase components x_0, x_1 are computed by shifting and downsampling (in that order) yielding

$$\begin{aligned} x_0 &= (\downarrow 2)x[n] \\ &= (\downarrow 2)(\ldots, a_{-4}, a_{-3}, a_{-2}, a_{-1}, \boxed{a_0}, a_1, a_2, a_3, a_4, \ldots) \\ &= (\ldots, a_{-4}, a_{-2}, \boxed{a_0}, a_2, a_4, \ldots), \quad \text{and} \end{aligned}$$

$$\begin{aligned} x_1 &= (\downarrow 2)(x[n-1]) \\ &= (\downarrow 2)(\ldots, a_{-4}, a_{-3}, a_{-2}, \boxed{a_{-1}}, a_0, a_1, a_2, a_3, a_4, \ldots) \\ &= (\ldots, a_{-3}, \boxed{a_{-1}}, a_1, a_3, \ldots). \end{aligned}$$

One can confirm that (3.16) holds. Let $\mathcal{S}$ denote a shift operator (e.g., $\mathcal{S}x[n] = x[n-1]$). We have

$$\begin{aligned} x &= ((\uparrow 2)x_0)[n+0] + ((\uparrow 2)x_1)[n+1] \\ &= (\ldots, 0, a_{-4}, 0, a_{-2}, 0, \boxed{a_0}, 0, a_2, 0, a_4, 0, \ldots) + \mathcal{S}^{-1}(\ldots, 0, a_{-3}, 0, \boxed{a_{-1}}, 0, a_1, 0, a_3, 0, \ldots) \\ &= (\ldots, 0, a_{-4}, 0, a_{-2}, 0, \boxed{a_0}, 0, a_2, 0, a_4, 0, \ldots) + (\ldots, 0, a_{-3}, 0, a_{-1}, \boxed{0}, a_1, 0, a_3, 0, \ldots) \\ &= (\ldots, a_{-4}, a_{-3}, a_{-2}, a_{-1}, \boxed{a_0}, a_1, a_2, a_3, a_4, \ldots). \end{aligned}$$

The above polyphase representation can also be expressed in the $\mathcal{Z}$ domain. Let $X(z) = \mathcal{Z}\{x[n]\}$ and $X_k(z) = \mathcal{Z}\{x_k[n]\}$ for $k = 0, 1, \ldots, M-1$. Then, by taking the $\mathcal{Z}$ transform of (3.14), we obtain

$$X(z) = \sum_{k=0}^{M-1} z^{m_k} X_k(z^M) \tag{3.17a}$$

where

$$X_k(z) = (\downarrow M) z^{-m_k} X(z). \tag{3.17b}$$

Examining (3.15), we see that many choices are possible for the sampling factor M and set of coset offsets $\{m_k\}_{k=0}^{M-1}$. Even for a fixed choice of M, the choice of $\{m_k\}_{k=0}^{M-1}$ is not unique. Therefore, many variations on the polyphase representation exist.

Although, for a given sampling factor M, many different choices are possible for the set of coset offsets $\{m_k\}_{k=0}^{M-1}$, four specific choices are most frequently used in practice. Two of these choices are commonly designated as type-1 and type-2 (e.g., as in [31], [29]). The other two choices do not have standard names associated with them, and for convenience will be referred to as type-3 and type-4. In the case of these four commonly used types of polyphase representations, the $\{m_k\}_{k=0}^{M-1}$ in equation (3.14) or (3.17) are chosen as

$$m_k = \begin{cases} -k & \text{type 1} \\ k-(M-1) & \text{type 2} \\ k & \text{type 3} \\ (M-1)-k & \text{type 4} \end{cases}$$

for $k = 0, 1, \ldots, M-1$.

One can show that different choices of $\{m_k\}_{k=0}^{M-1}$ serve only to time shift and permute the polyphase components. For example, for a fixed choice of M, the type-2 polyphase components of a signal are simply a permutation of its type-1 polyphase components. More specifically, the orders of the components are reversed with respect to one another. The particular type of polyphase decomposition to be used is often dictated by practical considerations or notational convenience.

In passing, we note that if $X(z)$ is rational (but not a Laurent polynomial), the polyphase components of an M-phase polyphase representation need not be *rational* [12, p. 95]. This fact leads to some potential practical problems when dealing with polyphase decompositions of rational functions.

Example 3.10. Suppose that we have a signal with $\mathcal{Z}$ transform

$$X(z) = 1 + 2z^{-1} + 3z^{-2} + 4z^{-3} + 5z^{-4} + 6z^{-5} + 7z^{-6}.$$

Find the type-1 three-phase polyphase representation of $X(z)$.

Solution. A type-1 three-phase polyphase representation of $X(z)$ is of the form

$$\begin{aligned} X(z) &= \sum_{k=0}^{M-1} z^{m_k} X_k(z^M) \\ &= X_0(z^3) + z^{-1}X_1(z^3) + z^{-2}X_2(z^3). \end{aligned}$$

By straightforward manipulation, we can express $X(z)$ as

$$\begin{aligned} X(z) &= \left(1 + 4z^{-3} + 7z^{-6}\right) + \left(2z^{-1} + 5z^{-4}\right) + \left(3z^{-2} + 6z^{-5}\right) \\ &= \left(1 + 4z^{-3} + 7z^{-6}\right) + z^{-1}\left(2 + 5z^{-3}\right) + z^{-2}\left(3 + 6z^{-3}\right). \end{aligned}$$

Comparing this equation to the form of the desired polyphase representation, we conclude

$$\begin{gathered} X_0(z) = 1 + 4z^{-1} + 7z^{-2}, \\ X_1(z) = 2 + 5z^{-1}, \text{ and} \\ X_2(z) = 3 + 6z^{-1}. \end{gathered}$$

□

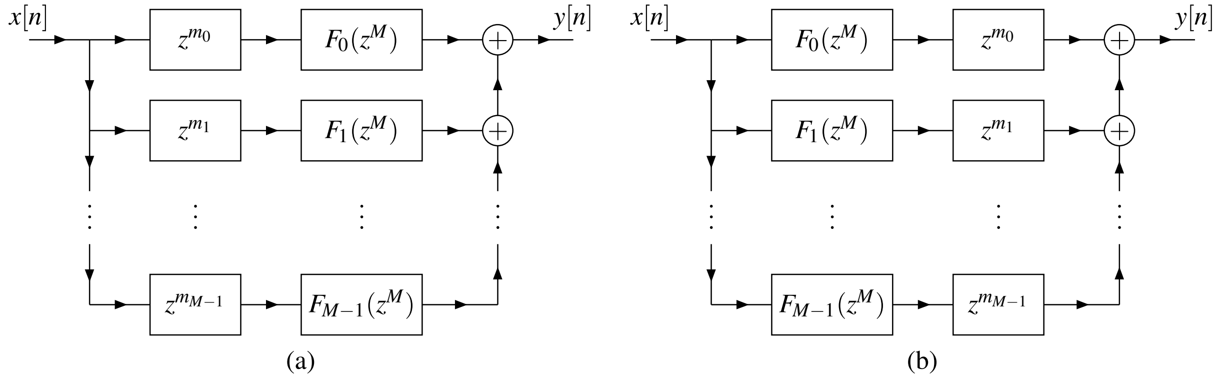


Figure 3.15: Polyphase representation of a filter.

Polyphase representations can also be applied in the context of filters. In particular, we can find a polyphase representation of the impulse response of a filter. This leads naturally to the polyphase representation of filters. Suppose that we have a filter with transfer function $F(z)$. We can express $F(z)$ in terms of the polyphase representation

$$F(z) = \sum_{k=0}^{M-1} z^{m_k} F_k(z^M).$$

This representation suggests an implementation strategy for a filter known as the polyphase realization.

The polyphase realization of a filter is shown in Figure 3.15. Two slight variations are shown but they are clearly mathematically equivalent. When realized in its M-phase polyphase form, a filter is implemented using delay/advance units, adders, and M filters each having a transfer function that is a rational polynomial expression in z^M.

For type 1, 2, 3, and 4 polyphase representations, the coset offsets $\{m_k\}_{k=0}^{M-1}$ are chosen to be consecutive integers with either the largest or smallest value being zero. This allows all of the delays/advances in these polyphase realizations to be implemented with a chain of M unit-delays/unit-advances. For example, in the case of a type-1 polyphase representation, we have the equivalences shown in Figures 3.16 and 3.17.

Example 3.11. Suppose that we have a filter with transfer function

$$F(z) = -\tfrac{1}{8}z^2 + \tfrac{1}{4}z + \tfrac{3}{4} + \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2}.$$

Find each of the following two-phase polyphase representations of $F(z)$: (a) type-1 and (b) type-3.

Solution. (a) Consider the type-1 two-phase polyphase representation. We need to express $F(z)$ in the form

$$F(z) = \sum_{k=0}^{1} z^{-k} F_k(z^2) = F_0(z^2) + z^{-1} F_1(z^2).$$

By using straightforward manipulation, we can express $F(z)$ as

$$F(z) = \left(-\tfrac{1}{8}z^2 + \tfrac{3}{4} - \tfrac{1}{8}z^{-2}\right) + z^{-1}\left(\tfrac{1}{4}z^2 + \tfrac{1}{4}\right).$$

Comparing this to the desired form, we conclude

$$F_0(z) = -\tfrac{1}{8}z + \tfrac{3}{4} - \tfrac{1}{8}z^{-1} \text{ and}$$
$$F_1(z) = \tfrac{1}{4}z + \tfrac{1}{4}.$$

The corresponding filter implementation is depicted in Figure 3.18(a).

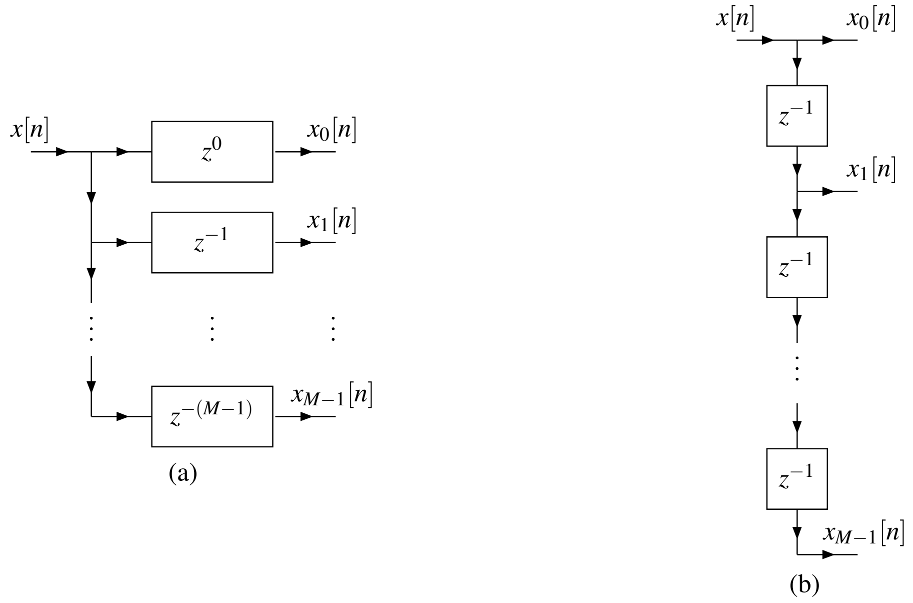


Figure 3.16: Implementation of type-1 polyphase decomposition. (a) Conceptual and (b) practical computational structures.

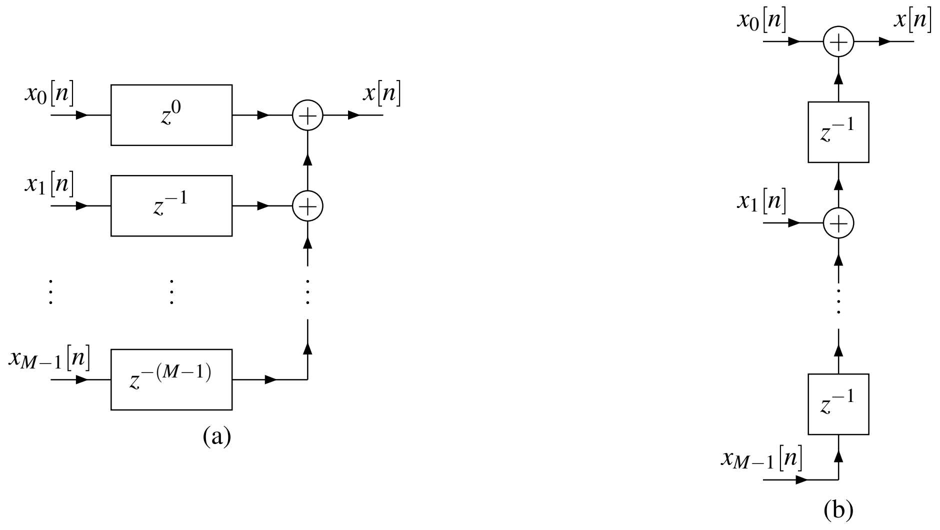
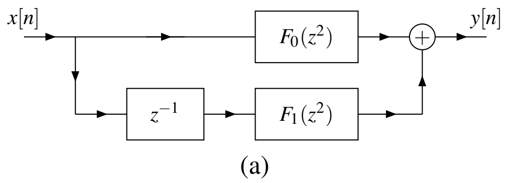


Figure 3.17: Implementation of type-1 polyphase recomposition. (a) Conceptual and (b) practical computational structures.



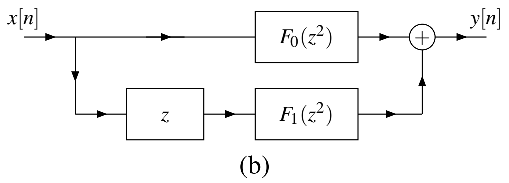


Figure 3.18: Polyphase realizations of the filter in Example 3.11. (a) Type-1 and (b) type-3 polyphase realizations.

(b) Consider the type-3 two-phase polyphase representation. We need to express $F(z)$ in the form

$$F(z) = \sum_{k=0}^{1} z^k F_k(z^2) = F_0(z^2) + zF_1(z^2).$$

From the given expression for $F(z)$, we can write

$$F(z) = \left(-\tfrac{1}{8}z^2 + \tfrac{3}{4} - \tfrac{1}{8}z^{-2}\right) + z\left(\tfrac{1}{4} + \tfrac{1}{4}z^{-2}\right).$$

Thus, we conclude

$$F_0(z) = -\tfrac{1}{8}z + \tfrac{3}{4} - \tfrac{1}{8}z^{-1} \quad \text{and}$$
$$F_1(z) = \tfrac{1}{4} + \tfrac{1}{4}z^{-1}.$$

The corresponding filter realization is illustrated in Figure 3.18(b). □

Example 3.12. Suppose that we have a filter with transfer function

$$F(z) = \tfrac{21}{128}z^6 + \tfrac{65}{128}z^5 + \tfrac{57}{128}z^4 - \tfrac{3}{128}z^3 - \tfrac{17}{128}z^2 + \tfrac{3}{128}z^1 + \tfrac{3}{128}z^0 - \tfrac{1}{128}z^{-1}.$$

Find each of the following three-phase polyphase representations of $F(z)$: (a) type-1 and (b) type-2.

Solution. (a) Consider the type-1 three-phase polyphase representation. We need to express $F(z)$ in the form

$$F(z) = \sum_{k=0}^{M-1} z^{m_k} X(z^M)$$

where $M = 3$ and $m_k = -k$. That is, we seek a representation of the form

$$F(z) = \sum_{k=0}^{2} z^{-k} F_k(z^3) = F_0(z^3) + z^{-1}F_1(z^3) + z^{-2}F_2(z^3).$$

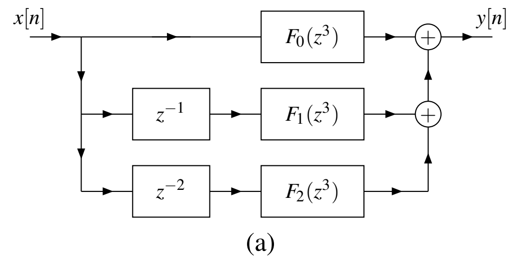
From the given expression for $F(z)$, we can write

$$F(z) = \left(\tfrac{21}{128}z^6 - \tfrac{3}{128}z^3 + \tfrac{3}{128}\right) + z^{-1}\left(\tfrac{65}{128}z^6 - \tfrac{17}{128}z^3 - \tfrac{1}{128}\right) + z^{-2}\left(\tfrac{57}{128}z^6 + \tfrac{3}{128}z^3\right).$$

Thus, we conclude

$$F_0(z) = \tfrac{21}{128}z^2 - \tfrac{3}{128}z + \tfrac{3}{128},$$
$$F_1(z) = \tfrac{65}{128}z^2 - \tfrac{17}{128}z - \tfrac{1}{128}, \quad \text{and}$$
$$F_2(z) = \tfrac{57}{128}z^2 + \tfrac{3}{128}z.$$

The corresponding filter realization is shown in Figure 3.19(a).



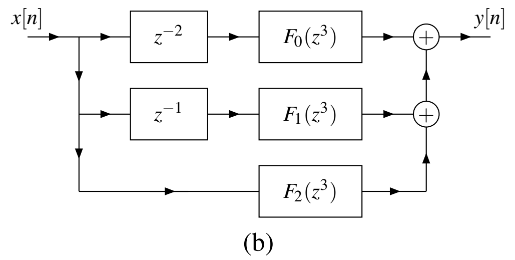


Figure 3.19: Polyphase realizations of the filter in Example 3.12. (a) Type-1 and (b) type-2 polyphase realizations.

(b) Consider the type-2 three-phase polyphase representation. We must express $F(z)$ in the form

$$F(z) = \sum_{k=0}^{2} z^{k-M+1} F_k(z^3) = z^{-2}F_0(z^3) + z^{-1}F_1(z^3) + F_2(z^3).$$

We observe that this is the same sum as in part (a), except that the polyphase components have simply been permuted. Thus, the type-2 polyphase representation is obtained by simply permuting the polyphase components from part (a). More specifically, we have

$$\begin{aligned} F_0(z) &= \tfrac{57}{128}z^2 + \tfrac{3}{128}z, \\ F_1(z) &= \tfrac{65}{128}z^2 - \tfrac{17}{128}z - \tfrac{1}{128}, \text{ and} \\ F_2(z) &= \tfrac{21}{128}z^2 - \tfrac{3}{128}z + \tfrac{3}{128}. \end{aligned}$$

The corresponding filter realization is shown in Figure 3.19(b). □

Example 3.13. Suppose that we have two filters, one with transfer function $F(z)$ and one with transfer function $G(z) = z^{-1}F(z)$. Let $\{F_k(z)\}_{k=0}^{M-1}$ and $\{G_k(z)\}_{k=0}^{M-1}$ denote the type-1 M-phase polyphase components of $F(z)$ and $G(z)$, respectively. Determine the relationship between $\{G_k(z)\}_{k=0}^{M-1}$ and $\{F_k(z)\}_{k=0}^{M-1}$.

Solution. From the definition of a type-1 polyphase decomposition, we have

$$\begin{aligned} F(z) &= \sum_{k=0}^{M-1} z^{-k}F_k(z^M), \text{ and} \\ G(z) &= \sum_{k=0}^{M-1} z^{-k}G_k(z^M). \end{aligned} \tag{3.18}$$

Using the fact that $G(z) = z^{-1}F(z)$, we can write

$$\begin{aligned} G(z) &= z^{-1}F(z) \\ &= z^{-1}\sum_{k=0}^{M-1} z^{-k}F_k(z^M) \\ &= \sum_{k=0}^{M-1} z^{-(k+1)}F_k(z^M) \\ &= \sum_{k=1}^{M} z^{-k}F_{k-1}(z^M) \\ &= [z^{-M}F_{M-1}(z^M)] + \sum_{k=1}^{M-1} z^{-k}F_{k-1}(z^M). \end{aligned}$$

This equation, however, is simply a type-1 polyphase decomposition of $G(z)$. Therefore, by comparison to (3.18), we can conclude

$$G_k(z) = \begin{cases} z^{-1}F_{M-1}(z) & \text{for } k=0 \\ F_{k-1}(z) & \text{for } k=1,2,\ldots,M-1. \end{cases}$$

□

So far, we have only considered finding polyphase representations of FIR filters. Finding polyphase representations of IIR filters is more difficult. This is demonstrated by the following example.

Example 3.14 (Polyphase representation of IIR filter)**.** Find the type-1 two-phase polyphase representation of the filter with transfer function

$$F(z) = \frac{1}{1-az^{-1}}.$$

Solution. We can rewrite $F(z)$ as

$$\begin{aligned} F(z) &= \left(\frac{1}{1-az^{-1}}\right)\left(\frac{1+az^{-1}}{1+az^{-1}}\right) \\ &= \frac{1+az^{-1}}{1-a^2z^{-2}} \\ &= \frac{1}{1-a^2z^{-2}} + \frac{az^{-1}}{1-a^2z^{-2}} \\ &= \frac{1}{1-a^2z^{-2}} + z^{-1}\frac{a}{1-a^2z^{-2}}. \end{aligned}$$

Thus, we can express $F(z)$ as $F(z) = F_0(z^2) + z^{-1}F_1(z^2)$, where

$$F_0(z) = \frac{1}{1-a^2z^{-1}} \quad \text{and} \quad F_1(z) = \frac{a}{1-a^2z^{-1}}.$$

□

As we shall see, the polyphase representation is often a mathematically convenient form in which to express filtering operations in multirate systems, facilitating easier analysis of such systems and greatly simplifying many theoretical results. Also, this representation leads to an efficient means for implementing filtering operations in a multirate framework, as elaborated upon in the next section.

3.2.10 Efficient Decimation and Interpolation

The polyphase representation is particularly important in the context of filtering operations in multirate systems. In multirate systems, filters are often connected in cascade with upsamplers/downsamplers. For reasons of computational efficiency, it is usually preferable to perform any filtering on the side of the upsampler/downsampler with the lower sampling rate. We have seen how the noble identities can be used to move a filtering operation across M-fold upsamplers/downsamplers, but in order to move filtering to the side with the lower sampling rate, the transfer function of the filter must be an expression in z^M which is not generally the case. The polyphase representation, however, provides us with a means to express any filter as a set of filters with transfer functions in z^M so that filtering can be moved to the more desirable side of an upsampler/downsampler.

One commonly occurring structure in multirate systems is the ***M*-fold decimator** which consists of an **anti-aliasing filter** followed by an M-fold downsampler as shown in Figure 3.20(a). Here, we have a filter with transfer function $F(z)$ followed by an M-fold downsampler. Obviously, this structure is computationally inefficient as the filtering is performed on the side of the downsampler with the higher sampling rate. Suppose now that we realize the filter in its M-phase polyphase form. This leads to the new system shown in Figure 3.20(b). We can equivalently

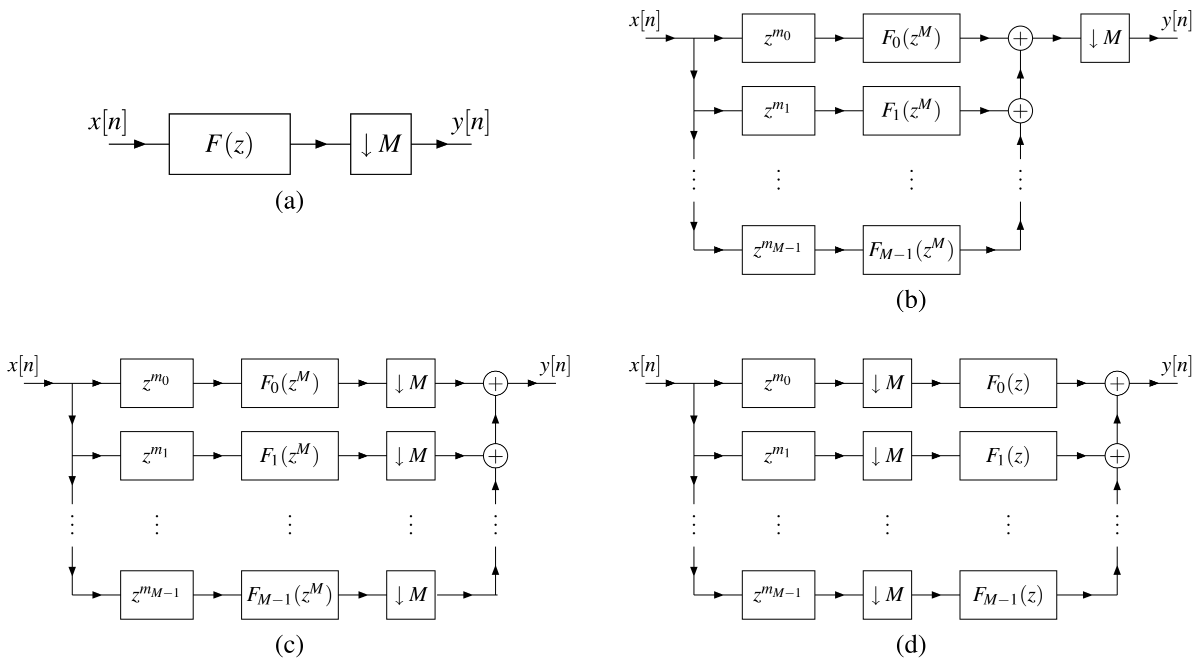


Figure 3.20: Efficient implementation of a decimator using the noble identities. (a) M-fold decimator. Transformed system obtained after (b) implementing filtering in polyphase form, (c) interchanging downsamplers and adders, and (d) interchanging downsamplers and filters.

move the downsampling operation before the adders to obtain the result shown in Figure 3.20(c). Now, the first noble identity can be used to move the downsampling operations before the polyphase filtering operations as illustrated by Figure 3.20(d). In this way, we achieve a more efficient structure for decimation. The filtering is now being performed at the lower sampling rate.

Another frequently encountered structure in multirate systems is the **M-fold interpolator** which consists of an M-fold upsampler followed by an **anti-imaging filter** as shown in Figure 3.21(a). Here, we have an M-fold upsampler followed by a filter with transfer function $F(z)$. Again, this structure is computationally inefficient as the filtering is performed on the side of the upsampler with the higher sampling rate. Using a strategy similar to the case of the decimator above, we can obtain the equivalent system shown in Figure 3.21(d). In such a case, the second noble identity can be used to move the upsampling operation after the polyphase filtering operations. This results in a new system in which filtering is performed on the side of the upsampler with the lower sampling rate.

Multirate systems often contain interpolators and decimators as building blocks. By using the approach above, we can use polyphase methods to obtain efficient decimators and interpolators.

3.2.11 Efficient Rational Sampling Rate Changes

Earlier, we considered how to change the sampling rate by a rational factor L/M. This led to the system shown in Figure 3.22. For the reasons discussed earlier, the system as shown is very computationally inefficient. In what follows, we will consider how polyphase techniques can be employed in order to improve efficiency.

To improve computational efficiency, we can implement the filter H in polyphase form and then use the noble identities to move the filtering operation across the upsampler or downsampler. If $\frac{L}{M} < 1$, it would make the most sense to move the filtering operation across the downsampler, since the sampling rate is lowest at the downsampler output. If $\frac{L}{M} > 1$, it would make the most sense to move the filtering operation across the upsampler, since the sampling rate is lowest at the upsampler input. In the first case, we can represent the filtering operation H in M-phase polyphase form and then move the filtering operation to the side of downsampler with the lower sampling rate using the noble identities. This process is illustrated in Figure 3.23. In the second case, we can represent the filtering operation H in

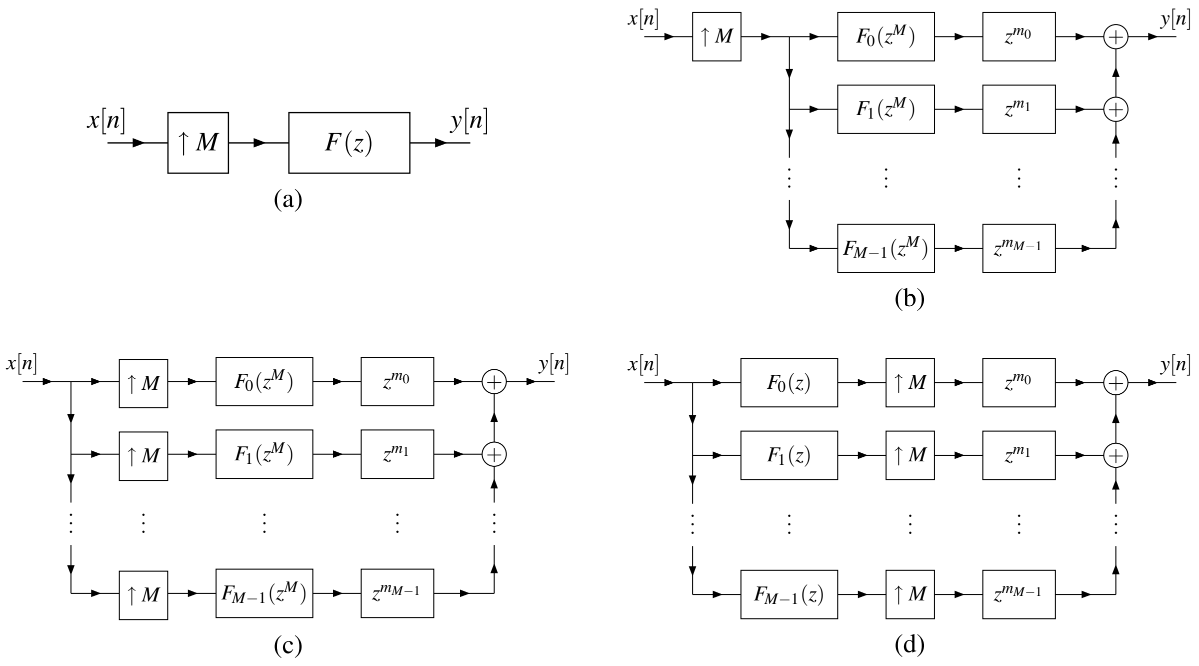


Figure 3.21: Efficient implementation of an interpolator using the noble identities. (a) M-fold interpolator. Transformed system obtained after (b) implementing filter in polyphase form, (c) moving upsamplers, and (d) interchanging upsamplers and filters.

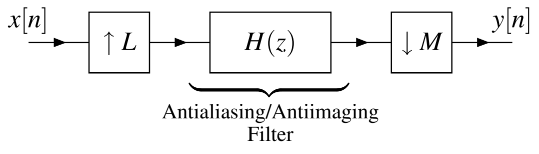


Figure 3.22: Rational sampling rate conversion system.

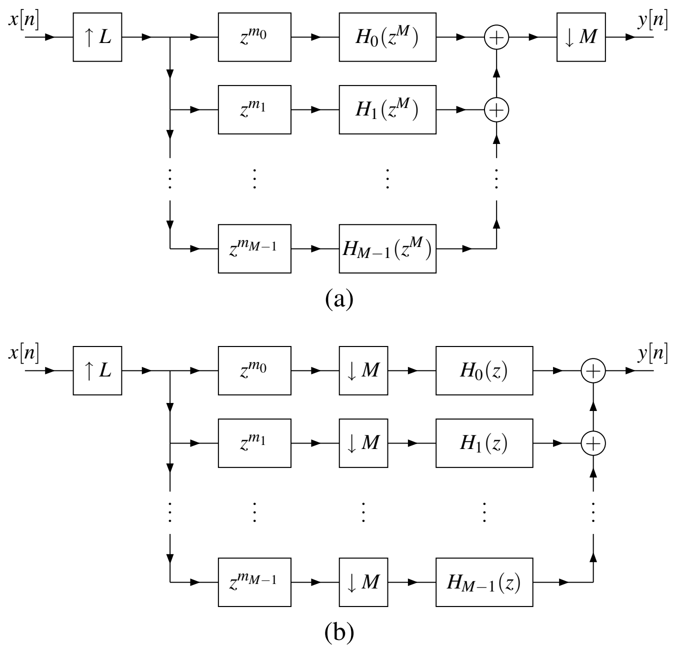


Figure 3.23: Rational sampling rate conversion system (a) before and (b) after moving the filtering operations across the downsamplers.

L-phase polyphase form and then move the filtering operation to the side of the upsampler with the lower sampling rate using the noble identities. This process is shown in Figure 3.24.

Fortunately, we can achieve even better computational efficiency by using polyphase techniques to further manipulate the systems from Figures 3.23(b) and 3.24(b). How might we improve upon this situation? In both of the preceding systems, the filtering is performed either on the side of the downsampler with the higher rate or the side of the upsampler with the higher rate. The computational efficiency could be improved if we could have the filtering performed on the side of the downsampler and the side of the upsampler with the lower sampling rate. At first glance, this may seem impossible, since downsampling follows upsampling. Appearances can be deceiving, however. Let us consider this problem more carefully (i.e., as in [16]).

Without loss of generality in what follows, we assume that L and M are coprime. (If L and M are not coprime, we can simply cancel any common factors in order to obtain new values that are coprime.) Since L and M are coprime, every integer can be expressed as an integer linear combination of L and M. This follows from Bezout's identity (i.e., Theorem E.1). In particular, for each $l_k \in \mathbb{Z}$, there exist $l'_k, m'_k \in \mathbb{Z}$ such that $l_k = Ll'_k + Mm'_k$. We start with the system obtained by moving the filtering operation across upsampling, shown in Figure 3.24(b). We then transform the system as shown in Figure 3.25. Finally, we obtain the system shown in Figure 3.26. Observe that in this system, all filtering is performed at the lowest possible sampling rate (i.e., after downsampling and before upsampling).

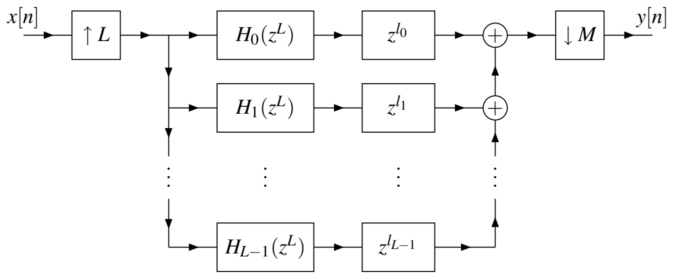
3.2.12 Polyphase Identity

In multirate systems, we sometimes encounter an upsampling operation followed by a downsampling operation (using the same sampling factor) with filtering in between. One useful identity in relation to such situations is given by the theorem below.

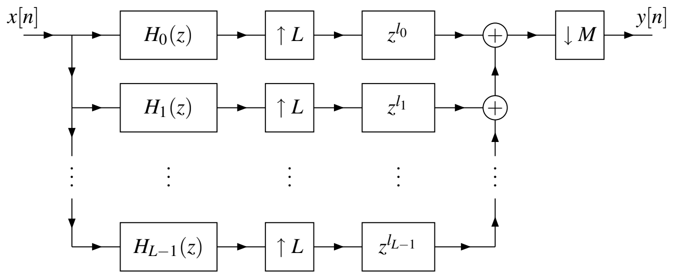
Theorem 3.6 (Polyphase identity)**.** *Let $F(z)$ and $X(z)$ denote $\mathcal{Z}$ transforms. Then, we have*

$$(\downarrow M)\left(F(z)\left[(\uparrow M)X(z)\right]\right) = F_0(z)X(z),$$

where $F_0(z) = (\downarrow M)F(z)$. This relationship is known as the ***polyphase identity****, and has the interpretation shown in Figure 3.27.*



(a)



(b)

Figure 3.24: Rational sampling rate conversion system (a) before and (b) after moving filtering operations across the upsamplers.

Proof. Since $(\uparrow M)X(z) = X(z^M)$, we have

$$(\downarrow M)(F(z)[(\uparrow M)X(z)]) = (\downarrow M)V(z),$$

where $V(z) = F(z)X(z^M)$. So, we have

$$\begin{aligned}(\downarrow M)(F(z)[(\uparrow M)X(z)]) &= \tfrac{1}{M}\sum_{k=0}^{M-1} V(z^{1/M}W_M^k)\\ &= \tfrac{1}{M}\sum_{k=0}^{M-1} F(z^{1/M}W_M^k)X([z^{1/M}W_M^k]^M)\\ &= \tfrac{1}{M}\sum_{k=0}^{M-1} F(z^{1/M}W_M^k)X(zW_M^{Mk})\\ &= \tfrac{1}{M}\sum_{k=0}^{M-1} F(z^{1/M}W_M^k)X(z)\\ &= \left[\tfrac{1}{M}\sum_{k=0}^{M-1} F(z^{1/M}W_M^k)\right]X(z)\\ &= [(\downarrow M)F(z)]X(z).\end{aligned}$$

Thus, the polyphase identity holds. □

3.2.13 Filter Banks

A collection of filters having either a common input or common output is called a **filter bank**. When the filters share a common input, they form what is called an **analysis bank**. When they share a common output, they form a **synthesis**

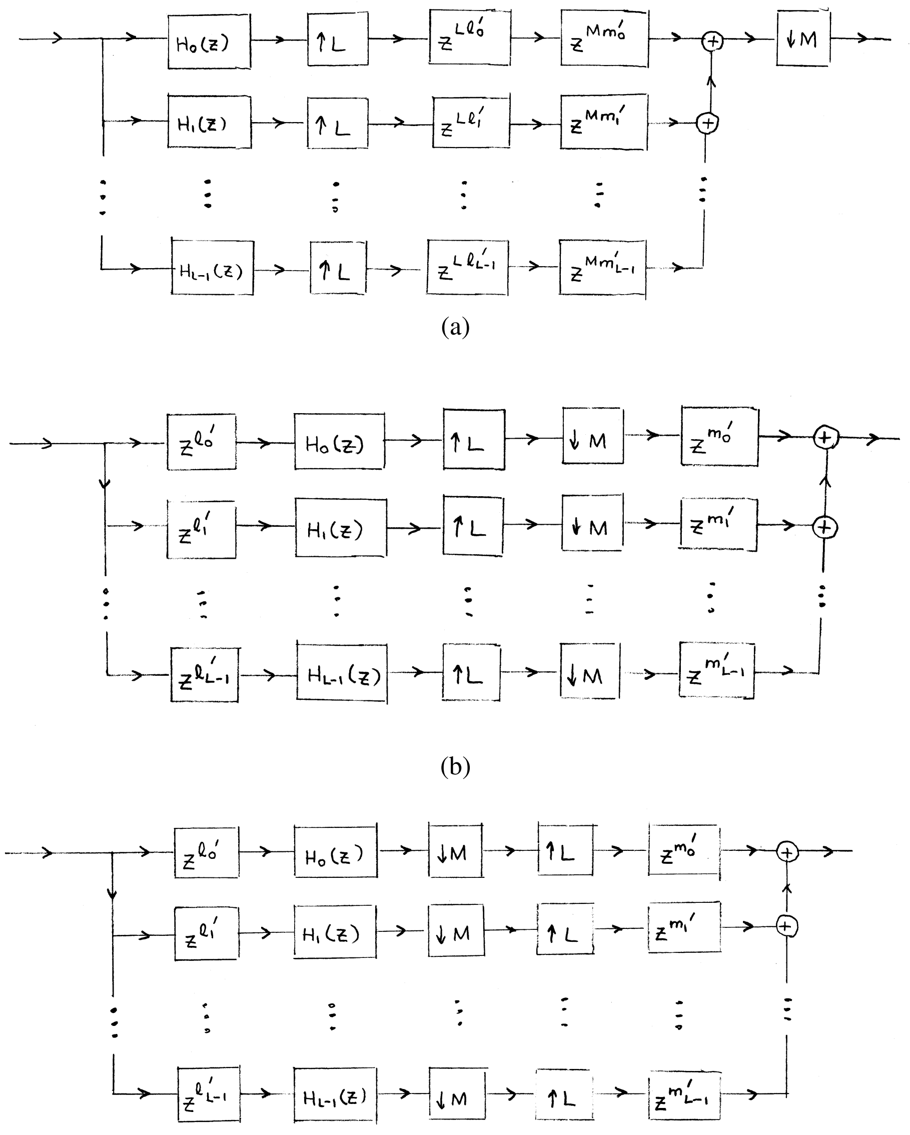


Figure 3.25: Transformation of rational sampling rate conversion system. System after (a) each delay has been split into two, (b) the delays have been moved across the downsamplers/upsamplers, (c) the downsamplers/upsamplers have been interchanged.

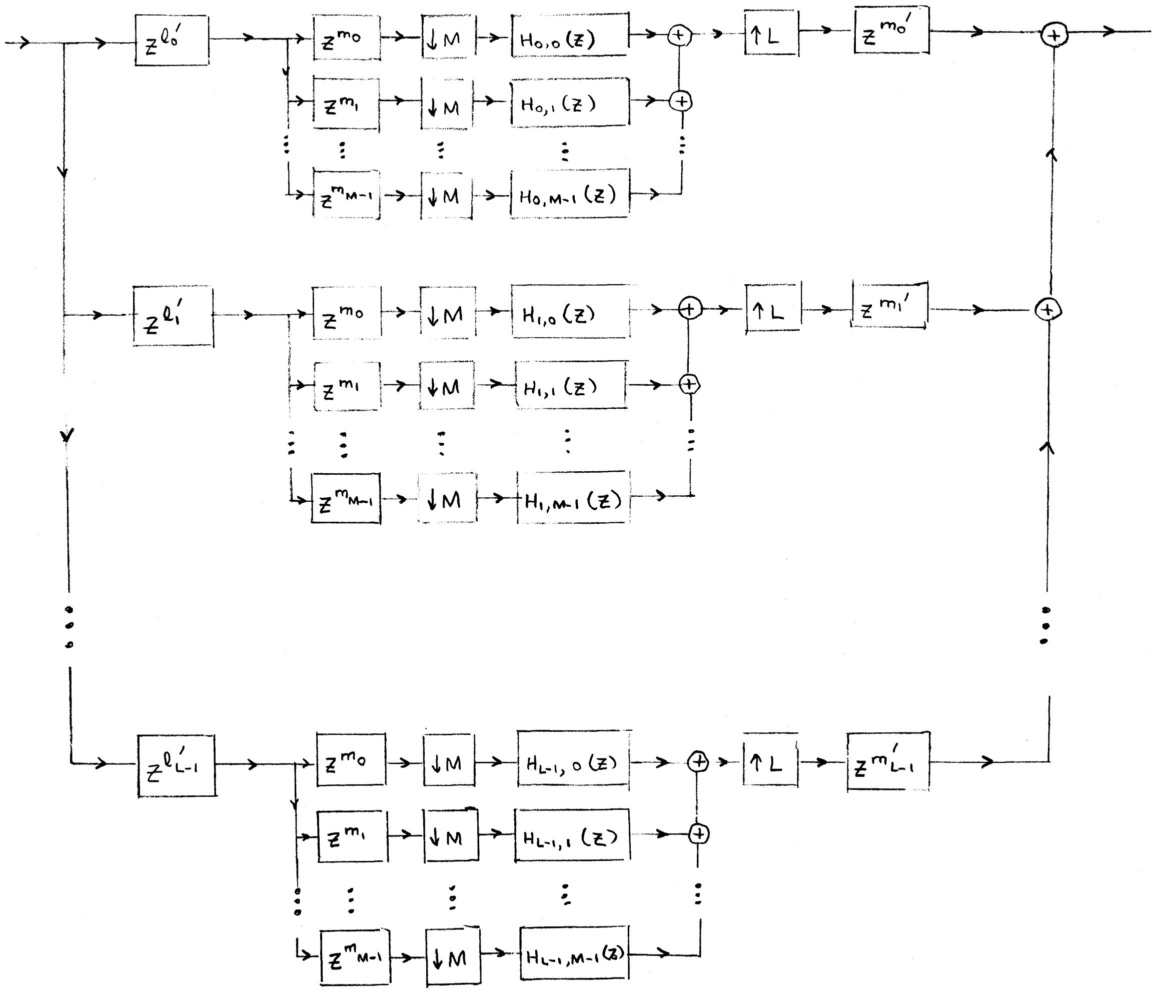


Figure 3.26: Efficient rational sampling rate conversion system.

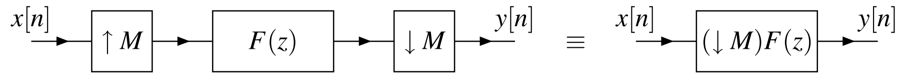


Figure 3.27: Polyphase identity.

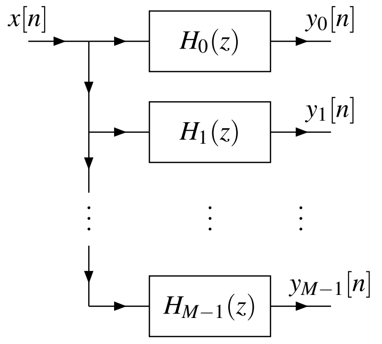


Figure 3.28: Analysis bank.

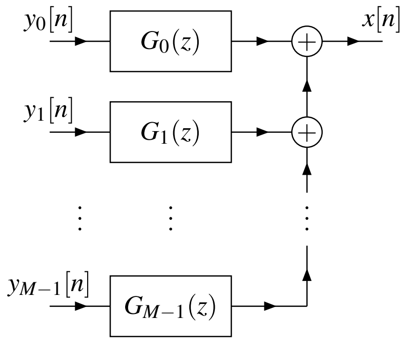


Figure 3.29: Synthesis bank.

bank. These two types of filter banks are depicted in Figures 3.28 and 3.29. Each of the filter banks shown consists of M filters. The filters $\{H_k\}_{k=0}^{M-1}$ belonging to the analysis bank are called **analysis filters** and the filters $\{F_k\}_{k=0}^{M-1}$ comprising the synthesis bank are referred to as **synthesis filters**. The signals $\{y_k\}_{k=0}^{M-1}$ are called **subband signals**. The frequency responses of the analysis/synthesis filters may be non-overlapping, marginally overlapping, or greatly overlapping depending on the application.

3.3 Uniformly Maximally Decimated (UMD) Filter Banks

Although many filter bank configurations exist, an extremely useful one is the so called **uniformly maximally decimated** (UMD) filter bank[3]. The general structure of an M-channel UMD filter bank is shown in Figure 3.30. The analysis bank and downsamplers collectively form the analysis side of the UMD filter bank. Similarly, the upsamplers and synthesis bank together constitute the synthesis side of the UMD filter bank.

The UMD filter bank (shown in Figure 3.30) operates in the following manner. On the analysis side, the input signal x is processed by the analysis filters $\{H_k\}_{k=0}^{M-1}$, and the resulting filter outputs are then downsampled by M, yielding the subband signals $\{y_k\}_{k=0}^{M-1}$. On the synthesis side, the subband signals $\{y_k\}_{k=0}^{M-1}$ are upsampled by M. Then, the upsampler outputs are transformed by the synthesis filters $\{G_k\}_{k=0}^{M-1}$, and the resulting filter outputs are summed to produce the output signal $\hat{x}$.

Each of the M subband signals $\{y_k\}_{k=0}^{M-1}$ has $\frac{1}{M}$-th the sampling density of the original input signal x. Consequently, the subband signals collectively possess the same number of samples as the original input signal. For this reason, the filter bank is referred to as **maximally decimated**. Furthermore, since all of the subband signals have the same sampling density, the filter bank is said to be **uniformly decimated**. Hence, a filter bank of the above form is referred to as **uniformly maximally decimated**.

3.3.1 Alias-Free and Perfect Reconstruction (PR) UMD Filter Banks

Recall the general structure of an M-channel UMD filter bank as shown in Figure 3.30. Since downsampling takes place on the analysis side of the filter bank, the potential exists for aliasing to occur. In principle, this problem could be solved by using filters with ideal frequency responses (such as ideal lowpass, highpass, and bandpass filters). In practice, however, we cannot realize such ideal filters and must find another way in which to address the aliasing problem. In the presence of aliasing, the overall system with input $x[n]$ and output $\hat{x}[n]$ can be shown to be a periodically time varying system with period M.

Fortunately, another solution to the aliasing problem exists, one that does not require the use of filters with ideal frequency responses. More specifically, for a given set of analysis filters, it is often possible to choose the synthesis filters in such a way that aliasing is completely cancelled. In other words, we can sometimes design a filter bank so

[3]Sometimes, the qualifier "quadrature mirror-image" (abbreviated QM) is used as a synonym for "uniformly maximally decimated", but strictly speaking this is an abuse of terminology.

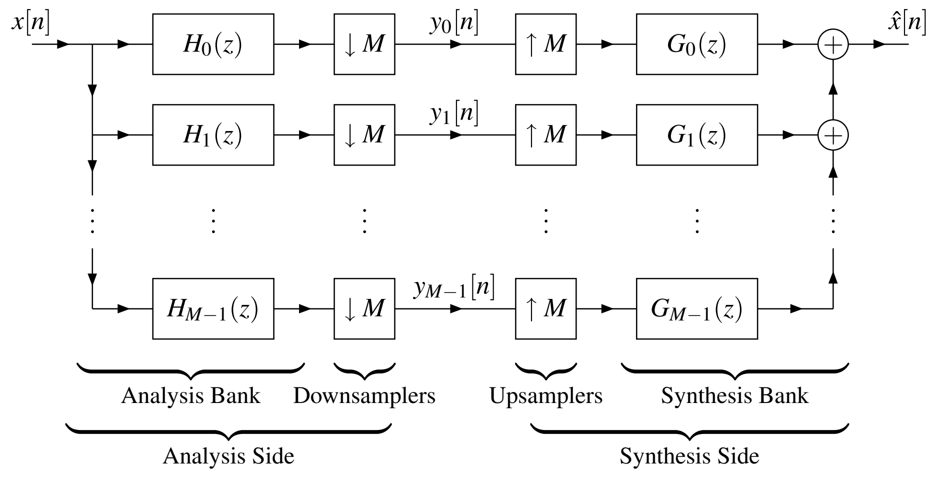


Figure 3.30: Canonical form of an M-channel UMD filter bank.

that the aliasing components obtained at the outputs of the synthesis filters will always cancel each other (i.e., add to zero). A filter bank in which the aliasing components cancel in this way is said to be **alias free**.

In the alias-free case, the overall system with input $x[n]$ and output $\hat{x}[n]$ can be shown to be LTI with transfer function (or distortion function) $T(z)$, where $T(z)$ depends on the analysis and synthesis filter transfer functions. If a system is alias free and, in addition, has a transfer function $T(z)$ of the form

$$T(z) = z^{-n_0} \quad \text{where } n_0 \in \mathbb{Z},$$

the system is said to have the **perfect reconstruction** (PR) property. In other words, a PR system can reproduce the input signal exactly, except for a possible shift. In the special case that $n_0 = 0$, the system is said to possess the **shift-free PR** property.

In light of the above discussion, we can see that there are three reasons that the reconstructed signal $\hat{x}[n]$ can differ from $x[n]$: aliasing distortion, amplitude distortion, and phase distortion. The analysis and synthesis filters can be designed in such a way so as to eliminate some or all of these distortions. In the case of an alias-free system, aliasing distortion is eliminated. In the case of a PR system, aliasing distortion, amplitude distortion, and phase distortion are all eliminated.

For obvious reasons, alias-free and PR systems are often of great practical interest. In particular, as we shall see later, UMD filter banks with the shift-free PR property play an important role in the context of wavelet systems.

3.3.2 Time-Domain Input-Output Relationship for a UMD Filter Bank

Consider the UMD filter bank shown in Figure 3.31 with input $x[n]$, subband signals $\{y_k[n]\}$, and various intermediate signals $\{u_k[n]\}$, $\{v_k[n]\}$, and $\{w_k[n]\}$. Let $g_k[n]$ and $h_k[n]$ denote the inverse $\mathcal{Z}$ transforms of $G_k(z)$ and $H_k(z)$, respectively. The action of the filter bank has an interesting mathematical interpretation which we explore below.

First, consider the analysis side of the filter bank. The output of the kth analysis filter can be expressed as

$$\begin{aligned} u_k[n] &= x * h_k[n] \\ &= \sum_{l \in \mathbb{Z}} x[l] h_k[n-l]. \end{aligned}$$

Thus, the kth subband signal $y_k[n]$ is given by

$$\begin{aligned} y_k[n] &= u_k[Mn] \\ &= \sum_{l \in \mathbb{Z}} x[l] h_k[Mn-l]. \end{aligned} \tag{3.19}$$

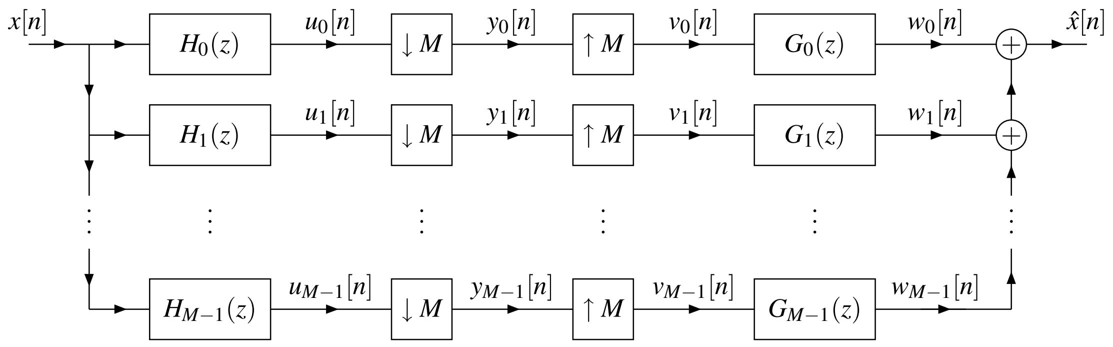


Figure 3.31: *M*-channel UMD filter bank.

Now, consider the synthesis side of the filter bank. The input of the *k*th synthesis filter can be expressed as

$$v_k[n] = \begin{cases} y_k[n/M] & \text{if } M \mid n \\ 0 & \text{otherwise.} \end{cases}$$

From this, we can deduce that the output of the *k*th synthesis filter is given by

$$\begin{aligned} w_k[n] &= v_k * g_k[n] \\ &= \sum_{l \in \mathbb{Z}} v_k[l] g_k[n-l] \\ &= \sum_{\substack{l \in \mathbb{Z} \\ M \mid l}} y_k[l/M] g_k[n-l] \\ &= \sum_{l \in \mathbb{Z}} y_k[l] g_k[n-Ml]. \end{aligned} \tag{3.20}$$

Thus, the system output $\hat{x}[n]$ can be expressed as

$$\begin{aligned} \hat{x}[n] &= \sum_{k=0}^{M-1} w_k[n] \\ &= \sum_{k=0}^{M-1} \sum_{l \in \mathbb{Z}} y_k[l] g_k[n-Ml]. \end{aligned} \tag{3.21}$$

The input-output behavior of the system is completely characterized by (3.19) and (3.21) above. We can also express these equations in matrix form. Let us define

$$\boldsymbol{x} = \begin{bmatrix} \cdots & x[-1] & x[0] & x[1] & \cdots \end{bmatrix}^T, \quad \hat{\boldsymbol{x}} = \begin{bmatrix} \cdots & \hat{x}[-1] & \hat{x}[0] & \hat{x}[1] & \cdots \end{bmatrix}^T,$$
$$\boldsymbol{y}_k = \begin{bmatrix} \cdots & y_k[-1] & y_k[0] & y_k[1] & \cdots \end{bmatrix}^T, \quad \text{and} \quad \boldsymbol{w}_k = \begin{bmatrix} \cdots & w_k[-1] & w_k[0] & w_k[1] & \cdots \end{bmatrix}^T.$$

Then, we can write (3.19) in matrix form as

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x} \tag{3.22}$$

where $(\boldsymbol{H}_k)_{p,q} = h_k[Mp-q]$. For example, in the case that $M = 2$, the matrix $\boldsymbol{H}_k$ is of the form

$$\boldsymbol{H}_k = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & h_k[0] & h_k[-1] & h_k[-2] & h_k[-3] & h_k[-4] & \cdots \\ \cdots & h_k[2] & h_k[1] & h_k[0] & h_k[-1] & h_k[-2] & \cdots \\ \cdots & h_k[4] & h_k[3] & h_k[2] & h_k[1] & h_k[0] & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

(i.e., the rows of $\boldsymbol{H}_k$ are two-shifts of the sequence $h_k[n]$). We can write (3.20) in matrix form as

$$\boldsymbol{w}_k = \boldsymbol{G}_k \boldsymbol{y}_k \tag{3.23}$$

where $(\boldsymbol{G}_k)_{p,q} = g_k[p - Mq]$. For example, in the case that $M = 2$, the matrix $\boldsymbol{G}_k$ is of the form

$$\boldsymbol{G}_k = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & g_k[0] & g_k[-2] & g_k[-4] & \cdots \\ \cdots & g_k[1] & g_k[-1] & g_k[-3] & \cdots \\ \cdots & g_k[2] & g_k[0] & g_k[-2] & \cdots \\ \cdots & g_k[3] & g_k[1] & g_k[-1] & \cdots \\ \cdots & g_k[4] & g_k[2] & g_k[0] & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

(i.e., the columns of $\boldsymbol{G}_k$ are the two-shifts of the sequence $g_k[n]$). We can write (3.21) in matrix form as

$$\hat{\boldsymbol{x}} = \sum_{k=0}^{M-1} \boldsymbol{w}_k.$$

Substituting (3.22) and (3.23) into the above equation, we obtain

$$\begin{aligned} \hat{\boldsymbol{x}} &= \sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{y}_k \\ &= \sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k \boldsymbol{x}. \end{aligned} \tag{3.24}$$

Thus, we can characterize the input-output behavior of the system by the preceding matrix equation.

Again, we consider (3.19) and (3.21). We can also express these equations in terms of a series expansion involving inner products. Let us define

$$\varphi_{Ml+k}[n] = g_k[n - Ml] \quad \text{and} \tag{3.25}$$

$$\tilde{\varphi}_{Ml+k}[n] = h_k^*[Ml - n] \tag{3.26}$$

for $k \in \{0, 1, \ldots, M-1\}$ and $l \in \mathbb{Z}$. (Note that (3.26) implies that $h_k[Mn - l] = \tilde{\varphi}^*_{Mn+k}[l]$.) Using this new definition, we can rewrite the expression for $y_k[n]$ in (3.19) as

$$\begin{aligned} y_k[n] &= \sum_{l \in \mathbb{Z}} x[l] \tilde{\varphi}^*_{Mn+k}[l] \\ &= \langle x, \tilde{\varphi}_{Mn+k} \rangle. \end{aligned} \tag{3.27}$$

From this equation, we can see that the analysis side of the UMD filter bank simply computes the inner product of the input signal $x[n]$ with each of the sequences $\{\tilde{\varphi}_k\}_{k \in \mathbb{Z}}$. Furthermore, by examining (3.26), we see that these sequences are simply the M-shifts of the conjugated time-reversed analysis filter impulse responses.

We can rewrite the expression for $\hat{x}[n]$ in (3.21) as

$$\begin{aligned} \hat{x}[n] &= \sum_{k=0}^{M-1} \sum_{l \in \mathbb{Z}} y_k[l] \varphi_{Ml+k}[n] \\ &= \sum_{k=0}^{M-1} \sum_{l \in \mathbb{Z}} \langle x, \tilde{\varphi}_{Ml+k} \rangle \varphi_{Ml+k}[n] \\ &= \sum_{k \in \mathbb{Z}} \langle x, \tilde{\varphi}_k \rangle \varphi_k[n]. \end{aligned} \tag{3.28}$$

From this equation, we can see that the synthesis side of the filter bank simply computes a weighted sum of the sequences $\{\varphi_k\}_{k\in\mathbb{Z}}$. Furthermore, by examining (3.25), we can see that these sequences are simply the M-shifts of the synthesis filter impulse responses.

We can also rewrite (3.28) in matrix form. Let us define

$$\tilde{\boldsymbol{\varphi}}_k = \begin{bmatrix}\cdots & \tilde{\varphi}_k[-1] & \tilde{\varphi}_k[0] & \tilde{\varphi}_k[1] & \cdots\end{bmatrix}^T \quad\text{and}\quad \boldsymbol{\varphi}_k = \begin{bmatrix}\cdots & \varphi_k[-1] & \varphi_k[0] & \varphi_k[1] & \cdots\end{bmatrix}^T.$$

Then, we can write

$$\hat{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\cdots & \boldsymbol{\varphi}_{-1} & \boldsymbol{\varphi}_0 & \boldsymbol{\varphi}_1 & \cdots\end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix}\vdots \\ \tilde{\boldsymbol{\varphi}}_{-1}^\dagger \\ \tilde{\boldsymbol{\varphi}}_0^\dagger \\ \tilde{\boldsymbol{\varphi}}_1^\dagger \\ \vdots\end{bmatrix}}_{\tilde{\boldsymbol{A}}} \boldsymbol{x}.$$

For example, in the case that $M = 2$, the matrices $\tilde{\boldsymbol{A}}$ and $\boldsymbol{A}$ have the respective forms

$$\boldsymbol{A} = \begin{bmatrix}
\ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\
\cdots & g_0[0] & g_1[0] & g_0[-2] & g_1[-2] & g_0[-4] & g_1[-4] & \cdots \\
\cdots & g_0[1] & g_1[1] & g_0[-1] & g_1[-1] & g_0[-3] & g_1[-3] & \cdots \\
\cdots & g_0[2] & g_1[2] & g_0[0] & g_1[0] & g_0[-2] & g_1[-2] & \cdots \\
\cdots & g_0[3] & g_1[3] & g_0[1] & g_1[1] & g_0[-1] & g_1[-1] & \cdots \\
\cdots & g_0[4] & g_1[4] & g_0[2] & g_1[2] & g_0[0] & g_1[0] & \cdots \\
\iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix} \quad\text{and}$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix}
\ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\
\cdots & h_0[0] & h_0[-1] & h_0[-2] & h_0[-3] & h_0[-4] & \cdots \\
\cdots & h_1[0] & h_1[-1] & h_1[-2] & h_1[-3] & h_1[-4] & \cdots \\
\cdots & h_0[2] & h_0[1] & h_0[0] & h_0[-1] & h_0[-2] & \cdots \\
\cdots & h_1[2] & h_1[1] & h_1[0] & h_1[-1] & h_1[-2] & \cdots \\
\cdots & h_0[4] & h_0[3] & h_0[2] & h_0[1] & h_0[0] & \cdots \\
\cdots & h_1[4] & h_1[3] & h_1[2] & h_1[1] & h_1[0] & \cdots \\
\iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}.$$

So, we have

$$\hat{\boldsymbol{x}} = \boldsymbol{A}\tilde{\boldsymbol{A}}\boldsymbol{x}. \tag{3.29}$$

We will use (3.24) and (3.29) in the next section to study the conditions for PR and alias cancellation.

3.3.3 Filter Banks and Series Expansions of Discrete-Time Signals

Let us reconsider (3.28). Suppose now that $\{\varphi_k\}$ and $\{\tilde{\varphi}_k\}$ are biorthonormal. This implies

$$\langle \tilde{\varphi}_k, \varphi_l \rangle = \delta[k-l].$$

From the definitions of φ_k and $\tilde{\varphi}_l$ in (3.25) and (3.26), we can rewrite this relationship as

$$\left\langle h_k^*[Ml-\cdot], g_p[\cdot - Mq] \right\rangle = \delta[k-p]\delta[l-q]$$

or equivalently

$$\left\langle h_k^*[Ml-\cdot], g_p[\cdot]\right\rangle = \delta[k-p]\delta[l]. \tag{3.30}$$

Now, consider the special case that $\tilde{\varphi}_k = \varphi_k$ for all $k \in \mathbb{Z}$. That is, $\{\varphi_k\}$ is orthonormal. From the definitions of φ_k and $\tilde{\varphi}_k$, we see that orthonormality implies

$$h_k[n] = g_k^*[-n].$$

Substituting this result into (3.30), we obtain

$$\left\langle g_k[\cdot - Ml], g_p[\cdot]\right\rangle = \delta[k-p]\delta[l]. \tag{3.31}$$

As a matter of terminology, a UMD filter bank that computes a biorthonormal series expansion is called a biorthonormal filter bank. Similarly, a UMD filter bank that computes an orthonormal series expansion is called an orthonormal filter bank.

3.3.4 Time-Domain Conditions for Alias-Free and PR Systems

Often, we are interested in UMD filter banks with the alias-free or PR property. Therefore, one might wonder how these properties manifest themselves in the time-domain representations of such filter banks developed earlier. First, we will consider alias-free systems. The theorem below offers necessary and sufficient conditions for alias cancellation, expressed in the time domain.

Theorem 3.7 (Necessary and sufficient conditions for alias cancellation)**.** *Suppose that we have a UMD filter bank characterized by the matrices $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_{M-1}$ and $\boldsymbol{H}_0, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_{M-1}$ as defined in (3.24) and the matrices $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ as defined in (3.29). Then, the system is alias free if and only if any one of the following equivalent conditions is satisfied:*

$$\sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k \text{ is a Toeplitz matrix};$$
$$\boldsymbol{A}\tilde{\boldsymbol{A}} \text{ is a Toeplitz matrix}.$$

Proof. A UMD filter bank with input $\boldsymbol{x}$ and output $\hat{\boldsymbol{x}}$ is alias free if and only if it corresponds to a LTI system. In turn, we have that a LTI system is one that can be characterized by a convolution operation. When sequences are expressed in vector form, convolution is equivalent to multiplication by a Toeplitz matrix. Since $\hat{\boldsymbol{x}} = (\sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k)\boldsymbol{x}$, we have that the system is alias free if and only if $\sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k$ is a Toeplitz matrix. Similarly, since $\hat{\boldsymbol{x}} = \boldsymbol{A}\tilde{\boldsymbol{A}}\boldsymbol{x}$, we have that the system is LTI if and only if $\boldsymbol{A}\tilde{\boldsymbol{A}}$ is a Toeplitz matrix. □

Now, we consider UMD filter banks with the PR property. The theorem below gives necessary and sufficient conditions for a PR system.

Theorem 3.8 (Necessary and sufficient conditions for PR)**.** *Suppose that we have a UMD filter bank characterized by the matrices $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_{M-1}$ and $\boldsymbol{H}_0, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_{M-1}$ as defined in (3.24) and the matrices $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ as defined in (3.29). Let $\boldsymbol{S}$ be a matrix such that $[\boldsymbol{S}]_{p,q} = \delta[p-q+1]$ (i.e., $\boldsymbol{S}$ corresponds to a unit-advance operator). Then, the system has the PR property if and only if any one of the following equivalent conditions is satisfied:*

$$\sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k = \boldsymbol{S}^{-n_0};$$
$$\boldsymbol{A}\tilde{\boldsymbol{A}} = \boldsymbol{S}^{-n_0};$$
$$\langle h_k^*[Mm-\cdot], g_l[\cdot]\rangle = \delta[k-l+n_0]\delta[m];$$

where $n_0 \in \mathbb{Z}$ is the reconstruction delay of the system. Furthermore, the system has the shift-free PR property if and only if any one of the following equivalent conditions is satisfied:

$$\sum_{k=0}^{M-1} \boldsymbol{G}_k \boldsymbol{H}_k = \boldsymbol{I}; \tag{3.32a}$$

$$\boldsymbol{A}\tilde{\boldsymbol{A}} = \boldsymbol{I}; \tag{3.32b}$$

$$\langle h_k^*[Mm-\cdot], g_l[\cdot] \rangle = \delta[k-l]\delta[m]. \tag{3.32c}$$

Proof. Equations (3.32a) and (3.32b) follow directly from (3.24) and (3.29), respectively. Now, we need to show that (3.32c) holds. We have

$$\tilde{\boldsymbol{A}}\boldsymbol{A} = \boldsymbol{I},$$

or equivalently

$$\begin{bmatrix} \vdots \\ \tilde{\boldsymbol{\varphi}}_{-1}^{\dagger} \\ \tilde{\boldsymbol{\varphi}}_{0}^{\dagger} \\ \tilde{\boldsymbol{\varphi}}_{1}^{\dagger} \\ \vdots \end{bmatrix} \begin{bmatrix} \cdots & \boldsymbol{\varphi}_{-1} & \boldsymbol{\varphi}_0 & \boldsymbol{\varphi}_1 & \cdots \end{bmatrix} = \boldsymbol{I}.$$

From this, we may deduce

$$\tilde{\boldsymbol{\varphi}}_l^{\dagger} \boldsymbol{\varphi}_k = \delta[k-l].$$

This can be rewritten in terms of an inner product as

$$\langle \varphi_k, \tilde{\varphi}_l \rangle = \delta[k-l].$$

Using the definition of φ_k and $\tilde{\varphi}_l$, we have

$$\langle g_k[Mm-\cdot], h_l^*[\cdot] \rangle = \delta[k-l]\delta[m].$$

□

3.3.5 $\mathcal{Z}$-Domain Input-Output Relationship for a UMD Filter Bank

For obvious reasons, we are often interested in the relationship between the input and output of a UMD filter bank. In this section, we will derive this relationship. Consider the M-channel UMD filter bank shown in Figure 3.32. The system has analysis filters $\{H_k(z)\}_{k=0}^{M-1}$, synthesis filters $\{G_k(z)\}_{k=0}^{M-1}$, input $x[n]$, output $\hat{x}[n]$, and various intermediate signals $\{u_k[n]\}$, $\{y_k[n]\}$, $\{v_k[n]\}$.

First, let us consider the analysis side of the filter bank. To begin, we observe that

$$U_k(z) = H_k(z)X(z).$$

Using the preceding relationship and the $\mathcal{Z}$-domain downsampling formula, we have that

$$\begin{aligned} Y_k(z) &= \tfrac{1}{M}\sum_{l=0}^{M-1} U_k(z^{1/M}W^l) \\ &= \tfrac{1}{M}\sum_{l=0}^{M-1} H_k(z^{1/M}W^l)X(z^{1/M}W^l). \end{aligned}$$

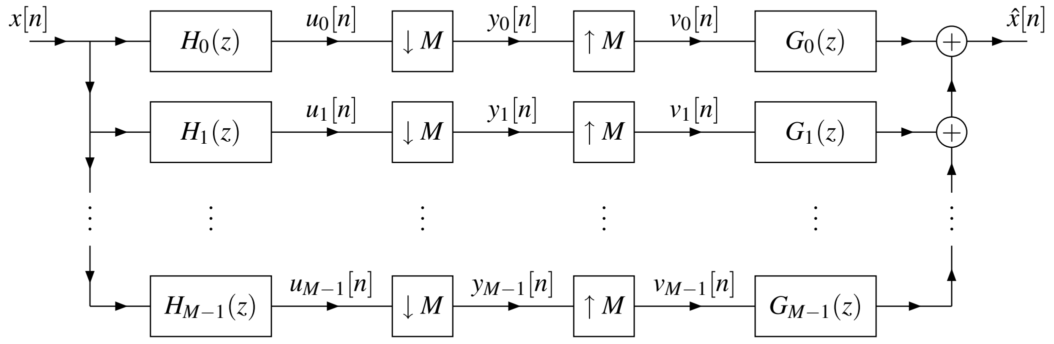


Figure 3.32: An M-channel UMD filter bank.

Thus, the analysis side of the filter bank is characterized by the equation

$$Y_k(z) = \tfrac{1}{M} \sum_{l=0}^{M-1} H_k(z^{1/M} W^l) X(z^{1/M} W^l). \tag{3.33}$$

Next, let us consider the synthesis side of the filter bank. Using the $\mathcal{Z}$-domain upsampling formula, we have that

$$\begin{aligned}\hat{X}(z) &= \sum_{k=0}^{M-1} G_k(z) V_k(z) \\ &= \sum_{k=0}^{M-1} G_k(z) Y_k(z^M).\end{aligned}$$

Thus, the synthesis side of the filter bank is characterized by the equation

$$\hat{X}(z) = \sum_{k=0}^{M-1} G_k(z) Y_k(z^M). \tag{3.34}$$

Finally, we consider the combined behavior of the analysis and synthesis sides of the filter bank. Substituting (3.33) into (3.34) and using relationship (3.11), we have

$$\begin{aligned}\hat{X}(z) &= \sum_{k=0}^{M-1} G_k(z) \left[\tfrac{1}{M} \sum_{l=0}^{M-1} H_k(\lambda^{1/M} W^l) X(\lambda^{1/M} W^l) \right] \Bigg|_{\lambda = z^M} \\ &= \sum_{k=0}^{M-1} G_k(z) \left[\tfrac{1}{M} \sum_{l=0}^{M-1} H_k(zW^l) X(zW^l) \right] \\ &= \sum_{l=0}^{M-1} X(zW^l) \tfrac{1}{M} \sum_{k=0}^{M-1} H_k(zW^l) G_k(z).\end{aligned}$$

Rewriting the preceding equation in a slightly different form, we obtain

$$\hat{X}(z) = \sum_{l=0}^{M-1} A_l(z) X(zW^l) \tag{3.35a}$$

where

$$A_l(z) = \tfrac{1}{M} \sum_{k=0}^{M-1} G_k(z) H_k(zW^l). \tag{3.35b}$$

Thus, we see that the output of the system is formed by filtering the input and its $(M-1)$ aliased versions and then summing the result. The terms in the summation for which $l \neq 0$ (i.e., the terms containing $X(\alpha z)$ where $\alpha \neq 1$) correspond to aliasing (as they involve aliased versions of the input).

From (3.35), we can deduce that a system is alias free if and only if

$$A_l(z) \equiv 0 \;\text{ for } l = 1, \ldots, M-1. \tag{3.36}$$

Furthermore, if the system is alias free, it has the transfer function $A_0(z)$. Thus, we can conclude that a system has the PR property if and only if the system is alias free and $A_0(z)$ is of the form $A_0(z) = z^{-n_0}$ where $n_0 \in \mathbb{Z}$.

Example 3.15 (Two-channel UMD filter bank**.** Consider the case of a two-channel UMD filter bank (i.e., $M = 2$). From (3.35), the relationship between the input and output is given by

$$\begin{aligned}\hat{X}(z) &= A_0(z)X(z) + A_1(z)X(-z) \\ &= \tfrac{1}{2}\left[H_0(z)G_0(z) + H_1(z)G_1(z)\right]X(z) + \tfrac{1}{2}\left[H_0(-z)G_0(z) + H_1(-z)G_1(z)\right]X(-z).\end{aligned}$$

Thus, the system is alias free if and only if

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0.$$

The system has the PR property if and only if the preceding condition is satisfied along with

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2z^{-n_0},$$

where $n_0 \in \mathbb{Z}$ is the reconstruction delay. If $n_0 = 0$, the system has the shift-free PR property.

3.3.6 Modulation Representation of UMD Filter Banks

In the previous section, we found the $\mathcal{Z}$-domain relationship between the input and output of a UMD filter bank to be given by (3.35). This equation can be written in matrix form as

$$\hat{X}(z) = \begin{bmatrix} X(z) & X(zW) & \cdots & X(zW^{M-1}) \end{bmatrix} \boldsymbol{A}(z) \tag{3.37a}$$

where $W = W_M = e^{-j2\pi/M}$ and

$$\underbrace{\begin{bmatrix} A_0(z) \\ A_1(z) \\ \vdots \\ A_{M-1}(z) \end{bmatrix}}_{\boldsymbol{A}(z)} = \frac{1}{M} \underbrace{\begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \\ H_0(zW) & H_1(zW) & \cdots & H_{M-1}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(zW^{M-1}) & H_1(zW^{M-1}) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{a}}(z)} \underbrace{\begin{bmatrix} G_0(z) \\ G_1(z) \\ \vdots \\ G_{M-1}(z) \end{bmatrix}}_{\boldsymbol{g}(z)}. \tag{3.37b}$$

with the left-hand side $\boldsymbol{A}(z) =$ as printed.

As a matter of terminology, $\boldsymbol{H}_{\mathsf{a}}(z)$ is called the **analysis alias component (AC) matrix** and its transpose

$$\boldsymbol{H}_{\mathsf{m}}(z) = \boldsymbol{H}_{\mathsf{a}}^T(z) = \begin{bmatrix} H_0(z) & H_0(zW) & \cdots & H_0(zW^{M-1}) \\ H_1(z) & H_1(zW) & \cdots & H_1(zW^{M-1}) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1}(z) & H_{M-1}(zW) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix}$$

is referred to as the **analysis modulation matrix**. We can rewrite (3.37b) in terms of $\boldsymbol{H}_{\mathsf{m}}(z)$ (rather than $\boldsymbol{H}_{\mathsf{a}}(z)$) as

$$\boldsymbol{A}(z) = \tfrac{1}{M}\boldsymbol{H}_{\mathsf{m}}^T(z)\boldsymbol{g}(z). \tag{3.38}$$

Using the modulation matrix, we can formulate a necessary and sufficient condition for a UMD filter bank to be alias free. From our earlier results, we know that a system is alias-free if and only if $A_k(z) \equiv 0$ for $k = 1, 2, \ldots, M-1$.

Furthermore, we know that the PR property is achieved if and only if the system is alias free and $A_0(z)$ is of the form $A(z) = z^{-n_0}$ for some $n_0 \in \mathbb{Z}$. Thus, we have that a UMD filter bank is alias free if and only if

$$\tfrac{1}{M}\boldsymbol{H}_{\mathsf{m}}^T(z)\boldsymbol{g}(z) = \boldsymbol{t}(z)$$

where

$$\boldsymbol{t}(z) = \begin{bmatrix} T(z) & 0 & \cdots & 0 \end{bmatrix}^T.$$

Furthermore, we also have that the filter bank has the PR property if and only if the above condition is satisfied and $T(z)$ is of the form $T(z) = z^{-n_0}$ for some $n_0 \in \mathbb{Z}$.

Another matrix that is sometimes of interest is called the synthesis modulation matrix. The **synthesis modulation matrix $\boldsymbol{G}_{\mathsf{m}}(z)$** is defined as

$$\boldsymbol{G}_{\mathsf{m}}(z) = \begin{bmatrix} G_0(z) & G_1(z) & \cdots & G_{M-1}(z) \\ G_0(zW) & G_1(zW) & \cdots & G_{M-1}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ G_0(zW^{M-1}) & G_1(zW^{M-1}) & \cdots & G_{M-1}(zW^{M-1}) \end{bmatrix}.$$

The definition of $\boldsymbol{G}_{\mathsf{m}}(z)$ is analogous to that of $\boldsymbol{H}_{\mathsf{m}}(z)$ with one minor (but noteworthy) difference. In the case of $\boldsymbol{H}_{\mathsf{m}}(z)$, $H_k(z)$ and its aliased versions appear in the kth *row* of $\boldsymbol{H}_{\mathsf{m}}(z)$, whereas in the case of $\boldsymbol{G}_{\mathsf{m}}(z)$, $G_k(z)$ and its aliased versions appear in the kth *column* of $\boldsymbol{G}_{\mathsf{m}}(z)$. Although the synthesis modulation matrix does not have a direct interpretation in terms of (3.38), this matrix is often useful in characterizing filter bank behavior. The analysis and synthesis modulation matrices, taken together, completely characterize the UMD filter bank. That is, if we know these two matrices, we can determine the analysis and synthesis filters, and these filters completely characterize the input-output behavior of the system.

Example 3.16. Suppose that we have a two-channel UMD filter bank with analysis filters $\{H_k(z)\}_{k=0}^{1}$ and synthesis filters $\{G_k(z)\}_{k=0}^{1}$, where

$$\begin{aligned} H_0(z) &= \tfrac{1}{2}z + \tfrac{1}{2}, \\ H_1(z) &= z - 1, \\ G_0(z) &= 1 + z^{-1}, \text{ and} \\ G_1(z) &= -\tfrac{1}{2} + \tfrac{1}{2}z^{-1}. \end{aligned}$$

Find the analysis and synthesis modulation matrices of this system.

Solution. From the definition of the analysis modulation matrix, we can write

$$\boldsymbol{H}_{\mathsf{m}}(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}z + \frac{1}{2} & -\frac{1}{2}z + \frac{1}{2} \\ z - 1 & -z - 1 \end{bmatrix}.$$

From the definition of the synthesis modulation matrix, we can write

$$\boldsymbol{G}_{\mathsf{m}}(z) = \begin{bmatrix} G_0(z) & G_1(z) \\ G_0(-z) & G_1(-z) \end{bmatrix} = \begin{bmatrix} 1 + z^{-1} & -\frac{1}{2} + \frac{1}{2}z^{-1} \\ 1 - z^{-1} & -\frac{1}{2} - \frac{1}{2}z^{-1} \end{bmatrix}.$$

□

Although the modulation matrices can often prove beneficial in the study of UMD filter banks, the use of such matrices does have one potential drawback. In the case of M-channel systems where $M > 2$, even when the analysis filters have transfer functions with real coefficients, the modulation matrices can have entries with complex coefficients. For this reason, the modulation representation can sometimes be cumbersome to use. This shortcoming is illustrated by the example below.

Example 3.17. Suppose that we have a three-channel UMD filter bank with analysis filters $\{H_k(z)\}_{k=0}^{2}$, where

$$\begin{aligned} H_0(z) &= \tfrac{1}{3}z^2 + \tfrac{1}{3}z + \tfrac{1}{3}, \\ H_1(z) &= z - 1, \\ H_2(z) &= z^2 - z. \end{aligned}$$

Find the analysis modulation matrix of this system.

Solution. Let $W = W_3 = e^{-j2\pi/3}$. From the definition of the analysis modulation matrix, we can write

$$\begin{aligned} \boldsymbol{H}_{\mathsf{m}}(z) &= \begin{bmatrix} H_0(z) & H_0(zW) & H_0(zW^2) \\ H_1(z) & H_1(zW) & H_1(zW^2) \\ H_2(z) & H_2(zW) & H_2(zW^2) \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{3}z^2 + \frac{1}{3}z + \frac{1}{3} & \frac{1}{3}z^2W^2 + \frac{1}{3}zW + \frac{1}{3} & \frac{1}{3}z^2W^4 + \frac{1}{3}zW^2 + \frac{1}{3} \\ z-1 & zW-1 & zW^2-1 \\ z^2-z & z^2W^2 - zW & z^2W^4 - zW^2 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{3}z^2 + \frac{1}{3}z + \frac{1}{3} & \frac{1}{3}e^{-j4\pi/3}z^2 + \frac{1}{3}e^{-j2\pi/3}z + \frac{1}{3} & \frac{1}{3}e^{-j2\pi/3}z^2 + \frac{1}{3}e^{-j4\pi/3}z + \frac{1}{3} \\ z-1 & e^{-j2\pi/3}z - 1 & e^{-j4\pi/3}z - 1 \\ z^2-z & e^{-j4\pi/3}z^2 - e^{-j2\pi/3}z & e^{-j2\pi/3}z^2 - e^{-j4\pi/3}z \end{bmatrix}. \end{aligned}$$

Observe that the coefficients of the entries in $\boldsymbol{H}_{\mathsf{m}}(z)$ are complex in spite of the fact that the $H_k(z)$ all have real coefficients. □

Suppose now that we have a set of analysis filters $\{H_k(z)\}$ and want to find a corresponding set of synthesis filters $\{G_k(z)\}$ to achieve an alias-free or PR system with a particular $\{A_k(z)\}$. To do this, we can solve for $\boldsymbol{g}(z)$ in terms of $\boldsymbol{H}_{\mathsf{m}}(z)$ and $\boldsymbol{A}(z)$ in (3.38) to obtain

$$\boldsymbol{g}(z) = M\boldsymbol{H}_{\mathsf{m}}^{-T}(z)\boldsymbol{A}(z).$$

This equation is valid, provided that $\det\boldsymbol{H}_{\mathsf{m}}(z) \not\equiv 0$. There are, however, a few problems with the above design approach. First, the synthesis filters could be IIR even if the analysis filters are FIR. Second, since the synthesis filters can be IIR, they may be unstable. Third, the synthesis filters could be of considerably higher order than the analysis filters.

3.3.7 Polyphase Representation of UMD Filter Banks

Although the structure for the UMD filter bank shown in Figure 3.30 may be intuitively appealing, it is often not the most convenient structure with which to work. This leads us to the polyphase representation of a UMD filter bank. The polyphase representation has many advantages, but most importantly it simplifies many theoretical results and suggests an efficient means by which to implement filter banks. The polyphase representation of a UMD filter bank is based on the polyphase representation of filters introduced earlier.

Suppose that we have an M-channel UMD filter bank with analysis filters $\{H_k\}$ and synthesis filters $\{G_k\}$. First, let us consider the analysis filters of the UMD filter bank. We can express the transfer functions of these filters in polyphase form as

$$H_k(z) = \sum_{p=0}^{M-1} z^{m_p} H_{k,p}(z^M)$$

for $k = 0, 1, \ldots, M-1$. (Note that each filter in the analysis bank is represented with the same type of polyphase representation; that is, $\{m_k\}_{k=0}^{M-1}$ is fixed for all analysis filters.) The resulting set of equations can be rewritten in matrix form as

$$\underbrace{\begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix}}_{\boldsymbol{h}(z)} = \underbrace{\begin{bmatrix} H_{0,0}(z^M) & H_{0,1}(z^M) & \cdots & H_{0,M-1}(z^M) \\ H_{1,0}(z^M) & H_{1,1}(z^M) & \cdots & H_{1,M-1}(z^M) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1,0}(z^M) & H_{M-1,1}(z^M) & \cdots & H_{M-1,M-1}(z^M) \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z^M)} \underbrace{\begin{bmatrix} z^{m_0} \\ z^{m_1} \\ \vdots \\ z^{m_{M-1}} \end{bmatrix}}_{\boldsymbol{v}(z)}$$

or more compactly as

$$\boldsymbol{h}(z) = \boldsymbol{H}_\mathsf{p}(z^M)\boldsymbol{v}(z). \tag{3.39}$$

Equation (3.39) completely characterizes the analysis bank and is called the polyphase representation of the analysis bank. The quantity $\boldsymbol{H}_\mathsf{p}(z)$ is referred to as the **analysis polyphase matrix**.

Now, let us consider the synthesis filters of the UMD filter bank. We can express the transfer functions of these filters in polyphase form as

$$G_k(z) = \sum_{p=0}^{M-1} z^{l_p} G_{p,k}(z^M)$$

for $k = 0, 1, \ldots, M-1$. (Note that the same type of polyphase decomposition is used for each of the synthesis filters; that is, $\{l_k\}_{k=0}^{M-1}$ is the same for all of the synthesis filters. The type of decomposition used for the synthesis bank need not be the same as the one used for the analysis bank, however.) In matrix form, the above equation becomes

$$\underbrace{\begin{bmatrix} G_0(z) & G_1(z) & \cdots & G_{M-1}(z) \end{bmatrix}}_{\boldsymbol{g}^T(z)} = \underbrace{\begin{bmatrix} z^{l_0} & z^{l_1} & \cdots & z^{l_{M-1}} \end{bmatrix}}_{\boldsymbol{u}^T(z)} \underbrace{\begin{bmatrix} G_{0,0}(z^M) & G_{0,1}(z^M) & \cdots & G_{0,M-1}(z^M) \\ G_{1,0}(z^M) & G_{1,1}(z^M) & \cdots & G_{1,M-1}(z^M) \\ \vdots & \vdots & \ddots & \vdots \\ G_{M-1,0}(z^M) & G_{M-1,1}(z^M) & \cdots & G_{M-1,M-1}(z^M) \end{bmatrix}}_{\boldsymbol{G}_\mathsf{p}(z^M)}$$

which can be written more concisely as

$$\boldsymbol{g}^T(z) = \boldsymbol{u}^T(z)\boldsymbol{G}_\mathsf{p}(z^M) \tag{3.40}$$

Equation (3.40) completely characterizes the synthesis bank and is called the polyphase representation of the synthesis bank. The quantity $\boldsymbol{G}_\mathsf{p}(z)$ is referred to as the **synthesis polyphase matrix**.

Observe that (3.39) and (3.40) provide an alternative way in which to express the analysis and synthesis banks of the UMD filter bank. Suppose now that we have a UMD filter bank where the analysis and synthesis banks have been represented in this manner. In this case, these equations give us the transformed, but mathematically equivalent, system shown in Figure 3.33. Using the noble identities, however, we can move the analysis polyphase filtering to the right of the downsamplers and the synthesis polyphase filtering to the left of the upsamplers. This gives us the polyphase form of the filter bank shown in Figure 3.34.

To show in more detail the transformation of a filter bank from its canonical to polyphase representation, we consider the case of $M = 2$. In this case the analysis side of the filter bank is transformed as shown in Figure 3.35. Similarly, the synthesis side of the filter bank is transformed as shown in Figure 3.36.

In effect, the polyphase representation reorganizes a filter bank so that it operates on the polyphase components of the input signal. The analysis and synthesis polyphase filtering is performed by M-input M-output networks. On the analysis side of the filter bank, the shift operators and downsamplers form what is referred to as the **forward polyphase transform** (FPT). Similarly, the upsamplers and shift operators form what is called the **inverse polyphase transform** (IPT).

It is important to note that with the polyphase form of a UMD filter bank, analysis filtering is performed after downsampling, and synthesis filtering is performed before upsampling. In other words, all filtering is performed in the downsampled domain (i.e., at the lower sampling rate). This leads to improved computational efficiency over a UMD filter bank implemented in its canonical form. Consequently, most practical implementations of UMD filter banks make use of the polyphase form.

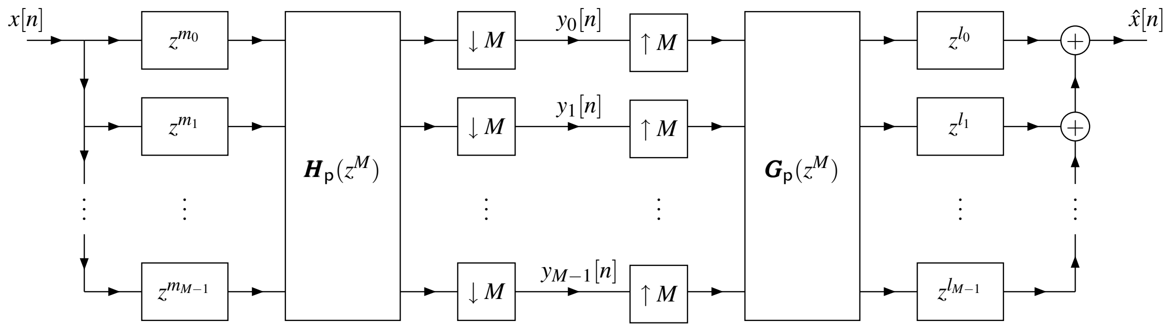


Figure 3.33: Polyphase representation of an M-channel UMD filter bank before simplification with the noble identities.

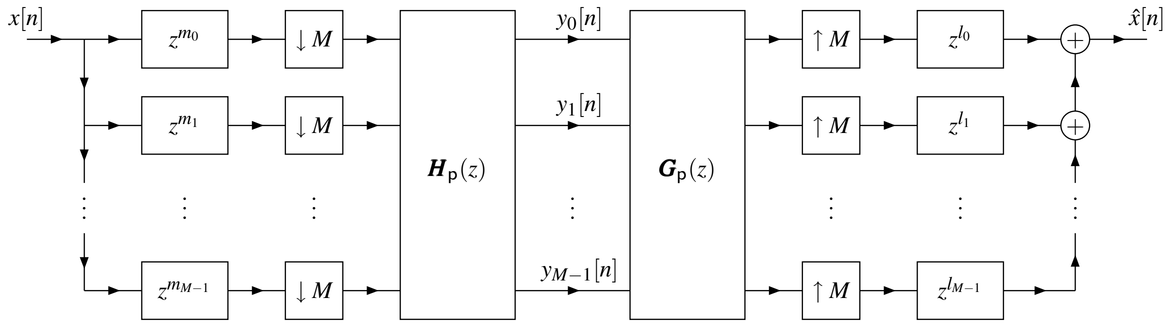


Figure 3.34: Polyphase representation of an M-channel UMD filter bank.

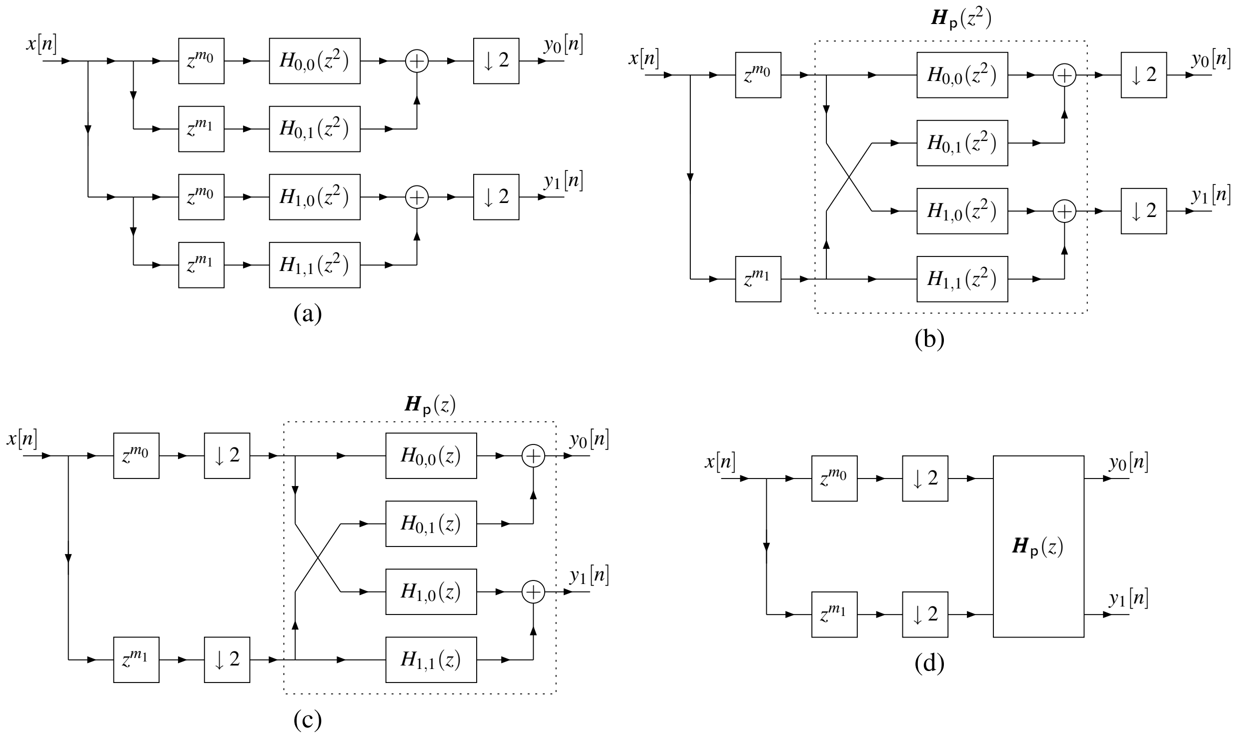


Figure 3.35: Constructing the polyphase representation of the analysis side of a two-channel UMD filter bank. The system obtained after (a) implementing the analysis filters in polyphase form, (b) combining equivalent shifts, (c) using the noble identities to move the downsamplers across the filters, (d) replacing the individual polyphase filters by a network with transfer matrix $\boldsymbol{H}_\mathsf{p}(z)$.

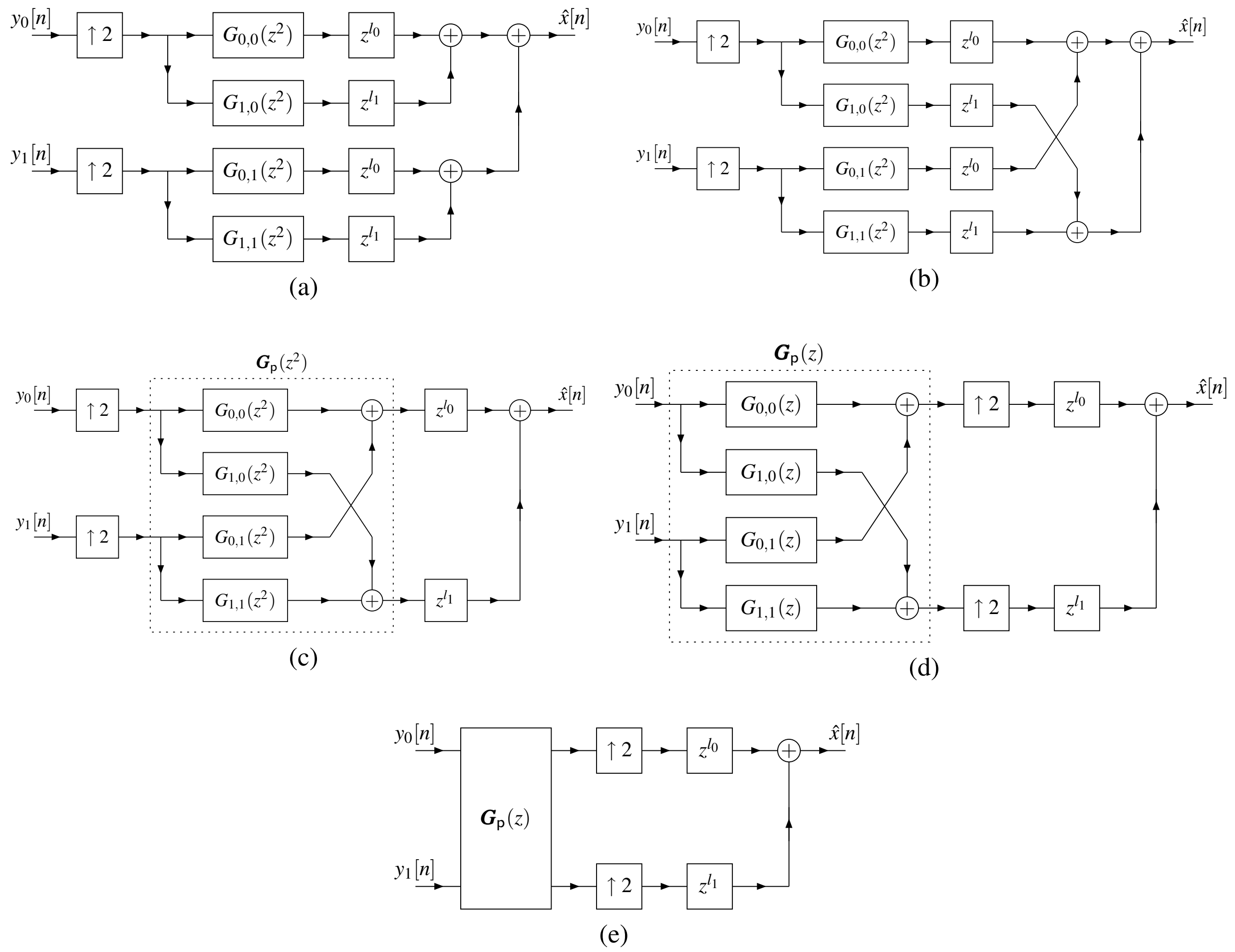


Figure 3.36: Constructing the polyphase representation of the synthesis side of a two-channel UMD filter bank. The system obtained after (a) implementing the synthesis filters in polyphase form, (b) changing the order of addition, (c) combining common shifts, (d) using the noble identities to move the upsamplers across the filters, (e) replacing the individual polyphase filters with a network with transfer matrix $\boldsymbol{G}_\mathsf{p}(z)$.

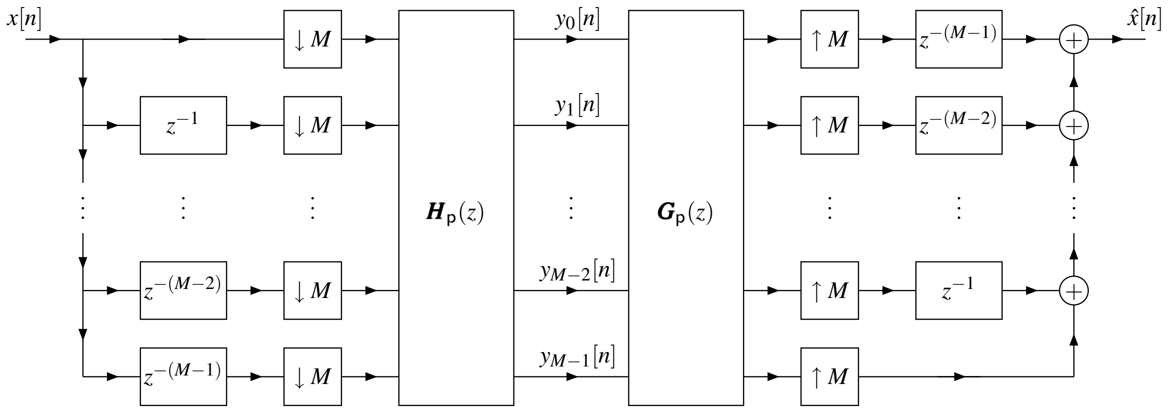


Figure 3.37: The $(1,2)$-type polyphase representation of an M-channel UMD filter bank.

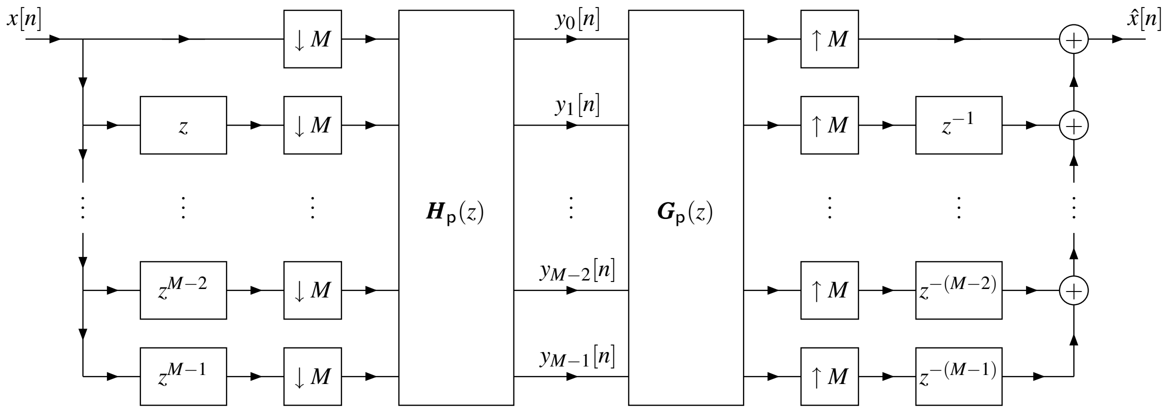


Figure 3.38: The $(3,1)$-type polyphase representation of an M-channel UMD filter bank.

From our previous experience with polyphase representations of filters, we know that some freedom exists in the choice of the parameters $\{m_k\}$ and $\{l_k\}$. For this reason, many variations on the polyphase representation of a UMD filter bank are possible. In practice, however, two particular variants are most commonly used. Since no standard naming convention exists in the literature to distinguish between these variants, the author introduces his own terminology in what follows. The first variant represents the filters of the analysis bank in their type-1 polyphase form and the filters of the synthesis bank in their type-2 polyphase form. This variant will be referred to as a type-$(1,2)$ polyphase representation. The corresponding realization is shown in Figure 3.37. The second variant employs type-3 and type-1 polyphase decompositions for the analysis and synthesis banks, respectively. This variant will be referred to as a type-$(3,1)$ polyphase representation. The corresponding realization is shown in Figure 3.38.

In the remainder of this book, we will focus primarily on the $(3,1)$-type polyphase representation. This particular representation is often the most convenient to use in the context of wavelets. One must be careful when interpreting results involving polyphase representation from different authors, since different polyphase representations might be used. For example, [33] focuses primarily on the $(3,1)$-type representation, while [31] uses mostly the $(1,2)$-type representation.

Example 3.18. Suppose that we have a two-channel UMD filter bank with analysis filter transfer functions $\{H_k(z)\}$

and synthesis filter transfer functions $\{G_k(z)\}$, where

$$\begin{gathered}H_0(z) = \tfrac{1}{2}z + \tfrac{1}{2}, \\ H_1(z) = -\tfrac{1}{8}z^3 - \tfrac{1}{8}z^2 + z - 1 + \tfrac{1}{8}z^{-1} + \tfrac{1}{8}z^{-2}, \\ G_0(z) = -\tfrac{1}{8}z^2 + \tfrac{1}{8}z + 1 + z^{-1} + \tfrac{1}{8}z^{-2} - \tfrac{1}{8}z^{-3}, \quad \text{and} \\ G_1(z) = -\tfrac{1}{2} + \tfrac{1}{2}z^{-1}.\end{gathered}$$

Find the $(3,1)$-type polyphase representation of the filter bank.

Solution. First, we find the type-3 polyphase representation of each of the analysis filters. We can write

$$\begin{gathered}H_0(z) = \left(\tfrac{1}{2}\right) + z\left(\tfrac{1}{2}\right) \quad \text{and} \\ H_1(z) = \left(-\tfrac{1}{8}z^2 - 1 + \tfrac{1}{8}z^{-2}\right) + z\left(-\tfrac{1}{8}z^2 + 1 + \tfrac{1}{8}z^{-2}\right).\end{gathered}$$

Thus, the analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \tfrac{1}{2} & \tfrac{1}{2} \\ -\tfrac{1}{8}z - 1 + \tfrac{1}{8}z^{-1} & -\tfrac{1}{8}z + 1 + \tfrac{1}{8}z^{-1} \end{bmatrix}.$$

Now, we find the type-1 polyphase representation of each of the synthesis filters. We can write

$$\begin{gathered}G_0(z) = \left(-\tfrac{1}{8}z^2 + 1 + \tfrac{1}{8}z^{-2}\right) + z^{-1}\left(\tfrac{1}{8}z^2 + 1 - \tfrac{1}{8}z^{-2}\right) \quad \text{and} \\ G_1(z) = \left(-\tfrac{1}{2}\right) + z^{-1}\left(\tfrac{1}{2}\right).\end{gathered}$$

Thus, the synthesis polyphase matrix $\boldsymbol{G}_\mathsf{p}(z)$ is given by

$$\boldsymbol{G}_\mathsf{p}(z) = \begin{bmatrix} -\tfrac{1}{8}z + 1 + \tfrac{1}{8}z^{-1} & -\tfrac{1}{2} \\ \tfrac{1}{8}z + 1 - \tfrac{1}{8}z^{-1} & \tfrac{1}{2} \end{bmatrix}.$$

□

Example 3.19. Suppose that we have a two-channel UMD filter bank with analysis filter transfer functions $\{H_k(z)\}$ and synthesis filter transfer functions $\{G_k(z)\}$, where

$$\begin{gathered}H_0(z) = -\tfrac{1}{8}z^2 + \tfrac{1}{4}z + \tfrac{3}{4} + \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2}, \\ H_1(z) = -\tfrac{1}{2}z^2 + z - \tfrac{1}{2}, \\ G_0(z) = \tfrac{1}{2}z + 1 + \tfrac{1}{2}z^{-1}, \text{ and} \\ G_1(z) = -\tfrac{1}{8}z - \tfrac{1}{4} + \tfrac{3}{4}z^{-1} - \tfrac{1}{4}z^{-2} - \tfrac{1}{8}z^{-3}.\end{gathered}$$

Find the $(3,1)$-type polyphase representation of the filter bank.

Solution. First, we find the type-3 polyphase representation of each of the analysis filters. We can express $H_0(z)$ and $H_1(z)$ as follows:

$$\begin{aligned}H_0(z) &= H_{0,0}(z^2) + zH_{0,1}(z^2) \\ &= \left[-\tfrac{1}{8}z^2 + \tfrac{3}{4} - \tfrac{1}{8}z^{-2}\right] + z\left[\tfrac{1}{4} + \tfrac{1}{4}z^{-2}\right] \text{ and}\end{aligned}$$

$$\begin{aligned}H_1(z) &= H_{1,0}(z^2) + zH_{1,1}(z^2) \\ &= \left[-\tfrac{1}{2}z^2 - \tfrac{1}{2}\right] + z[1].\end{aligned}$$

Thus, the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ is given by

$$\boldsymbol{H}_{\mathsf{p}}(z) = \begin{bmatrix} -\frac{1}{8}z + \frac{3}{4} - \frac{1}{8}z^{-1} & \frac{1}{4} + \frac{1}{4}z^{-1} \\ -\frac{1}{2}z - \frac{1}{2} & 1 \end{bmatrix}.$$

Now, we find the type-1 polyphase representation of each of the synthesis filters. We can express $G_0(z)$ and $G_1(z)$ as follows:

$$\begin{aligned} G_0(z) &= G_{0,0}(z^2) + z^{-1}G_{1,0}(z^2) \\ &= [1] + z^{-1}\left[\tfrac{1}{2}z^2 + \tfrac{1}{2}\right] \quad \text{and} \end{aligned}$$

$$\begin{aligned} G_1(z) &= G_{0,1}(z^2) + z^{-1}G_{1,1}(z^2) \\ &= \left[-\tfrac{1}{4} - \tfrac{1}{4}z^{-2}\right] + z^{-1}\left[-\tfrac{1}{8}z^2 + \tfrac{3}{4} - \tfrac{1}{8}z^{-2}\right]. \end{aligned}$$

Thus, the synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ is given by

$$\boldsymbol{G}_{\mathsf{p}}(z) = \begin{bmatrix} 1 & -\frac{1}{4} - \frac{1}{4}z^{-1} \\ \frac{1}{2}z + \frac{1}{2} & -\frac{1}{8}z + \frac{3}{4} - \frac{1}{8}z^{-1} \end{bmatrix}.$$

□

3.3.8 Relationship Between Modulation and Polyphase Matrices

Earlier, we introduced the modulation and polyphase representations of UMD filter banks. In the modulation case, the system is completely characterized by its analysis and synthesis modulation matrices. In the polyphase case, the system is completely described by its analysis and synthesis polyphase matrices. Since both pairs of matrices contain essentially the same information, one might wonder if some simple relationship exists between these matrices. This is, in fact, the case, as demonstrated by the theorem below.

Theorem 3.9 (Relationship between modulation and polyphase matrices)**.** *Suppose that an M-channel UMD filter bank has analysis modulation matrix $\boldsymbol{H}_{\mathsf{m}}(z)$, synthesis modulation matrix $\boldsymbol{G}_{\mathsf{m}}(z)$, analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$, and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$. Then, these matrices are related as*

$$\boldsymbol{H}_{\mathsf{m}}(z) = \begin{cases} \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z^{-1})\boldsymbol{W}^\dagger & \textit{for } (1,\cdot) \textit{ type} \\ z^{-(M-1)}\boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z)\boldsymbol{J}\boldsymbol{W}^\dagger & \textit{for } (2,\cdot) \textit{ type} \\ \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z)\boldsymbol{W} & \textit{for } (3,\cdot) \textit{ type} \\ z^{M-1}\boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z^{-1})\boldsymbol{J}\boldsymbol{W} & \textit{for } (4,\cdot) \textit{ type} \end{cases} \tag{3.41}$$

$$\boldsymbol{G}_{\mathsf{m}}(z) = \begin{cases} \boldsymbol{W}^\dagger\boldsymbol{D}(z^{-1})\boldsymbol{G}_{\mathsf{p}}(z^M) & \textit{for } (\cdot,1) \textit{ type} \\ z^{-(M-1)}\boldsymbol{W}^\dagger\boldsymbol{J}\boldsymbol{D}(z)\boldsymbol{G}_{\mathsf{p}}(z^M) & \textit{for } (\cdot,2) \textit{ type} \\ \boldsymbol{W}\boldsymbol{D}(z)\boldsymbol{G}_{\mathsf{p}}(z^M) & \textit{for } (\cdot,3) \textit{ type} \\ z^{M-1}\boldsymbol{W}\boldsymbol{J}\boldsymbol{D}(z^{-1})\boldsymbol{G}_{\mathsf{p}}(z^M) & \textit{for } (\cdot,4) \textit{ type} \end{cases} \tag{3.42}$$

where $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix}1 & z & \cdots & z^{M-1}\end{bmatrix}$ and $\boldsymbol{W} = \boldsymbol{W}_M$. (Note that $\boldsymbol{J}$ denotes the anti-identity matrix and $\boldsymbol{W}_M$ denotes the $M \times M$ DFT matrix as defined by (2.13) *on page 68.)*

Proof. Denote the analysis filter transfer functions as $H_k(z)$ and define

$$\boldsymbol{h}(z) = \begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \end{bmatrix}^T.$$

From the definition of the modulation matrix, we can write

$$\boldsymbol{H}_{\mathsf{m}}(z) = \begin{bmatrix} \boldsymbol{h}(z) & \boldsymbol{h}(zW) & \cdots & \boldsymbol{h}(zW^{M-1}) \end{bmatrix}. \tag{3.43}$$

From the definition of the polyphase matrix, we have

$$\boldsymbol{h}(z) = \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{v}(z)$$

where $\boldsymbol{v}(z)$ is as defined earlier (i.e., $\boldsymbol{v}(z) = \begin{bmatrix} z^{m_0} & z^{m_1} & \cdots & z^{m_{M-1}} \end{bmatrix}^T$). Substituting this expression for $\boldsymbol{h}(z)$ into (3.43), we obtain

$$\begin{aligned} \boldsymbol{H}_{\mathsf{m}}(z) &= \begin{bmatrix} \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{v}(z) & \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{v}(zW) & \cdots & \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{v}(zW^{M-1}) \end{bmatrix} \\ &= \boldsymbol{H}_{\mathsf{p}}(z^M) \begin{bmatrix} \boldsymbol{v}(z) & \boldsymbol{v}(zW) & \cdots & \boldsymbol{v}(zW^{M-1}) \end{bmatrix}. \end{aligned} \tag{3.44}$$

Now, we consider an expression of the form $\boldsymbol{v}(zW^k)$, as such appears in the preceding equation.

Consider the case of a type $(1,\cdot)$ polyphase representation. In this case, $\boldsymbol{v}(z) = \begin{bmatrix} 1 & z^{-1} & \cdots & z^{-(M-1)} \end{bmatrix}^T$. So, we have

$$\boldsymbol{v}(zW^k) = \begin{bmatrix} 1 \\ (zW^k)^{-1} \\ \vdots \\ (zW^k)^{-(M-1)} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & z^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z^{-(M-1)} \end{bmatrix}}_{\boldsymbol{D}(z^{-1})} \underbrace{\begin{bmatrix} 1 \\ W^{-k} \\ \vdots \\ W^{-(M-1)k} \end{bmatrix}}_{\boldsymbol{a}_k}.$$

Thus, by substituting the above expression for $\boldsymbol{v}(zW^k)$ into (3.44), we obtain

$$\begin{aligned} \boldsymbol{H}_{\mathsf{m}}(z) &= \boldsymbol{H}_{\mathsf{p}}(z^M) \begin{bmatrix} \boldsymbol{D}(z^{-1})\boldsymbol{a}_0 & \boldsymbol{D}(z^{-1})\boldsymbol{a}_1 & \cdots & \boldsymbol{D}(z^{-1})\boldsymbol{a}_{M-1} \end{bmatrix} \\ &= \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z^{-1}) \begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_{M-1} \end{bmatrix}. \end{aligned}$$

Now, we observe that the rightmost matrix in the preceding equation can be expressed as

$$\begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_{M-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W^{-1} & \cdots & W^{-(M-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{-(M-1)} & \cdots & W^{-(M-1)^2} \end{bmatrix} = \boldsymbol{W}^{\dagger}.$$

Thus, we have

$$\boldsymbol{H}_{\mathsf{m}}(z) = \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z^{-1})\boldsymbol{W}^{\dagger}.$$

This proves that (3.41) holds in the type $(1,\cdot)$ case.

Now, consider the case of a $(3,\cdot)$ type polyphase representation. In this case, $\boldsymbol{v}(z) = \begin{bmatrix} 1 & z & \cdots & z^{M-1} \end{bmatrix}^T$. So, we have

$$\boldsymbol{v}(zW^k) = \begin{bmatrix} 1 \\ zW^k \\ \cdots \\ (zW^k)^{M-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & z & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z^{M-1} \end{bmatrix}}_{\boldsymbol{D}(z)} \underbrace{\begin{bmatrix} 1 \\ W^{k} \\ \vdots \\ W^{(M-1)k} \end{bmatrix}}_{\boldsymbol{a}_k}.$$

Proceeding in a similar fashion as above, we can write

$$\begin{aligned} \boldsymbol{H}_{\mathsf{m}}(z) &= \boldsymbol{H}_{\mathsf{p}}(z^M) \begin{bmatrix} \boldsymbol{D}(z)\boldsymbol{a}_0 & \boldsymbol{D}(z)\boldsymbol{a}_1 & \cdots & \boldsymbol{D}(z)\boldsymbol{a}_{M-1} \end{bmatrix} \\ &= \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z) \begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_{M-1} \end{bmatrix}. \end{aligned}$$

Now, we observe

$$\begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_{M-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W & \cdots & W^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{M-1} & \cdots & W^{(M-1)^2} \end{bmatrix} = \boldsymbol{W}.$$

So, we have

$$\boldsymbol{H}_{\mathsf{m}}(z) = \boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z)\boldsymbol{W}.$$

This proves that (3.41) holds in the type $(3,\cdot)$ case.

The proof of (3.41) is similar for the remaining types of polyphase decompositions. The result (3.42) follows trivially from (3.41) by observing that we are required to prove the same assertions as in (3.41) with $\boldsymbol{H}_{\mathsf{m}}(z)$ and $\boldsymbol{H}_{\mathsf{p}}(z)$ replaced by $\boldsymbol{G}_{\mathsf{m}}^T(z)$ and $\boldsymbol{G}_{\mathsf{p}}^T(z)$, respectively. □

3.3.9 Modulation-Domain Conditions for Alias-Free and PR Systems

Since a UMD filter bank is completely characterized by its modulation matrices, it must be possible to use these matrices in order to determine whether the system has the alias-free or PR properties. Below, we provide necessary and sufficient conditions for an alias-free system formulated in terms of the modulation matrices.

Theorem 3.10 (Necessary and sufficient conditions for alias cancellation)**.** *Suppose that an M-channel UMD filter bank has the analysis and synthesis modulation matrices $\boldsymbol{H}_{\mathsf{m}}(z)$ and $\boldsymbol{G}_{\mathsf{m}}(z)$, respectively. Then, the system is alias free if and only if the product $\boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{H}_{\mathsf{m}}(z)$ is of the form*

$$\boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{H}_{\mathsf{m}}(z) = M\boldsymbol{D}(z)$$

where $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix} T(z) & T(zW) & \cdots & T(zW^{M-1}) \end{bmatrix}$ and $W = W_M$. If the system is alias free, then it is LTI with the transfer function $T(z)$.

Proof. Taking the transpose of both sides of equation (3.38), we obtain

$$\boldsymbol{A}^T(z) = \tfrac{1}{M}\boldsymbol{g}^T(z)\boldsymbol{H}_{\mathsf{m}}(z).$$

Thus, we have

$$\boldsymbol{A}^T(zW^k) = \tfrac{1}{M}\boldsymbol{g}^T(zW^k)\boldsymbol{H}_{\mathsf{m}}(zW^k). \tag{3.45}$$

Now, let us further examine the factor $\boldsymbol{H}_{\mathsf{m}}(zW^k)$ in the preceding equation. First, we consider $\boldsymbol{H}_{\mathsf{m}}(zW)$. From the definition of the modulation matrix, we can write

$$\boldsymbol{H}_{\mathsf{m}}(zW) = \begin{bmatrix} H_0(zW) & H_0(zW^2) & \cdots & H_0(zW^{M-1}) & H_0(z) \\ H_1(zW) & H_1(zW^2) & \cdots & H_1(zW^{M-1}) & H_1(z) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ H_{M-1}(zW) & H_{M-1}(zW^2) & \cdots & H_{M-1}(zW^{M-1}) & H_{M-1}(z) \end{bmatrix}.$$

Thus, $\boldsymbol{H}_{\mathsf{m}}(zW)$ is formed by circularly shifting $\boldsymbol{H}_{\mathsf{m}}(z)$ left by one column, which we can express as

$$\boldsymbol{H}_{\mathsf{m}}(zW) = \boldsymbol{H}_{\mathsf{m}}(z)\underbrace{\begin{bmatrix} \boldsymbol{0} & 1 \\ \boldsymbol{I}_{M-1} & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{C}}.$$

By recursively applying this identity k times, we obtain

$$\boldsymbol{H}_{\mathsf{m}}(zW^k) = \boldsymbol{H}_{\mathsf{m}}(z)\boldsymbol{C}^k.$$

Substituting this expression for $\boldsymbol{H}_\mathsf{m}(zW^k)$ into (3.45) yields

$$\boldsymbol{A}^T(zW^k) = \tfrac{1}{M}\boldsymbol{g}^T(zW^k)\boldsymbol{H}_\mathsf{m}(z)\boldsymbol{C}^k.$$

Since $\boldsymbol{C}$ is obviously invertible, we can rewrite the above equation as

$$M\boldsymbol{A}^T(zW^k)\boldsymbol{C}^{-k} = \boldsymbol{g}^T(zW^k)\boldsymbol{H}_\mathsf{m}(z).$$

Now, the set of equations obtained with $k = 0, 1, \ldots, M-1$ can be expressed in matrix form as

$$\begin{bmatrix} \boldsymbol{g}^T(z)\boldsymbol{H}_\mathsf{m}(z) \\ \boldsymbol{g}^T(zW)\boldsymbol{H}_\mathsf{m}(z) \\ \vdots \\ \boldsymbol{g}^T(zW^{M-1})\boldsymbol{H}_\mathsf{m}(z) \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{g}^T(z) \\ \boldsymbol{g}^T(zW) \\ \vdots \\ \boldsymbol{g}^T(zW^{M-1}) \end{bmatrix}}_{\boldsymbol{G}_\mathsf{m}(z)} \boldsymbol{H}_\mathsf{m}(z) = M \begin{bmatrix} \boldsymbol{A}^T(z) \\ \boldsymbol{A}^T(zW)\boldsymbol{C}^{-1} \\ \vdots \\ \boldsymbol{A}^T(zW^{M-1})\boldsymbol{C}^{-(M-1)} \end{bmatrix}.$$

Observe that $\boldsymbol{A}^T(zW^k)\boldsymbol{C}^{-k}$ is simply $\boldsymbol{A}^T(zW^k)$ circularly shifted right by k columns. So, we have

$$\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) = M \begin{bmatrix} A_0(z) & A_1(z) & \cdots & A_{M-1}(z) \\ A_{M-1}(zW) & A_0(zW) & \cdots & A_{M-2}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ A_1(zW^{M-1}) & A_2(zW^{M-1}) & \cdots & A_0(zW^{M-1}) \end{bmatrix}.$$

From (3.36), we know that the system is alias free if and only if $A_k(z) \equiv 0$ for $k = 1, \ldots, M-1$. In this case, we have

$$\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) = M\boldsymbol{D}(z)$$

where $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix} A_0(z) & A_0(zW) & \cdots & A_0(zW^{M-1}) \end{bmatrix}$. Thus, the product $\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z)$ is of the form stated in the theorem. Furthermore, the system must be LTI with the transfer function $T(z) = A_0(z)$. □

Now, we consider the PR property. Below, we give necessary and sufficient conditions for the PR property formulated in terms of the modulation matrices.

Theorem 3.11 (Necessary and sufficient conditions for PR)**.** *Suppose that an M-channel UMD filter bank has the analysis and synthesis modulation matrices $\boldsymbol{H}_\mathsf{m}(z)$ and $\boldsymbol{G}_\mathsf{m}(z)$, respectively. Then, the system has the PR property if and only if the product $\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z)$ is of the form*

$$\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) = Mz^{-n_0}\boldsymbol{D}(z)$$

where $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix} 1 & W^{-n_0} & W^{-2n_0} & \cdots & W^{-(M-1)n_0} \end{bmatrix}$, n_0 is the reconstruction shift, and $W = W_M$. This implies that a system has the shift-free PR property if and only if

$$\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) = M\boldsymbol{I}.$$

Proof. The proof of this theorem is trivial and follows immediately from Theorem 3.10 for the alias-free case with transfer function $T(z) = z^{-n_0}$. □

Corollary 3.1. *Any M-channel PR UMD filter bank with analysis and synthesis modulation matrices, $\boldsymbol{H}_\mathsf{m}(z)$ and $\boldsymbol{G}_\mathsf{m}(z)$, respectively, must be such that the product $\boldsymbol{P}(z) = \boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z)$ has a determinant of the form*

$$\det\boldsymbol{P}(z) = M^M W^{-Mn_0(M-1)/2} z^{-Mn_0},$$

where $W = W_M$ and n_0 is the reconstruction shift of the filter bank. Moreover, if the analysis and synthesis filters are of the FIR type, this condition implies that the modulation matrices must have determinants of the form

$$\det\boldsymbol{H}_\mathsf{m}(z) = a_0 z^{-L_0} \quad \text{and} \quad \det\boldsymbol{G}_\mathsf{m}(z) = a_1 z^{-L_1},$$

where $L_0, L_1 \in \mathbb{Z}$ and $a_0, a_1 \in \mathbb{C} \setminus \{0\}$. That is, the determinants of the modulation matrices must be monomials.

Proof. We observe that the matrix $\boldsymbol{P}(z)$ must have the form specified in Theorem 3.11. Namely, $\boldsymbol{P}(z)$ is of the form

$$\boldsymbol{P}(z) = Mz^{-n_0}\boldsymbol{D}(z),$$

where $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix}1 & W^{-n_0} & W^{-2n_0} & \cdots & W^{-(M-1)n_0}\end{bmatrix}$. We have

$$\begin{aligned}\det\boldsymbol{P}(z) &= \det(Mz^{-n_0}\boldsymbol{I})\det\boldsymbol{D}(z)\\ &= M^M W^{-Mn_0(M-1)/2}z^{-Mn_0}.\end{aligned}$$

If the analysis and synthesis filters are of the FIR type, then $\det\boldsymbol{H}_\mathsf{m}(z)$ and $\det\boldsymbol{G}_\mathsf{m}(z)$ must be Laurent polynomials. The only way that the product $\det\boldsymbol{H}_\mathsf{m}(z)\det\boldsymbol{G}_\mathsf{m}(z)$ can be a monomial is if $\det\boldsymbol{H}_\mathsf{m}(z)$ and $\det\boldsymbol{G}_\mathsf{m}(z)$ are each monomials. □

Example 3.20. Suppose that we have a two-channel UMD filter bank with analysis filters $\{H_k(z)\}_{k=0}^{1}$ and synthesis filters $\{G_k(z)\}_{k=0}^{1}$, respectively, where

$$\begin{aligned}H_0(z) &= \tfrac{1}{2}z+\tfrac{1}{2},\\ H_1(z) &= z-1,\\ G_0(z) &= 1+z^{-1}, \text{ and}\\ G_1(z) &= -\tfrac{1}{2}+\tfrac{1}{2}z^{-1}.\end{aligned}$$

Use the modulation matrices to determine if the filter bank has the alias free and PR properties.

Solution. The analysis and synthesis modulations matrices are given by

$$\boldsymbol{H}_\mathsf{m}(z) = \begin{bmatrix}\frac{1}{2}z+\frac{1}{2} & -\frac{1}{2}z+\frac{1}{2}\\ z-1 & -z-1\end{bmatrix} \quad\text{and}\quad \boldsymbol{G}_\mathsf{m}(z) = \begin{bmatrix}1+z^{-1} & -\frac{1}{2}+\frac{1}{2}z^{-1}\\ 1-z^{-1} & -\frac{1}{2}-\frac{1}{2}z^{-1}\end{bmatrix}.$$

We compute the product of the modulation matrices as

$$\begin{aligned}\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) &= \begin{bmatrix}1+z^{-1} & -\frac{1}{2}+\frac{1}{2}z^{-1}\\ 1-z^{-1} & -\frac{1}{2}-\frac{1}{2}z^{-1}\end{bmatrix}\begin{bmatrix}\frac{1}{2}z+\frac{1}{2} & -\frac{1}{2}z+\frac{1}{2}\\ z-1 & -z-1\end{bmatrix}\\ &= \begin{bmatrix}(1+z^{-1})(\frac{1}{2}z+\frac{1}{2})+(-\frac{1}{2}+\frac{1}{2}z^{-1})(z-1) & (1+z^{-1})(-\frac{1}{2}z+\frac{1}{2})+(-\frac{1}{2}+\frac{1}{2}z^{-1})(-z-1)\\ (1-z^{-1})(\frac{1}{2}z+\frac{1}{2})+(-\frac{1}{2}-\frac{1}{2}z^{-1})(z-1) & (1-z^{-1})(-\frac{1}{2}z+\frac{1}{2})+(-\frac{1}{2}+\frac{1}{2}z^{-1})(-z-1)\end{bmatrix}\\ &= \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}\\ &= 2\boldsymbol{I}.\end{aligned}$$

Therefore, by the earlier theorems, the system is alias free and has the shift-free PR property. □

3.3.10 Polyphase-Domain Conditions for Alias-Free and PR Systems

Since the polyphase matrices of a UMD filter bank completely characterize the system, it must be possible to use these matrices in order to determine whether the alias-free or PR properties hold. First, we consider the alias-free property. Before proceeding further, we need to introduce the notion of a pseudocirculant matrix.

Definition 3.4 (Pseudocirculant matrices)**.** A matrix $\boldsymbol{P}(z)$ that is formed by taking a circulant matrix and multiplying each element below its main diagonal by z is said to be an **A-type pseudocirculant matrix**. A matrix $\boldsymbol{P}(z)$ that is formed by taking a circulant matrix and multiplying each element below its main diagonal by z^{-1} is said to be a **B-type pseudocirculant matrix**. For example, the following two matrices are, respectively, A- and B-type pseudocirculant:

$$\begin{bmatrix}P_0(z) & P_1(z) & P_2(z)\\ zP_2(z) & P_0(z) & P_1(z)\\ zP_1(z) & zP_2(z) & P_0(z)\end{bmatrix} \quad\text{and}\quad \begin{bmatrix}P_0(z) & P_1(z) & P_2(z)\\ z^{-1}P_2(z) & P_0(z) & P_1(z)\\ z^{-1}P_1(z) & z^{-1}P_2(z) & P_0(z)\end{bmatrix}.$$

Each successive row in a pseudocirculant matrix is the (right) circular shift of the preceding row with the circulated element multiplied by z or z^{-1} for an A- or B-type pseudocirculant, respectively. Equivalently, each successive column in a pseudocirculant matrix is the downward circular shift of the preceding column with the circulated element multiplied by z^{-1} or z for an A- or B-type pseudocirculant, respectively. Clearly, a pseudocirculant matrix is completely characterized by the elements of its top row. In particular, an $M \times M$ pseudocirculant matrix $\boldsymbol{P}(z)$ can be expressed as

$$\boldsymbol{P}(z) = \begin{cases} \sum\limits_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^k & \text{for A type} \\ \sum\limits_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^k & \text{for B type,} \end{cases} \tag{3.47}$$

where $P_k(z)$ denotes the kth element in the top row of $\boldsymbol{P}(z)$.

Example 3.21. Let $\boldsymbol{P}(z)$ be a 3×3 A-type pseudocirculant matrix with the elements of its top row given by $P_0(z), P_1(z), P_2(z)$. Confirm the validity of (3.47). That is, show that

$$\boldsymbol{P}(z) = \sum_{k=0}^{2} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_2 \\ z & \mathbf{0} \end{bmatrix}^k .$$

Solution. We compute $\left[\begin{smallmatrix} \mathbf{0} & \boldsymbol{I}_2 \\ z & \mathbf{0} \end{smallmatrix}\right]^k$ for $k = 1,2$ to obtain

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{I}_2 \\ z & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ z & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_2 \\ z & \mathbf{0} \end{bmatrix}^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ z & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ z & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ z & 0 & 0 \\ 0 & z & 0 \end{bmatrix}.$$

Using the preceding results, we can write

$$\begin{aligned} \sum_{k=0}^{2} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_2 \\ z & \mathbf{0} \end{bmatrix}^k &= \begin{bmatrix} P_0(z) & 0 & 0 \\ 0 & P_0(z) & 0 \\ 0 & 0 & P_0(z) \end{bmatrix} + \begin{bmatrix} 0 & P_1(z) & 0 \\ 0 & 0 & P_1(z) \\ zP_1(z) & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & P_2(z) \\ zP_z(z) & 0 & 0 \\ 0 & zP_2(z) & 0 \end{bmatrix} \\ &= \begin{bmatrix} P_0(z) & P_1(z) & P_2(z) \\ zP_2(z) & P_0(z) & P_1(z) \\ zP_1(z) & zP_2(z) & P_0(z) \end{bmatrix}. \end{aligned}$$

Thus, the stated relationship holds. □

Having defined the notion of pseudocirculant matrices, we are now ready to state a necessary and sufficient condition for a UMD filter bank to be alias free.

Theorem 3.12 (Necessary and sufficient conditions for alias cancellation)**.** *An M-channel UMD filter bank in either $(1,2)$ or $(3,1)$ polyphase form with analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ is alias free if and only if the product $\boldsymbol{P}(z) = \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z)$ is such that*

$$\boldsymbol{P}(z) \text{ is } \begin{cases} \textit{B-type pseudocirculant} & \textit{for } (1,2) \textit{ type} \\ \textit{A-type pseudocirculant} & \textit{for } (3,1) \textit{ type.} \end{cases} \tag{3.48}$$

If the system is alias free, it has the transfer function

$$T(z) = \begin{cases} z^{-(M-1)} \sum_{k=0}^{M-1} z^{-k} P_k(z^M) & \textit{for } (1,2) \textit{ type} \\ \sum_{k=0}^{M-1} z^k P_k(z^M) & \textit{for } (3,1) \textit{ type} \end{cases} \tag{3.49}$$

where $P_0(z), P_1(z), \ldots, P_{M-1}(z)$ are the elements of the top row of $\boldsymbol{P}(z)$.

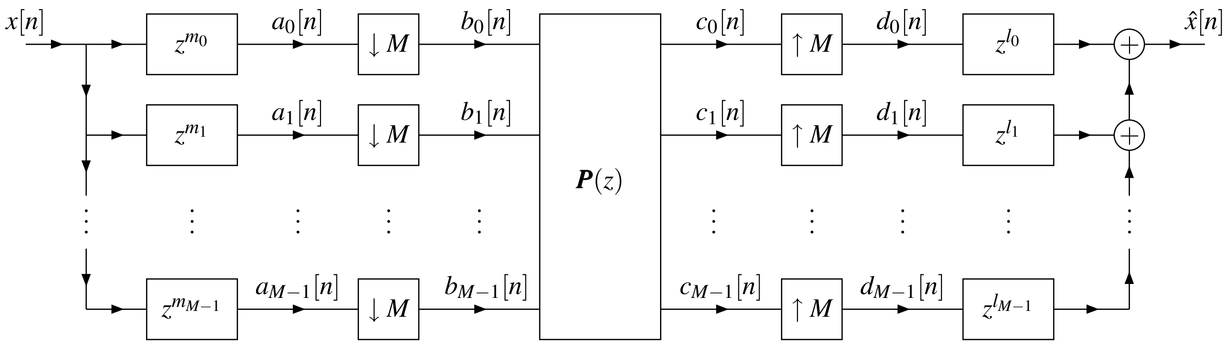


Figure 3.39: Polyphase representation of a UMD filter bank with the analysis and synthesis polyphase filtering combined.

Proof. To begin, we redraw the UMD filter bank as shown in Figure 3.39.

$(1,2)$-TYPE POLYPHASE. First, we consider the case of a $(1,2)$-type polyphase decomposition. In this case, we have $m_k = -k$ and $l_k = k-(M-1)$ for $k = 0,1,\ldots,M-1$. From the diagram, we find equations relating the various signals. The output of the kth shift unit is given by

$$A_k(z) = z^{-k}X(z).$$

The output of the kth downsampler is given by

$$B_k(z) = \tfrac{1}{M}\sum_{l=0}^{M-1} A_k(z^{1/M}W^l).$$

The input of the kth upsampler is given by

$$C_k(z) = \sum_{q=0}^{M-1} P_{k,q}(z)B_q(z).$$

The output of the kth upsampler is given by

$$D_k(z) = C_k(z^M).$$

The output of the system is given by

$$\hat{X}(z) = \sum_{s=0}^{M-1} z^{-(M-1-s)}D_s(z).$$

Combining the preceding equations, we have

$$\begin{aligned}
\hat{X}(z) &= \sum_{s=0}^{M-1} z^{-(M-1-s)} D_s(z) \\
&= \sum_{s=0}^{M-1} z^{-(M-1-s)} C_s(z^M) \\
&= \sum_{s=0}^{M-1} z^{-(M-1-s)} \sum_{q=0}^{M-1} P_{s,q}(z^M) B_q(z^M) \\
&= \sum_{s=0}^{M-1} z^{-(M-1-s)} \sum_{q=0}^{M-1} P_{s,q}(z^M) \left[\tfrac{1}{M} \sum_{l=0}^{M-1} A_q(zW^l) \right] \\
&= \tfrac{1}{M} \sum_{s=0}^{M-1} z^{-(M-1-s)} \sum_{q=0}^{M-1} P_{s,q}(z^M) \sum_{l=0}^{M-1} A_q(zW^l) \\
&= \tfrac{1}{M} \sum_{s=0}^{M-1} z^{-(M-1-s)} \sum_{q=0}^{M-1} P_{s,q}(z^M) \sum_{l=0}^{M-1} (zW^l)^{-q} X(zW^l) \\
&= \tfrac{1}{M} \sum_{s=0}^{M-1} z^{-(M-1-s)} \sum_{q=0}^{M-1} P_{s,q}(z^M) \sum_{l=0}^{M-1} z^{-q} W^{-lq} X(zW^l) \\
&= \tfrac{1}{M} \sum_{l=0}^{M-1} \sum_{q=0}^{M-1} \sum_{s=0}^{M-1} z^{-(M-1-s)} P_{s,q}(z^M) z^{-q} W^{-lq} X(zW^l) \\
&= \tfrac{1}{M} \sum_{l=0}^{M-1} X(zW^l) \sum_{q=0}^{M-1} W^{-lq} \sum_{s=0}^{M-1} z^{-q} z^{-(M-1-s)} P_{s,q}(z^M).
\end{aligned}$$

So, we have

$$\hat{X}(z) = \tfrac{1}{M} \sum_{l=0}^{M-1} X(zW^l) \sum_{q=0}^{M-1} W^{-lq} V_q(z) \tag{3.50a}$$

where

$$V_q(z) = \sum_{s=0}^{M-1} z^{-q} z^{-(M-1-s)} P_{s,q}(z^M). \tag{3.50b}$$

Clearly, in order to achieve alias cancellation, we must have

$$\sum_{q=0}^{M-1} W^{-lq} V_q(z) \equiv 0 \quad \text{for } l = 1, \ldots, M-1.$$

We can rewrite these conditions on aliasing in matrix form as

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W^{-1} & \cdots & W^{-(M-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{-(M-1)} & \cdots & W^{-(M-1)^2} \end{bmatrix} \begin{bmatrix} V_0(z) \\ V_1(z) \\ \vdots \\ V_{M-1}(z) \end{bmatrix} = \begin{bmatrix} \lambda(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Now, we observe that the matrix in the above equation is $\boldsymbol{W}_M^\dagger$. Premultiplying both sides of the above equation by $\boldsymbol{W}$

and using the fact that $\boldsymbol{W}\boldsymbol{W}^\dagger = M\boldsymbol{I}$, we obtain

$$\begin{bmatrix} V_0(z) \\ V_1(z) \\ \vdots \\ V_{M-1}(z) \end{bmatrix} = \frac{1}{M} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W^1 & \cdots & W^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W^{M-1} & \cdots & W^{(M-1)^2} \end{bmatrix} \begin{bmatrix} \lambda(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\lambda(z) \\ \frac{1}{M}\lambda(z) \\ \vdots \\ \frac{1}{M}\lambda(z) \end{bmatrix}.$$

This implies, however, that

$$V_l(z) = V_0(z) = \tfrac{1}{M}\lambda(z)$$

for $l = 0, 1, \ldots, M-1$. Therefore, the system is alias free if and only if $V_l(z)$ is the same for all l.

Examining the expression for $V_l(z)$ in (3.50), we see that this is a polyphase decomposition. Furthermore, since $V_0(z) = V_1(z) = \cdots = V_{M-1}(z)$, each of these are polyphase decompositions of the same function. With this in mind, we write

$$\begin{aligned} V_l(z) &= \sum_{s=0}^{M-1} z^{-l} z^{-(M-1-s)} P_{s,l}(z^M) \\ &= \sum_{s=0}^{M-1} z^{-M+1-l+s} P_{s,l}(z^M) \\ &= z^{-l} P_{M-1,l}(z^M) + \sum_{s=0}^{M-2} z^{-M+1-l+s} P_{s,l}(z^M) \end{aligned} \tag{3.51}$$

and

$$\begin{aligned} V_{l+1}(z) &= \sum_{s=0}^{M-1} z^{-(l+1)} z^{-(M-1-s)} P_{s,l+1}(z^M) \\ &= \sum_{s=0}^{M-1} z^{-M-l+s} P_{s,l+1}(z^M) \\ &= z^{-M-l} P_{0,l+1}(z^M) + \sum_{s=1}^{M-1} z^{-M-l+s} P_{s,l+1}(z^M) \\ &= z^{-l}[z^{-M} P_{0,l+1}(z^M)] + \sum_{s=0}^{M-2} z^{-M+1-l+s} P_{s+1,l+1}(z^M). \end{aligned} \tag{3.52}$$

Comparing (3.51) and (3.52), we see that

$$P_{s,l}(z) = \begin{cases} P_{s+1,l+1}(z) & \text{for } s = 0, 1, \ldots, M-2 \\ z^{-1} P_{0,l+1}(z) & \text{for } s = M-1. \end{cases}$$

This implies that the lth column of $\boldsymbol{P}(z)$ is formed by circularly shifting the $(l+1)$th column upwards and multiplying the circulated element by z^{-1}. Thus, $\boldsymbol{P}(z)$ is B-type pseudocirculant.

Assuming that aliasing is cancelled, the system is LTI with the output determined (from (3.50a)) by

$$\begin{aligned} \hat{X}(z) &= \tfrac{1}{M} \sum_{l=0}^{M-1} X(zW^l) \sum_{q=0}^{M-1} W^{-lq} V_q(z) \\ &= \tfrac{1}{M} X(z) \sum_{q=0}^{M-1} V_q(z) \\ &= \tfrac{1}{M} X(z)[M V_0(z)] \\ &= V_0(z) X(z). \end{aligned}$$

Therefore, the transfer function $T(z) = V_0(z)$. From (3.50b), this gives us

$$T(z) = \sum_{s=0}^{M-1} z^{-(M-1-s)} P_{s,0}(z^M).$$

This expression for $T(z)$ is in terms of the elements from the zeroth column of $\boldsymbol{P}(z)$. Using the properties of a pseudocirculant matrix, however, we can rewrite $T(z)$ in terms of the elements from the zeroth row of $\boldsymbol{P}(z)$. Since $\boldsymbol{P}(z)$ is (B-type) pseudocirculant, we have

$$P_{k,0}(z) = \begin{cases} P_{0,k}(z) & \text{for } k = 0 \\ z^{-1}P_{0,M-k}(z) & \text{for } k = 1,2,\ldots,M-1. \end{cases}$$

Substituting into the equation for $T(z)$ above, we obtain

$$\begin{aligned} T(z) &= z^{-(M-1)}P_{0,0}(z^M) + \sum_{s=1}^{M-1} z^{-(M-1-s)}P_{s,0}(z^M) \\ &= z^{-(M-1)}P_{0,0}(z^M) + \sum_{s=1}^{M-1} z^{-(M-1-s)}z^{-M}P_{0,M-s}(z^M) \\ &= z^{-(M-1)}P_{0,0}(z^M) + \sum_{s=1}^{M-1} z^{-(2M-1-s)}P_{0,M-s}(z^M) \\ &= z^{-(M-1)}\left[P_{0,0}(z^M) + \sum_{s=1}^{M-1} z^{-(M-s)}P_{0,M-s}(z^M)\right] \end{aligned}$$

Now, we employ a change of variable. Let $s' = M-s$ so that $s = M-s'$. Applying the change of variable and dropping the primes, we obtain

$$\begin{aligned} T(z) &= z^{-(M-1)}\left[P_{0,0}(z^M) + \sum_{s=1}^{M-1} z^{-s}P_{0,s}(z^M)\right] \\ &= z^{-(M-1)}\sum_{s=0}^{M-1} z^{-s}P_{0,s}(z^M) \\ &= z^{-(M-1)}\sum_{s=0}^{M-1} z^{-s}P_s(z^M). \end{aligned}$$

Thus, the transfer function $T(z)$ has the stated form.

$(3,1)$-TYPE POLYPHASE. The proof of the theorem in the case of a $(3,1)$-type polyphase system follows an approach similar to that above, and is left as an exercise to the reader. □

Example 3.22. Suppose that we have a two-channel UMD filter bank with analysis filter transfer functions $H_0(z)$ and $H_1(z)$ and synthesis filter transfer functions $G_0(z)$ and $G_1(z)$ given by

$$\begin{aligned} H_0(z) &= -\tfrac{1}{8} + \tfrac{1}{4}z^{-1} + \tfrac{3}{4}z^{-2} + \tfrac{1}{4}z^{-3} - \tfrac{1}{8}z^{-4}, \\ H_1(z) &= -\tfrac{1}{2} + z^{-1} - \tfrac{1}{2}z^{-2}, \\ G_0(z) &= \tfrac{1}{2} + z^{-1} + \tfrac{1}{2}z^{-2} \quad \text{and} \\ G_1(z) &= -\tfrac{1}{8} - \tfrac{1}{4}z^{-1} + \tfrac{3}{4}z^{-2} - \tfrac{1}{4}z^{-3} - \tfrac{1}{8}z^{-4}. \end{aligned}$$

Using polyphase matrices, determine whether this system is alias free.

Solution. Consider a $(3,1)$-type polyphase representation of the system. We have

$$\begin{gathered}
H_0(z) = z^0\left(-\tfrac{1}{8}+\tfrac{3}{4}z^{-2}-\tfrac{1}{8}z^{-4}\right)+z^1\left(\tfrac{1}{4}z^{-2}+\tfrac{1}{4}z^{-4}\right),\\
H_1(z) = z^0\left(-\tfrac{1}{2}-\tfrac{1}{2}z^{-2}\right)+z^1\left(z^{-2}\right),\\
G_0(z) = z^0\left(\tfrac{1}{2}+\tfrac{1}{2}z^{-2}\right)+z^{-1}(1), \quad \text{and}\\
G_1(z) = z^0\left(-\tfrac{1}{8}+\tfrac{3}{4}z^{-2}-\tfrac{1}{8}z^{-4}\right)+z^{-1}\left(-\tfrac{1}{4}-\tfrac{1}{4}z^{-2}\right).
\end{gathered}$$

So, the analysis and synthesis polyphase matrices are given by

$$\begin{gathered}
\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} -\frac{1}{8}+\frac{3}{4}z^{-1}-\frac{1}{8}z^{-2} & \frac{1}{4}z^{-1}+\frac{1}{4}z^{-2} \\ -\frac{1}{2}-\frac{1}{2}z^{-1} & z^{-1} \end{bmatrix} \quad \text{and}\\
\boldsymbol{G}_\mathsf{p}(z) = \begin{bmatrix} \frac{1}{2}+\frac{1}{2}z^{-1} & -\frac{1}{8}+\frac{3}{4}z^{-1}-\frac{1}{8}z^{-2} \\ 1 & -\frac{1}{4}-\frac{1}{4}z^{-1} \end{bmatrix}.
\end{gathered}$$

The product $\boldsymbol{P}(z) = \boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\boldsymbol{P}(z) = \begin{bmatrix} 0 & z^{-2} \\ z^{-1} & 0 \end{bmatrix}.$$

By inspection, we have that $\boldsymbol{P}(z)$ is an A-type pseudocirculant matrix. Thus, the system is alias free. Furthermore, we can compute the distortion function $T(z)$ for the system as

$$T(z) = \sum_{k=0}^{1} z^k P_k(z^2) = z^0(0)+z^1(z^{-4}) = z^{-3}.$$

(Note that P_k denotes the kth element in the top row of $\boldsymbol{P}(z)$.) Thus, the system also happens to have the PR property with an associated reconstruction delay of 3. □

Alternative Solution. Consider a $(1,2)$-type polyphase representation of the system. We can rewrite the analysis and synthesis filter transfer functions as

$$\begin{gathered}
H_0(z) = z^0\left(-\tfrac{1}{8}+\tfrac{3}{4}z^{-2}-\tfrac{1}{8}z^{-4}\right)+z^{-1}\left(\tfrac{1}{4}+\tfrac{1}{4}z^{-2}\right),\\
H_1(z) = z^0\left(-\tfrac{1}{2}-\tfrac{1}{2}z^{-2}\right)+z^{-1}(1),\\
G_0(z) = z^{-1}(1)+z^0\left(\tfrac{1}{2}+\tfrac{1}{2}z^{-2}\right), \quad \text{and}\\
G_1(z) = z^{-1}\left(-\tfrac{1}{4}-\tfrac{1}{4}z^{-2}\right)+z^0\left(-\tfrac{1}{8}+\tfrac{3}{4}z^{-2}-\tfrac{1}{8}z^{-4}\right).
\end{gathered}$$

So, the analysis and synthesis polyphase matrices are given by

$$\begin{gathered}
\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} -\frac{1}{8}+\frac{3}{4}z^{-1}-\frac{1}{8}z^{-2} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -\frac{1}{2}-\frac{1}{2}z^{-1} & 1 \end{bmatrix} \quad \text{and}\\
\boldsymbol{G}_\mathsf{p} = \begin{bmatrix} 1 & -\frac{1}{4}-\frac{1}{4}z^{-1} \\ \frac{1}{2}+\frac{1}{2}z^{-1} & -\frac{1}{8}+\frac{3}{4}z^{-1}-\frac{1}{8}z^{-2} \end{bmatrix}.
\end{gathered}$$

The product $\boldsymbol{P}(z) = \boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} z^{-1} & 0 \\ 0 & z^{-1} \end{bmatrix}.$$

Since $\boldsymbol{P}(z)$ is a B-type pseudocirculant matrix, the system is alias free. Furthermore, the distortion function $T(z)$ for the system is given by

$$T(z) = z^{-1}\sum_{k=0}^{1} z^{-k}P_k(z^2) = z^{-1}\left[z^0(z^{-2})+z^{-1}(0)\right] = z^{-3}.$$

(Note that P_k denotes the kth element in the top row of $\boldsymbol{P}(z)$.) Thus, the system also happens to have the PR property with an associated reconstruction delay of 3. □

Now, we introduce a necessary and sufficient condition for the PR property formulated in terms of the polyphase matrices.

Theorem 3.13 (Necessary and sufficient conditions for PR)**.** *An M-channel UMD filter bank in either $(1,2)$ or $(3,1)$ polyphase form with analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ has the PR property if and only if the product $\boldsymbol{P}(z) \triangleq \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z)$ has the form*

$$\boldsymbol{P}(z) = \begin{cases} \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^K & \text{for } (1,2) \text{ type} \\ \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^K & \text{for } (3,1) \text{ type} \end{cases}$$

for some $K \in \mathbb{Z}$. If this condition is satisfied, the relationship between the input signal $x[n]$ and the reconstructed signal $\hat{x}[n]$ is given by

$$\hat{x}[n] = x[n-n_0]$$

where

$$n_0 = \begin{cases} K+M-1 & \text{for } (1,2) \text{ type} \\ K & \text{for } (3,1) \text{ type.} \end{cases}$$

Thus, the shift-free PR property holds if and only if

$$\boldsymbol{P}(z) = \begin{cases} \begin{bmatrix} \mathbf{0} & z\boldsymbol{I}_{M-1} \\ 1 & \mathbf{0} \end{bmatrix} & \text{for } (1,2) \text{ type} \\ \boldsymbol{I} & \text{for } (3,1) \text{ type.} \end{cases}$$

This theorem holds regardless of whether the analysis/synthesis filters are of the FIR or IIR type.

Proof. TYPE $(3,1)$ POLYPHASE. Consider a UMD filter bank represented in $(3,1)$-type polyphase form. From (3.48), we know that such a system is alias free if and only if $\boldsymbol{P}(z)$ is an A-type pseudocirculant matrix. That is, $\boldsymbol{P}(z)$ is of the form

$$\boldsymbol{P}(z) = \sum_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^k, \tag{3.53}$$

where $P_k(z)$ denotes the kth element in the top row of $\boldsymbol{P}(z)$. (Here, we have used the fact that a pseudocirculant matrix can be expressed as in (3.47).) Furthermore, from (3.49), such an alias-free system has the distortion function

$$T(z) = \sum_{k=0}^{M-1} z^k P_k(z^M). \tag{3.54}$$

Now, suppose that the system has the PR property. Then, we have that $T(z)$ is of the form

$$T(z) = z^{-n_0} \tag{3.55}$$

for some $n_0 \in \mathbb{Z}$. This implies that exactly one of the $P_k(z)$, say $P_N(z)$, is not identically zero. In particular, we have

$$P_k(z) = \begin{cases} z^{-L} & \text{for } k = N \\ 0 & \text{for } k \neq N, \end{cases} \tag{3.56}$$

where $L \in \mathbb{Z}$. Combining (3.54), (3.55), and (3.56), we have

$$\begin{aligned} T(z) &= \sum_{k=0}^{M-1} z^k P_k(z^M) \\ &= z^N P_N(z^M) \\ &= z^N z^{-ML} \\ &= z^{-(ML-N)} \\ &= z^{-n_0}, \end{aligned} \tag{3.57}$$

where $n_0 = ML - N$. Combining (3.53) and (3.56), we obtain

$$\begin{aligned} \boldsymbol{P}(z) &= \sum_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^k \\ &= P_N(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^N \\ &= z^{-L} \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^N \\ &= z^{-L} \boldsymbol{I}_M \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix}^N . \end{aligned}$$

Now, we observe that

$$z^{-L} \boldsymbol{I}_M = \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{ML} \quad \text{and} \quad \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{-1} .$$

So, we have

$$\begin{aligned} \boldsymbol{P}(z) &= \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{ML} \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{-N} \\ &= \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{ML-N} . \end{aligned}$$

From (3.57), we have that $n_0 = ML - N$. Thus, we have

$$\boldsymbol{P}(z) = \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{n_0} .$$

TYPE $(1,2)$ POLYPHASE. Consider a UMD filter bank represented in $(1,2)$-type polyphase form. From (3.48), we know that such a system is alias free if and only if $\boldsymbol{P}(z)$ is a B-type pseudocirculant matrix. That is, $\boldsymbol{P}(z)$ is of the form

$$\boldsymbol{P}(z) = \sum_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^k , \tag{3.58}$$

where $P_k(z)$ denotes the kth element in the top row of $\boldsymbol{P}(z)$. Furthermore, from (3.49), such an alias-free system has the distortion function

$$T(z) = z^{-(M-1)} \sum_{k=0}^{M-1} z^{-k} P_k(z^M). \tag{3.59}$$

Now, suppose that the system has the PR property. Then, we have that $T(z)$ is of the form

$$T(z) = z^{-n_0} \tag{3.60}$$

for some $n_0 \in \mathbb{Z}$. This implies that exactly one of the $P_k(z)$, say $P_N(z)$, is not identically zero. In particular, we have

$$P_k(z) = \begin{cases} z^{-L} & \text{for } k = N \\ 0 & \text{for } k \neq N, \end{cases} \tag{3.61}$$

where $L \in \mathbb{Z}$. Combining (3.59), (3.60), and (3.61), we have

$$\begin{aligned} T(z) &= z^{-(M-1)} \sum_{k=0}^{M-1} z^{-k} P_k(z^M) \qquad (3.62)\\ &= z^{-(M-1)} z^{-N} P_N(z^M) \\ &= z^{-(M-1)} z^{-N} z^{-ML} \\ &= z^{-(M-1+ML+N)} \\ &= z^{-n_0}, \qquad (3.63) \end{aligned}$$

where $n_0 = M-1+ML+N$. Combining (3.58) and (3.61), we obtain

$$\begin{aligned} \boldsymbol{P}(z) &= \sum_{k=0}^{M-1} P_k(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^k \\ &= P_N(z) \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^N \\ &= z^{-L} \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^N \\ &= z^{-L} \boldsymbol{I}_M \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^N . \end{aligned}$$

Now, we observe that

$$z^{-L} \boldsymbol{I}_M = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{ML} .$$

So, we have

$$\begin{aligned} \boldsymbol{P}(z) &= \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{ML} \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^N \\ &= \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{ML+N} . \end{aligned}$$

From (3.62), we have that $n_0 = M-1+ML+N$. Thus, we have

$$\boldsymbol{P}(z) = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{n_0-(M-1)} .$$

Thus, if the UMD filter bank has the PR property, $\boldsymbol{P}(z)$ must have the form stated in the theorem. Lastly, we observe that

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{-(M-1)} = \begin{bmatrix} \mathbf{0} & z\boldsymbol{I}_{M-1} \\ 1 & \mathbf{0} \end{bmatrix} .$$

So, in the case that $n_0 = 0$, the above expression for $\boldsymbol{P}(z)$ simplifies to

$$\begin{bmatrix} \mathbf{0} & z\boldsymbol{I}_{M-1} \\ 1 & \mathbf{0} \end{bmatrix}.$$

□

One comment is in order concerning Theorem 3.13. If we let $K = 0$, $\boldsymbol{P}(z)$ becomes the $M \times M$ identity matrix. Since the identity matrix is sometimes an attractive matrix with which to work, one might wonder what degree of freedom is lost by constraining K to be zero. As it turns out, not much is sacrificed. This quantity only serves to introduce additional delay into the analysis/synthesis filters. For this reason, we often only consider the case of $\boldsymbol{P}(z) = \boldsymbol{I}$ when designing PR filter banks. Delay (or advance) can always be added to the analysis and synthesis filters after the fact if required. Finally, with $\boldsymbol{P}(z) = \boldsymbol{I}$, the problem of designing PR filter banks, in some sense, reduces to a problem of factorizing the identity matrix.

From Theorem 3.13, it follows that a type $(3,1)$ polyphase system has the shift-free PR property if and only if

$$\boldsymbol{P}(z) = \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z) = \boldsymbol{I} \tag{3.64}$$

(or equivalently $\boldsymbol{G}_{\mathsf{p}}(z) = \boldsymbol{H}_{\mathsf{p}}^{-1}(z)$). By examining the polyphase form of a UMD filter bank, we can see the reason behind the above condition for shift-free PR. If (3.64) is satisfied, then the synthesis polyphase filtering (represented by $\boldsymbol{G}_{\mathsf{p}}(z)$) cancels the effects of the analysis polyphase filtering (represented by $\boldsymbol{H}_{\mathsf{p}}(z)$). This being the case, the filter bank, in effect, only serves to split the input signal into its polyphase components and then recombine these components, yielding the original input signal with no shift.

The preceding theorem has some interesting implications—the most important of which is stated in the following corollary:

Corollary 3.2. *Any M-channel PR UMD filter bank in either $(1,2)$ or $(3,1)$ polyphase form with analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ must be such that the product $\boldsymbol{P}(z) \triangleq \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z)$ has a determinant of the form*

$$\det \boldsymbol{P}(z) = (-1)^{K(M-1)} z^{-K}$$

where K is an integer. Moreover, if the analysis and synthesis filters of the UMD filter bank are of the FIR type, this condition implies that the polyphase matrices must have determinants of the form

$$\det \boldsymbol{H}_{\mathsf{p}}(z) = \alpha_0 z^{-L_0} \quad \text{and}$$
$$\det \boldsymbol{G}_{\mathsf{p}}(z) = \alpha_1 z^{-L_1}$$

where the α_i are nonzero constants and the L_i are integers. That is, the determinants of the polyphase matrices must be monomials.

Proof. From Theorem 3.13, we know that $\boldsymbol{P}(z)$ has the form

$$\boldsymbol{P}(z) = \begin{cases} \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^K & \text{for } (1,2) \text{ type} \\ \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^K & \text{for } (3,1) \text{ type,} \end{cases}$$

where K is an integer. Taking the determinant of $\boldsymbol{P}(z)$ simply yields

$$\det \boldsymbol{P}(z) = (-1)^{K(M-1)} z^{-K}. \tag{3.65}$$

Thus, $\det \boldsymbol{P}(z)$ must be of the form stated in the theorem.

Using both (3.65) and the fact that $\boldsymbol{P}(z) = \boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z)$, we have

$$\det \boldsymbol{G}_\mathsf{p}(z) \det \boldsymbol{H}_\mathsf{p}(z) = (-1)^{K(M-1)} z^{-K}. \tag{3.66}$$

Assume now that the analysis and synthesis filters are of the FIR type. This implies that $\det \boldsymbol{H}_\mathsf{p}(z)$ and $\det \boldsymbol{G}_\mathsf{p}(z)$ are Laurent polynomials. The only way that the product in (3.66) can be a monomial is if both $\det \boldsymbol{H}_\mathsf{p}(z)$ and $\det \boldsymbol{G}_\mathsf{p}(z)$ are themselves monomials. Thus, the last part of the theorem is proven. □

Example 3.23. Suppose that we have a two-channel UMD filter bank with analysis filter transfer functions $H_0(z)$ and $H_1(z)$, where

$$H_0(z) = \tfrac{1}{2}z + \tfrac{1}{2} \quad \text{and}$$
$$H_1(z) = -\tfrac{1}{8}z^3 - \tfrac{1}{8}z^2 + z - 1 + \tfrac{1}{8}z^{-1} + \tfrac{1}{8}z^{-2}.$$

Using polyphase-domain analysis, determine the transfer functions $G_0(z)$ and $G_1(z)$ of the synthesis filters that will yield a shift-free PR system.

Solution. In what follows, we choose to employ a type (3,1) polyphase representation of the filter bank. This choice is convenient since the shift-free PR condition is then equivalent to the analysis and synthesis polyphase matrices being inverses of one another (i.e., $\boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z) = \boldsymbol{I}$). First, we express the analysis filters in type-3 polyphase form, yielding

$$H_0(z) = z^0\left(\tfrac{1}{2}\right) + z^1\left(\tfrac{1}{2}\right) \quad \text{and}$$
$$H_1(z) = z^0\left(-\tfrac{1}{8}z^2 - 1 + \tfrac{1}{8}z^{-2}\right) + z^1\left(-\tfrac{1}{8}z^2 + 1 + \tfrac{1}{8}z^{-2}\right).$$

The analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{8}z - 1 + \frac{1}{8}z^{-1} & -\frac{1}{8}z + 1 + \frac{1}{8}z^{-1} \end{bmatrix}.$$

Now, we observe that, for the shift-free PR condition to hold, $\boldsymbol{G}_\mathsf{p}(z) = \boldsymbol{H}_\mathsf{p}^{-1}(z)$. We compute the determinant of $\boldsymbol{H}_\mathsf{p}(z)$ as

$$\begin{aligned}\det \boldsymbol{H}_\mathsf{p}(z) &= \tfrac{1}{2}\left(-\tfrac{1}{8}z + 1 + \tfrac{1}{8}z^{-1}\right) - \tfrac{1}{2}\left(-\tfrac{1}{8}z - 1 + \tfrac{1}{8}z^{-1}\right) \\ &= -\tfrac{1}{16}z + \tfrac{1}{2} + \tfrac{1}{16}z^{-1} + \tfrac{1}{16}z + \tfrac{1}{2} - \tfrac{1}{16}z^{-1} \\ &= 1.\end{aligned}$$

Since $\det \boldsymbol{H}_\mathsf{p}(z) \not\equiv 0$, $\boldsymbol{H}_\mathsf{p}^{-1}(z)$ is well defined. So, we have

$$\begin{aligned}\boldsymbol{G}_\mathsf{p}(z) &= [\det \boldsymbol{H}_\mathsf{p}(z)]^{-1} \operatorname{Adj} \boldsymbol{H}_\mathsf{p}(z) \\ &= \begin{bmatrix} -\frac{1}{8}z + 1 + \frac{1}{8}z^{-1} & (-1)\left(-\frac{1}{8}z - 1 + \frac{1}{8}z^{-1}\right) \\ (-1)\left(\frac{1}{2}\right) & \frac{1}{2} \end{bmatrix}^T \\ &= \begin{bmatrix} -\frac{1}{8}z + 1 + \frac{1}{8}z^{-1} & -\frac{1}{2} \\ \frac{1}{8}z + 1 - \frac{1}{8}z^{-1} & \frac{1}{2} \end{bmatrix}.\end{aligned}$$

From $\boldsymbol{G}_\mathsf{p}(z)$, we can easily determine the synthesis filter transfer functions $G_0(z)$ and $G_1(z)$ to be

$$\begin{aligned}G_0(z) &= z^0\left(-\tfrac{1}{8}z^2 + 1 + \tfrac{1}{8}z^{-2}\right) + z^{-1}\left(\tfrac{1}{8}z^2 + 1 - \tfrac{1}{8}z^{-2}\right) \\ &= -\tfrac{1}{8}z^2 + \tfrac{1}{8}z + 1 + z^{-1} + \tfrac{1}{8}z^{-2} - \tfrac{1}{8}z^{-3} \quad \text{and} \\ G_1(z) &= z^0\left(-\tfrac{1}{2}\right) + z^{-1}\left(\tfrac{1}{2}\right) \\ &= -\tfrac{1}{2} + \tfrac{1}{2}z^{-1}.\end{aligned}$$

□

Example 3.24. Suppose that we have a two-channel UMD filter bank with analysis filter transfer functions $H_0(z)$ and $H_1(z)$, where

$$H_0(z) = z + 1 + z^{-1} \quad \text{and} \quad H_1(z) = 1 - z^{-1}.$$

Determine whether a shift-free PR system can be constructed with FIR synthesis filters.

Solution. Consider a $(3,1)$-type polyphase representation. We have

$$\begin{aligned} H_0(z) &= z^0(1) + z^1\left(1 + z^{-2}\right) \quad \text{and} \\ H_1(z) &= z^0(1) + z^1\left(-z^{-2}\right). \end{aligned}$$

The analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ is given by

$$\boldsymbol{H}_{\mathsf{p}}(z) = \begin{bmatrix} 1 & 1 + z^{-1} \\ 1 & -z^{-1} \end{bmatrix}.$$

In order for PR to be achieved with FIR synthesis filters, we must have that $\det \boldsymbol{H}_{\mathsf{p}}(z)$ is a monomial. Computing the determinant, we have

$$\begin{aligned} \det \boldsymbol{H}_{\mathsf{p}}(z) &= -z^{-1} - \left(1 + z^{-1}\right) \\ &= -z^{-1} - 1 - z^{-1} \\ &= -1 - 2z^{-1}. \end{aligned}$$

Since $\det \boldsymbol{H}_{\mathsf{p}}(z)$ is not a monomial, it is not possible to construct a shift-free PR system with FIR synthesis filters. □

3.3.11 Biorthonormal and Orthonormal UMD Filter Banks

A UMD filter bank may be associated with a biorthonormal or orthonormal signal expansion. In particular, we have the results stated below.

Theorem 3.14 (Biorthonormality)**.** *Suppose that we have an M-channel UMD filter bank with analysis filters $\{H_k(z)\}$, synthesis filters $\{G_k(z)\}$, analysis modulation matrix $\boldsymbol{H}_{\mathsf{m}}(z)$, synthesis modulation matrix $\boldsymbol{G}_{\mathsf{m}}(z)$, analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$, and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$. Let $h_k[n]$ and $g_k[n]$ denote the inverse $\mathcal{Z}$ transforms of $H_k(z)$ and $G_k(z)$, respectively, for $k = 0,1,\ldots,M-1$. Then, the above system computes a biorthonormal signal expansion if and only if*

$$\langle h_k^*[Mm - \cdot], g_l[\cdot] \rangle = \delta[k-l]\delta[m].$$

Furthermore, this condition is equivalent to each of the following:

$$\boldsymbol{H}_{\mathsf{m}}(z)\boldsymbol{g}(z) = \begin{bmatrix} M & 0 & \cdots & 0 \end{bmatrix}^T, \tag{3.67}$$

$$\boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{H}_{\mathsf{m}}(z) = M\boldsymbol{I}, \ \textit{and} \tag{3.68}$$

$$\boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z) = \begin{cases} \begin{bmatrix} \boldsymbol{0} & z\boldsymbol{I}_{M-1} \\ 1 & \boldsymbol{0} \end{bmatrix} & \textit{for } (1,2) \textit{ type} \\ \boldsymbol{I} & \textit{for } (3,1) \textit{ type.} \end{cases} \tag{3.69}$$

Theorem 3.15 (Orthonormality conditions)**.** *Suppose that we have an M-channel UMD filter bank with analysis filters $\{H_k(z)\}$, synthesis filters $\{G_k(z)\}$, analysis modulation matrix $\boldsymbol{H}_{\mathsf{m}}(z)$, synthesis modulation matrix $\boldsymbol{G}_{\mathsf{m}}(z)$, analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$, and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$. Let $h_k[n]$ and $g_k[n]$ denote the inverse $\mathcal{Z}$ transforms of $H_k(z)$ and $G_k(z)$, respectively, for $k = 0,1,\ldots,M-1$. Then, the above system computes an orthonormal signal expansion if and only if*

$$\langle g_k[\cdot - Mm], g_l[\cdot] \rangle = \delta[k-l]\delta[m], \ \ h_k[n] = g_k^*[-n].$$

Furthermore, this condition is equivalent to each of the following:

$$\boldsymbol{G}_{\mathsf{m}}^T(z^{-1})\boldsymbol{g}_*(z^{-1}) = \begin{bmatrix} M & 0 & \cdots & 0 \end{bmatrix}^T, \quad \boldsymbol{H}_{\mathsf{m}}(z) = \boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1}); \tag{3.70}$$

$$\boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1}) = M\boldsymbol{I}, \quad \boldsymbol{H}_{\mathsf{m}}(z) = \boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1});\ \textit{and} \tag{3.71}$$

$$\boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{G}_{\mathsf{p}_*}^T(z^{-1}) = \begin{cases} \begin{bmatrix} \boldsymbol{0} & z\boldsymbol{I}_{M-1} \\ 1 & \boldsymbol{0} \end{bmatrix} & \textit{for } (1,2) \textit{ type} \\ \boldsymbol{I} & \textit{for } (3,1) \textit{ type}, \end{cases} \quad \boldsymbol{H}_{\mathsf{p}}(z) = \boldsymbol{G}_{\mathsf{p}_*}^T(z^{-1}). \tag{3.72}$$

Proof. An orthonormal system is simply a special case of a biorthonormal system where each basis function and its dual are equal. Thus, we have $g_k[n] = h_k^*[-n]$. In the $\mathcal{Z}$ domain, this is equivalent to

$$G_k(z) = H_{k_*}(z^{-1}).$$

Thus, an orthonormal system is a biorthonormal system that also satisfies each of the following equivalent conditions:

$$\begin{aligned} \boldsymbol{h}(z) &= \boldsymbol{g}_*(z^{-1}), \\ \boldsymbol{H}_{\mathsf{m}}(z) &= \boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1}), \quad \text{and} \\ \boldsymbol{H}_{\mathsf{p}}(z) &= \boldsymbol{G}_{\mathsf{p}_*}^T(z^{-1}). \end{aligned}$$

(In the case of the polyphase matrices, we are assuming a $(3,1)$ type polyphase representation.) Using Theorem 3.14, we have

$$\begin{aligned} \boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{g}_*(z^{-1}) &= \begin{bmatrix} M & 0 & \cdots & 0 \end{bmatrix}^T, \\ \boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{H}_{\mathsf{m}}(z) = M\boldsymbol{I} &\Leftrightarrow \boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1}) = M\boldsymbol{I}, \quad \text{and} \\ \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{H}_{\mathsf{p}}(z) = \boldsymbol{I} &\Leftrightarrow \boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{G}_{\mathsf{p}_*}^T(z^{-1}) = \boldsymbol{I}. \end{aligned}$$

□

As an aside, we observe that the condition $\boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{G}_{\mathsf{m}_*}^T(z^{-1}) = M\boldsymbol{I}$ in (3.71) is equivalent to $\frac{1}{\sqrt{M}}\boldsymbol{G}_{\mathsf{m}}(z)$ being paraunitary. Similarly, the condition $\boldsymbol{G}_{\mathsf{p}}(z)\boldsymbol{G}_{\mathsf{p}_*}^T(z^{-1}) = \boldsymbol{I}$ in (3.72) is equivalent to $\boldsymbol{G}_{\mathsf{p}}(z)$ being paraunitary. For the above reason, we sometime refer to an orthonormal UMD filter bank as a paraunitary filter bank.

3.4 Two-Channel UMD Filter Banks

Now, we consider the special case of two-channel UMD filter banks. Recall that the general z-domain input-output relationship for an M-channel UMD filter bank is given by

$$\boldsymbol{A}(z) = \tfrac{1}{M}\boldsymbol{H}_{\mathsf{m}}^T(z)\boldsymbol{g}(z).$$

In the case of a two-channel PR system with reconstruction delay n_0, we have

$$\underbrace{\begin{bmatrix} z^{-n_0} \\ 0 \end{bmatrix}}_{\boldsymbol{A}(z)} = \tfrac{1}{2}\underbrace{\begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{m}}^T(z)}\underbrace{\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix}}_{\boldsymbol{g}(z)}.$$

Rearranging the above equation and solving for $\boldsymbol{g}(z)$, we have

$$\begin{bmatrix} 2z^{-n_0} \\ 0 \end{bmatrix} = \begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix}\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix}$$

$$\Rightarrow \quad \begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \underbrace{\begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix}^{-1}}_{\boldsymbol{H}_{\mathsf{m}}^{-T}(z)}\begin{bmatrix} 2z^{-n_0} \\ 0 \end{bmatrix}. \tag{3.73}$$

Now, we compute $\boldsymbol{H}_{\mathsf{m}}^{-T}(z)$ in the preceding equation. The determinant of $\boldsymbol{H}_{\mathsf{m}}^{T}(z)$ is given by

$$\det \boldsymbol{H}_{\mathsf{m}}^{T}(z) = H_0(z)H_1(-z) - H_0(-z)H_1(z).$$

In what follows, we assume that $\det \boldsymbol{H}_{\mathsf{m}}^{T}(z) \not\equiv 0$ so that $\boldsymbol{H}_{\mathsf{m}}^{-T}(z)$ is well defined. In this case, we have

$$\begin{aligned}
\boldsymbol{H}_{\mathsf{m}}^{-T}(z) &= \frac{1}{\det \boldsymbol{H}_{\mathsf{m}}^{T}(z)} \operatorname{Adj} \boldsymbol{H}_{\mathsf{m}}^{T}(z) \\
&= \frac{1}{\det \boldsymbol{H}_{\mathsf{m}}(z)} \begin{bmatrix} H_1(-z) & -H_0(-z) \\ -H_1(z) & H_0(z) \end{bmatrix}^T \\
&= \frac{1}{\det \boldsymbol{H}_{\mathsf{m}}(z)} \begin{bmatrix} H_1(-z) & -H_1(z) \\ -H_0(-z) & H_0(z) \end{bmatrix}.
\end{aligned}$$

Substituting this expression for $\boldsymbol{H}_{\mathsf{m}}^{-T}(z)$ into (3.73) yields

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \frac{1}{\det \boldsymbol{H}_{\mathsf{m}}(z)} \begin{bmatrix} H_1(-z) & -H_1(z) \\ -H_0(-z) & H_0(z) \end{bmatrix} \begin{bmatrix} 2z^{-n_0} \\ 0 \end{bmatrix}.$$

Thus, we have that the synthesis filter transfer functions $G_0(z)$ and $G_1(z)$ are given by

$$G_0(z) = \left(\frac{2}{z^{n_0} \det \boldsymbol{H}_{\mathsf{m}}(z)} \right) H_1(-z) \quad \text{and} \tag{3.74a}$$

$$G_1(z) = -\left(\frac{2}{z^{n_0} \det \boldsymbol{H}_{\mathsf{m}}(z)} \right) H_0(-z). \tag{3.74b}$$

Suppose now that the analysis filters are FIR and the synthesis filters required for PR are also FIR. Then, we have that $\det \boldsymbol{H}_{\mathsf{m}}(z)$ must be a monomial. Consequently, the synthesis filter transfer functions must be of the form

$$G_0(z) = az^{-b}H_1(-z) \quad \text{and} \quad G_1(z) = -az^{-b}H_0(-z),$$

where $a \in \mathbb{C} \setminus \{0\}$ and $b \in \mathbb{Z}$ are constants determined by n_0 and $\det \boldsymbol{H}_{\mathsf{m}}(z)$. Thus, the transfer functions of the analysis and synthesis filters are very similar. That is, $G_0(z)$ and $G_1(z)$ are modulated, shifted, and scaled versions of $H_1(z)$ and $H_0(z)$, respectively.

Example 3.25. Consider the two-channel UMD filter bank having analysis filters with transfer functions $H_0(z), H_1(z)$ given by

$$H_0(z) = -\tfrac{1}{8}z^2 + \tfrac{1}{4}z + \tfrac{3}{4} + \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2} \quad \text{and} \quad H_1(z) = -\tfrac{1}{2}z^2 + z - \tfrac{1}{2}.$$

Determine the transfer functions $G_0(z)$ and $G_1(z)$ of the synthesis filters required for shift-free PR.

Solution. First, we compute the determinant of the analysis modulation matrix $\boldsymbol{H}_{\mathsf{m}}(z)$. This yields

$$\begin{aligned}
\det \boldsymbol{H}_{\mathsf{m}}(z) &= H_0(z)H_1(-z) - H_0(-z)H_1(z) \\
&= -2z.
\end{aligned}$$

Now, we can compute the transfer functions of the synthesis filters as follows:

$$\begin{aligned}
G_0(z) &= \left(\frac{2}{\det \boldsymbol{H}_{\mathsf{m}}(z)} \right) H_1(-z) \\
&= \left(\tfrac{2}{-2z} \right) \left(-\tfrac{1}{2}z^2 - z - \tfrac{1}{2} \right) \\
&= -z^{-1} \left(-\tfrac{1}{2}z^2 - z - \tfrac{1}{2} \right) \\
&= \tfrac{1}{2}z + 1 + \tfrac{1}{2}z^{-1} \quad \text{and}
\end{aligned}$$

$$\begin{aligned}G_1(z) &= -\left(\frac{2}{\det \boldsymbol{H}_{\mathsf{m}}(z)}\right) H_0(-z) \\ &= -\left(\tfrac{2}{-2z}\right)\left(-\tfrac{1}{8}z^2 - \tfrac{1}{4}z + \tfrac{3}{4} - \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2}\right) \\ &= z^{-1}\left(-\tfrac{1}{8}z^2 - \tfrac{1}{4}z + \tfrac{3}{4} - \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2}\right) \\ &= -\tfrac{1}{8}z - \tfrac{1}{4} + \tfrac{3}{4}z^{-1} - \tfrac{1}{4}z^{-2} - \tfrac{1}{8}z^{-3}.\end{aligned}$$

□

Theorem 3.16. *Suppose that a two-channel orthonormal UMD filter bank has linear-phase FIR analysis and synthesis filters with real coefficients. Then, the number of nonzero coefficients in each of the analysis and synthesis filters cannot exceed two.*

Proof. See [29, p. 159]. □

From the above theorem, we can see that, except in the trivial case of filter banks with two-tap filters, orthonormality and symmetry are mutually exclusive. This result has important consequences. In many applications, both orthonormality and symmetry are desirable. Unfortunately, we cannot have both (except for the trivial case noted above). In order to simultaneously achieve orthonormality and symmetry with longer filters, we must drop one or more of the following constraints: real filter coefficients, FIR filters, two channels.

It is helpful to note that linear-phase FIR PR filter banks can only assume a limited number of forms in the two-channel case. The possible forms are identified by the theorem below.

Theorem 3.17. *Suppose that a two-channel PR UMD filter bank is such that all of its filters are FIR and have linear phase. Then, the analysis filters have one of the following forms:*

1. *Both filters are symmetric and of odd lengths, differing by an odd multiple of two.*
2. *One filter is symmetric and the other is antisymmetric, and both are of even length with their lengths differing by an even (possibly zero) multiple of two.*
3. *One filter is of odd length, while the other is of even length. Both filters have all of their zeros on the unit circle. Either both filters are symmetric, or one is symmetric and the other is antisymmetric.*

Proof. For a sketch of a proof, see [33, Proposition 3.11, p. 134]. (See also [29, Theorem 4.3, p. 111].) □

It is worth noting that the last of the three cases in the above theorem is of little practical interest, as the analysis filters cannot have good frequency responses in this case.

3.5 Design of UMD Filter Banks

In this section, we briefly consider a design technique for shift-free PR filter banks. We consider both the biorthonormal and orthonormal types of shift-free PR systems. Before we begin, however, we introduce the definition below.

Definition 3.5 (Nyquist filter)**.** A filter with impulse response $h[n]$ is said to be a **Nyquist (M) filter** (or **M-th band filter**) if

$$(\downarrow M)h = c\delta_n$$

for some constant c. That is, all of the elements of the sequence h with indices divisible by M are zero, except for the element with index zero. A filter satisfying the above condition with $M = 2$ is called a **halfband filter**.

It follows from the definition of a halfband filter that, if $H(z)$ is halfband, then $H(z) + H(-z) = 2c$ (where c is as defined above).

Now, let us consider a two-channel PR UMD filter bank with analysis and synthesis filter transfer functions $H_0(z), H_1(z)$ and $G_0(z), G_1(z)$, respectively. Since the system has the PR property, we have

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2z^{-n_0}, \tag{3.75}$$

where n_0 is the reconstruction delay of the system. Now, we use (3.74) to rewrite the above equation exclusively in terms of $H_0(z)$ and $G_0(z)$. From (3.74), we have

$$H_1(z) = \frac{(-z)^{n_0} \det \boldsymbol{H}_\mathsf{m}(-z)}{2} G_0(-z).$$

Now, we substitute the expression for $H_1(z)$ from above and the expression for $G_1(z)$ from (3.74) into (3.75). This process yields

$$\begin{aligned} & H_0(z)G_0(z) - \left(\left(\frac{(-z)^{n_0} \det \boldsymbol{H}_\mathsf{m}(-z)}{2}\right) G_0(-z)\right)\left(\left(\frac{2}{z^{n_0} \det \boldsymbol{H}_\mathsf{m}(z)}\right) H_0(-z)\right) = 2z^{-n_0} \\ \Rightarrow \quad & H_0(z)G_0(z) - \left(\frac{(-1)^{n_0} \det \boldsymbol{H}_\mathsf{m}(-z)}{\det \boldsymbol{H}_\mathsf{m}(z)}\right) H_0(-z)G_0(-z) = 2z^{-n_0}. \end{aligned}$$

Observing that $\det \boldsymbol{H}_\mathsf{m}(z) = -\det(\boldsymbol{H}_\mathsf{m}(-z))$, we can simplify the above equation to obtain

$$H_0(z)G_0(z) + (-1)^{n_0} H_0(-z)G_0(-z) = 2z^{-n_0}. \tag{3.76}$$

Let us define

$$P(z) = H_0(z)G_0(z).$$

Then, we can rewrite (3.76) as

$$P(z) + (-1)^{n_0} P(-z) = 2z^{-n_0}.$$

Let us further suppose that the system has the shift-free PR property (i.e., $n_0 = 0$). In this case, we have

$$P(z) + P(-z) = 2. \tag{3.77}$$

This implies that all of the terms in $P(z)$ with even-indexed powers of z are zero except for the z^0 term. In other words, $P(z)$ is halfband. Thus, we trivially have that $H_0(z)G_0(z)$ is halfband. This result suggests a simple design technique for shift-free PR UMD filter banks.

From the above results, we can see that the following process can be used to design a shift-free PR system:

1. Find a $P(z)$ satisfying (3.77) (i.e., $P(z)$ is halfband).
2. Decompose $P(z)$ into two factors. Assign one factor to $H_0(z)$ and one factor to $G_0(z)$.
3. Using (3.74), determine $H_1(z)$ and $G_1(z)$ from $H_0(z)$ and $G_0(z)$.

Example 3.26 (Biorthonormal filter bank design)**.** Consider the halfband filter with transfer function

$$P(z) = -\tfrac{1}{16}z^3 + \tfrac{9}{16}z + 1 + \tfrac{9}{16}z^{-1} - \tfrac{1}{16}z^{-3}.$$

From $P(z)$, construct two distinct shift-free PR UMD filter banks.

Solution. Clearly, we have that $P(z) + P(-z) = 2$. We begin by factoring $P(z)$ to obtain

$$\begin{aligned} P(z) &= -\tfrac{1}{16}(z+1)^4(z^{-1} - 4z^{-2} + z^{-3}) \\ &= -\tfrac{1}{16}z^{-3}(z+1)^4(z - 2 + \sqrt{3})(z - (2 + \sqrt{3})). \end{aligned}$$

Now, we consider two distinct factorizations of $P(z)$.

FIRST SYSTEM. Consider the factorization $P(z) = H_0(z)G_0(z)$, where

$$\begin{aligned}H_0(z) &= -\tfrac{1}{8}z^{-2}(z+1)^2(z-2+\sqrt{3})(z-(2+\sqrt{3}))\\&= -\tfrac{1}{8}z^2+\tfrac{1}{4}z+\tfrac{3}{4}+\tfrac{1}{4}z^{-1}-\tfrac{1}{8}z^{-2}\\G_0(z) &= \tfrac{1}{2}z^{-1}(z+1)^2\\&= \tfrac{1}{2}z+1+\tfrac{1}{2}z^{-1}.\end{aligned}$$

For this particular system, H_0 and G_0 have linear phase. Moreover, H_0 and G_0 each have a second order zero at the Nyquist frequency.

SECOND SYSTEM. Consider the factorization $P(z) = H_0(z)G_0(z)$, where

$$\begin{aligned}H_0(z) &= -\tfrac{1}{2}z^{-1}(z+1)(z-2+\sqrt{3})\\&= -\tfrac{1}{2}z+\left(\tfrac{1-\sqrt{3}}{2}\right)+\left(\tfrac{2-\sqrt{3}}{2}\right)z^{-1}\\G_0(z) &= \tfrac{1}{8}z^{-2}(z+1)^3(z-(2+\sqrt{3}))\\&= \tfrac{1}{8}z^2+\left(\tfrac{1-\sqrt{3}}{8}\right)z-\left(\tfrac{3+3\sqrt{3}}{8}\right)-\left(\tfrac{5+3\sqrt{3}}{8}\right)z^{-1}-\left(\tfrac{2+\sqrt{3}}{8}\right)z^{-2}.\end{aligned}$$

For this particular system, neither H_0 nor G_0 have linear phase. Also, H_0 has a first order zero at the Nyquist frequency, and G_0 has a third order zero at the Nyquist frequency.

In the case of each of the above systems, the transfer functions of the remaining filters can be obtained via (3.74).

□

In the case of an orthonormal filter bank, we have $G_0(z) = H_0(z^{-1})$ (Here, we assume the filter coefficients to be real.) Suppose that we could factor $P(z)$ as

$$P(z) = C(z)C(z^{-1})$$

for some $C(z)$. Then, we could choose $H_0(z) = C(z)$ and $G_0(z) = C(z^{-1})$. If $P(z)$ is a symmetric halfband filter, we can always decompose $P(z)$ in this way. This process is known as spectral factorization. Thus, we have a means for constructing an orthonormal filter bank.

The following process can be used to construct an orthonormal filter bank:

1. Select a $P(z)$ satisfying the halfband condition (3.77) that is also symmetric.
2. Decompose $P(z)$ into the form $P(z) = C(z)C(z^{-1})$. Then, choose $H_0(z) = C(z)$ and $G_0(z) = C(z^{-1})$.
3. Using (3.74), determine $H_1(z)$ and $G_1(z)$ from $H_0(z)$ and $G_0(z)$.

Example 3.27 (Orthonormal filter bank design)**.** Use the spectral factorization method in order to design a two-channel orthonormal UMD filter bank that satisfies the following constraints: 1) the analysis and synthesis filters are of length four; and 2) each of the analysis and synthesis lowpass filters has a second order zero at the Nyquist frequency.

Solution. First, we need to determine a symmetric halfband $P(z)$ to use as the basis of the design. Since H_0 and G_0 must each have a second order zero at the Nyquist frequency, we need to select a $P(z)$ of the form

$$P(z) = (z+1)^2(1+z^{-1})^2Q(z), \tag{3.78}$$

where $Q(z)$ is to be determined. To obtain length four filters, $P(z)$ should have terms in $z^3, z^2, \ldots, z^{-3}$. Thus, $Q(z)$ must have terms in z^{-1}, z^0, z^1. Furthermore, since $P(z)$ must be symmetric, $Q(z)$ must also be symmetric. Thus, $Q(z)$ is of the form

$$Q(z) = a_1z + a_0 + a_1z^{-1}.$$

Now, we must solve for the coefficients a_0, a_1. Expanding $P(z)$ we have

$$\begin{aligned}P(z) &= z^{-2}(z+1)^4(a_1z+a_0+a_1z^{-1})\\ &= (z+1)^2(1+z^{-1})^2(a_1z+a_0+a_1z^{-1})\\ &= (z^2+4z+6+4z^{-1}+z^{-2})(a_1z+a_0+a_1z^{-1})\\ &= a_1z^3+(4a_1+a_0)z^2+(7a_1+4a_0)z+(8a_1+6a_0)+(7a_1+4a_0)z^{-1}+(4a_1+a_0)z^{-2}+a_1z^{-3}.\end{aligned}$$

Equating the coefficients of z^2 and z^{-2} with zero and the coefficient of z^0 with one, we have

$$4a_1+a_0=0 \quad \text{and} \quad 8a_1+6a_0=1.$$

Solving for a_0 and a_1, we obtain

$$a_0=\tfrac{1}{4} \quad \text{and} \quad a_1=-\tfrac{1}{16}.$$

Thus, we have

$$Q(z)=-\tfrac{1}{16}z+\tfrac{1}{4}-\tfrac{1}{16}z^{-1}. \tag{3.79}$$

Now that $Q(z)$ is known in (3.78), we need to factor the resulting $P(z)$. To do this, we need to factor $Q(z)$ in the form $Q(z)=C(z)C(z^{-1})$.

We want to express $Q(z)$ in the form

$$\begin{aligned}Q(z) &= (az^{-1}+b)(az+b)\\ &= (ab)z+(a^2+b^2)+(ab)z^{-1}.\end{aligned}$$

Comparing this expression for $Q(z)$ to (3.79), we have

$$ab=-\tfrac{1}{16} \quad \text{and} \quad a^2+b^2=\tfrac{1}{4}.$$

Now, we must solve for a and b. First, we solve for a. We have

$$\begin{aligned}& a^2+(-\tfrac{1}{16a})^2=\tfrac{1}{4}\\ \Rightarrow\quad & a^2+\tfrac{1}{256a^2}=\tfrac{1}{4}\\ \Rightarrow\quad & 256a^4+1=64a^2\\ \Rightarrow\quad & 256a^4-64a^2+1=0.\end{aligned}$$

Solving the quadratic equation in a^2 for a^2 yields

$$\begin{aligned}a^2 &= \tfrac{64\pm\sqrt{64^2-4(256)}}{2(256)}\\ &= \tfrac{64\pm32\sqrt{3}}{512}\\ &= \tfrac{2\pm\sqrt{3}}{16}.\end{aligned}$$

We arbitrarily choose the solution $a^2=\frac{2-\sqrt{3}}{16}$. Solving for a from a^2, we obtain

$$\begin{aligned}a &= \pm\sqrt{\tfrac{2-\sqrt{3}}{16}}\\ &= \pm\tfrac{1}{4}\sqrt{2-\sqrt{3}}\\ &= \pm\tfrac{1}{4\sqrt{2}}\sqrt{(\tfrac{1-\sqrt{3}}{2})^2}\\ &= \pm\left(\tfrac{1-\sqrt{3}}{4\sqrt{2}}\right).\end{aligned}$$

We arbitrarily choose the solution $a = \frac{1-\sqrt{3}}{4\sqrt{2}}$. Now, we solve for b. We have

$$\begin{aligned} b &= -\frac{1}{16\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right)} \\ &= -\frac{1}{2\sqrt{2}(1-\sqrt{3})} \\ &= -\frac{1+\sqrt{3}}{2\sqrt{2}(1-\sqrt{3})(1+\sqrt{3})} \\ &= \frac{1+\sqrt{3}}{4\sqrt{2}}. \end{aligned}$$

From above, we have that one solution for a and b is given by

$$a = \frac{1-\sqrt{3}}{4\sqrt{2}} \quad \text{and} \quad b = \frac{1+\sqrt{3}}{4\sqrt{2}}.$$

We obtain

$$\begin{aligned} Q(z) &= \left(\frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{1-\sqrt{3}}{4\sqrt{2}}z^{-1}\right)\left(\frac{1-\sqrt{3}}{4\sqrt{2}}z + \frac{1+\sqrt{3}}{4\sqrt{2}}\right) \\ &= \left(\frac{1}{4\sqrt{2}}\right)^2 \left(1+\sqrt{3}+(1-\sqrt{3})z^{-1}\right)\left((1-\sqrt{3})z+1+\sqrt{3}\right). \end{aligned}$$

Combining this result with (3.78), we have

$$P(z) = \left(\frac{1}{4\sqrt{2}}\right)^2 (z+1)^2(1+z^{-1})^2(1+\sqrt{3}+(1-\sqrt{3})z)(1+\sqrt{3}+(1-\sqrt{3})z^{-1}).$$

From this, we can trivially factor $P(z)$ in the form of $P(z) = G_0(z)G_0(z^{-1})$. This yields

$$\begin{aligned} G_0(z) &= \frac{1}{4\sqrt{2}}(1+z^{-1})^2(1+\sqrt{3}+(1-\sqrt{3})z^{-1}) \\ &= \frac{1}{4\sqrt{2}}\left((1+\sqrt{3})+(3+\sqrt{3})z^{-1}+(3-\sqrt{3})z^{-2}+(1-\sqrt{3})z^{-3}\right). \end{aligned}$$

We have that $H_0(z) = G_0(z^{-1})$. The remaining filters are easily deduced from $G_0(z)$ and $H_0(z)$. (As it turns out, this particular filter bank is quite a famous one. This filter bank is associated with one member of a family of orthonormal wavelet systems proposed by Daubechies.) □

In addition to the relatively simple filter-bank design technique introduced above, many other design methods have also been proposed. Many of these methods are based on optimization [3, 8, 9, 10, 15, 20, 24, 25, 26, 32].

3.6 Implementation of UMD Filter Banks

In principle, the design of a PR UMD filter bank amounts to decomposing the identity matrix into two factors with desired properties. These two factors are simply the polyphase matrices of the filter bank. Once we have determined the analysis and synthesis polyphase matrices $\boldsymbol{H}_\mathsf{p}(z)$ and $\boldsymbol{G}_\mathsf{p}(z)$, respectively, we are ready to proceed to the implementation of the filter bank. Of course, the filter bank could be realized by directly implementing the filtering operations in each of the polyphase matrices, but it is often beneficial to break the filtering process into a number of smaller and simpler cascaded stages.

Consider for a moment the analysis side of the filter bank. Instead of implementing $\boldsymbol{H}_\mathsf{p}(z)$ directly, we further decompose $\boldsymbol{H}_\mathsf{p}(z)$ as follows

$$\boldsymbol{H}_\mathsf{p}(z) = \boldsymbol{E}_{n-1}(z)\cdots\boldsymbol{E}_1(z)\boldsymbol{E}_0(z).$$

Each of the $\{\boldsymbol{E}_i(z)\}$ can then be taken to represent a single filtering stage in the final implementation as depicted in Figure 3.40. Similarly, $\boldsymbol{G}_\mathsf{p}(z)$ can be decomposed to produce

$$\boldsymbol{G}_\mathsf{p}(z) = \boldsymbol{R}_{m-1}(z)\cdots\boldsymbol{R}_1(z)\boldsymbol{R}_0(z).$$

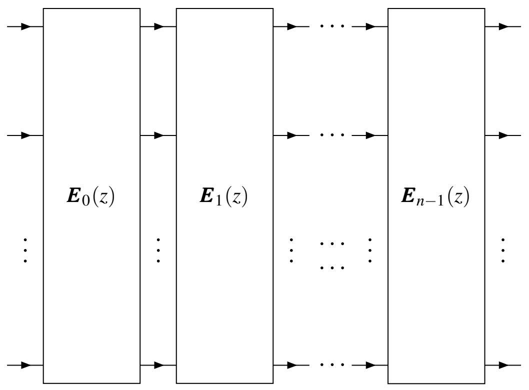


Figure 3.40: Block cascade realization of the analysis polyphase matrix.

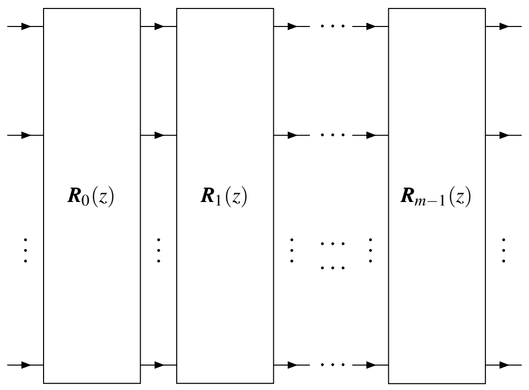


Figure 3.41: Block cascade realization of the synthesis polyphase matrix.

This corresponds to the cascade realization of the synthesis polyphase matrix shown in Figure 3.41. In the event that $\boldsymbol{G}_\mathsf{p}(z)\boldsymbol{H}_\mathsf{p}(z) = \boldsymbol{I}$, we have $\boldsymbol{G}_\mathsf{p}(z) = \boldsymbol{H}_\mathsf{p}^{-1}(z)$. Thus, we could choose $\boldsymbol{G}_\mathsf{p}(z)$ as

$$\boldsymbol{G}_\mathsf{p}(z) = \boldsymbol{E}_0^{-1}(z)\boldsymbol{E}_1^{-1}(z)\cdots\boldsymbol{E}_{n-1}^{-1}(z).$$

This factorization results in a certain symmetry between the analysis and synthesis sides of the filter bank which can often be advantageous.

Assuming that we want to realize the polyphase matrices with a number of cascaded blocks, one might wonder what type of polyphase matrix factorization should be employed. In the sections that follow, we will introduce a few possibilities.

3.6.1 Lattice Realization of Paraunitary PR UMD Filter Banks

In the case of paraunitary (i.e., orthonormal) filter banks, one very popular implementation strategy is the lattice realization. The lattice realization is a polyphase scheme based on a so called lattice factorization of the polyphase matrices. In a lattice factorization, the matrix factors correspond to rotations (up to scale factor) and delays. To avoid unnecessary complexity in what follows, we consider the case of a two-channel filter bank and $(1,2)$-type polyphase representation.

A lattice realization is parameterized by set of $J+1$ rotation angles $\{\theta_k\}_{k=0}^{J}$. Consider a rotation matrix $\boldsymbol{R}$ associated with the rotation angle $\theta \in [-\pi,\pi)$. We have

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

The matrix $\boldsymbol{R}$ can be rewritten as

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{cases} \cos\theta \begin{bmatrix} 1 & \tan\theta \\ -\tan\theta & 1 \end{bmatrix} & \theta \notin \{-\frac{\pi}{2},\frac{\pi}{2}\} \\ \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & \theta = \frac{\pi}{2} \\ \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} & \theta = -\frac{\pi}{2}. \end{cases}$$

By expressing $\boldsymbol{R}$ in this manner, the number of elements in the matrix part of the expression that are not trivial multipliers (i.e., not zero or plus/minus one) can be reduced to at most two. This transformation is exploited in the lattice realization in order to reduce the number of multiplication operations needed.

With a lattice realization, the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ and synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ are factored as

$$\boldsymbol{H}_{\mathsf{p}}(z) = S\boldsymbol{R}_J\boldsymbol{\Lambda}(z)\boldsymbol{R}_{J-1}\boldsymbol{\Lambda}(z)\cdots\boldsymbol{R}_1\boldsymbol{\Lambda}(z)\boldsymbol{R}_0\boldsymbol{\Lambda}(\pm 1) \quad \text{and} \tag{3.80a}$$

$$\boldsymbol{G}_{\mathsf{p}}(z) = S\boldsymbol{\Lambda}(\pm 1)\boldsymbol{R}_0^T\boldsymbol{\Gamma}(z)\boldsymbol{R}_1^T\cdots\boldsymbol{R}_{J-1}^T\boldsymbol{\Gamma}(z)\boldsymbol{R}_J^T, \tag{3.80b}$$

where

$$\boldsymbol{\Lambda}(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}, \quad \boldsymbol{\Gamma}(z) = \begin{bmatrix} z^{-1} & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{R}_k = \begin{cases} \begin{bmatrix} 1 & \alpha_k \\ -\alpha_k & 1 \end{bmatrix} & \theta_k \notin \{-\frac{\pi}{2}, \frac{\pi}{2}\} \\ \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} & \theta_k = -\frac{\pi}{2} \\ \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & \theta_k = \frac{\pi}{2}, \end{cases} \quad \beta_k = \begin{cases} \cos\theta_k & \theta_k \notin \{-\frac{\pi}{2}, \frac{\pi}{2}\} \\ 1 & \theta_k \in \{-\frac{\pi}{2}, \frac{\pi}{2}\}, \end{cases}$$

$$S = \prod_{k=0}^{J} \beta_k, \quad \text{and} \quad \alpha_k = \tan\theta_k.$$

The matrix decompositions of $\boldsymbol{H}_{\mathsf{p}}(z)$ and $\boldsymbol{G}_{\mathsf{p}}(z)$ in (3.80) are called **lattice factorizations**. Note that if $\theta_k \notin \{-\frac{\pi}{2}, \frac{\pi}{2}\}$, then

$$\det \boldsymbol{R}_k = 1 + \alpha_k^2 \quad \text{and} \quad \boldsymbol{R}_k^{-1} = \tfrac{1}{1+\alpha_k^2}\boldsymbol{R}_k^T = \tfrac{1}{1+\alpha_k^2}\begin{bmatrix} 1 & -\alpha_k \\ \alpha_k & 1 \end{bmatrix};$$

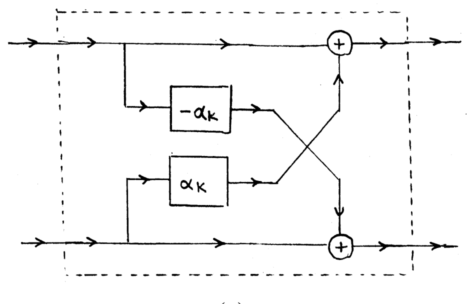
and if $\theta_k \in \{-\frac{\pi}{2}, \frac{\pi}{2}\}$, then

$$\det \boldsymbol{R}_k = 1 \quad \text{and} \quad \boldsymbol{R}_k^{-1} = \boldsymbol{R}_k^T.$$

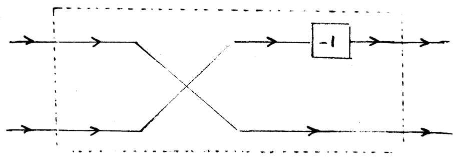
Examining the various matrix factors, we see that $\boldsymbol{R}_k$ is a rotation matrix (up to scale factor), and $\boldsymbol{\Lambda}(z)$ and $\boldsymbol{\Gamma}(z)$ are delay matrices. The matrix factors $\boldsymbol{R}_k$, $\boldsymbol{\Lambda}(z)$, and $\boldsymbol{\Gamma}(z)$ correspond to the computational structures shown in Figures 3.42 and 3.43. The lattice realization has the general structure shown in Figure 3.44. Each of the analysis and synthesis sides consist of $J+1$ rotations separated by J delays. Observe that each stage on the synthesis side essentially undoes the effect of the corresponding stage on the analysis side. Therefore, it should not be surprising that the PR property is achieved.

The lattice realization provides a complete parameterization of two-channel paraunitary filter banks. That is, any 2×2 real coefficient causal FIR paraunitary matrix admits a lattice factorization. For a proof, see [31, Section 14.3, pp. 727–731] or [29, Theorem 4.7, pp. 139–140]. In addition to completeness, the lattice realization has a number of other desirable characteristics:

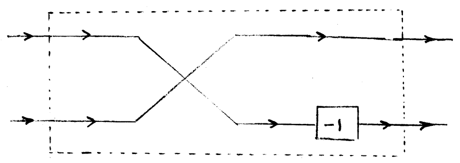
- It structurally imposes the paraunitary and PR properties (i.e., the properties hold regardless of the choice of the parameters $\{\theta_k\}_{k=0}^{J}$).
- It has a hierarchical structure in the sense that, if the lattice structure is truncated after only N stages, the resulting filter bank will still have the paraunitary and PR properties.
- It is robust to coefficient quantization error (i.e., the paraunitary and PR properties will still hold even if the $\{\alpha_k\}$ are quantized).
- It is such that the analysis and synthesis sides have high degree of symmetry (i.e., have very similar computational structures).
- It has the lowest implementation complexity amongst all known structures for paraunitary filter banks.



(a)

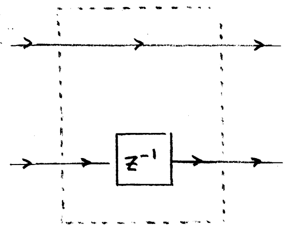


(b)

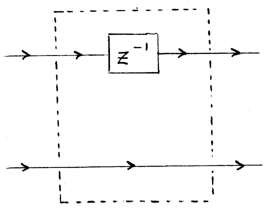


(c)

Figure 3.42: Computational structures associated with the matrix factor $\boldsymbol{R}_k$ for (a) $\theta \notin \{-\frac{\pi}{2}, \frac{\pi}{2}\}$, (b) $\theta = -\frac{\pi}{2}$, and (c) $\theta = \frac{\pi}{2}$.

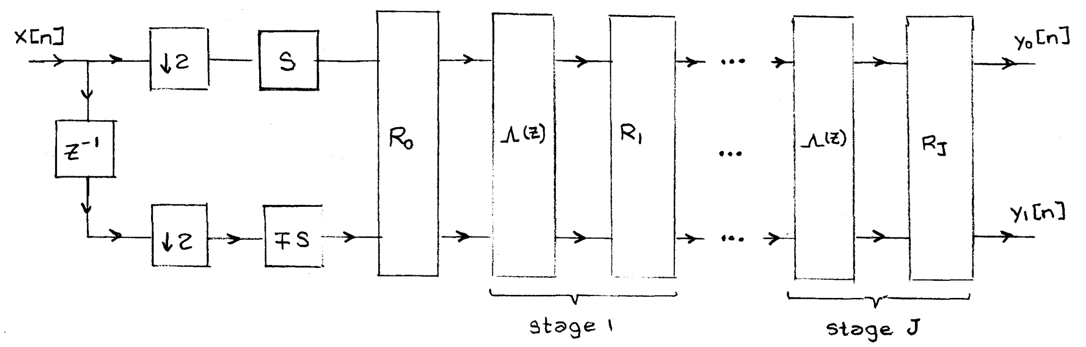


(a)

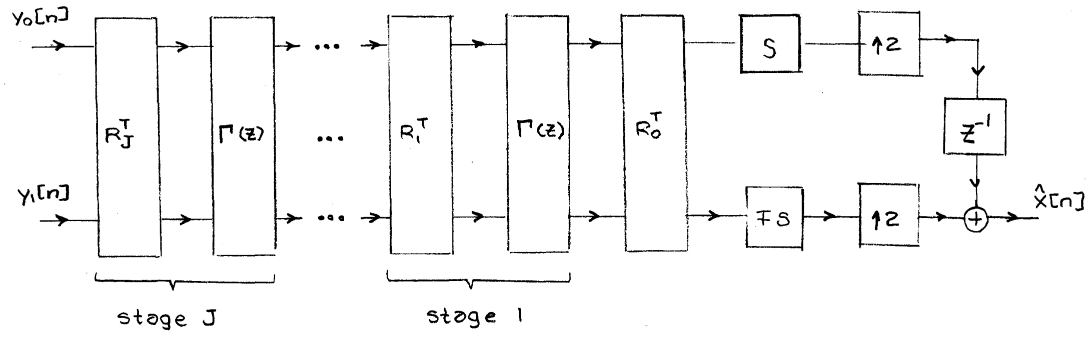


(b)

Figure 3.43: Computational structures associated with the matrix factors (a) $\boldsymbol{\Lambda}(z)$ and (b) $\boldsymbol{\Gamma}(z)$.



(a)



(b)

Figure 3.44: General structure of the lattice realization of a paraunitary filter bank. (a) Analysis and (b) synthesis sides.

In order to construct a lattice realization of a filter bank, we need to determine the lattice coefficients $\{\alpha_k\}$ associated with given analysis filters $H_0(z)$ and $H_1(z)$. For simplicity, in what follows, we assume that $\theta_k \notin \{-\frac{\pi}{2}, \frac{\pi}{2}\}$. Let $H_0^{(m)}(z)$ and $H_1^{(m)}(z)$ denote the transfer functions of the analysis filters obtained from the m-stage lattice. So, we have

$$\begin{aligned} H_0^{(m)}(z) &= H_0^{(m-1)}(z) + \alpha_m z^{-2} H_1^{(m-1)}(z) \quad \text{and} \\ H_1^{(m)}(z) &= -\alpha_m H_0^{(m-1)}(z) + z^{-2} H_1^{(m-1)}(z). \end{aligned}$$

We can then determine the coefficient α_m by inverting the preceding recursion to obtain

$$\begin{aligned} H_0^{(m-1)}(z) &= \tfrac{1}{1+\alpha_m^2}\left(H_0^{(m)}(z) - \alpha_m H_1^{(m)}(z)\right) \quad \text{and} \\ H_1^{(m-1)}(z) &= \tfrac{1}{1+\alpha_m^2} z^2 \left(\alpha_m H_0^{(m)}(z) + H_1^{(m)}(z)\right). \end{aligned}$$

To find the lattice coefficients $\{a_k\}$, we initially set $H_0^{(J)}(z) = H_0(z)$ and $H_1^{(J)}(z) = H_1(z)$; then, we iterate. In each step, we choose α_m so that the highest power of z^{-1} in $H_0^{(m)}(z) - \alpha_m H_1^{(m)}(z)$ is cancelled.

Although, in our discussion above, we have exclusively considered the case of two-channel filter banks, the lattice realization can be generalized to the M-channel case (where $M > 2$). For additional information on lattice realizations of paraunitary filter banks (including the generalization to the M-channel case), the reader is referred to [31, Sections 6.4 and 6.5, pp. 302–322] and [31, Sections 14.3 and 14.4, pp. 727–740].

3.6.2 Lattice Realization of Linear-Phase PR UMD Filter Banks

A lattice structure can also be used for realizing linear-phase filter banks. In this case, the lattice realization is parameterized by a set of $J+1$ lattice coefficients $\{\alpha_k\}_{k=0}^{J}$. The analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ and synthesis polyphase matrix $\boldsymbol{G}_\mathsf{p}(z)$ are factored as

$$\begin{aligned} \boldsymbol{H}_\mathsf{p}(z) &= \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right] \boldsymbol{T}_J \boldsymbol{\Lambda}(z) \cdots \boldsymbol{\Lambda}(z) \boldsymbol{T}_1 \boldsymbol{\Lambda}(z) \boldsymbol{T}_0 \quad \text{and} \\ \boldsymbol{G}_\mathsf{p}(z) &= \tfrac{1}{\beta} \boldsymbol{S}_0 \boldsymbol{\Gamma}(z) \boldsymbol{S}_1 \boldsymbol{\Gamma}(z) \cdots \boldsymbol{\Gamma}(z) \boldsymbol{S}_J \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right], \end{aligned}$$

where

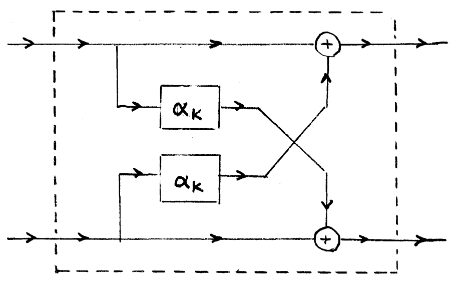
$$\boldsymbol{T}_k = \begin{bmatrix} 1 & \alpha_k \\ \alpha_k & 1 \end{bmatrix}, \quad \boldsymbol{S}_k = \begin{bmatrix} 1 & -\alpha_k \\ -\alpha_k & 1 \end{bmatrix}, \quad \boldsymbol{\Lambda}(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}, \quad \boldsymbol{\Gamma}(z) = \begin{bmatrix} z^{-1} & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and}$$
$$\beta = \prod_{k=0}^{J} (1-\alpha_k^2).$$

Note that $\boldsymbol{S}_k = (1-\alpha_k^2)\boldsymbol{T}_k^{-1}$ (i.e., $\boldsymbol{S}_k$ is the inverse of $\boldsymbol{T}_k$ up to a scale factor). The filtering networks associated with the matrix factors $\boldsymbol{T}_k$ and $\boldsymbol{S}_k$ are depicted in Figure 3.45. The general structure of the lattice realization is shown in Figure 3.46.

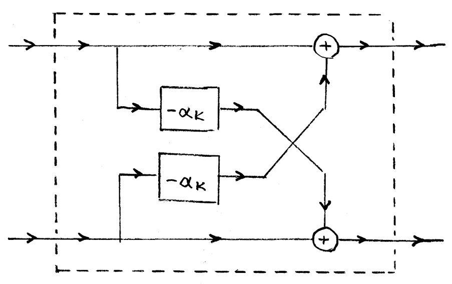
As it turns out, every linear-phase PR filter bank with filters of equal and even length has a lattice factorization. (See [29, Theorem 4.8, p. 140].) Unfortunately, not all linear-phase PR filter banks have filters of equal and even length. Therefore, not all linear-phase PR filter banks can be realized using the lattice structure from above. This said, however, the lattice realization is still quite useful. It structurally imposes the PR and linear-phase properties and has other advantages similar to the case of the paraunitary lattice structure. For additional information on lattice realizations for linear-phase filter banks, the reader is referred to [31, Chapter 7, pp. 337–352].

3.6.3 Lifting Realization of PR UMD Filter Banks

Suppose we have an M-channel PR UMD filter bank with FIR filters. Denote the transfer functions of the analysis and synthesis filters as $H_k(z)$ and $G_k(z)$, respectively. Further, assume that the system is represented in either type

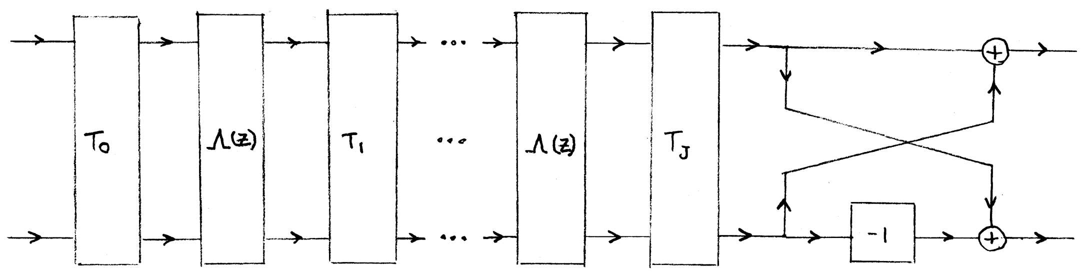


(a)

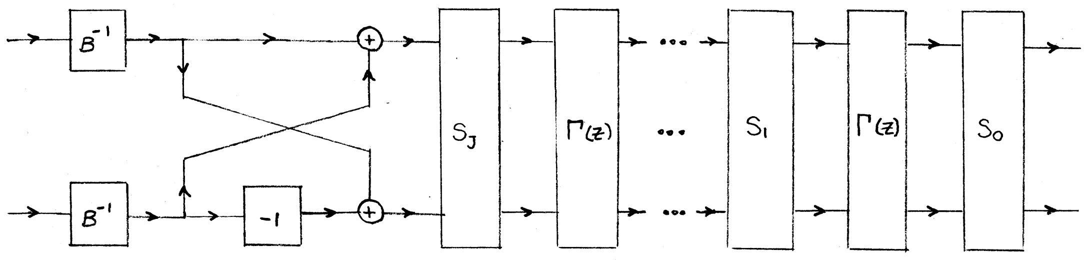


(b)

Figure 3.45: Polyphase filtering networks associated with the matrices (a) $\boldsymbol{T}_k$ and (b) $\boldsymbol{S}_k$.



(a)



(b)

Figure 3.46: General structure of the lattice realization for a linear-phase filter bank. (a) Analysis and (b) synthesis side.

$(1,2)$ or type $(3,1)$ polyphase form with analysis and synthesis polyphase matrices $\boldsymbol{H}_{\mathsf{p}}(z)$ and $\boldsymbol{G}_{\mathsf{p}}(z)$, respectively. In a lifting realization, a UMD filter bank is implemented in its polyphase form. The distinguishing characteristic of this realization is the type of network used to implement the polyphase filtering.

Fundamental to the idea of the lifting realization is the lifting factorization which decomposes a matrix into two types of factors:

1. Scaling (type S). This type of $M \times M$ matrix has all of the elements on the main diagonal equal to one except the (k,k) entry which is α and all off-main-diagonal elements equal to zero. Such a matrix is completely characterized by the pair (k,α) and is denoted $\mathcal{S}_M(k,\alpha)$. In cases where the size of the matrix is clear from the context, the subscript M is omitted.

2. Adding (type A). This type of $M \times M$ matrix has all ones on the main diagonal, α at position (k,l), and all other entries zero. Such a matrix is completely characterized by the triple (k,l,α) and is denoted $\mathcal{A}_M(k,l,\alpha)$. Again, the subscript M may be omitted in cases where the size of the matrix is clear from the context.

More specifically, the lifting factorization decomposes a matrix into zero or more constant type S factors that either premultiply or postmultiply zero or more type A factors.

In a lifting realization, the polyphase filtering is implemented directly from lifting factorization of the analysis and synthesis polyphase matrices. More specifically, the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ is decomposed using a lifting factorization as

$$\boldsymbol{H}_{\mathsf{p}}(z) = \boldsymbol{S}_{\sigma-1} \cdots \boldsymbol{S}_1 \boldsymbol{S}_0 \boldsymbol{A}_{\lambda-1}(z) \cdots \boldsymbol{A}_1(z) \boldsymbol{A}_0(z) \tag{3.81}$$

where the $\boldsymbol{S}_k$ are type S matrices with all constant entries and the $\boldsymbol{A}_k(z)$ are type A elementary matrices (which can depend on z). In a lifting realization, the decomposition of the synthesis polyphase matrix $\boldsymbol{G}_{\mathsf{p}}(z)$ is completely determined by the decomposition used for the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ and is given by

$$\boldsymbol{G}_{\mathsf{p}}(z) = \boldsymbol{A}_0^{-1}(z) \boldsymbol{A}_1^{-1}(z) \cdots \boldsymbol{A}_{\lambda-1}^{-1}(z) \boldsymbol{S}_0^{-1} \boldsymbol{S}_1^{-1} \cdots \boldsymbol{S}_{\sigma-1}^{-1} \tag{3.82}$$

Furthermore, this choice of decomposition for $\boldsymbol{G}_{\mathsf{p}}(z)$ implies that $\boldsymbol{G}_{\mathsf{p}}(z) = \boldsymbol{H}_{\mathsf{p}}^{-1}(z)$. Note that $\mathcal{A}^{-1}(k,l,\alpha) = \mathcal{A}(k,l,-\alpha)$ and $\mathcal{S}^{-1}(k,\alpha) = \mathcal{S}(k,\alpha^{-1})$. That is, the inverse of a type A matrix is another type A matrix, and the inverse of a type S matrix is another type S matrix. Thus, in equation (3.82) the $\boldsymbol{A}_k^{-1}(z)$ are type A matrices and the $\boldsymbol{S}_k^{-1}$ are type S matrices. Hence, the decomposition given for $\boldsymbol{G}_{\mathsf{p}}(z)$ is, in fact, a lifting factorization of the matrix. Moreover, each pair of corresponding matrices $\boldsymbol{A}_k(z)$ and $\boldsymbol{A}_k^{-1}(z)$ are identical except for one element which differs only in sign. Similarly, each pair of corresponding matrices $\boldsymbol{S}_k$ and $\boldsymbol{S}_k^{-1}$ are identical except for one element which differ in a reciprocal relationship.

From the definition of the lifting realization, it follows that such a realization exists if and only if a lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ exists and $\boldsymbol{G}_{\mathsf{p}}(z) = \boldsymbol{H}_{\mathsf{p}}^{-1}(z)$. We shall revisit this issue later, but for the time being it suffices to say that any UMD filter bank can be made to satisfy these conditions through an appropriate normalization of its analysis and synthesis filters.

The decompositions in (3.81) and (3.82) correspond to block cascade realizations of the analysis and synthesis filtering, respectively. Since the UMD filter bank has M channels, each filtering block has M inputs and M outputs and is characterized by an $M \times M$ matrix. Each of the type A factors corresponds to a block that adds a filtered version of a signal in one channel to a signal in another channel. This corresponds to a single ladder step in a ladder network as depicted in Figure 3.47(a). To simplify the diagram only the kth and lth inputs and outputs are shown. All other inputs pass directly through to their corresponding outputs unchanged. Clearly, the inverse of this network is another network of the same form, namely the one shown in Figure 3.47(b). Each of the type S factors corresponds to a block that scales the signal in a single channel. Such a scaling unit is shown in Figure 3.48(a). Only the kth input and output are shown as all other inputs pass directly through to their corresponding outputs without modification. The inverse of this network is another network of the same form, namely the one shown in Figure 3.48(b). Since the lifting factorization consists of only type A and type S factors, this decomposition yields a ladder structure with some additional scaling elements. The ladder structure is followed by a scaling on the analysis side and preceded by a scaling on the synthesis side of the filter bank. Due to the similarities between the factors in both of these decompositions, the structures used to perform the analysis and synthesis filtering possess a certain degree of symmetry. This symmetry has important consequences as will become evident later.

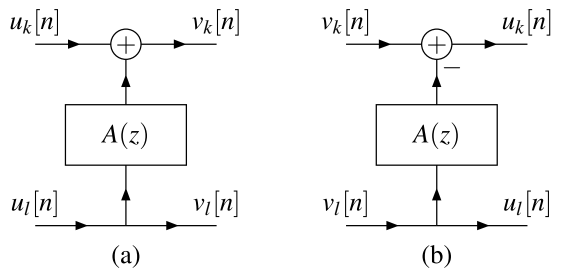


Figure 3.47: Lifting step. (a) A lifting step and (b) its inverse.

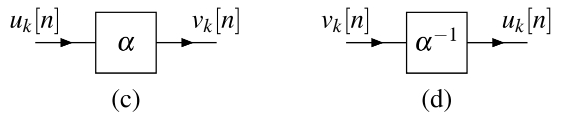


Figure 3.48: Scaling step. (a) A scaling step and (b) its inverse.

The resulting structure for the general two-channel case is shown in Figure 3.49 (where some of the $A_i(z)$ may be identically zero). Notice the symmetry between the forward and inverse transform structures. By inspection, it is obvious that the filter bank has PR. The synthesis side simply undoes step-by-step each operation performed on the analysis side.

In the M-channel case, the lifting realization is completely analogous to the 2-band case, and has the general form shown in Figure 3.50. The $\boldsymbol{S}_k$ and $\boldsymbol{S}_k^{-1}$ blocks corresponds to scaling operations. The $\boldsymbol{A}_k(z)$ and $\boldsymbol{A}_k^{-1}(z)$ blocks correspond to ladder steps. Moreover, the scaling and ladder steps on the analysis side are simply the inverses of the scaling and ladder steps on the synthesis side.

The lifting realization is a particularly interesting one, primarily because of its use of ladder networks. Such networks have the remarkable property that they can be made to maintain their invertibility even in the presence of certain types of quantization error, particularly the rounding error introduced by finite-precision arithmetic. To see why this is so, consider the two ladder steps shown in Figure 3.47. Clearly, if exact arithmetic is employed, these two networks invert one another. Suppose now that the filtering operations (associated with $A(z)$) are implemented using finite-precision arithmetic and some roundoff error is incurred. When we cascade these two networks together, both ladder-step filters are presented with the same input. Since the two filters are identical (and assumed to use the same implementation strategy), they will both incur the same rounding error, and their outputs will be identical. Therefore, whatever value is added by the adder in the first network will be subtracted by the adder in the second network. Consequently, the two networks continue to invert one another, even in the presence of rounding error. It is not difficult to see that we can apply this argument repeatedly in order to show that a ladder network with any number of ladder steps can maintain its invertibility even in the presence of rounding error. In other words, such networks are fundamentally reversible in nature.

3.6.3.1 Existence of Lifting Realization

Having decided to use a lifting realization, we must now concern ourselves with the existence of such a realization. That is, we would like to know if any UMD filter bank can be implemented using lifting. In order to determine this, we must revisit the conditions for existence stated earlier. That is, a UMD filter bank can be realized using lifting if and only if the following conditions are satisfied:

$$\text{a lifting factorization of } \boldsymbol{H}_\mathsf{p}(z) \text{ exists} \quad \text{and} \tag{3.83}$$

$$\boldsymbol{G}_\mathsf{p}(z) = \boldsymbol{H}_\mathsf{p}^{-1}(z). \tag{3.84}$$

Assuming that the realization exists, its uniqueness is also of concern.

First, let us consider condition (3.83). We would like to know what conditions must hold in order for the analysis

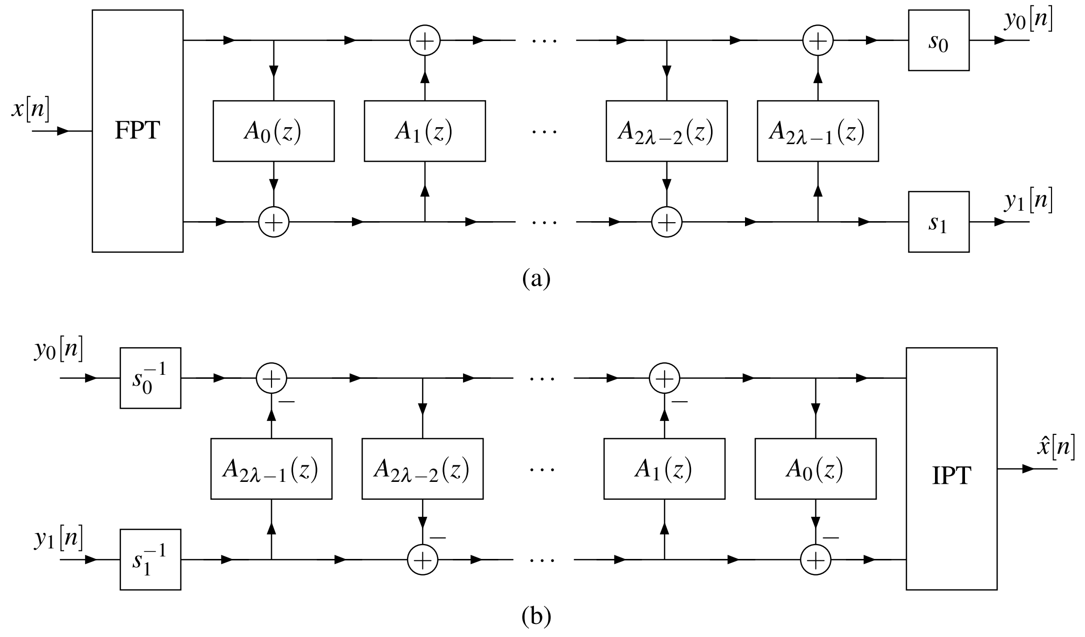


Figure 3.49: Lifting realization of a two-channel UMD filter bank. (a) Analysis side. (b) Synthesis side.

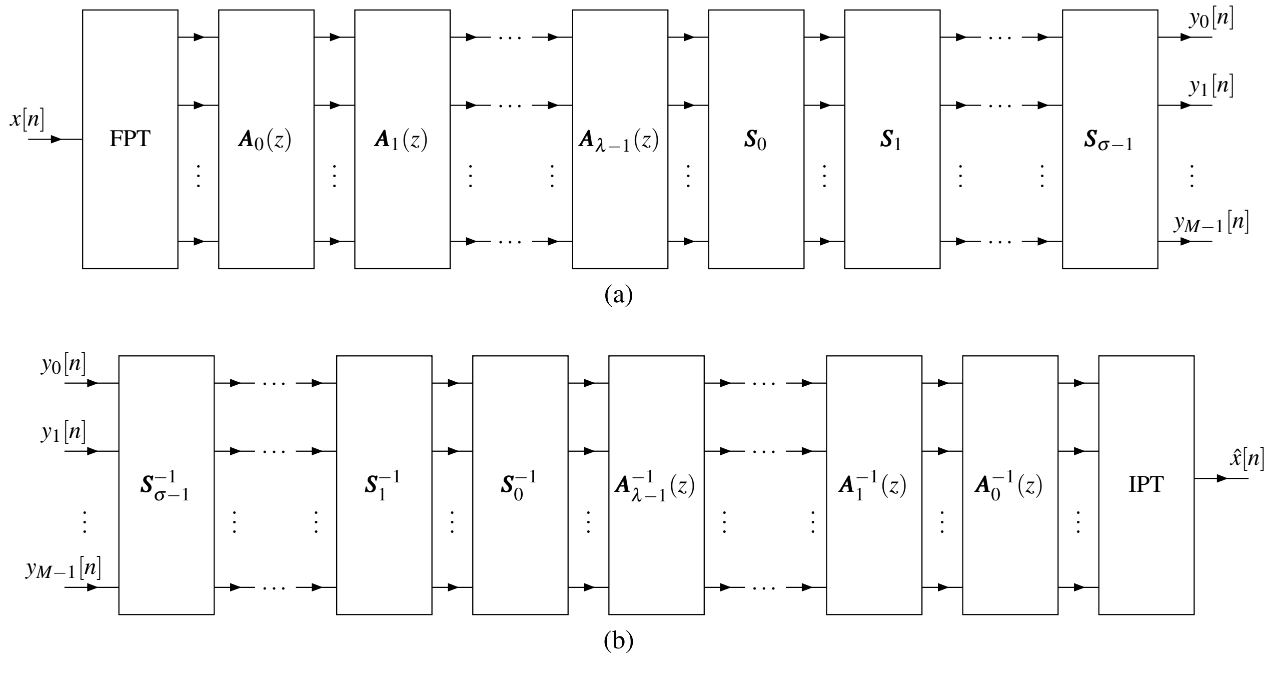


Figure 3.50: General M-band lifting structure. (a) Analysis side (forward transform). (b) Synthesis side (inverse transform).

polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ to have a lifting factorization. To help us answer this question, we rely on the following theorem:

Theorem 3.18. *Any $M \times M$ Laurent polynomial matrix $\boldsymbol{U}(z)$ admits a factorization of the form*

$$\boldsymbol{U}(z) = \boldsymbol{S}_0(z)\boldsymbol{S}_1(z)\cdots\boldsymbol{S}_{Q-1}(z)\boldsymbol{A}_0(z)\boldsymbol{A}_1(z)\cdots\boldsymbol{A}_{P-1}(z)$$

where the $\boldsymbol{S}_k(z)$ are type S elementary matrices and the $\boldsymbol{A}_k(z)$ are type A elementary matrices and all factors are Laurent polynomial matrices. For any particular set of $\boldsymbol{S}_k(z)$, a set of $\boldsymbol{A}_k(z)$ can be found to complete the factorization if and only if

$$\prod_{k=0}^{Q-1} \det \boldsymbol{S}_k(z) = \det \boldsymbol{U}(z)$$

When the factorization can be completed, the choice of $\boldsymbol{A}_k(z)$ is not unique.

Proof. See [1]. □

Recall, the form of the lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ is given by equation (3.81). Clearly, this factorization is a special case of the decomposition considered in Theorem 3.18. That is, the lifting factorization is a special case where the type S matrices are constrained to be constant (i.e., independent of z). If in the theorem, we assume that the $\boldsymbol{S}_k$ are constant, this implies that $\det \boldsymbol{U}(z)$ must also be constant in order for the decomposition to exist. Relating this back to the original problem, a lifting factorization of the analysis polyphase matrix exists if and only if $\det \boldsymbol{H}_{\mathsf{p}}(z)$ is a nonzero constant. Moreover, the theorem also tells us that the factorization is never unique when it exists. Since the lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ determines the structure of the lifting realization, the nonuniqueness of the factorization also implies the nonuniqueness of the lifting realization structure.

For a lifting realization to exist, we therefore require

$$\det \boldsymbol{H}_{\mathsf{p}}(z) = \alpha \quad \text{and} \tag{3.85}$$

$$\boldsymbol{G}_{\mathsf{p}}(z) = \boldsymbol{H}_{\mathsf{p}}^{-1}(z) \tag{3.86}$$

where α is a nonzero constant. Since we are considering FIR PR UMD filter banks, the analysis and synthesis polyphase matrices must satisfy the conditions

$$\det \boldsymbol{H}_{\mathsf{p}}(z) = \alpha z^{-K_0} \quad \text{and} \tag{3.87}$$

$$\boldsymbol{G}_{\mathsf{p}}(z) = \begin{cases} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \boldsymbol{0} \end{bmatrix}^{K_1} \boldsymbol{H}_{\mathsf{p}}^{-1}(z) & \text{type } (1,2) \\ \begin{bmatrix} \boldsymbol{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \boldsymbol{0} \end{bmatrix}^{K_1} \boldsymbol{H}_{\mathsf{p}}^{-1}(z) & \text{type } (3,1) \end{cases} \tag{3.88}$$

where α is a nonzero constant and the K_k are integers. From (3.87) and (3.88), we can see that conditions (3.85) and (3.86) are not generally satisfied. In order for the conditions to be satisfied, we must have $K_0 = K_1 = 0$. Fortunately, it is always possible to normalize the analysis and synthesis filters so that these constraints are met. To show this, we rely on the following theorem:

Theorem 3.19. *Suppose we are given an M-channel PR UMD filter bank with FIR filters. Denote the transfer functions of the analysis and synthesis filters of the UMD filter bank as $H_k(z)$ and $G_k(z)$, respectively. Assume the system is represented in either type $(1,2)$ or type $(3,1)$ polyphase form and has analysis and synthesis polyphase matrices $\boldsymbol{H}_{\mathsf{p}}(z)$ and $\boldsymbol{G}_{\mathsf{p}}(z)$, respectively. Since the system is PR and has FIR filters, we have*

$$\det \boldsymbol{H}_{\mathsf{p}}(z) = \alpha_0 z^{-K_0}$$

and

$$\boldsymbol{G}_{\mathsf{p}}(z) = \begin{cases} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \boldsymbol{0} \end{bmatrix}^{K_1} \boldsymbol{H}_{\mathsf{p}}^{-1}(z) & \textit{type } (1,2) \\ \begin{bmatrix} \boldsymbol{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \boldsymbol{0} \end{bmatrix}^{K_1} \boldsymbol{H}_{\mathsf{p}}^{-1}(z) & \textit{type } (3,1) \end{cases}$$

where α_0 is a nonzero constant, and $K_0, K_1 \in \mathbb{Z}$.

Suppose we normalize the analysis and synthesis filters to obtain a new set of analysis and synthesis filters with transfer functions $H'_k(z)$ and $G'_k(z)$, respectively, where this normalization is defined as follows:

$$H'_k(z) = \beta_0 z^{-L_0} H_k(z) \quad \textit{and}$$
$$G'_k(z) = \beta_1 z^{-L_1} G_k(z).$$

Denote the analysis and synthesis polyphase matrices associated with the new filters as $\boldsymbol{H}'_{\mathsf{p}}(z)$ and $\boldsymbol{G}'_{\mathsf{p}}(z)$, respectively. Given the above normalization, the following assertions are true:

$$\det \boldsymbol{H}'_{\mathsf{p}}(z) = \alpha_0 \beta_0^M (-1)^{L_0(M-1)} z^{L_0+K_0} \quad \textit{and}$$
$$\boldsymbol{G}'_{\mathsf{p}}(z) = \begin{cases} \beta_0\beta_1 \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{M-1} \\ z^{-1} & \mathbf{0} \end{bmatrix}^{L_0+L_1+K_1} (\boldsymbol{H}'_{\mathsf{p}}(z))^{-1} & \textit{type } (1,2) \\ \beta_0\beta_1 \begin{bmatrix} \mathbf{0} & z^{-1} \\ \boldsymbol{I}_{M-1} & \mathbf{0} \end{bmatrix}^{L_0+L_1+K_1} (\boldsymbol{H}'_{\mathsf{p}}(z))^{-1} & \textit{type } (3,1). \end{cases}$$

This implies that with an appropriate normalization, we can force $\det \boldsymbol{H}'_{\mathsf{p}}(z)$ to be any arbitrary nonzero constant up to sign while simultaneously satisfying the constraint $\boldsymbol{G}'_{\mathsf{p}}(z) = [\boldsymbol{H}'_{\mathsf{p}}(z)]^{-1}$.

Proof. See [1]. □

Let us now normalize the filters of the system to obtain new analysis and synthesis filters with transfer functions $H'_k(z)$ and $G'_k(z)$, respectively, as specified by

$$H'_k(z) = \beta_0 z^{K_0} H_k(z) \quad \text{and} \tag{3.89}$$
$$G'_k(z) = \beta_1 z^{K_1-K_0} G_k(z) \tag{3.90}$$

where $\beta_0 = 1/\beta_1$. Denote the new analysis and synthesis polyphase matrices obtained from this normalization as $\boldsymbol{H}'_{\mathsf{p}}(z)$ and $\boldsymbol{G}'_{\mathsf{p}}(z)$, respectively. From the above theorem, this normalization will always yield a new system satisfying

$$\det \boldsymbol{H}'_{\mathsf{p}}(z) = \alpha_0 \beta_0^M (-1)^{K_0(M-1)} \quad \text{and} \tag{3.91}$$
$$\boldsymbol{G}'_{\mathsf{p}}(z) = [\boldsymbol{H}'_{\mathsf{p}}(z)]^{-1}. \tag{3.92}$$

Clearly, this new system satisfies conditions (3.85) and (3.86). Therefore, a lifting realization of the new system must always exist. Moreover, we can see from above that it is always possible to further force $\det \boldsymbol{H}'_{\mathsf{p}}(z) \in \{-1, 1\}$ by choosing $\beta_0 = |\alpha|^{-1/M}$. If we can always force $\det \boldsymbol{H}'_{\mathsf{p}}(z) \in \{-1, 1\}$, this implies we can always choose the scaling factors on the analysis side in the lifting realization to be ± 1. This turns out to be useful in some applications (e.g., constructing reversible transforms).

It is important to note that the normalization process does not affect the fundamental behavior of the UMD filter bank. The relative magnitude and phase responses of the analysis filters remain unchanged since all have an equal gain and equal phase shift added. The same statement also holds true for the synthesis filters. The PR property is not affected although the reconstruction delay will generally change as the normalization process forces a particular reconstruction delay.

The normalization suggested above is not the only one possible. Another possibility is suggested by Daubechies and Sweldens in [13]. In this case, they propose changing the phase delay and gain of a single analysis filter. The author felt that in some cases it might be more desirable to distribute the phase delay and gain change equally across all of the analysis filters—hence, the reason for proposing the above normalization scheme.

By using the lifting realization of a UMD filter bank, we obtain a transform with several desirable properties:

- The transform can be calculated in place without the need for auxiliary memory.
- Even in the presence of coefficient quantization error or roundoff error (for the ladder step filters), the PR property is retained.

- The inverse transform has the same computational complexity as the forward transform.
- Asymptotically, for long filters, the lifting realization yields a more computationally efficient structure than the standard realization [13].
- The lifting realization can be used to easily construct reversible transforms.
- The PR property is structurally imposed, regardless of the choice of lifting filters.

3.6.3.2 Lifting Factorization Algorithm

Lifting factorizations can be computed using a matrix Euclidean algorithm. To compute the lifting factorization of the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$, we proceed as follows:

1. Assume that the filter bank has been normalized (as described previously) so that $\det \boldsymbol{H}_{\mathsf{p}}(z)$ is a nonzero constant.
2. Choose any $\boldsymbol{S}_k$ for $k = 0, 1, \ldots, \sigma - 1$ from $\mathcal{S}(\cdot)$ that satisfy
$$\prod_{k=0}^{\sigma-1} \det \boldsymbol{S}_k = \det \boldsymbol{H}_{\mathsf{p}}(z).$$
For example, if $\det \boldsymbol{H}_{\mathsf{p}}(z) = d$, we can simply select $\sigma = 1$ and $\boldsymbol{S}_0 = \mathcal{S}(0, d)$.
3. Calculate the quantity $\boldsymbol{B}(z)$ as
$$\boldsymbol{B}(z) = \left[\prod_{k=0}^{\sigma-1} \boldsymbol{S}_k^{-1} \right] \boldsymbol{H}_{\mathsf{p}}(z).$$
4. Perform operations from $\mathcal{A}(\cdot)$ (i.e., type A row/column operations) on $\boldsymbol{B}(z)$ until it is reduced to an identity matrix. The idea here is similar to Gaussian elimination except that division of arbitrary Laurent polynomials is avoided and replaced by division with remainder (via the Euclidean algorithm). This process yields
$$\boldsymbol{D}_{\nu-1}(z) \cdots \boldsymbol{D}_1(z) \boldsymbol{D}_0(z) \boldsymbol{B}(z) \boldsymbol{C}_0(z) \boldsymbol{C}_1(z) \cdots \boldsymbol{C}_{\mu-1}(z) = \boldsymbol{I}$$
where the $\boldsymbol{D}_k(z)$ and $\boldsymbol{C}_k(z)$ correspond to type A elementary row and column operations on $\boldsymbol{B}(z)$, respectively. From this, we can write the factorization of $\boldsymbol{B}(z)$ as
$$\boldsymbol{B}(z) = \boldsymbol{D}_0^{-1}(z) \boldsymbol{D}_1^{-1}(z) \cdots \boldsymbol{D}_{\nu-1}^{-1}(z) \boldsymbol{C}_{\mu-1}^{-1}(z) \cdots \boldsymbol{C}_1^{-1}(z) \boldsymbol{C}_0^{-1}(z). \tag{3.93}$$
5. The lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ is then
$$\boldsymbol{H}_{\mathsf{p}}(z) = \left[\prod_{k=0}^{\sigma-1} \boldsymbol{S}_k \right] \boldsymbol{B}(z)$$
where $\boldsymbol{B}(z)$ is given by (3.93).

Since the choice of $\boldsymbol{S}_k(z)$,$\boldsymbol{C}_k(z)$, and $\boldsymbol{D}_k(z)$ are not unique for a given $\boldsymbol{H}_{\mathsf{p}}(z)$, the factorization is also not unique. In particular, the $\boldsymbol{C}_k(z)$ and $\boldsymbol{D}_k(z)$ are influenced by the terms cancelled during Euclidean division and the sequence of rows and columns operated on during the reduction of $\boldsymbol{B}(z)$.

It is important to emphasize that in a lifting decomposition all factors are Laurent polynomial matrices. Although the factorization algorithm is similar in spirit to Gaussian elimination, Gaussian elimination cannot be used since it requires closure on division. Laurent polynomials are not closed on division. That is, the quotient of two Laurent polynomials is generally a rational polynomial expression and not another Laurent polynomial. For more information on matrix Euclidean algorithms, the reader is referred to [14] (see Theorem 22.8).

To demonstrate the technique for finding lifting realizations of filter banks, some examples are provided below.

Example 3.28. Consider the two-channel shift-free PR UMD filter bank having analysis transfer functions $H_0(z), H_1(z)$, where

$$H_0(z) = \tfrac{1}{2}(1+z), \quad \text{and} \quad H_1(z) = 1-z.$$

Find a lifting realization of the system.

Solution. We find the analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ corresponding to a $(3,1)$-type decomposition. Calculating the analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$, we obtain

$$\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{bmatrix}.$$

Moreover, we also have that $\det \boldsymbol{H}_\mathsf{p}(z) \equiv -1$. Since $\det \boldsymbol{H}_\mathsf{p}(z)$ is a constant, a lifting realization exists and no renormalization of the analysis filters is required.

Now, we apply elementary row and column operations to $\boldsymbol{H}_\mathsf{p}(z)$. Since $\boldsymbol{H}_\mathsf{p}(z)$ is not unimodular, we must first apply a scaling operation. We multiply row 1 of $\boldsymbol{H}_\mathsf{p}(z)$ by -1. This forces the resulting product to be unimodular. This process yields

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}_{\boldsymbol{S}_0(z)} \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)} = \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{bmatrix}}_{\boldsymbol{S}_0(z)\boldsymbol{H}_\mathsf{p}(z)}.$$

We add a multiple of column 1 to column 0 to obtain

$$\underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{bmatrix}}_{\boldsymbol{S}_0(z)\boldsymbol{H}_\mathsf{p}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{S}_0(z)\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)}.$$

We add a multiple of column 0 to column 1 yielding

$$\underbrace{\begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{S}_0(z)\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{I}}.$$

Combining the above results, we have

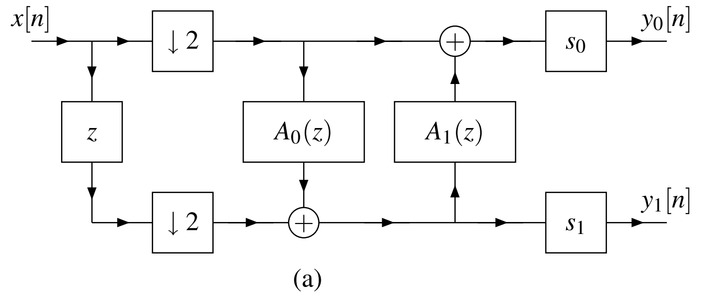
$$\boldsymbol{S}_0(z)\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z) = \boldsymbol{I}.$$

Thus, a lifting factorization of $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\begin{aligned} \boldsymbol{H}_\mathsf{p}(z) &= \boldsymbol{S}_0^{-1}(z)\boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\ &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\ &= \mathcal{S}(1,-1)\mathcal{A}(0,1,\tfrac{1}{2})\mathcal{A}(1,0,-1). \end{aligned}$$

From the preceding result, it immediately follows that a lifting factorization of the synthesis polyphase matrix $\boldsymbol{G}_\mathsf{p}(z)$ is given by

$$\begin{aligned} \boldsymbol{G}_\mathsf{p}(z) &= \boldsymbol{C}_0(z)\boldsymbol{C}_1(z)\boldsymbol{S}_0(z) \\ &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &= \mathcal{A}(1,0,1)\mathcal{A}(0,1,-\tfrac{1}{2})\mathcal{S}(1,-1). \end{aligned}$$



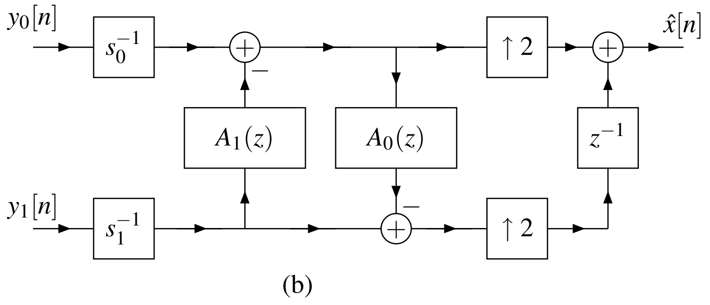


Figure 3.51: Example lifting realization. (a) Analysis side. (b) Synthesis side.

These lifting factorizations yield the realization shown in Figure 3.51, where

$$A_0(z) = -1, \quad A_1(z) = \tfrac{1}{2}, \quad s_0 = 1, \quad \text{and} \quad s_1 = -1.$$

By inspection, it is obvious that the UMD filter bank has the shift-free PR property. The synthesis side undoes step-by-step each operation performed on the analysis side. □

Example 3.29. Consider the two-channel shift-free PR UMD filter bank with analysis filter transfer functions $H_0(z), H_1(z)$, where

$$H_0(z) = -\tfrac{1}{8}z^2 + \tfrac{1}{4}z + \tfrac{3}{4} + \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2} \quad \text{and} \quad H_1(z) = -\tfrac{1}{2}z^2 + z - \tfrac{1}{2}.$$

Find two distinct lifting realizations of this system.

Solution. Let us consider the $(3,1)$-type polyphase representation of the system. First, we determine the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$. We can rewrite the analysis filter transfer functions as

$$H_0(z) = z^0\left(-\tfrac{1}{8}z^2 + \tfrac{3}{4} - \tfrac{1}{8}z^{-2}\right) + z^1\left(\tfrac{1}{4} + \tfrac{1}{4}z^{-2}\right) \quad \text{and} \quad H_1(z) = z^0\left(-\tfrac{1}{2}z^2 - \tfrac{1}{2}\right) + z^1(1).$$

From this, we can trivially write the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ as

$$\boldsymbol{H}_{\mathsf{p}}(z) = \begin{bmatrix} -\tfrac{1}{8}z + \tfrac{3}{4} - \tfrac{1}{8}z^{-1} & \tfrac{1}{4} + \tfrac{1}{4}z^{-1} \\ -\tfrac{1}{2}z - \tfrac{1}{2} & 1 \end{bmatrix}.$$

One can confirm that $\det \boldsymbol{H}_{\mathsf{p}}(z)$ is unimodular (i.e., $\det \boldsymbol{H}_{\mathsf{p}}(z) \equiv 1$). Therefore, no scaling operations are necessary in what follows.

FIRST REALIZATION. We perform elementary column operations on $\boldsymbol{H}_{\mathsf{p}}(z)$ in order to transform it into an identity matrix. First, we add a multiple of column 1 to column 0 as follows:

$$\underbrace{\begin{bmatrix} -\frac{1}{8}z+\frac{3}{4}-\frac{1}{8}z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -\frac{1}{2}z-\frac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ a_1z+a_0 & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \begin{bmatrix} -\frac{1}{8}z+\frac{3}{4}-\frac{1}{8}z^{-1}+(a_1z+a_0)\left(\frac{1}{4}+\frac{1}{4}z^{-1}\right) & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -\frac{1}{2}z-\frac{1}{2}+(a_1z+a_0) & 1 \end{bmatrix} \tag{3.94}$$

$$= \begin{bmatrix} \left(-\frac{1}{8}+\frac{1}{4}a_1\right)z+\left(\frac{3}{4}+\frac{1}{4}a_1+\frac{1}{4}a_0\right)+\left(-\frac{1}{8}+\frac{1}{4}a_0\right)z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ \left(-\frac{1}{2}+a_1\right)z+\left(-\frac{1}{2}+a_0\right) & 1 \end{bmatrix}$$

Let us choose the coefficients a_0, a_1 to cancel the terms in z and z^{-1}. Clearly, we require

$$-\tfrac{1}{8}+\tfrac{1}{4}a_1 = 0 \quad \text{and} \quad -\tfrac{1}{8}+\tfrac{1}{4}a_0 = 0.$$

Solving for a_0, a_1, we obtain

$$a_0 = \tfrac{1}{2} \quad \text{and} \quad a_1 = \tfrac{1}{2}.$$

Thus, we have

$$\underbrace{\begin{bmatrix} -\frac{1}{8}z+\frac{3}{4}-\frac{1}{8}z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -\frac{1}{2}z-\frac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{1}{2}z+\frac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} 1 & \frac{1}{4}+\frac{1}{4}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)}.$$

Now, we can trivially obtain

$$\underbrace{\begin{bmatrix} 1 & \frac{1}{4}+\frac{1}{4}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & -\frac{1}{4}-\frac{1}{4}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{I}}.$$

So, in summary, we have

$$\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z) = \boldsymbol{I}.$$

Thus, the lifting factorization of $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\begin{aligned} \boldsymbol{H}_\mathsf{p}(z) &= \boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\ &= \begin{bmatrix} 1 & \frac{1}{4}+\frac{1}{4}z^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z-\frac{1}{2} & 1 \end{bmatrix} \\ &= \mathcal{A}(0,1,\tfrac{1}{4}+\tfrac{1}{4}z^{-1})\mathcal{A}(1,0,-\tfrac{1}{2}-\tfrac{1}{2}z). \end{aligned}$$

This factorization yields the filtering structure shown in Figure 3.51, where

$$A_0(z) = -\tfrac{1}{2}z-\tfrac{1}{2}, \quad A_1(z) = \tfrac{1}{4}+\tfrac{1}{4}z^{-1}, \quad \text{and} \quad s_0 = s_1 = 1.$$

SECOND REALIZATION. Again, we apply elementary column operations to $\boldsymbol{H}_\mathsf{p}(z)$. We commence as before, leading to (3.94). Unlike before, however, we will choose to cancel the terms in z^1 and z^0. So, we need to choose the coefficients a_0, a_1 to satisfy

$$-\tfrac{1}{8}+\tfrac{1}{4}a_1 = 0 \quad \text{and} \quad \tfrac{3}{4}+\tfrac{1}{4}a_1+\tfrac{1}{4}a_0 = 0.$$

Solving for a_0, a_1, we obtain

$$a_0 = -\tfrac{7}{2} \quad \text{and} \quad a_1 = \tfrac{1}{2}.$$

Thus, we have

$$\underbrace{\begin{bmatrix} -\frac{1}{8}z+\frac{3}{4}-\frac{1}{8}z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -\frac{1}{2}z-\frac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{1}{2}z-\frac{7}{2} & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} -z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -4 & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)}.$$

We add a multiple of column 0 to column 1 to obtain

$$\begin{aligned} \underbrace{\begin{bmatrix} -z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -4 & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & a_1z+a_0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} &= \begin{bmatrix} -z^{-1} & (a_1z+a_0)(-z^{-1})+\frac{1}{4}+\frac{1}{4}z^{-1} \\ -4 & (a_1z+a_0)(-4)+1 \end{bmatrix} \\ &= \begin{bmatrix} -z^{-1} & \left(\frac{1}{4}-a_1\right)+\left(\frac{1}{4}-a_0\right)z^{-1} \\ -4 & (-4a_1)z+(1-4a_0) \end{bmatrix}. \end{aligned}$$

Let us choose to cancel the terms in z^0 and z^{-1}. So, we want a_0, a_1 to satisfy

$$\tfrac{1}{4}-a_1=0 \quad \text{and} \quad \tfrac{1}{4}-a_0=0.$$

Solving for a_0, a_1, we obtain

$$a_0=\tfrac{1}{4} \quad \text{and} \quad a_1=\tfrac{1}{4}.$$

Thus, we have

$$\underbrace{\begin{bmatrix} -z^{-1} & \frac{1}{4}+\frac{1}{4}z^{-1} \\ -4 & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & \frac{1}{4}z+\frac{1}{4} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \underbrace{\begin{bmatrix} -z^{-1} & 0 \\ -4 & -z \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)}.$$

We add a multiple of column 1 to column 0 to produce

$$\begin{aligned} \underbrace{\begin{bmatrix} -z^{-1} & 0 \\ -4 & -z \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ a_{-1}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{C}_2(z)} &= \begin{bmatrix} -z^{-1} & 0 \\ -4-z(a_{-1}z^{-1}) & z \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} -z^{-1} & 0 \\ -4-a_{-1} & z \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)\boldsymbol{C}_2(z)}. \end{aligned}$$

Let us cancel the terms in z^0. So, we require

$$-4-a_{-1}=0 \quad \Rightarrow \quad a_{-1}=-4.$$

Thus, we have

$$\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)\underbrace{\begin{bmatrix} 1 & 0 \\ -4z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{C}_2(z)} = \underbrace{\begin{bmatrix} -z^{-1} & 0 \\ 0 & -z \end{bmatrix}}_{\boldsymbol{D}(z)}.$$

So, in summary, we have

$$\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)\boldsymbol{C}_2(z)=\boldsymbol{D}(z).$$

Thus, a factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ is given by

$$\begin{aligned}\boldsymbol{H}_{\mathsf{p}}(z) &= \boldsymbol{D}(z)\boldsymbol{C}_2^{-1}(z)\boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\ &= \begin{bmatrix} -z^{-1} & 0 \\ 0 & -z \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 4z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -\frac{1}{4}z-\frac{1}{4} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z+\frac{7}{2} & 1 \end{bmatrix}.\end{aligned}$$

This is not quite of the form of a lifting factorization, however (due the form of the leftmost factor). To deal with the diagonal matrix factor, we observe that

$$\begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} = \begin{bmatrix} 1 & K-K^2 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -1/K & 1 \end{bmatrix}\begin{bmatrix} 1 & K-1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

So, we have that

$$\begin{bmatrix} -z^{-1} & 0 \\ 0 & -z \end{bmatrix} = \begin{bmatrix} 1 & -z^{-1}-z^{-2} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -1-z^{-1} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Using the preceding relationship, we obtain the lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ given by

$$\begin{aligned}\boldsymbol{H}_{\mathsf{p}}(z) &= \begin{bmatrix} 1 & -z^{-1}-z^{-2} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -1-z^{-1} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 4z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -\frac{1}{4}z-\frac{1}{4} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z+\frac{7}{2} & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -z^{-1}-z^{-2} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -1-z^{-1} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1+4z^{-1} & 1 \end{bmatrix}\begin{bmatrix} 1 & -\frac{1}{4}z-\frac{1}{4} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z+\frac{7}{2} & 1 \end{bmatrix} \\ &= \mathcal{A}(0,1,-z^{-1}-z^{-2})\mathcal{A}(1,0,z^{-1})\mathcal{A}(0,1,-1-z^{-1})\mathcal{A}(1,0,1+4z^{-1})\mathcal{A}(0,1,-\tfrac{1}{4}z-\tfrac{1}{4})\mathcal{A}(1,0,-\tfrac{1}{2}z+\tfrac{7}{2}).\end{aligned}$$

The lifting realization of the system is then obtained as in the earlier examples. □

Example 3.30. Consider the two-channel shift-free PR UMD filter bank with analysis filter transfer functions $H_0(z), H_1(z)$, where

$$H_0(z) = \tfrac{1}{2}z+\tfrac{1}{2} \quad \text{and} \quad H_1(z) = -\tfrac{1}{8}z^3-\tfrac{1}{8}z^2+z-1+\tfrac{1}{8}z^{-1}+\tfrac{1}{8}z^{-2}.$$

Find a lifting realization of this system.

Solution. First, we find the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ corresponding to a $(3,1)$-type decomposition. We rearrange the analysis filter transfer functions to obtain

$$H_0(z) = z^0\left(\tfrac{1}{2}\right)+z^1\left(\tfrac{1}{2}\right) \quad \text{and} \quad H_1(z) = z^0\left(-\tfrac{1}{8}z^2-1+\tfrac{1}{8}z^{-2}\right)+z^1\left(-\tfrac{1}{8}z^2+1+\tfrac{1}{8}z^{-2}\right).$$

From this, we can trivially write the analysis polyphase matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ as

$$\boldsymbol{H}_{\mathsf{p}}(z) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{8}z-1+\frac{1}{8}z^{-1} & -\frac{1}{8}z+1+\frac{1}{8}z^{-1} \end{bmatrix}.$$

Since $\boldsymbol{H}_{\mathsf{p}}(z)$ is unimodular, no scaling operations are required in what follows.

Now, we apply A-type elementary column operations in order to transform the matrix $\boldsymbol{H}_{\mathsf{p}}(z)$ into an identity matrix. First, we perform column operations in an attempt to force an entry of one on the first main diagonal element. We add a multiple of the 1th column to the 0th column. We have

$$\underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{8}z-1+\frac{1}{8}z^{-1} & -\frac{1}{8}z+1+\frac{1}{8}z^{-1} \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)}\underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} 1 & \frac{1}{2} \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & -\frac{1}{8}z+1+\frac{1}{8}z^{-1} \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)}.$$

Now, we add a multiple of the 0th column to the 1th column.

$$\underbrace{\begin{bmatrix} 1 & \frac{1}{2} \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & -\frac{1}{8}z+1+\frac{1}{8}z^{-1} \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & -\frac{1}{2}\left(-\frac{1}{4}z+\frac{1}{4}z^{-1}\right)+\left(-\frac{1}{8}z+1+\frac{1}{8}z^{-1}\right) \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)}.$$

We obtain

$$\underbrace{\begin{bmatrix} 1 & 0 \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{1}{4}z-\frac{1}{4}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{C}_2(z)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{I}}.$$

So, in summary, we have

$$\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z)\boldsymbol{C}_2(z) = \boldsymbol{I}.$$

Thus, the lifting factorization of $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\begin{aligned} \boldsymbol{H}_\mathsf{p}(z) &= \boldsymbol{C}_2^{-1}(z)\boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\ &= \begin{bmatrix} 1 & 0 \\ -\frac{1}{4}z+\frac{1}{4}z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\ &= \mathcal{A}(1,0,-\tfrac{1}{4}z+\tfrac{1}{4}z^{-1})\mathcal{A}(0,1,\tfrac{1}{2})\mathcal{A}(1,0,-1). \end{aligned}$$

The lifting realization of the filter bank is trivially obtained from the above factorization of $\boldsymbol{H}_\mathsf{p}(z)$. □

Example 3.31. Consider the two-channel shift-free PR UMD filter bank with analysis filter transfer functions $H_0(z), H_1(z)$, where

$$\begin{aligned} H_0(z) &= \tfrac{1}{64}z^4 - \tfrac{8}{64}z^2 + \tfrac{16}{64}z + \tfrac{46}{64} + \tfrac{16}{64}z^{-1} - \tfrac{8}{64}z^{-2} + \tfrac{1}{64}z^{-4} \quad \text{and} \\ H_1(z) &= \tfrac{1}{16}z^4 - \tfrac{9}{16}z^2 + z - \tfrac{9}{16} + \tfrac{1}{16}z^{-2}. \end{aligned}$$

Find a lifting realization of this system. Choose the lifting filters to be symmetric.

Solution. First, we find the analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ corresponding to a $(3,1)$-type decomposition. We rearrange the analysis filter transfer functions to obtain

$$\begin{aligned} H_0(z) &= z^0\left(\tfrac{1}{64}z^4 - \tfrac{8}{64}z^2 + \tfrac{46}{64} - \tfrac{8}{64}z^{-2} + \tfrac{1}{64}z^{-4}\right) + z^1\left(\tfrac{16}{64} + \tfrac{16}{64}z^{-2}\right) \quad \text{and} \\ H_1(z) &= z^0\left(\tfrac{1}{16}z^4 - \tfrac{9}{16}z^2 - \tfrac{9}{16} + \tfrac{1}{16}z^{-2}\right) + z^1(1). \end{aligned}$$

From this, we can trivially write the analysis polyphase matrix

$$\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \frac{1}{64}z^2 - \frac{8}{64}z + \frac{46}{64} - \frac{8}{64}z^{-1} + \frac{1}{64}z^{-2} & \frac{16}{64} + \frac{16}{64}z^{-1} \\ \frac{1}{16}z^2 - \frac{9}{16}z - \frac{9}{16} + \frac{1}{16}z^{-1} & 1 \end{bmatrix}.$$

One can confirm that $\det \boldsymbol{H}_\mathsf{p}(z) \equiv 1$. Therefore, we do not need to concern ourselves with scaling operations.

Now, we apply A-type elementary column operations to $\boldsymbol{H}_\mathsf{p}(z)$ in order to obtain an identity matrix. First, we perform column operations in an attempt to force a value of one on the first element of the main diagonal. We add a multiple of column 1 to column 0. This yields

$$\underbrace{\begin{bmatrix} \frac{1}{64}z^2 - \frac{8}{64}z + \frac{46}{64} - \frac{8}{64}z^{-1} + \frac{1}{64}z^{-2} & \frac{16}{64} + \frac{16}{64}z^{-1} \\ \frac{1}{16}z^2 - \frac{9}{16}z - \frac{9}{16} + \frac{1}{16}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ a_2z^2 + a_1z + a_0 + a_{-1}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)}$$

$$= \begin{bmatrix} \frac{1}{64}z^2 - \frac{8}{64}z + \frac{46}{64} - \frac{8}{64}z^{-1} + \frac{1}{64}z^{-2} + \left(a_2z^2 + a_1z + a_0 + a_{-1}z^{-1}\right)\left(\frac{16}{64} + \frac{16}{64}z^{-1}\right) & \frac{16}{64} + \frac{16}{64}z^{-1} \\ \frac{1}{16}z^2 - \frac{9}{16}z - \frac{9}{16} + \frac{1}{16}z^{-1} + \left(a_2z^2 + a_1z + a_0 + a_{-1}z^{-1}\right) & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \left(\frac{1}{64} + \frac{16}{64}a_2\right)z^2 + \left(-\frac{8}{64} + \frac{16}{64}a_1 + \frac{16}{64}a_2\right)z + \left(\frac{46}{64} + \frac{16}{64}a_0 + \frac{16}{64}a_1\right) + \left(-\frac{8}{64} + \frac{16}{64}a_{-1} + \frac{16}{64}a_0\right)z^{-1} + \left(\frac{1}{64} + \frac{16}{64}a_{-1}\right)z^{-2} & \frac{16}{64} + \frac{16}{64}z^{-1} \\ \left(\frac{1}{16} + a_2\right)z^2 + \left(-\frac{9}{16} + a_1\right)z + \left(-\frac{9}{16} + a_0\right) + \left(\frac{1}{16} + a_{-1}\right)z^{-1} & 1 \end{bmatrix}.$$

Since we want a symmetric lifting filter, we must choose to cancel the terms in z^2, z^1, z^{-1}, z^{-2}. So, the coefficients a_{-1}, a_0, a_1, a_2 must satisfy

$$\tfrac{1}{64} + \tfrac{16}{64}a_2 = 0, \quad -\tfrac{8}{64} + \tfrac{16}{64}a_1 + \tfrac{16}{64}a_2 = 0, \quad -\tfrac{8}{64} + \tfrac{16}{64}a_0 + \tfrac{16}{64}a_{-1} = 0, \quad \text{and} \quad \tfrac{1}{64} + \tfrac{16}{64}a_{-1} = 0.$$

Solving for a_{-1}, a_0, a_1, a_2, we obtain

$$a_2 = -\tfrac{1}{16}, \quad a_1 = \tfrac{9}{16}, \quad a_0 = \tfrac{9}{16}, \quad \text{and} \quad a_{-1} = -\tfrac{1}{16}.$$

Thus, we have

$$\underbrace{\begin{bmatrix} \frac{1}{64}z^2 - \frac{8}{64}z + \frac{46}{64} - \frac{8}{64}z^{-1} + \frac{1}{64}z^{-2} & \frac{16}{64} + \frac{16}{64}z^{-1} \\ \frac{1}{16}z^2 - \frac{9}{16}z - \frac{9}{16} + \frac{1}{16}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ -\frac{1}{16}z^2 + \frac{9}{16}z + \frac{9}{16} - \frac{1}{16}z^{-1} & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} 1 & \frac{16}{64} + \frac{16}{64}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)}.$$

Trivially, we have

$$\underbrace{\begin{bmatrix} 1 & \frac{16}{64} + \frac{16}{64}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & -\frac{16}{64} - \frac{16}{64}z^{-1} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{I}}.$$

So, in summary, we have

$$\boldsymbol{H}_{\mathsf{p}}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z) = \boldsymbol{I}.$$

Thus, the lifting factorization of $\boldsymbol{H}_{\mathsf{p}}(z)$ is given by

$$\begin{aligned} \boldsymbol{H}_{\mathsf{p}}(z) &= \boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\ &= \begin{bmatrix} 1 & \frac{16}{64} + \frac{16}{64}z^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{16}z^2 - \frac{9}{16}z - \frac{9}{16} + \frac{1}{16}z^{-1} & 1 \end{bmatrix} \\ &= \mathcal{A}\left(0, 1, \tfrac{16}{64} + \tfrac{16}{64}z^{-1}\right)\mathcal{A}\left(1, 0, \tfrac{1}{16}z^2 - \tfrac{9}{16}z - \tfrac{9}{16} + \tfrac{1}{16}z^{-1}\right). \end{aligned}$$

□

Example 3.32. Consider the two-channel shift-free PR UMD filter bank with analysis filter transfer functions $H_0(z), H_1(z)$, where

$$H_0(z) = \tfrac{3}{128}z^4 - \tfrac{6}{128}z^3 - \tfrac{16}{128}z^2 + \tfrac{38}{128}z + \tfrac{90}{128} + \tfrac{38}{128}z^{-1} - \tfrac{16}{128}z^{-2} - \tfrac{6}{128}z^{-3} + \tfrac{3}{128}z^{-4} \quad \text{and} \quad H_1(z) = -\tfrac{1}{2}z^2 + z - \tfrac{1}{2}.$$

Find a lifting realization of this system with symmetric lifting filters.

Solution. First, we find the analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}(z)$ corresponding to a $(3,1)$-type decomposition. We rearrange the analysis filter transfer functions to obtain

$$H_0(z) = z^0\left(\tfrac{3}{128}z^4 - \tfrac{16}{128}z^2 + \tfrac{90}{128} - \tfrac{16}{128}z^{-2} + \tfrac{3}{128}z^{-4}\right) + z\left(-\tfrac{6}{128}z^2 + \tfrac{38}{128} + \tfrac{38}{128}z^{-2} - \tfrac{6}{128}z^{-4}\right) \quad \text{and}$$
$$H_1(z) = z^0\left(-\tfrac{1}{2}z^2 - \tfrac{1}{2}\right) + z(1).$$

From this, we can trivially write the analysis polyphase matrix

$$\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \tfrac{3}{128}z^2 - \tfrac{16}{128}z + \tfrac{90}{128} - \tfrac{16}{128}z^{-1} + \tfrac{3}{128}z^{-2} & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ -\tfrac{1}{2}z - \tfrac{1}{2} & 1 \end{bmatrix}.$$

We have that $\det \boldsymbol{H}_\mathsf{p}(z) \equiv 1$. Therefore, no scaling operations are required.

Now, we apply A-type elementary column operations to $\boldsymbol{H}_\mathsf{p}(z)$ in order to reduce it to an identity matrix. First, we add a multiple of column 1 to column 0. This process yields

$$\begin{aligned}
&\underbrace{\begin{bmatrix} \tfrac{3}{128}z^2 - \tfrac{16}{128}z + \tfrac{90}{128} - \tfrac{16}{128}z^{-1} + \tfrac{3}{128}z^{-2} & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ -\tfrac{1}{2}z - \tfrac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ a_1z + a_0 & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} \\
&= \begin{bmatrix} \tfrac{3}{128}z^2 - \tfrac{16}{128}z + \tfrac{90}{128} - \tfrac{16}{128}z^{-1} + \tfrac{3}{128}z^{-2} + (a_1z+a_0)\left(-\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2}\right) & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ -\tfrac{1}{2}z - \tfrac{1}{2} + (a_1z+a_0) & 1 \end{bmatrix} \\
&= \begin{bmatrix} \left(\tfrac{3}{128} - \tfrac{6}{128}a_1\right)z^2 + \left(-\tfrac{16}{128} + \tfrac{38}{128}a_1 - \tfrac{6}{128}a_0\right)z + \left(\tfrac{90}{128} + \tfrac{38}{128}a_0 + \tfrac{38}{128}a_1\right) + \left(-\tfrac{16}{128} + \tfrac{38}{128}a_0 - \tfrac{6}{128}a_1\right)z^{-1} + \left(\tfrac{3}{128} - \tfrac{6}{128}a_0\right)z^{-2} & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ \left(-\tfrac{1}{2} + a_1\right)z + \left(-\tfrac{1}{2} + a_0\right) & 1 \end{bmatrix}.
\end{aligned}$$

Since we want a symmetric lifting filter, we must choose to cancel the terms in z^2 and z^{-2}. So, the coefficients a_0, a_1 must satisfy

$$\tfrac{3}{128} - \tfrac{6}{128}a_1 = 0 \quad \text{and} \quad \tfrac{3}{128} - \tfrac{6}{128}a_0 = 0.$$

Solving for a_0, a_1, we obtain

$$a_0 = \tfrac{1}{2} \quad \text{and} \quad a_1 = \tfrac{1}{2}.$$

Thus, we have

$$\underbrace{\begin{bmatrix} \tfrac{3}{128}z^2 - \tfrac{16}{128}z + \tfrac{90}{128} - \tfrac{16}{128}z^{-1} + \tfrac{3}{128}z^{-2} & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ -\tfrac{1}{2}z - \tfrac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)} \underbrace{\begin{bmatrix} 1 & 0 \\ \tfrac{1}{2}z + \tfrac{1}{2} & 1 \end{bmatrix}}_{\boldsymbol{C}_0(z)} = \underbrace{\begin{bmatrix} 1 & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)}.$$

Trivially, we have

$$\underbrace{\begin{bmatrix} 1 & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)} \underbrace{\begin{bmatrix} 1 & \tfrac{6}{128}z - \tfrac{38}{128} - \tfrac{38}{128}z^{-1} + \tfrac{6}{128}z^{-2} \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_1(z)} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{I}}.$$

So, in summary, we have

$$\boldsymbol{H}_\mathsf{p}(z)\boldsymbol{C}_0(z)\boldsymbol{C}_1(z) = \boldsymbol{I}.$$

Thus, a lifting factorization of $\boldsymbol{H}_\mathsf{p}(z)$ is given by

$$\begin{aligned}
\boldsymbol{H}_\mathsf{p}(z) &= \boldsymbol{C}_1^{-1}(z)\boldsymbol{C}_0^{-1}(z) \\
&= \begin{bmatrix} 1 & -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tfrac{1}{2}z - \tfrac{1}{2} & 1 \end{bmatrix} \\
&= \mathcal{A}\left(0, 1, -\tfrac{6}{128}z + \tfrac{38}{128} + \tfrac{38}{128}z^{-1} - \tfrac{6}{128}z^{-2}\right)\mathcal{A}\left(1, 0, -\tfrac{1}{2}z - \tfrac{1}{2}\right).
\end{aligned}$$

□

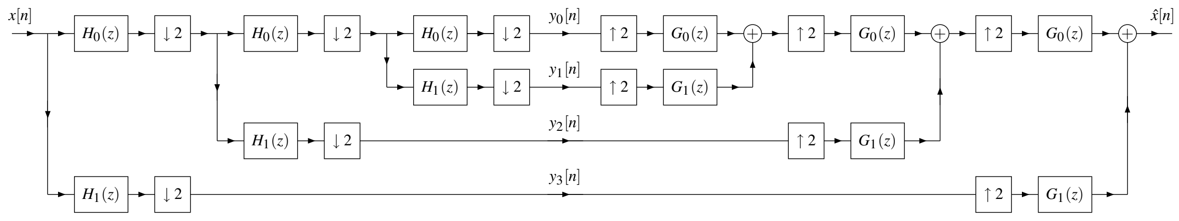


Figure 3.52: Example of a tree-structured filter bank.

3.7 Filter Banks and Series Expansions of Signals

Although a UMD filter bank can be seen as a structure that computes a series expansion of a discrete-time signal, this is not the only interpretation. In fact, as we will see later, such a filter bank can also be viewed as computing coefficients for a series expansion of a *continuous-time* signal from the coefficients of a related series expansion of the same signal. In this case, $x[n]$ is interpreted as the expansion coefficients of the original series, and the subband signals $\{y_k[n]\}$ represent the expansion coefficients of the new series. As we shall see in next chapter, this scenario corresponds to a single-level M-band wavelet decomposition. In other words, UMD filter banks compute wavelet series expansions (i.e., wavelet transforms).

3.8 Tree-Structured Filter Banks

As shown in Figure 3.32, the analysis side of an M-channel UMD filter bank decomposes the input signal $x[n]$ into M subband signals $\{y_k[n]\}$. The synthesis side then recombines these subband signals to obtain $\hat{x}[n]$, the reconstructed version of the original signal. There is nothing, however, to prevent the use of additional UMD filter banks to further decompose some or all of the subband signals $\{y_k[n]\}$. Of course, some or all of the resulting subband signals can again be decomposed with even more UMD filter banks. In other words, this idea can be applied recursively, and the final result is a filter bank with a tree structure.

In Figure 3.52, we show an example of a tree-structured filter bank obtained by recursively decomposing the lowpass subband signal of a two-channel UMD filter bank. Using the noble identities, this filter bank can also be equivalently expressed in the form shown in Figure 3.53, where

$$\begin{aligned}
H'_0(z) &= H_0(z)H_0(z^2)H_0(z^4),\\
H'_1(z) &= H_0(z)H_0(z^2)H_1(z^4),\\
H'_2(z) &= H_0(z)H_1(z^2),\\
H'_3(z) &= H_1(z),\\
G'_0(z) &= G_0(z)G_0(z^2)G_0(z^4),\\
G'_1(z) &= G_0(z)G_0(z^2)G_1(z^4),\\
G'_2(z) &= G_0(z)G_1(z^2), \quad \text{and}\\
G'_3(z) &= G_1(z).
\end{aligned}$$

In passing, we note that if a tree-structured filter bank is such that 1) only the lowpass subband signal is decomposed at each level in the tree, 2) the same basic UMD filter bank building block is used for decomposition at all levels, and 3) this basic block has PR and satisfies certain regularity conditions, then the tree-structured filter bank can be shown to compute a wavelet transform. In the two-channel case, such a tree-structured filter bank is called an **octave-band filter bank**. The analysis side of the octave-band filter bank calculates the forward wavelet transform and the synthesis side calculates the inverse wavelet transform.

At this point, our motivation for studying UMD filter banks becomes apparent. Under the conditions stated above, a UMD filter bank can be directly linked to a wavelet decomposition. Thus, UMD filter banks can be used to both design and implement wavelet transforms.

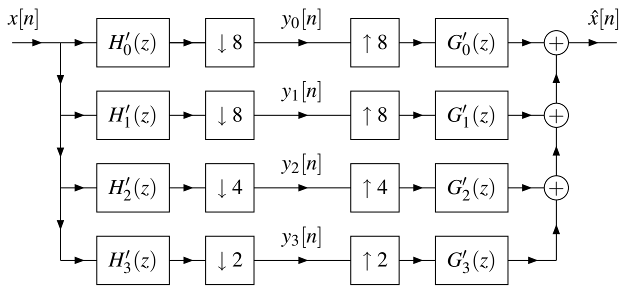


Figure 3.53: Equivalent non-uniformly decimated filter bank.

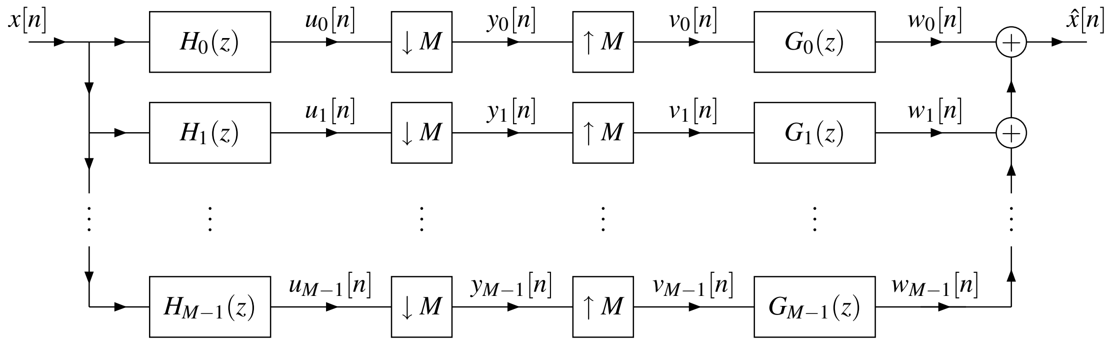


Figure 3.54: An M-channel UMD filter bank.

3.9 Filter Banks and Finite-Extent Signals

Consider an M-channel UMD filter bank like the one shown in Figure 3.54. So far, we have assumed that such a system operates on sequences of infinite extent (i.e., sequences defined on $\mathbb{Z}$). In many practical applications, however, the sequences of interest are defined only on some bounded domain such as a finite subset of the integers (e.g., $\{0,1,\ldots,N-1\}$). Consequently, a means is needed to facilitate the handling of finite-extent sequences by filter banks. As we shall see, a number of complications arise when dealing with finite-extent sequences.

For example, let us consider a sequence $x[n]$ defined only for $n \in \{0,1,2,3\}$ Suppose that we wish to process x using the filter bank from Figure 3.54 with $M = 2$. Defining downsampling and upsampling for a finite-extent sequence is not too problematic. For example, we could simply define these operations as:

$$x[0] \quad x[1] \quad x[2] \quad x[3] \quad \xrightarrow{\uparrow 2} \quad x[0] \quad 0 \quad x[1] \quad 0 \quad x[2] \quad 0 \quad x[3] \quad 0 \quad \text{and}$$
$$x[0] \quad x[1] \quad x[2] \quad x[3] \quad \xrightarrow{\downarrow 2} \quad x[0] \quad x[2].$$

In contrast, however, filtering a finite-extent sequence is problematic. Consider filtering the sequence x with a filter having impulse response h, where $h[n] = 0$ for $n \notin \{-1,0,1\}$ (i.e., h is a three-tap FIR filter). In general, the convolution result $y = h * x$ is given by

$$y[n] = h * x[n] = \sum_{k \in \mathbb{Z}} h[k]x[n-k] = h[-1]x[n+1] + h[0]x[n] + h[1]x[n-1].$$

Consider calculating y for the same domain on which x is defined (i.e., for $\{0,1,2,3\}$). First, consider computing $y[0]$. We have

$$y[0] = h[-1]x[1] + h[0]x[0] + h[1]\underbrace{x[-1]}_{\text{undefined}}.$$

Observe that the quantity $x[-1]$, appearing in the preceding equation, is not defined since -1 is not in the domain of x. Similarly, consider computing $y[3]$. We have

$$y[3] = h[-1]\underbrace{x[4]}_{\text{undefined}} + h[0]x[3] + h[1]x[2].$$

Observe that the quantity $x[4]$, appearing in the preceding equation, is not defined since 4 is not in the domain of x. In short, we have a problem. That is, the convolution result $y[n]$ is not well defined for every n in the domain of x. This is the so called boundary problem for filtering (since it occurs when computing the convolution result at points near the boundary of the sequence's domain). Generally, unless h is of the form $h[n] = K\delta[n]$ (i.e., the filter is trivial), the boundary filtering problem will always occur. If $h[n] \neq 0$ for some $n < 0$, the boundary problem will occur when filtering near the "right" edge of the sequence's domain. If $h[n] \neq 0$ for some $n > 0$, the boundary problem will occur when filtering near the "left" edge of the sequence's domain.

There are two general approaches to dealing with the boundary filtering problem. The first approach is signal extension. With this approach, we extend a finite-extent sequence so that the resulting sequence is defined for all $\mathbb{Z}$. Then, the filter bank is made to operate on the extended sequence. Since the extended sequence is defined for all $\mathbb{Z}$, the boundary problem is avoided. The second approach is filter modification. That is, we change the filter near the sequence boundaries (effectively shortening the length of the filter) so that samples outside the sequence's domain are not required in the convolution computation. Of the above two approaches, the second one is more general, in the sense that signal-extension methods can also be equivalently described in terms of filter modification. Unfortunately, modifying the filters is often more complicated, as one must be careful not to adversely affect properties of the filter bank (such as PR).

Another issue that arises when handling finite-extent sequences relates to the number of samples being processed. The analysis and synthesis sides of the filter bank each perform a linear transformation. The analysis side maps N samples to N' samples. If the net number of samples does not increase (i.e., $N' \leq N$), we say that transform is **nonexpansive**. In many applications, the nonexpansive property is extremely important. If a transform is expansive, the transformed representation of the sequence is inefficient/redundant (i.e., we are representing a fundamentally N-dimensional vector as a vector in a space of dimension greater than N). This is undesirable in many applications, especially signal compression. Moreover, many complicating factors often arise in practical systems when using expansive transforms. For example, since $N \neq N'$, the amounts of memory required to hold the original sequence and its corresponding subband signals differ. Thus, extra bookkeeping is required to keep track of the differing sizes of arrays used to hold sequence samples. Also, we cannot compute the transform in place, since the transformed sequence requires more storage than the original sequence.

In the sections that follow, we will study three techniques for handling finite-extent sequences with filter banks, namely, the zero-padding, periodic-extension, and symmetric-extension methods. All of these techniques are based on signal extension. Also, we will explain how the lifting realization can easily accommodate finite-extent sequences via either signal extension or filter modification. As we will see, some of the preceding methods lead to nonexpansive transforms, while others do not.

3.9.1 Zero-Padding Method

Of the various techniques for handling finite-extent signals with filter banks, arguably the simplest is zero padding. This method relies on the so called zero-padding signal-extension scheme. Suppose that we have a finite-extent sequence x defined on $\{0,1,\ldots,N-1\}$ (i.e., a sequence of length N). The process of **zero padding** transforms x to the new sequence y defined for all $\mathbb{Z}$, where

$$y[n] = \begin{cases} x[n] & n \in \{0,1,\ldots,N-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{3.95}$$

In other words, y is obtained by padding x with zeros before its first sample and after its last sample. The zero padding process is illustrated in Figure 3.55.

In passing, we note the following property of convolution, which will be important later.

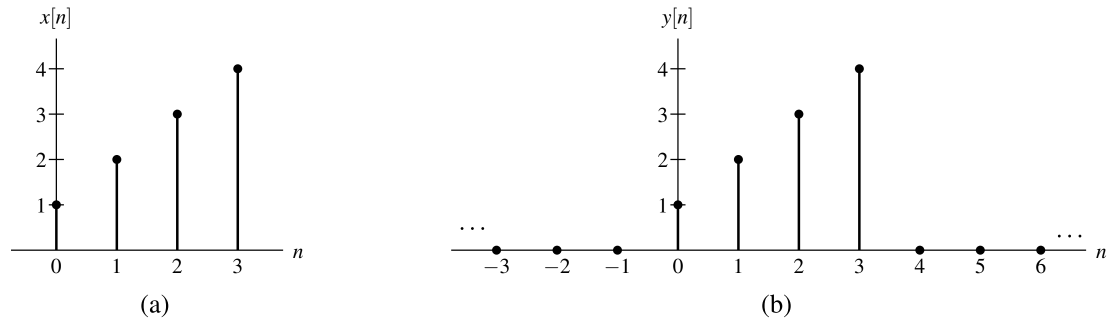


Figure 3.55: Zero padding example. (a) Original and (b) extended sequences.

Theorem 3.20. *Let x and h be sequences with support in* $\{M_0, M_0+1, \ldots, M_1\}$ *and* $\{N_0, N_0+1, \ldots, N_1\}$*, respectively, with the size of these sets being* $L_0 = M_1 - M_0 + 1$ *and* $L_1 = N_1 - N_0 + 1$*, respectively. Then,* $y = x * h$ *has support in* $\{M_0+N_0, M_0+N_0+1, \ldots, M_1+N_1\}$ *with size* $L_0 + L_1 - 1$.

From the above result, if a filtering operation is performed with a nontrivial filter (i.e., a filter with more than one tap), the number of samples needed to characterize the convolution result is greater than the number of samples needed to characterize the input to the filter.

Now, consider the M-channel filter bank from Figure 3.54 with FIR filters. Suppose that we have a finite-extent sequence x' defined on $\{0, 1, \ldots, N-1\}$ (i.e., a sequence of length N) that we want to process with the filter bank. To eliminate the problems associated with finite-extent sequences, we use zero padding to transform x' into the new sequence x, and then use this new sequence as the input to the analysis side of the filter bank. When using zero padding, the output of the analysis side of the filter bank is completely characterized by a finite number N' of samples (namely, the samples which can possibly be nonzero). Unfortunately, due to the result of Theorem 3.20, we have that $N' > N$ (unless the analysis filters are trivial). Since the analysis side of the filter bank maps N samples to N' samples, where $N' > N$, the transformation associated with the analysis side of the filter bank is expansive. As mentioned previously, expansive transforms are highly undesirable in many applications. Furthermore, the zero padding process can potentially introduce a significant amount of high-frequency spectral content in the extended sequence, due to large jumps where the padding starts. For the preceding reasons, the use of zero padding is often avoided.

3.9.2 Periodic-Extension Method

Another technique for handling finite-extent sequences with filter banks is the periodic-extension method. This approach makes use of the so called periodic-extension signal-extension method. Let x denote a finite-extent sequence defined on $\{0, 1, \ldots, N-1\}$ (i.e., a sequence of length N). The process of **periodic extension** transforms x to the new N-periodic sequence y defined on $\mathbb{Z}$, where

$$y[n] = x[\operatorname{mod}(n, N)] \quad \text{for all } n \in \mathbb{Z}. \tag{3.96}$$

In other words, periodic extension transforms a sequence containing N elements to an N-periodic sequence defined on $\mathbb{Z}$. An example of a sequence x and its periodically extended version y is illustrated in Figure 3.56.

Before presenting the details of the periodic-extension method, we must also introduce some important properties of LTI systems and downsamplers/upsamplers in relation to periodic sequences. These properties are stated in the theorems below.

Theorem 3.21. *If the input to a LTI system is N-periodic, then output of the system is N-periodic.*

Theorem 3.22. *If the input to an M-fold downsampler is N-periodic and* $M|N$*, then the output of the downsampler is* $(\frac{N}{M})$*-periodic.*

Theorem 3.23. *If the input to an M-fold upsampler is N-periodic, then the output of the upsampler is MN-periodic.*

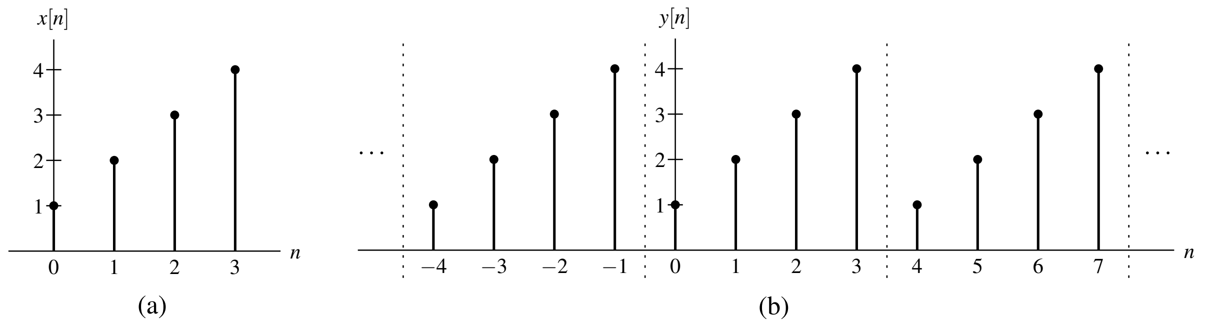


Figure 3.56: Periodic extension example. (a) Original and (b) extended sequences.

With the above properties in mind, we are now in a position to introduce the periodic-extension method. Consider a finite-extent sequence x' defined on $\{0,1,\ldots,N-1\}$ (i.e., a signal of length N). Suppose that we would like to process x' using the M-channel filter bank shown in Figure 3.54. With periodic extension, instead of using the system directly as it is shown in Figure 3.54, we employ the modified structure shown in Figure 3.57.

Consider the analysis side of the modified structure shown in Figure 3.57(a). For the input to the filter bank, we choose x as the periodic extension of x' (as given by (3.96)). Because $x[n]$ is defined for all $n \in \mathbb{Z}$, the boundary problem is eliminated. Since x is N-periodic (by construction) and H_k is LTI, u_k must be N-periodic (by Theorem 3.21). Suppose now that $M|N$. Then, it follows from the N-periodicity of u_k and Theorem 3.22 that y_k is $(\frac{N}{M})$-periodic, Since y_k is $(\frac{N}{M})$-periodic, it can be completely characterized by $\frac{N}{M}$ samples. For each y_k, we define a new sequence y'_k as $y'_k[n] = y_k[n]$ for $n \in \{0,1,\ldots,\frac{N}{M}-1\}$. In other words, y'_k is a finite-extent sequence consisting of the $\frac{N}{M}$ samples from one period of y_k. Since each sequence y'_k consists of $\frac{N}{M}$ samples and there are M such sequences, the output of the analysis side of the filter bank is comprised of $M\left(\frac{N}{M}\right) = N$ samples. Thus, the analysis side of the filter bank maps N samples to N' samples, where $N' = N$, and corresponds to a nonexpansive transform.

The synthesis side of the modified filter bank structure shown in Figure 3.57(b) works in a similar way as the analysis side. It is left as an exercise to the reader to consider the synthesis side in more detail. For this purpose, Theorem 3.23 is useful.

The periodic-extension method has a number of desirable characteristics. First, it can be used with a filter bank having any number of channels. Second, it works for any analysis and synthesis filters, including IIR filters (although one must be careful to handle the initial conditions appropriately). The periodic-extension method, however, has two significant disadvantages. First, it requires that $M|N$. (For example, if $M = 2$, N must be even.) Second, the process of periodically extending a sequence typically introduces large jumps at the splice points between periods, which artificially increases the amount of high-frequency content in the sequence. This increase in high-frequency content can be highly undesirable in some applications (e.g., signal compression).

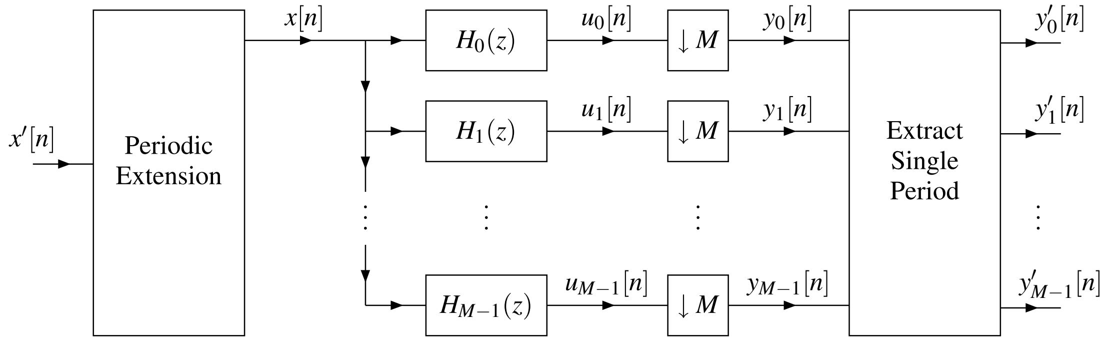
3.9.3 Symmetric-Extension Method

Another approach to handing finite-extent sequences with filter banks is the symmetric-extension method, which is based on a signal-extension scheme known as symmetric extension. Before presenting the details of the symmetric-extension method, we first provide some related background information.

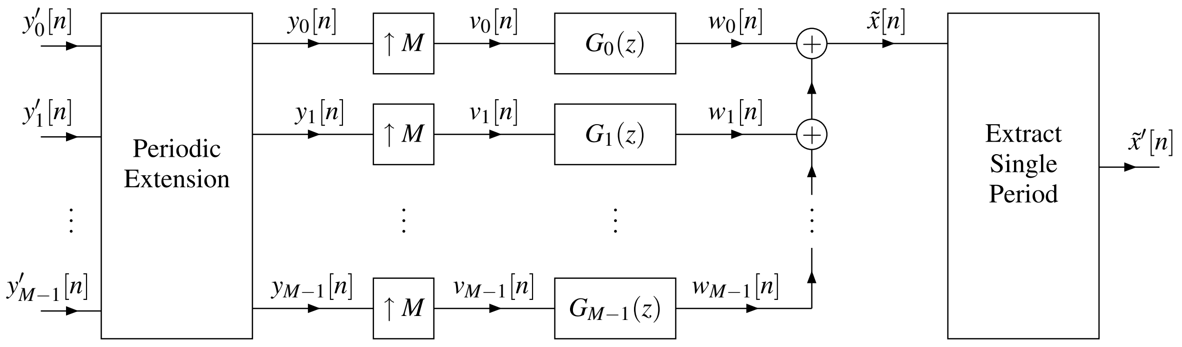
A sequence x of the form

$$x[n] = (-1)^s x[2c-n],$$

where $s \in \{0,1\}$ and $c \in \frac{1}{2}\mathbb{Z}$ (i.e., c is an integer multiple of $\frac{1}{2}$), is said to have symmetry about the point c. If $s = 0$ or $s = 1$, the sequence is said to be **symmetric** or **antisymmetric**, respectively. If the symmetry point $c \in \mathbb{Z}$ (i.e., c falls on a sample point) or $c \in \frac{1}{2}\mathbb{Z}_{\text{odd}}$ (i.e., c falls half way between sample points), the symmetry is referred to as **whole sample (WS)** or **half sample (HS)**, respectively. (Recall that $\mathbb{Z}_{\text{odd}}$ denotes the set of odd integers.) Some examples of sequences having symmetry are shown in Figure 3.58. In each case, the symmetry point is indicated with a dotted line.



(a)



(b)

Figure 3.57: Filter bank with periodic extension. (a) Analysis and (b) synthesis sides.

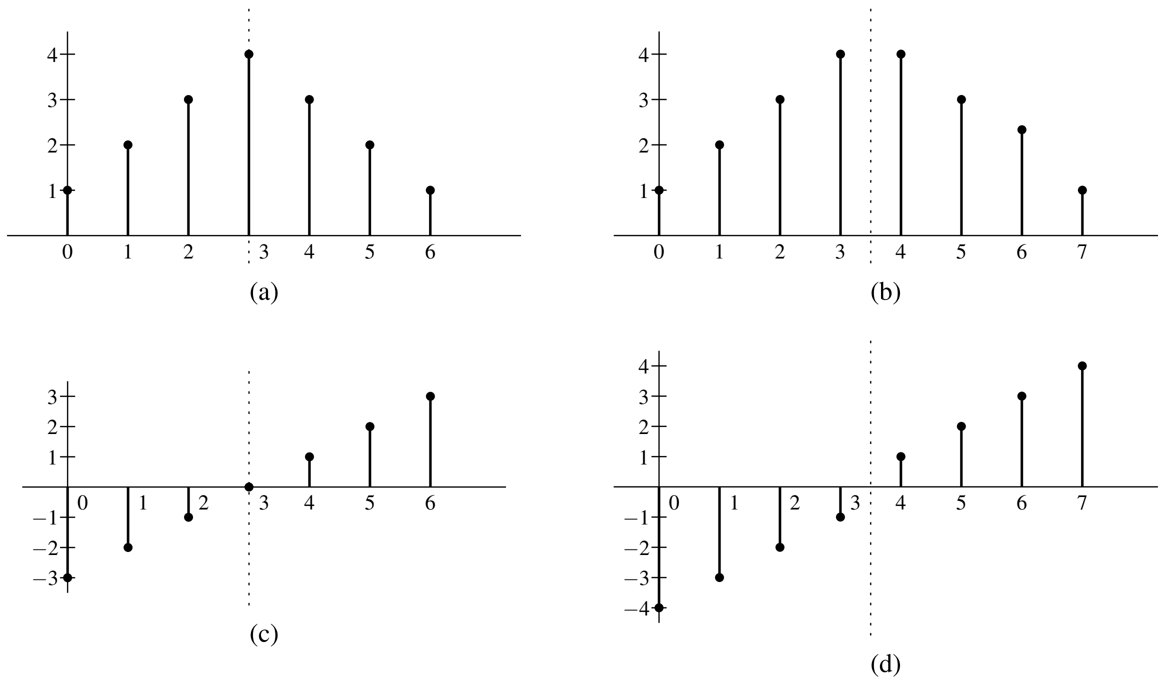


Figure 3.58: Examples of sequences with symmetry. An example of a sequence that is (a) WS symmetric, (b) HS symmetric, (c) WS antisymmetric, and (d) HS antisymmetric.

Let x be a (finite-extent) sequence defined on $\{0,1,\ldots,N-1\}$ (i.e., a sequence of length N). From x, we can construct a sequence defined on $\mathbb{Z}$ that is both periodic and symmetric. This is accomplished by applying a symmetry operation to the samples in x and then repeating the resulting samples periodically. In doing so, we have some flexibility in the number of times that we choose to repeat the first and last samples of x. To indicate that the first and last samples are repeated a and b times, respectively, we refer to the extension process as type (a,b). Here, we will consider the case of (a,b)-type extension for $a,b \in \{0,1\}$. This leads to four distinct symmetric extensions of x. Two cases are illustrated in Figure 3.59. In more precise terms, the (a,b)-type symmetric extension y of x is given by

$$y[n] = x\left[\min\left\{\operatorname{mod}(n,P), P-a-\operatorname{mod}(n,P)\right\}\right], \tag{3.97}$$

where $P = \max\{2N+a+b-2, 1\}$. Note that y is P-periodic and symmetric about 0 if $a=0$ or $-\frac{1}{2}$ if $a=1$.

Suppose that we have a sequence that is both periodic and has symmetry. For an N-periodic sequence with symmetry, the symmetry points always occur in pairs within each period. In particular, if one symmetry point is c_0, the other symmetry point c_1 within same period (with $c_1 > c_0$) is given by $c_1 = c_0 + \frac{N}{2}$. From this, it follows that:

- if N is even, the symmetry points are either both WS or both HS; and
- if N is odd, one symmetry point is WS and the other is HS.

This leads to six distinct symmetry types for periodic sequences as illustrated in Figure 3.60. Due to periodicity and symmetry, an N-periodic sequence with symmetry can be characterized by only a finite number N' of its samples, where $N' \approx \frac{1}{2}N$. The set of indices for the independent samples can always be chosen as a set of consecutive integers. In particular, given the symmetry point c, the indices of a complete set of independent samples is given by:

- $\lceil c \rceil, \lceil c \rceil + 1, \ldots, \left\lfloor c + \frac{N}{2} \right\rfloor$ if the sequence is symmetric; and
- $\left\lceil c + \frac{1}{2} \right\rceil, \left\lceil c + \frac{1}{2} \right\rceil + 1, \ldots, \left\lfloor c + \frac{N-1}{2} \right\rfloor$ if the sequence is antisymmetric.

Depending on the parity (i.e., evenness/oddness) of N and type of symmetry, several cases are possible. For each of these cases, the indices of the independent samples and N' are listed in Table 3.1.

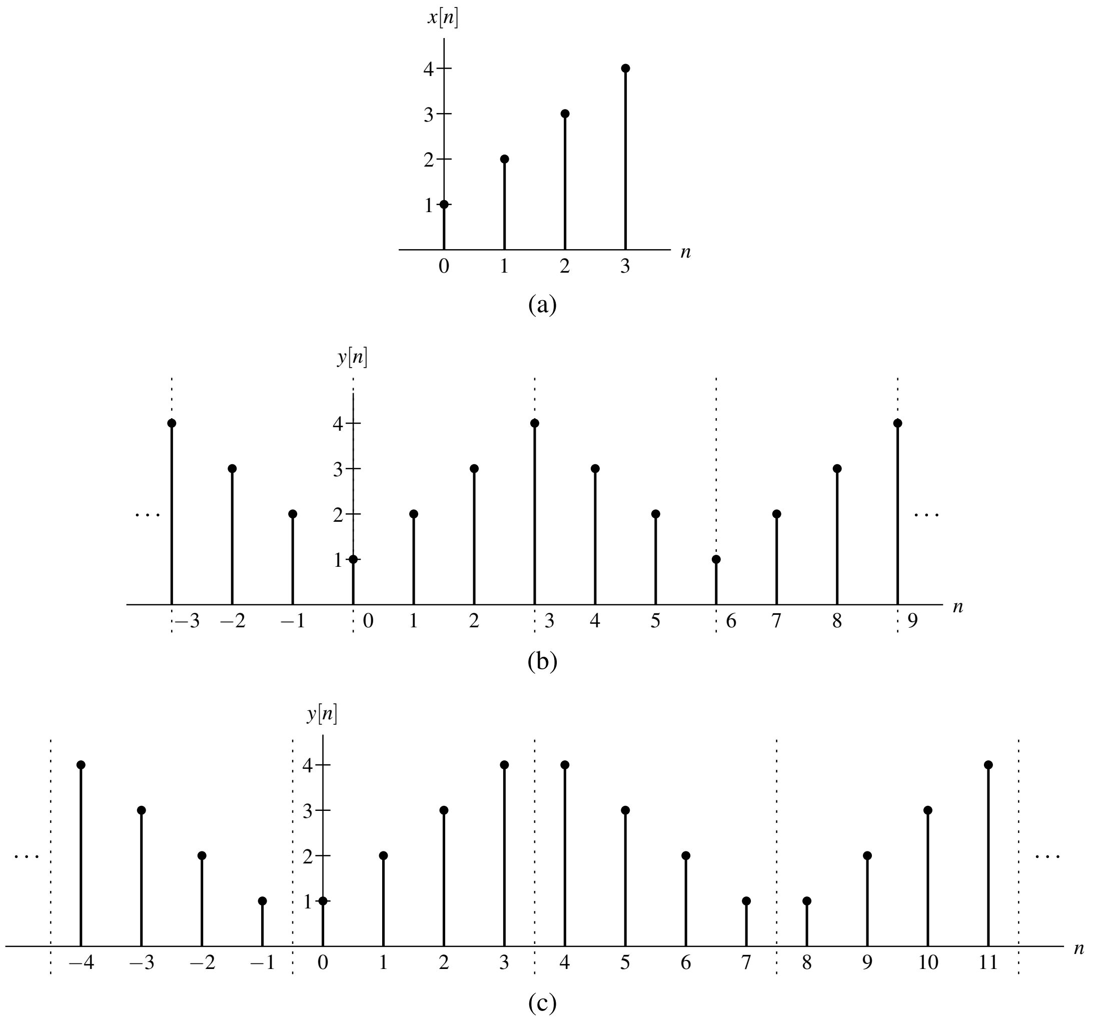


Figure 3.59: Symmetric extension example. (a) Original sequence and the result produced by applying (b) (0,0) and (c) (1,1) symmetric extension.

Table 3.1: The indices of the independent samples and their number for the various cases

N	Symmetry	c_0	c_1	Indices of Independent Samples	N'
even	symmetric or antisymmetric	HS	HS	$c_k+\frac{1}{2}, c_k+\frac{3}{2}, \ldots, c_k+\frac{N-1}{2}$	$\frac{N}{2}$
even	symmetric	WS	WS	$c_k, c_k+1, \ldots, c_k+\frac{N}{2}$	$\frac{N+2}{2}$
odd	symmetric	WS	HS	$c_0, c_0+1, \ldots, c_0+\frac{N-1}{2}$	$\frac{N+1}{2}$
				$c_1+\frac{1}{2}, c_1+\frac{3}{2}, \ldots, c_1+\frac{N}{2}$	
even	antisymmetric	WS	WS	$c_k+1, c_k+2, \ldots, c_k+\frac{N-2}{2}$	$\frac{N-2}{2}$
odd	antisymmetric	WS	HS	$c_0+1, c_0+2, \ldots, c_0+\frac{N-1}{2}$	$\frac{N-1}{2}$
				$c_1+\frac{1}{2}, c_1+\frac{3}{2}, \ldots, c_1+\frac{N-2}{2}$	

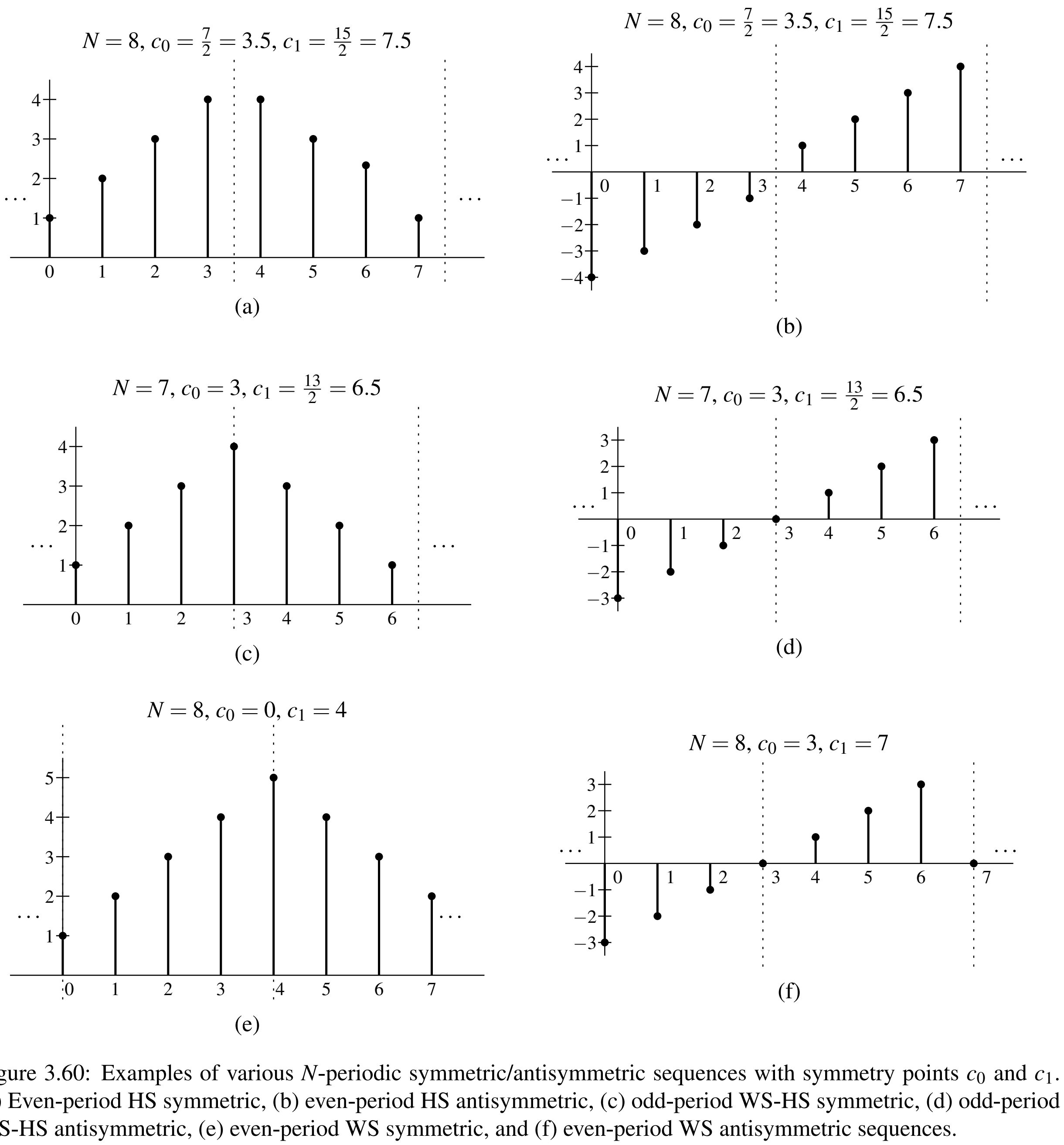
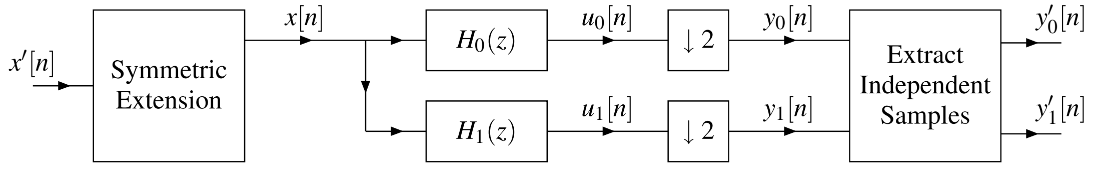
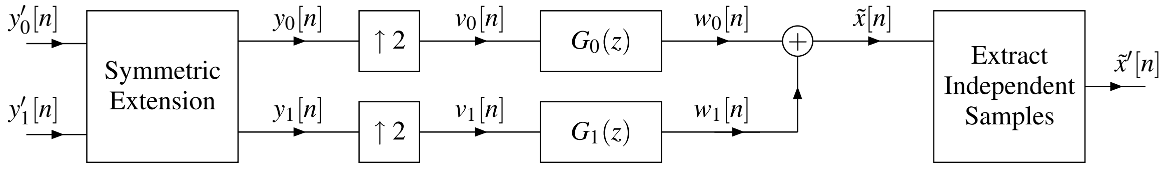


Figure 3.60: Examples of various N-periodic symmetric/antisymmetric sequences with symmetry points c_0 and c_1. (a) Even-period HS symmetric, (b) even-period HS antisymmetric, (c) odd-period WS-HS symmetric, (d) odd-period WS-HS antisymmetric, (e) even-period WS symmetric, and (f) even-period WS antisymmetric sequences.



(a)



(b)

Figure 3.61: Filter bank with symmetric extension. (a) Analysis and (b) synthesis sides.

Lastly, we introduce a few basic facts regarding the effects of downsampling and convolution on periodic sequences with symmetry. These facts are stated in the theorems below.

Theorem 3.24. *If a symmetric or antisymmetric sequence with the symmetry point c is downsampled by the factor M and $\frac{c}{M} \in \frac{1}{2}\mathbb{Z}$, then the resulting downsampled sequence is symmetric or antisymmetric, respectively, with the symmetry point $\frac{c}{M}$.*

Theorem 3.25. *If a symmetric sequence with the symmetry point c is input to a symmetric or antisymmetric filter with group delay d, then the output is symmetric or antisymmetric, respectively, with the symmetry point $c+d$.*

With the preceding background in place, we can now present the symmetric-extension method. The symmetric-extension method for handing finite-extent sequences with filter banks was first proposed by Smith and Eddins [27] and has subsequently received considerable attention (e.g., [2, 4, 5, 17, 19, 21, 22, 23, 28]). With this method, the input to the filter bank is extended to be both periodic and symmetric. Then, both the symmetry and periodicity properties are exploited to obtain a transform that is nonexpansive. The method can only be applied in the case of filter banks with filters having linear phase.

Consider the two-channel filter bank as shown in Figure 3.54, where $M = 2$. With the symmetric-extension method, we add preprocessing and postprocessing steps to each of the analysis and synthesis sides of the filter bank to obtain the modified system depicted in Figure 3.61. In what follows, we will consider only on the analysis side (as shown in Figure 3.61(a)), since the case of the synthesis side (shown in Figure 3.61(b)) involves the same principles.

Consider the analysis side of the structure shown in Figure 3.61(a). Suppose that we have a finite-length sequence x' defined on $\{0,1,\ldots,N-1\}$ (i.e., a sequence of length N). The basic idea behind the symmetric-extension method is as follows. We choose the input x to analysis side of filter bank as the symmetric extension of x'. Thus, x is periodic with a period of approximately $2N$. Recall that the analysis filters must have linear phase (i.e., their impulse responses have symmetry). Furthermore, we assume that their group delays satisfy certain constraints (to be introduced later). From the periodicity and symmetry of x, the symmetry of h_k, and Theorem 3.25, each u_k is symmetric/antisymmetric and is periodic with a period of approximately $2N$. We can conclude, from Theorem 3.24 and our assumption regarding the analysis-filter group delays, that each subband signal y_k is symmetric/antisymmetric and periodic with a period of approximately N. Since each subband signal y_k has symmetry, only about one half of the samples in a single period are required to characterize the sequence. In other words, we need approximately $\frac{N}{2}$ samples from each subband signal, for a total of approximately $2(\frac{N}{2}) = N$ samples. Thus, subject to the assumptions indicated above, we have a transform that is nonexpansive. Note that words like "approximately" frequently appear in the preceding discussion. This is because some technical details have been omitted in order to simplify the initial presentation of the method. Now, we will consider the method in more detail.

Table 3.2: The number of independent samples (for the case of odd-length filters)

N	$N-1$	Symmetry		N_0	N_1	N'
		y_0	y_1			
even	odd	WS-HS	HS-WS	$\frac{(N-1)+1}{2} = \frac{N}{2}$	$\frac{(N-1)+1}{2} = \frac{N}{2}$	N
even	odd	HS-WS	WS-HS	$\frac{(N-1)+1}{2} = \frac{N}{2}$	$\frac{(N-1)+1}{2} = \frac{N}{2}$	N
odd	even	WS	HS	$\frac{(N-1)+2}{2} = \frac{N+1}{2}$	$\frac{N-1}{2}$	N
odd	even	HS	WS	$\frac{N-1}{2}$	$\frac{(N-1)+2}{2} = \frac{N+1}{2}$	N

Note: The symmetry points for y_k are $\frac{\lambda+\gamma_k}{2}$ and $\frac{\lambda+\gamma_k+N-1}{2}$.

Recall Theorem 3.17 (on page 148). For a two-channel PR filter bank with linear-phase filters (having practically useful frequency responses), the filter lengths must be all even or all odd. Moreover, the analysis filters must be such that

1. both are of odd length and symmetric; or
2. both are of even length, with one being symmetric and the other being antisymmetric.

So, we consider each of these possibilities in turn.

CASE OF ODD-LENGTH FILTERS. First, we consider the case of odd-length analysis filters. Let γ_k denote the group delay of analysis filter H_k. In the case of odd-length filters, H_0 and H_1 are both symmetric and $\gamma_k \in \mathbb{Z}$. Without loss of generality, we assume that γ_0 and γ_1 have opposite parity (i.e., γ_0 and γ_1 are such that one is odd and one is even). Consider the analysis side of the filter bank in Figure 3.61(a). We choose the input x to the filter bank as the following $(0,0)$ symmetric extension of x':

$$x[n] = x'\left[\min\{\operatorname{mod}(n-\lambda, 2N-2), 2N-2-\operatorname{mod}(n-\lambda, 2N-2)\}\right] \quad \text{for all } n \in \mathbb{Z},$$

where $\lambda \in \mathbb{Z}$. For greater flexibility, we allow the extended sequence to be shifted via the parameter λ. (The preceding equation is identical to (3.97) when $\lambda = 0$.) We have that x is $(2N-2)$-periodic and symmetric with a WS symmetry point at λ. From the properties of convolution, u_k is $(2N-2)$-periodic with symmetry center $\lambda + \gamma_k \in \mathbb{Z}$. From the properties of downsampling, y_k is $(N-1)$-periodic with symmetry center $\frac{\lambda+\gamma_k}{2} \in \frac{1}{2}\mathbb{Z}$. Observe that one of $\{\frac{\lambda+\gamma_k}{2}\}_{k=0}^{1}$ is in $\mathbb{Z}$ and one is in $\frac{1}{2}\mathbb{Z}_{\mathsf{odd}}$ (i.e., one is WS and one is HS). Let N_0 and N_1 denote the number of samples needed to completely characterize y_0 and y_1, respectively, and let $N' = N_0 + N_1$. Using Table 3.1, we can find the indices of a set of independent samples for each of $\{y_k\}_{k=0}^{1}$ in order to determine $\{N_k\}_{k=0}^{1}$ and N'. The results obtained are shown in Table 3.2. From this table, we can see that $N' = N$ in every case. Thus, a nonexpansive transform is obtained.

CASE OF EVEN-LENGTH FILTERS. Next, we consider the case of even-length analysis filters. Let γ_k denote the group delay of analysis filter H_k. Without loss of generality, we assume that H_0 is symmetric, H_1 is antisymmetric, $\gamma_k \in \frac{1}{2}\mathbb{Z}_{\mathsf{odd}}$, and $\gamma_0 - \gamma_1 \in \mathbb{Z}$. Consider the analysis side of the filter bank in Figure 3.61(a). We choose the input x to the filter bank as the following $(1,1)$ symmetric extension of x':

$$x[n] = x'\left[\min\{\operatorname{mod}(n-\lambda, 2N), 2N-1-\operatorname{mod}(n-\lambda, 2N)\}\right] \quad \text{for all } n \in \mathbb{Z},$$

where $\lambda \in \mathbb{Z}$. (Again, the parameter λ simply introduces a shift in the extended sequence.) We have that x is $2N$-periodic and HS symmetric about $\lambda - \frac{1}{2}$. From properties of convolution, u_k is $2N$-periodic and WS symmetric/antisymmetric about $\frac{2\lambda-1}{2} + \gamma_k = \lambda + \gamma_k - \frac{1}{2} \in \mathbb{Z}$. From properties of downsampling, y_k is N-periodic and WS or HS symmetric/antisymmetric about $\frac{2\lambda+2\gamma_k-1}{4} \in \frac{1}{2}\mathbb{Z}$. Observe that $\{\frac{2\lambda+2\gamma_k-1}{4}\}_{k=0}^{1}$ are both even multiples or both odd multiples of $\frac{1}{2}$. Let N_0 and N_1 denote the number of samples needed to completely characterize y_0 and y_1, respectively, and let $N' = N_0 + N_1$. Using Table 3.1, we can find the indices of a set of independent samples for each of $\{y_k\}_{k=0}^{1}$ in order to determine $\{N_k\}_{k=0}^{1}$ and N'. The results obtained are shown in Table 3.3. From this table, we can see that $N' = N$ in every case. Thus, a nonexpansive transform is obtained.

Since we have shown that a nonexpansive transform is obtained in the case of even-length and odd-length filters, the symmetric-extension method clearly works in all cases.

Table 3.3: The number of independent samples (for the case of even-length filters)

N	Symmetry y_0	Symmetry y_1	N_0	N_1	N'
even	HS symmetric	HS antisymmetric	$\frac{N}{2}$	$\frac{N}{2}$	N
even	WS symmetric	WS antisymmetric	$\frac{N+2}{2}$	$\frac{N-2}{2}$	N
odd	WS-HS symmetric	WS-HS antisymmetric	$\frac{N+1}{2}$	$\frac{N-1}{2}$	N
odd	HS-WS symmetric	HS-WS antisymmetric	$\frac{N+1}{2}$	$\frac{N-1}{2}$	N

Note: The symmetry points for y_k are $\frac{2\lambda+2\gamma_k-1}{4}$ and $\frac{2\lambda+2\gamma_k+2N-1}{4}$.

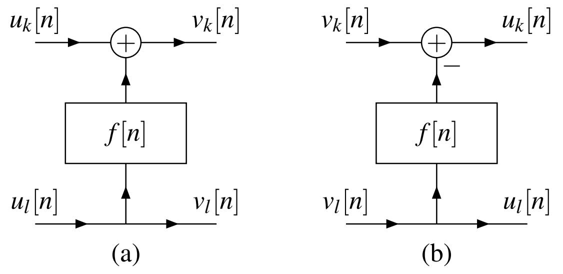


Figure 3.62: Lifting step. (a) A lifting step and (b) its inverse.

The symmetric-extension method has a number of desirable characteristics. First, it does not introduce as much high-frequency spectral content as periodic extension, since large jumps are always avoided at the splice points between periods. Second, it works regardless of the parity (i.e., evenness/oddness) of N. Unfortunately, the method also has some disadvantages. First, it requires that the filters have linear phase and satisfy certain constraints on their group delays. Second, the method does not extend cleanly to the M-channel case, where $M > 2$.

3.9.4 Per-Lifting-Step Extension Method

Another alternative for handling finite-extent sequences with filter banks is offered by the per-lifting-step (PLS) extension method. This particular technique is used in conjunction with the lifting realization of a filter bank. As it turns out, the lifting realization can very easily accommodate finite-extent sequences.

Recall that the lifting realization (introduced earlier in Section 3.6.3) makes use of two types of polyphase filtering networks, namely, lifting and scaling steps. These networks appear earlier in Figures 3.47 and 3.48. As far as finite-extent sequences are concerned, the scaling steps pose no problems. The lifting steps are potentially problematic, however, as they involve convolution. So, we must find a way in which to address the boundary problem during convolution in each lifting step.

The polyphase filtering networks for a lifting step and its inverse are shown in Figure 3.62. Consider the network for the lifting step shown in Figure 3.62(a). The lifting step is characterized by the equation

$$v_k[n] = u_k[n] + (u_l * f)[n]. \tag{3.98}$$

When computing $u_l * f$ in the preceding equation, we typically need to evaluate u_l at points outside of its domain. This leads to the so called boundary problem. This problem, however, is easily rectified. Let E denote an arbitrary signal-extension operator that maps a sequence defined on a finite subset of $\mathbb{Z}$ to a sequence defined on $\mathbb{Z}$. For example, E could be chosen as zero extension (3.95), periodic extension (3.96), or symmetric extension (3.97), amongst many other possibilities as well. We can then replace the computation in (3.98) by

$$v_k[n] = u_k[n] + (E(u_l) * f)[n].$$

In other words, we extend the sequence u_l before computing the convolution. In so doing, we avoid the boundary problem. Due to the symmetry in the analysis and synthesis sides of the lifting realization of a filter bank, as long as we employ the same extension operator E for corresponding lifting steps on the analysis and synthesis sides, the PR property will be maintained. The above approach also leads to nonexpansive transforms, since each lifting step preserves the total number of samples being processed.

In passing, we note that instead of extending u_l in (3.98), we could have modified f so as to avoid the boundary problem. This approach also would also maintain the PR property of the filter bank as long as the same filter-modification strategy is applied to corresponding lifting steps on the analysis and synthesis sides of the filter bank.

As it turns out, in some cases, by making an appropriate choice of the extension operator, one can achieve identical behavior to the symmetric-extension method discussed earlier. That is, for some filter banks, the symmetric-extension method can be implemented using the PLS-extension scheme [2].

The PLS-extension method has numerous desirable attributes. First, it is conceptually simple and relatively easy to implement. Second, it can be applied to any filter bank that has a lifting realization, regardless of the number of channels. Third, it provides great flexibility, since the extension operator can be chosen arbitrarily. Perhaps, the only disadvantage of the PLS-extension method is that is requires the use of the lifting realization. This restriction, however, is not so burdensome, since the lifting realization is often used in practice (due to its many advantages).

3.10 Filter Banks and Multidimensional Signals

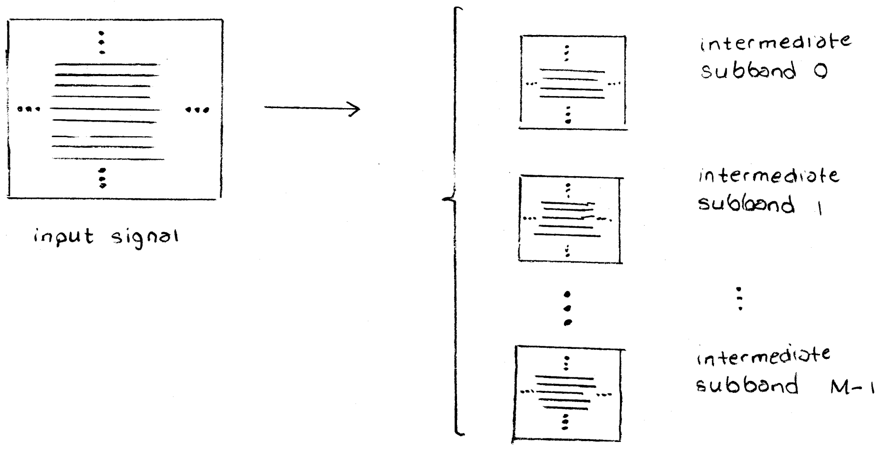
The UMD filter banks that we have considered so far are fundamentally one-dimensional in nature. That is, the filter banks are only capable of processing one-dimensional signals. Many types of signals, however, are multidimensional in nature (e.g., images, video, volumetric medical data, and so on). Clearly, a means is needed for handling such signals. For this reason, one might wonder how we can construct a filter bank that is capable of processing such multidimensional signals.

As it turns out, the easiest way in which to construct a multidimensional filter bank is from one-dimensional building blocks. In other words, we construct a multidimensional filter bank as a composition of one-dimensional filter banks. Or put another way, we view a multidimensional signal as being comprised of one-dimensional slices, which are then processed with one-dimensional filter banks. For example, consider a two-dimensional signal x. The kth (one-dimensional) horizontal slice of x is given by $x_{\mathrm{h},k}[n] = x[n,k]$ and the kth (one-dimensional) vertical slice of x is given by $x_{\mathrm{v},k}[n] = x[k,n]$. To begin, we apply a (one-dimensional) filter bank to each horizontal slice of x. For each horizontal slice that is processed, M one-dimensional subband signals are produced. Then, for each channel, we vertically stack the one-dimensional subband signals to produce M two-dimensional signals. Next, for each of the M two-dimensional subband signals, we apply the (one-dimensional) filter bank to each vertical slice. This yields M new subband signals for each of the M original subband signals, for a total of M^2 subbands. This process is illustrated in Figure 3.63.

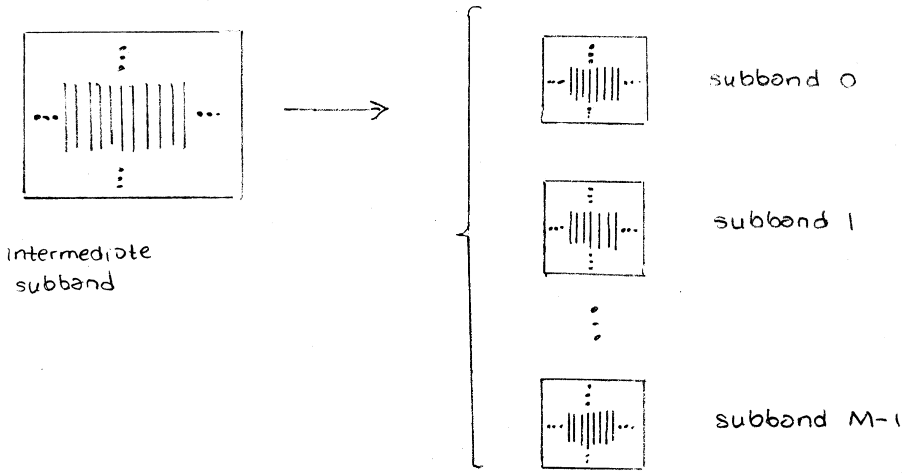
In effect, we have constructed the M^2-channel two-dimensional filter bank, shown in Figure 3.64 where $\mathfrak{J}(M) = M^2$. The filters $\{H_k\}_{k=0}^{M^2-1}$ and $\{G_k\}_{k=0}^{M^2-1}$ employ two-dimensional filtering operations that are composed from one-dimensional operations, and the two-dimensional downsamplers/upsamplers are composed from one-dimensional downsamplers/upsamplers.

Although we have considered the two-dimensional case here for simplicity, this idea trivially extends to any arbitrary number of dimensions. More specifically, we construct a D-dimensional filter bank by applying a one-dimensional filter bank in each of the D dimensions. The resulting D-dimensional filter bank has M^D channels (and the M^D subband signals are each D dimensional). where the analysis/synthesis filtering operations and downsampling/upsampling operations are defined in D dimensions (as a composition of one-dimensional operations). This system is shown in Figure 3.64, where $\mathfrak{J}(M) = M^D$. The analysis/synthesis filters $\{H_k\}_{k=0}^{M^D-1}$ and $\{G_k\}_{k=0}^{M^D-1}$ correspond to D-dimensional filtering operations composed from one-dimensional operations, and the D-dimensional downsampling/upsampling operations are composed from one-dimensional operations.

The above approach to constructing multidimensional filter banks has a number of desirable characteristics. It is conceptually simple and easy to analyze, requiring only one-dimensional signal-processing theory. The approach is also computationally efficient, as all operations are fundamentally one-dimensional in nature. Although the above approach is adequate for many applications, it also has some significant disadvantages. First, since the multidimensional filter bank is composed from one-dimensional operations, it cannot exploit the true multidimensional nature of the signal being processed. Second, the flexibility in the partitioning of the frequency-domain into subbands is quite



(a)



(b)

Figure 3.63: Processing two-dimensional signals with one-dimensional filter banks. Processing of (a) horizontal and (b) vertical slices.

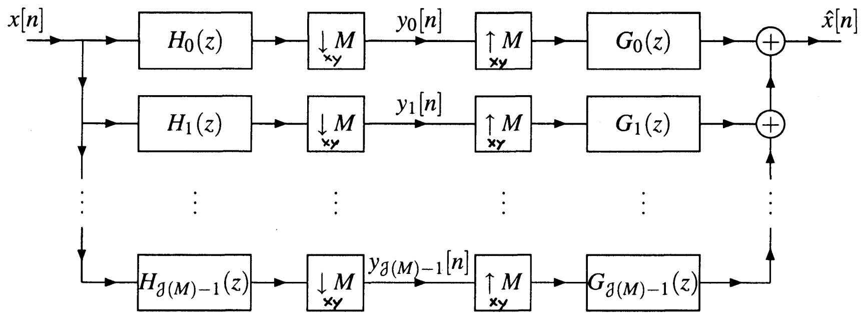


Figure 3.64: Separable multidimensional UMD filter bank.

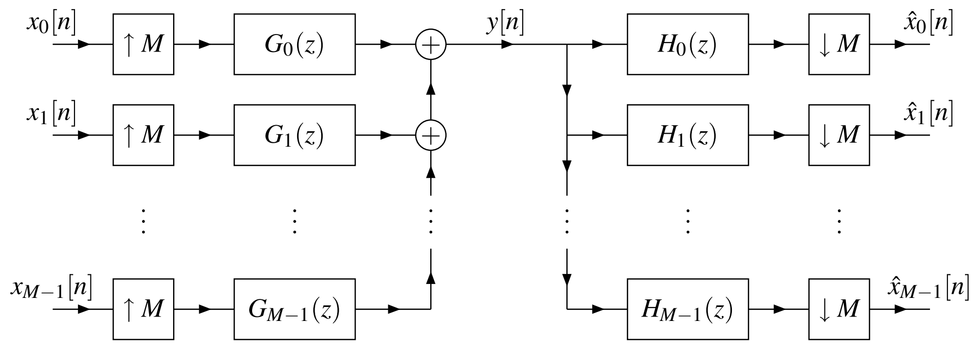


Figure 3.65: The M-channel transmultiplexer.

limited. Third, the number of channels possessed by the multidimensional filter bank is constrained to be M^D. Sometimes, this restriction on the number of channels might be overly constraining. For example, the number of channels can become quite large, depending on D and M. In particular, the number of channels increases exponentially with D and as a Dth-order polynomial with M (e.g., if $D = 2$, number of channels increases quadratically with M).

3.11 Transmultiplexers

Although the primary focus of this chapter is UMD filter banks, we would like to take a moment to briefly introduce another type of multirate system, called a transmultiplexer, that is very closely related to a UMD filter bank. A transmultiplexer is simply the system obtained by swapping the analysis and synthesis sides of a UMD filter bank. More specifically, the general structure of an **M-channel transmultiplexer** is shown in Figure 3.65. The synthesis side of the transmultiplexer (on the left side of the figure) multiplexes M signals $\{x_k\}_{k=0}^{M-1}$ onto single signal y, and the analysis side (on the right side of the figure) demultiplexes y into M signals $\{\hat{x}_k\}_{k=0}^{M-1}$. As a matter of terminology, a transmultiplexer is said to have the **PR property** if, for each $k \in \{0, 1, \ldots, M-1\}$,

$$\hat{x}_k[n] = x_k[n - d_k]$$

for some $d_0, d_1, \ldots, d_{M-1} \in \mathbb{Z}$ and all $n \in \mathbb{Z}$. One can show that a transmultiplexer has the PR property if its corresponding UMD filter bank has the PR property.

3.12 Applications of UMD Filter Banks and Other Multirate Systems

Although we have considered several types of multirate systems in this chapter, our primary focus has been UMD filter banks. An M-channel UMD filter bank can be used to decompose a signal into M frequency bands (determined by the frequency responses of the analysis filters $\{H_k\}_{k=0}^{M-1}$), as shown in Figure 3.66. Therefore, a filter bank is potentially useful whenever it is beneficial to process a signal in terms of its different frequency bands. In practice, we usually have some processing block inserted between the analysis and synthesis sides of the filter bank, as shown in Figure 3.67. Often a PR system is desired so that any difference between x and $\hat{x}$ is due to subband processing and not distortion caused by the filter bank itself. Some applications of filter banks are explored in Chapter 10. As it turns out, filter banks are also a basic building block in the computational structure for the discrete wavelet transform (DWT).

In addition to UMD filter banks, we have also discussed sampling-rate converters. In practice, many different sampling rates are often used for a particular type of data. For example, in the case of audio data, some commonly-used sampling rates include:

- studio recording: 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, 192 kHz
- MPEG-1 Audio Layer 3 (MP3): 44.1 kHz (typical), 32 kHz, 48 kHz

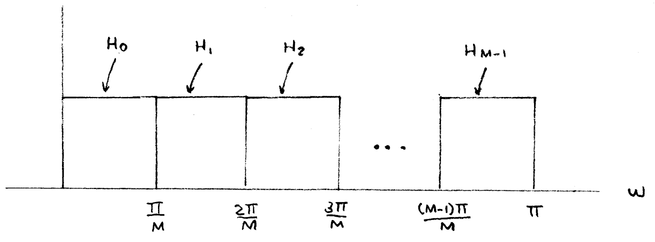


Figure 3.66: Partitioning of a signal into its frequency bands.

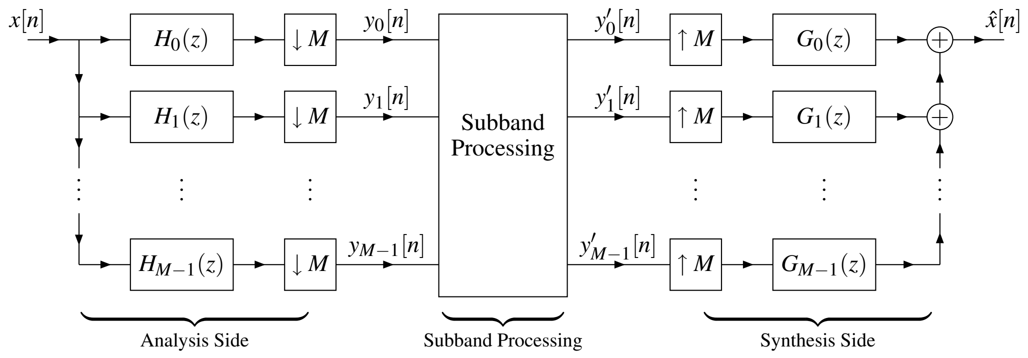


Figure 3.67: A UMD filter bank with processing performed on subband signals.

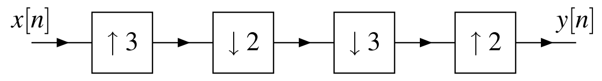
- Digital Audio Tape (DAT): 48 kHz (typical), 44.1 kHz, 32 kHz
- Compact Disc (CD): 44.1 kHz
- DVD Audio: 44.1 kHz, 192 kHz
- GSM-FR: 8 kHz

Since many different sampling rates are used, a means is needed to convert between sampling rates. In earlier sections, we examined how to efficiently perform sampling rate conversion when the conversion factor is rational (or integer).

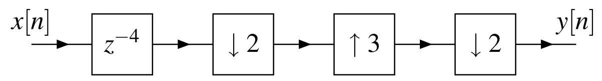
Lastly, we also briefly introduced transmultiplexers. Since transmultiplexers can be used to perform signal multiplexing, they are extremely useful in communication systems. Applications of transmultiplexers are examined in more depth in Chapter 10.

3.13 Problems

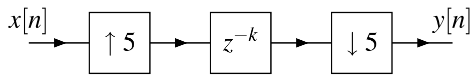
3.1 For the system shown in each of the figures below, find an expression for $y[n]$ in terms of $x[n]$. Use multirate identities in order to simplify your solution.



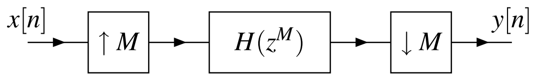
(a)



(b)



(c)



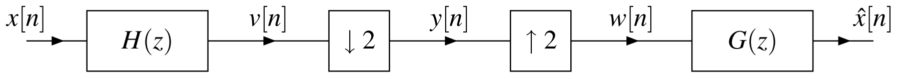
(d)

3.2 Suppose that we have the system shown in the figure (a) below, where

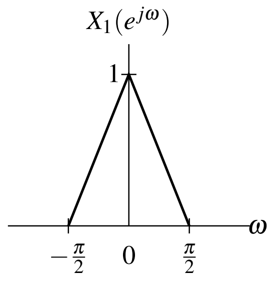
$$H(e^{j\omega}) = \begin{cases} 1 & \text{for } 0 \le |\omega| < \frac{\pi}{2} \\ 0 & \text{for } \frac{\pi}{2} \le |\omega| \le \pi \end{cases} \quad \text{and}$$

$$G(e^{j\omega}) = \begin{cases} 2 & \text{for } 0 \le |\omega| < \frac{\pi}{2} \\ 0 & \text{for } \frac{\pi}{2} \le |\omega| \le \pi. \end{cases}$$

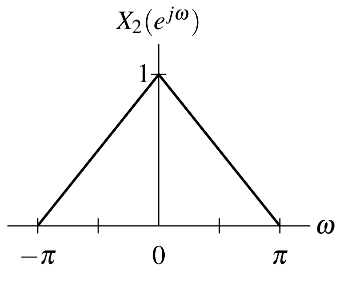
Let $X(e^{j\omega})$, $V(e^{j\omega})$, $Y(e^{j\omega})$, $W(e^{j\omega})$, and $\hat{X}(e^{j\omega})$ denote the Fourier transforms of $x[n]$, $v[n]$, $y[n]$, $w[n]$, and $\hat{x}[n]$, respectively. Let $X_1(e^{j\omega})$ and $X_2(e^{j\omega})$ be as given in figures (b) and (c), respectively. Sketch $V(e^{j\omega})$, $Y(e^{j\omega})$, $W(e^{j\omega})$, and $\hat{X}(e^{j\omega})$ if $X(e^{j\omega})$ is equal to each of the following:
(a) $X(e^{j\omega}) = X_1(e^{j\omega})$; and
(b) $X(e^{j\omega}) = X_2(e^{j\omega})$.



(a)

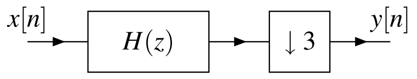


(b)

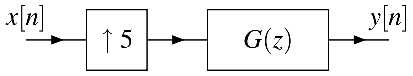


(c)

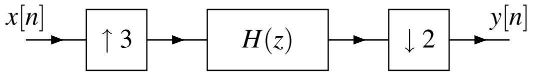
3.3 (a) For the decimator shown in the figure below, sketch the frequency response $H(e^{j\omega})$ of the ideal filter that prevents aliasing.



(b) For the interpolator shown in the figure below, sketch the frequency response $G(e^{j\omega})$ of the ideal filter that eliminates imaging.



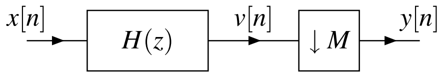
(c) Determine the frequency response $H(e^{j\omega})$ of the ideal lowpass filter H necessary to avoid aliasing and imaging in the system shown in the figure below.



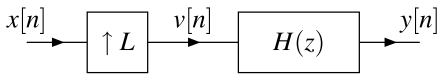
3.4 A system is said to be (N_2, N_1)-periodically time varying if shifting the input by N_1 shifts the output by N_2 for each possible input.
(a) Show that the M-fold downsampler is a linear $(1, M)$-periodically-time-varying system.
(b) Show that the M-fold upsampler is a linear $(M, 1)$-periodically-time-varying system.

3.5 Let $x[n]$ be periodic with (fundamental) period N. Let $y[n]$ be the M-fold downsampled version of $x[n]$ (i.e., $y[n] = x[Mn]$).
(a) Show that $y[n]$ is periodic.
(b) In the absence of any further knowledge about $x[n]$, find the smallest period L (expressed in terms of M and N) for which $y[n]$ is periodic.

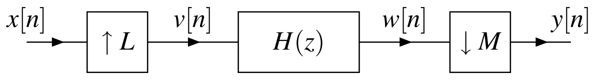
3.6 For each of the systems below with input x and output y, find an expression for y in terms of x. (The expression should be in the time domain.)
(a) an M-fold decimator (as shown in figure (a) below);
(b) an L-fold interpolator (as shown in figure (b) below); and
(c) a $\frac{M}{L}$-fold decimator/interpolator (as shown in figure (c) below).



(a)



(b)



(c)

3.7 Show that the relationship between the input and output of a two-fold downsampler can be expressed in matrix form as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output, respectively. Describe the form of $\boldsymbol{A}$. The relationship between the input and output of a two-fold upsampler can also be expressed in matrix form as $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$. Describe the relationship between $\boldsymbol{A}$ and $\boldsymbol{B}$.

3.8 Determine the matrix representation of the following operators:
(a) $(\uparrow 2)(\downarrow 2)$;
(b) $(\downarrow 2)(\uparrow 2)$; and
(c) $(\uparrow 3)(\downarrow 3)$.

3.9 In each of the cases below, find the specified polyphase representation of the given filter with the transfer function $F(z)$. Be sure to explicitly identify the polyphase components in each case.
(a) $F(z) = \frac{z^3+3z^2+2z-1}{8z^4+6z^2+1}$, type-1 two-phase;
(b) $F(z) = -\frac{1}{8}z^2 + \frac{1}{4}z + \frac{3}{4} + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}$, type-3 two-phase;
(c) $F(z) = \frac{3}{128}z^5 + \frac{3}{128}z^4 - \frac{22}{128}z^3 - \frac{22}{128}z^2 + z - 1 + \frac{22}{128}z^{-1} + \frac{22}{128}z^{-2} - \frac{3}{128}z^{-3} - \frac{3}{128}z^{-4}$, type-2 three-phase;
(d) $F(z) = \frac{30}{128} + \frac{73}{128}z^{-1} + \frac{42}{128}z^{-2} - \frac{12}{128}z^{-3} - \frac{8}{128}z^{-4} + \frac{3}{128}z^{-5}$, type-4 three-phase; and
(e) $F(z) = -\frac{3}{64}z^2 + \frac{15}{64}z + \frac{39}{64} + \frac{17}{64}z^{-1} - \frac{3}{64}z^{-2} - \frac{1}{64}z^{-3}$, type-1 two-phase.

3.10 Find the type-1 two-phase polyphase representation of the filters with each of the transfer functions given below. In your final answer, each polyphase component should be given in the form of a *single* rational expression.
(a) $F(z) = \dfrac{1}{1 - \frac{5}{6}z^{-1} + \frac{1}{6}z^{-2}}$;
(b) $F(z) = \dfrac{1+2z^{-1}}{1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}}$.
[Hint: Decompose the denominator of $F(z)$ into first order factors and then use the approach from Example 3.14.]

3.11 Find the type-1 four-phase polyphase representation of the filters with each of the following transfer functions:
(a) $F(z) = \dfrac{1}{1 - \frac{5}{6}z^{-1} + \frac{1}{6}z^{-2}}$;
(b) $F(z) = \dfrac{1+2z^{-1}+3z^{-2}}{1 - z^{-1} + \frac{3}{16}z^{-2}}$.
[Hint: Determine what polynomial multiplies $1-az^{-1}$ to produce $1-a^4z^{-4}$.]

3.12 Let $F(z)$ be the transfer function of a symmetric or antisymmetric filter. Suppose that $F(z)$ has the two-phase polyphase representation

$$F(z) = z^{m_0}F_0(z^2) + z^{m_1}F_1(z^2)$$

(where m_0 and m_1 are distinct modulo 2). Determine the symmetry properties (i.e., symmetry type and symmetry center) of each of the polyphase components $F_0(z)$ and $F_1(z)$ if:
(a) the filter F is symmetric and of even length;
(b) the filter F is symmetric and of odd length;
(c) the filter F is antisymmetric and of even length; and
(d) the filter F is antisymmetric and of odd length.
[Hint: A symmetric/antisymmetric sequence is of the form $f[n] = (-1)^s f[2c-n]$ where $s \in \{0,1\}$ and $2c \in \mathbb{Z}$. In particular, if f is symmetric then $s=0$, and if f is antisymmetric then $s=1$. If f has whole-sample symmetry then c is an integer, and if f has half-sample symmetry then c is an odd integer multiple of one half.] [Hint: It is probably easiest to consider all parts of the problem together, rather than separately.]

3.13 Determine the two-phase polyphase components of a symmetric half-band filter. Generalize the result to a symmetric M-th band filter.

3.14 Let $F(z)$ and $G(z)$ be transfer functions of filters. Suppose that $F(z)$ and $G(z)$ have the polyphase representations given by

$$F(z) = \sum_{k=0}^{M-1} z^k F_k(z^M) \quad \text{and} \quad G(z) = \sum_{k=0}^{M-1} z^k G_k(z^M).$$

Determine the relationship between the polyphase components of $F(z)$ and $G(z)$ if
(a) $G(z) = z^M F(z)$;
(b) $G(z) = zF(z)$;
(c) $G(z) = F(z^{-1})$; and
(d) $G(z) = F(az)$.

3.15 Let $\boldsymbol{H}_\mathsf{p}(z)$ and $\boldsymbol{G}_\mathsf{p}(z)$ denote the analysis and synthesis polyphase matrices, respectively, of a UMD filter bank. For each pair of polyphase matrices of the specified type given below, find the corresponding analysis and synthesis filters.
(a) type $(1,2)$, $\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$, $\boldsymbol{G}_\mathsf{p}(z) = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$;
(b) type $(3,1)$, $\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} \frac{1}{16}z+\frac{1}{2}-\frac{1}{16}z^{-1} & -\frac{1}{16}z+\frac{1}{2}+\frac{1}{16}z^{-1} \\ -1 & 1 \end{bmatrix}$, $\boldsymbol{G}_\mathsf{p}(z) = \begin{bmatrix} 1 & \frac{1}{16}z-\frac{1}{2}-\frac{1}{16}z^{-1} \\ 1 & \frac{1}{16}z+\frac{1}{2}-\frac{1}{16}z^{-1} \end{bmatrix}$; and
(c) type $(3,1)$, $\boldsymbol{H}_\mathsf{p}(z) = \begin{bmatrix} az+b & cz+d \\ -1 & 1 \end{bmatrix}$, $\boldsymbol{G}_\mathsf{p}(z) = \begin{bmatrix} 1 & e+fz^{-1} \\ 1 & g+hz^{-1} \end{bmatrix}$.

3.16 Let $H_k(z)$ and $G_k(z)$ denote the analysis and synthesis filter transfer functions of a UMD filter bank. For each of the systems given below, find the analysis and synthesis polyphase matrices, $\boldsymbol{H}_\mathsf{p}(z)$ and $\boldsymbol{G}_\mathsf{p}(z)$, respectively, for the specified polyphase representation.
(a) $H_0(z) = z+1$, $H_1(z) = z-1$, $G_0(z) = 1+z^{-1}$, $G_1(z) = -1+z^{-1}$, type $(1,2)$;
(b) $H_0(z) = \frac{1}{2}z+\frac{1}{2}$, $H_1(z) = \frac{3}{128}z^5+\frac{3}{128}z^4-\frac{22}{128}z^3-\frac{22}{128}z^2+z-1+\frac{22}{128}z^{-1}+\frac{22}{128}z^{-2}-\frac{3}{128}z^{-3}-\frac{3}{128}z^{-4}$, $G_0(z) = -z^{-1}H_1(-z)$, $G_1(z) = z^{-1}H_0(-z)$, type $(3,1)$;
(c) $H_0(z) = a_0+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}$, $H_1(z) = b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}$, $G_0(z) = H_0(z^{-1})$, $G_1(z) = H_1(z^{-1})$, type $(1,2)$.

3.17 Let $H_k(z)$ and $G_k(z)$ denote the analysis and synthesis filter transfer functions of a UMD filter bank. Find the analysis and synthesis modulation matrices, $\boldsymbol{H}_\mathsf{m}(z)$ and $\boldsymbol{G}_\mathsf{m}(z)$, respectively, for each of the systems given below.
(a) $H_0(z) = z+1$, $H_1(z) = z-1$, $G_0(z) = 1+z^{-1}$, $G_1(z) = 1+z^{-1}$,
(b) $H_0(z) = -\frac{1}{16}z^3+\frac{1}{16}z^2+\frac{1}{2}z+\frac{1}{2}+\frac{1}{16}z^{-1}-\frac{1}{16}z^{-2}$, $H_1(z) = z-1$, $G_0(z) = -z^{-1}H_1(-z)$, $G_1(z) = z^{-1}H_0(-z)$.

3.18 Let $H_0(z)$ and $H_1(z)$ denote the analysis filters of a 2-channel UMD filter bank. Let $\boldsymbol{H}_\mathsf{p}(z)$ denote the type-(3,1) analysis polyphase matrix. Let N denote the length of $H_0(z)$. Express $\boldsymbol{H}_\mathsf{p}(z)$ in terms of the polyphase components of $H_0(z)$ for each of the following cases:
(a) $H_1(z) = H_0(-z)$;
(b) $H_1(z) = z^{-(N-1)}H_0(z^{-1})$; and
(c) $H_1(z) = z^{-(N-1)}H_0(-z^{-1})$.

3.19 Consider a two-channel orthonormal FIR shift-free-PR UMD filter bank with lowpass analysis filter having the transfer function $H_0(z)$. Let $P(z)$ denote the $\mathcal{Z}$-transform of the autocorrelation of $\mathcal{Z}^{-1}H_0$ (i.e., $P(z) = H_0(z)H_0(z^{-1})$). One can show that $P(z)$ satisfies $P(z)+P(-z) = 2$. Use this fact to show that the length of H_0 must be even (aside from the trivial case when the length is one).

3.20 Prove that there are no real symmetric/antisymmetric orthonormal FIR 2-channel UMD filter banks, except the Haar/Haar-like case.

3.21 A two-channel orthonormal FIR UMD filter bank with real-valued filter coefficients cannot have linear-phase, except for the trivial two-tap (e.g., Haar) case. Show that other solutions exist if the filter coefficients can be complex valued.

3.22 For a two-channel linear-phase FIR PR UMD filter bank, show that:
(a) The analysis/synthesis filter lengths are either all odd or all even.
(b) The analysis filters must be both of odd length and symmetric, or both of even length with one symmetric and the other antisymmetric.
(c) If we let N_0 and N_1 denote the lengths of the analysis filters, then N_0+N_1+2 is divisible by four.

3.23 Consider a two-channel UMD filter bank with analysis filter transfer functions $H_0(z)$ and $H_1(z)$. Suppose that $H_0(z)$ corresponds to a symmetric lowpass filter with even length and $H_1(z) = H_0(-z)$.
(a) Verify that $H_1(z)$ corresponds to an antisymmetric highpass filter.
(b) Find the transfer functions, $G_0(z)$ and $G_1(z)$, of the synthesis filters that cancel aliasing.
(c) Determine whether the resulting system can have the PR property.

3.24 Suppose that we have a two-channel FIR UMD filter bank with analysis filters H_0 and H_1, where H_0 is symmetric and of even length, and H_1 is antisymmetric and of even length. Consider the four-channel tree-structured filter bank corresponding to the two-fold iteration of UMD filter bank (where iteration is done on the lowpass channel). Determine the lengths and symmetries of the filters of this new filter bank.

3.25 Let $H_k(z)$ for $k = 0, 1, \ldots, M-1$ denote the analysis filter transfer functions for an M-channel UMD filter bank. For each set of analysis filters given below, find a set of synthesis filters that result in a shift-free PR system. Indicate whether the resulting synthesis filters are stable (assuming that they have right-sided impulse responses). [Hint: Find the analysis polyphase matrix for a $(3,1)$ type polyphase decomposition. Then, find the corresponding synthesis polyphase matrix needed in order to achieve the shift-free PR property.]
(a) $H_0(z) = \frac{1}{\sqrt{2}}z + \frac{1}{\sqrt{2}}$, $H_1(z) = \frac{1}{\sqrt{2}}z - \frac{1}{\sqrt{2}}$
(b) $H_0(z) = 1$, $H_1(z) = 2 + z^{-1} + z^{-5}$, $H_2(z) = 3 + z^{-1} + 2z^{-2}$
(c) $H_0(z) = 1$, $H_1(z) = 2 + z^{-1} + z^{-5}$, $H_2(z) = 3 + 2z^{-1} + z^{-2}$.
(d) $H_0(z) = 1$, $H_1(z) = 2 + z^{-1}$, $H_2(z) = 3 + 2z^{-1} + z^{-2}$
(e) $H_0(z) = 2 + 6z^{-1} + z^{-2} + 5z^{-3} + z^{-5}$, $H_1(z) = H_0(-z)$
(f) $H_0(z) = \frac{1}{64}z^4 - \frac{8}{64}z^2 + \frac{16}{64}z + \frac{46}{64} + \frac{16}{64}z^{-1} - \frac{8}{64}z^{-2} + \frac{1}{64}z^{-4}$, $H_1(z) = \frac{1}{16}z^4 - \frac{9}{16}z^2 + z - \frac{9}{16} + \frac{1}{16}z^{-2}$.
(g) $H_0(z) = \frac{1}{3} + \frac{1}{3}z + \frac{1}{3}z^2$, $H_1(z) = -\frac{1}{3} + \frac{2}{3}z^2$, and $H_2(z) = -\frac{1}{3}z + \frac{2}{3}z^2$.

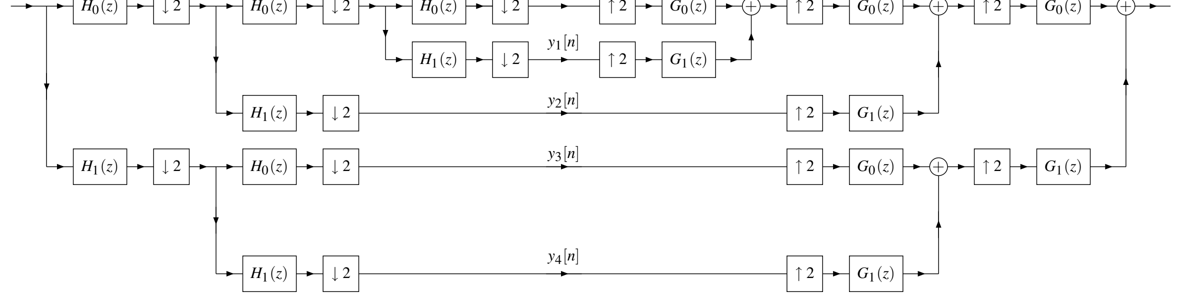
3.26 Suppose that we are given an alias-free M-channel UMD filter bank with distortion function $T(z)$, analysis filter transfer functions $\{H_k(z)\}$, and synthesis filter transfer functions $\{G_k(z)\}$. From this filter bank, we generate a new filter bank having analysis filter transfer functions $\{H'_k(z)\}$ and synthesis filter transfer functions $\{G'_k(z)\}$, where

$$H'_k(z) = G_k(z) \text{ and } G'_k(z) = H_k(z) \quad \text{for } k = 0, 1, \ldots, M-1.$$

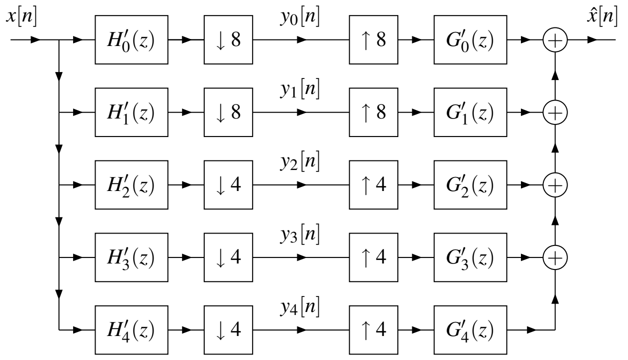
(In other words, the new filter bank is formed by exchanging the analysis and synthesis filters of the original filter bank.) Show that this new filter bank is also alias free with distortion function $T(z)$.

3.27 Suppose that we are given an M-channel UMD filter bank having analysis filter transfer functions $\{H_k(z)\}$ and synthesis filter transfer functions $\{G_k(z)\}$, respectively. From this filter bank, we derive a new filter bank having analysis filter transfer functions $\{H'_k(z)\}$ and synthesis filter transfer functions $\{G'_k(z)\}$. For each of the following choices of $\{H'_k(z)\}$ and $\{G'_k(z)\}$, determine under what circumstances (if any) the new filter bank will have PR:
(a) $H'_k(z) = H_k(z^2)$, $G'_k(z) = G_k(z^2)$;
(b) $H'_k(z) = H_k(-z)$, $G'_k(z) = G_k(-z)$;
(c) $H'_k(z) = H_k(z^L)$, $G'_k(z) = G_k(z^L)$;
(d) $H'_k(z) = \begin{cases} z^{-M}H_0(z) & \text{for } k = 0 \\ H_k(z) & \text{otherwise} \end{cases}$, $G'_k(z) = \begin{cases} z^{M}G_0(z) & \text{for } k = 0 \\ G_k(z) & \text{otherwise} \end{cases}$

3.28 Consider the system shown in figure (a) below. This system is equivalent to the system shown in figure (b).



(a)



(b)

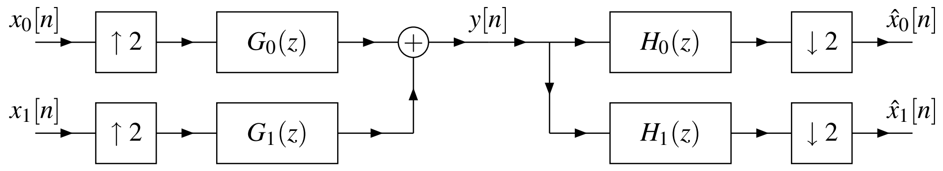
(a) Find $H_k'(z)$ and $G_k'(z)$ in terms of $H_k(z)$ and $G_k(z)$.
(b) Suppose that the two-channel UMD filter bank with analysis filters $H_0(z), H_1(z)$ and synthesis filters $G_0(z), G_1(z)$ has the shift-free PR property. Show that the system in figure (b) has the shift-free PR property.
(c) Suppose that the system in part (b) has the PR property but with a nonzero shift. Determine whether the system in figure (b) has the PR property.

3.29 Let $H_k(z)$ for $k = 0, 1, \ldots, M-1$ denote the analysis filter transfer functions for an M-channel UMD filter bank. For each set of analysis filters given below, determine whether the shift-free PR property can be achieved with FIR synthesis filters. [Note: It is not necessary to find the actual synthesis filters if they exist.]
(a) $H_0(z) = -\frac{1}{16}z^3 + \frac{1}{16}z^2 + \frac{1}{2}z + \frac{1}{2} + \frac{1}{16}z^{-1} - \frac{1}{16}z^{-2}$, $H_1(z) = z - 1$
(b) $H_0(z) = z + \frac{1}{4}$, $H_1(z) = 1 + z^3$
(c) $H_0(z) = \frac{1}{3} + \frac{1}{3}z^{-1} + \frac{1}{3}z^{-2}$, $H_1(z) = -1 - z^{-1} + 2z^{-2}$.

3.30 Use the biorthogonality condition for shift-free PR to derive the corresponding modulation matrix condition. In other words, show that

$$\left\langle g_p[\cdot], h_q^*[Mn - \cdot] \right\rangle = \delta[p-q]\delta[n] \;\Rightarrow\; \boldsymbol{G}_{\mathsf{m}}(z)\boldsymbol{H}_{\mathsf{m}}(z) = M\boldsymbol{I}.$$

3.31 Consider the system with inputs $x_0[n]$ and $x_1[n]$ and outputs $\hat{x}_0[n]$ and $\hat{x}_1[n]$ as shown in the figure below. Such a system is known as a two-channel transmultiplexer.



Let $X_0(z)$, $X_1(z)$, $\hat{X}_0(z)$, and $\hat{X}_1(z)$ denote the $\mathcal{Z}$ transforms of $x_0[n]$, $x_1[n]$, $\hat{x}_0[n]$, and $\hat{x}_1[n]$, respectively.
(a) Find expressions for $\hat{X}_0(z)$ and $\hat{X}_1(z)$ in terms of $X_0(z)$ and $X_1(z)$. [Hint: Use the polyphase identity in conjunction with the linearity of the system.]
(b) A two-channel transmultiplexer is said to have the shift-free PR property if

$$\hat{x}_k[n] \equiv x_k[n] \quad \text{for } k \in \{0, 1\}$$

(i.e., corresponding inputs and outputs are equal). Determine the conditions that $H_0(z)$, $H_1(z)$, $G_0(z)$, and $G_1(z)$ must satisfy in order for the system to have the shift-free PR property.

3.32 Suppose that we have a two-channel UMD filter bank with input $x[n]$ and output $\hat{x}[n]$. Denote the transfer functions of the analysis and synthesis filters as $H_0(z)$ and $H_1(z)$ and $G_0(z)$ and $G_1(z)$, respectively. Suppose that the analysis and synthesis filters are such that

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 0 \quad \text{and}$$
$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 2.$$

One can readily confirm that this system does not have the PR property. In spite of this, however, the input can be recovered exactly from the output. Explain how this is possible. Sketch a diagram of the system that would be used to recover the original input signal.

3.33 Suppose that we have an M-channel UMD filter bank. Let $\boldsymbol{H}_{\mathsf{p}}(z)$ and $\boldsymbol{G}_{\mathsf{p}}(z)$ denote the analysis and synthesis polyphase matrices, respectively. Let $\boldsymbol{H}_{\mathsf{m}}(z)$ and $\boldsymbol{G}_{\mathsf{m}}(z)$ denote the analysis and synthesis modulation matrices, respectively. Further, define $\boldsymbol{D}(z) = \operatorname{diag}\begin{bmatrix} 1 & z & \cdots & z^{M-1} \end{bmatrix}$ and $\boldsymbol{W} = \boldsymbol{W}_M$. Show that:
(a) $\boldsymbol{H}_{\mathsf{m}}(z) = z^{M-1}\boldsymbol{H}_{\mathsf{p}}(z^M)\boldsymbol{D}(z^{-1})\boldsymbol{J}\boldsymbol{W}$ for $(4,\cdot)$ type
(b) $\boldsymbol{G}_{\mathsf{m}}(z) = \boldsymbol{W}\boldsymbol{D}(z)\boldsymbol{G}_{\mathsf{p}}(z^M)$ for $(\cdot,3)$ type

3.34 Suppose that we have a two-channel UMD filter bank with analysis filters $H_0(z), H_1(z)$ and synthesis filters $G_0(z), G_1(z)$.
(a) Show that the synthesis filters required for a shift-free PR system are given by

$$G_0(z) = -\frac{z^{-1}}{\Delta(z^2)}H_1(-z) \quad \text{and}$$
$$G_1(z) = \frac{z^{-1}}{\Delta(z^2)}H_0(-z),$$

where $\Delta(z)$ is the determinant of the type-3 polyphase matrix of the analysis bank. [Hint: Consider a type $(3,1)$ polyphase decomposition of the system, and use Theorem 3.13.]
(b) Use the result of part (a) to show that, in a shift-free PR FIR UMD filter bank, the transfer functions of the analysis and synthesis filters are related as follows:

$$G_0(z) = -az^bH_1(-z) \quad \text{and}$$
$$G_1(z) = az^bH_0(-z).$$

where $a \in \mathbb{C}$ and $b \in \mathbb{Z}$.

3.35 Suppose that we have a two-channel shift-free PR UMD filter bank with analysis filters $H_0(z), H_1(z)$. For each of the following, find a lifting realization of the filter bank with symmetric lifting filters.
(a) $H_0(z) = -\frac{1}{8}z^2 + \frac{1}{4}z + \frac{3}{4} + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}$, $H_1(z) = -\frac{1}{128}z^6 + \frac{2}{128}z^5 + \frac{7}{128}z^4 - \frac{70}{128}z^2 + \frac{124}{128}z - \frac{70}{128} + \frac{7}{128}z^{-2} + \frac{2}{128}z^{-3} - \frac{1}{128}z^{-4}$.
(b) $H_0(z) = -\frac{1}{256}z^6 + \frac{18}{256}z^4 - \frac{16}{256}z^3 - \frac{63}{256}z^2 + \frac{144}{256}z + \frac{348}{256} + \frac{144}{256}z^{-1} - \frac{63}{256}z^{-2} - \frac{16}{256}z^{-3} + \frac{18}{256}z^{-4} + \frac{1}{256}z^{-6}$, $H_1(z) = -\frac{1}{16}z^4 + \frac{9}{16}z^2 - \frac{16}{16}z + \frac{9}{16} - \frac{1}{16}z^{-2}$.

3.36 Suppose that we have a two-channel shift-free PR UMD filter bank with analysis filters $H_0(z), H_1(z)$. For each of the following, find a lifting realization of the filter bank.
(a) $H_0(z) = \frac{1}{2}z + \frac{1}{2}$, $H_1(z) = -\frac{1}{8}z^3 - \frac{1}{8}z^2 + z - 1 + \frac{1}{8}z^{-1} + \frac{1}{8}z^{-2}$
(b) $H_0(z) = \frac{1}{2}z + \frac{1}{2}$, $H_1(z) = \frac{3}{128}z^5 + \frac{3}{128}z^4 - \frac{22}{128}z^3 - \frac{22}{128}z^2 + z - 1 + \frac{22}{128}z^{-1} + \frac{22}{128}z^{-2} - \frac{3}{128}z^{-3} - \frac{3}{128}z^{-4}$.

3.37 Consider a two-channel UMD filter bank with analysis filters having transfer functions $H_0(z)$ and $H_1(z)$, where

$$H_0(z) = a_0 + a_1z^{-1} + a_2z^{-2} \quad \text{and}$$
$$H_1(z) = b_0 + b_1z^{-1} + b_2z^{-2}.$$

Determine a set of constraints, expressed in terms of the coefficients $a_0, a_1, a_2, b_0, b_1, b_2$, that must be satisfied in order for PR to be possible with:
(a) FIR synthesis filters
(b) IIR synthesis filters (without concern for stability).

3.38 Let $\{H_k\}$ and $\{G_k\}$ respectively denote the transfer functions of the analysis and synthesis filters of a two-channel PR UMD filter bank with reconstruction delay n_0. Let $\boldsymbol{G}_\mathsf{m}(z)$ denote the synthesis modulation matrix for the system. Use the modulation-domain PR condition to show that

$$H_0(z) = \frac{2}{z^{n_0} \det \boldsymbol{G}_\mathsf{m}(z)} G_1(-z) \quad \text{and} \quad H_1(z) = \frac{-2}{z^{n_0} \det \boldsymbol{G}_\mathsf{m}(z)} G_0(-z).$$

3.39 Consider a two-channel UMD filter bank with analysis and synthesis filter transfer functions $\{H_k(z)\}_{k=0}^{1}$ and $\{G_k(z)\}_{k=0}^{1}$ and analysis and synthesis modulation matrices $\boldsymbol{H}_\mathsf{m}(z)$ and $\boldsymbol{G}_\mathsf{m}(z)$, all respectively. Show that the condition $\boldsymbol{G}_\mathsf{m}(z)\boldsymbol{H}_\mathsf{m}(z) = 2\boldsymbol{I}$ is equivalent to

$$\begin{gathered} H_k(z)G_k(z) + H_k(-z)G_k(-z) = 2 \quad \text{and} \\ H_{1-k}(z)G_k(z) + H_{1-k}(-z)G_k(-z) = 0 \end{gathered}$$

for $k \in \{0,1\}$.

3.40 Suppose that $F(z)$ has the M-phase polyphase components $\{P_k(z)\}_{k=0}^{M-1}$ and $\{Q_k(z)\}_{k=0}^{M-1}$ of the respective types indicated below. Find an expression for $Q_k(z)$ in terms of $P_k(z)$.
(a) type 1 and type 2;
(b) type 1 and type 3;
(c) type 3 and type 4.

3.41 Let $\{H_0, H_1\}$ and $\{G_0, G_1\}$ respectively denote the analysis and synthesis filters of a two-channel filter bank. Let $P(z)$ denote the transfer function of a halfband filter. For each of the cases below, use the spectral factorization of $P(z)$ to determine the filters H_0 and G_0 of a shift-free PR filter bank satisfying the given design criteria.
(a) $P(z) = -\frac{1}{16}z^{-3}(z+1)^4(z^2-4z+1)$; H_0 has a length of five, linear phase, a DC gain of one, and a second-order zero at the Nyquist frequency; and G_0 has a length of three, linear phase, and a second-order zero at the Nyquist frequency;
(b) $P(z) = \frac{1}{2}(z+1)(1+z^{-1})$; filter bank is orthonormal; H_0 and G_0 are each of length two and G_0 is causal.
(c) $P(z) = -\frac{1}{1024}z^{-7}(z+1)^6(z^2-4z+1)(z^6-2z^5-5z^4+28z^3-5z^2-2z+1)$; H_0 has a length of nine, linear phase, a DC gain of one, and a second-order zero at the Nyquist frequency; and G_0 has a length of seven, linear phase, and a fourth-order zero at the Nyquist frequency.
(d) $P(z) = \frac{1}{8192}z^{-9}(z+1)^8(z^2-4z+1)^2(z^6-9z^4+32z^3-9z^2+1)$; H_0 has a length of 13, linear phase, a DC gain of one, a fourth-order zero at the Nyquist frequency, and an impulse response with support in $\{0,1,\ldots,12\}$; and G_0 has a length of 7, linear phase, and a fourth-order zero at the Nyquist frequency;
(e) $P(z) = \frac{1}{2}z^{-3}(z+1)^2(z^2-z+1)^2 = \frac{1}{2}(z+1)(z^{-1}+1)(z^2-z+1)(z^{-2}-z^{-1}+1)$; the system is orthonormal; H_0 is causal, has a length of 4, linear phase, and a first-order zero at the Nyquist frequency.

3.42 Using the time domain, show that $\uparrow L$ and $\downarrow M$ commute if and only if L and M are coprime.

3.43 Consider an M-channel UMD filter bank with the analysis filter transfer functions $\{H_k\}_{k=0}^{M-1}$ and type-1 analysis polyphase matrix $\boldsymbol{H}_\mathsf{p}$. Given that $H_0(z) = 1 + z^{-1} + \ldots + z^{-(M-1)}$ and $H_k(z) = H_0(zW_M^k)$, find $\boldsymbol{H}_\mathsf{p}$.

3.44 Show that:
(a) the passband gain of the antialiasing filter used in an M-fold decimator should be one.
(b) the passband gain of the antiimaging filter used in an M-fold interpolator should be M.

3.14 Bibliography

[1] M. D. Adams. Reversible wavelet transforms and their application to embedded image compression. M.A.Sc. thesis, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, January 1998.

[2] M. D. Adams and R. K. Ward. Symmetric-extension-compatible reversible integer-to-integer wavelet transforms. *IEEE Trans. on Signal Processing*, 51(10):2624–2636, October 2003.

[3] R. Ansari, C. W. Kim, and M. Dedovic. Structure and design of two-channel filter banks derived from a triplet of halfband filters. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 46(12):1487–1496, December 1999.

[4] C. M. Brislawn. Preservation of subband symmetry in multirate signal coding. *IEEE Trans. on Signal Processing*, 43(12):3046–3050, December 1995.

[5] C. M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Harmonic Analysis*, 3(4):337–357, October 1996.

[6] F. A. M. L. Bruekers and A. W. M. van den Enden. New networks for perfect inversion and perfect reconstruction. *IEEE Journal on Selected Areas in Communications*, 10(1):130–137, January 1992.

[7] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5(3):332–369, July 1998.

[8] S. C. Chan, C. K. S. Pun, and K. L. Ho. The design of a class of perfect reconstruction two-channel FIR linear-phase filterbanks and wavelets bases using semidefinite programming. *IEEE Signal Processing Letters*, 11(2):297–300, February 2004.

[9] S. C. Chan, C. K. S. Pun, and K. L. Ho. New design and realization techniques for a class of perfect reconstruction two-channel FIR filterbanks and wavelet bases. *IEEE Trans. on Signal Processing*, 52(7):2135–2141, July 2004.

[10] S. C. Chan and K. S. Yeung. On the design and multiplierless realization of perfect reconstruction triplet-based FIR filter banks and wavelet bases. *IEEE Trans. on Circuits and Systems—I: Regular Papers*, 51(8):1476–1491, August 2004.

[11] L. Chen, T. Q. Nguyen, and K. Chan. Symmetric extension methods for m-channel linear-phase perfect-reconstruction filter banks. *IEEE Trans. on Signal Processing*, 43(11):2505–2511, December 1995.

[12] R. E. Crochiere and L. R. Rabiner. *Multirate Signal Signal Processing*. Prentice Hall, Upper Saddle River, NJ, USA, 1983.

[13] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4:247–269, 1998.

[14] D. F. Delchamps. *State Space and Input-Output Linear Systems*. Springer-Verlag, New York, 1988.

[15] C.-K. Goh and Y. C. Lim. An efficient algorithm to design weighted minimax perfect reconstruction quadrature mirror filter banks. *IEEE Trans. on Signal Processing*, 47(12):3303–3314, December 1999.

[16] C.-C. Hsiao. Polyphase filter matrix for rational sampling rate conversions. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2173–2176, April 1987.

[17] M. Iwahashi, H. Kiya, and K. Nishikawa. Subband coding of images with circular convolution. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1356–1359, March 1992.

[18] A. A. C. Kalker and I. A. Shah. Ladder structures for multidimensional linear phase perfect reconstruction filter banks and wavelets. In *Proc. of SPIE*, volume 1818, pages 12–20, Boston, MA, USA, November 1992.

[19] G. Karlsson and M. Vetterli. Extension of finite length signals for sub-band coding. *Signal Processing*, 17:161–168, June 1989.

[20] J. Katto, K. Komatsu, and Y. Yasuda. Short-tap and linear-phase PR filter banks for subband coding of images. In *Proc. of SPIE: Visual Communications and Image Processing*, volume 1818, pages 735–746, 1992.

[21] H. Kiya, K. Nishikawa, and M. Iwahashi. A development of symmetric extension methods for subband image coding. *IEEE Trans. on Image Processing*, 3(1):78–81, January 1994.

[22] S. A. Martucci and R. M. Mersereau. The symmetric convolution approach to the nonexpansive implementation of FIR filter banks for images. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 65–68, Minneapolis, MN, USA, April 1993.

[23] K. Nishikawa, H. Kiya, and M. Sagawa. Property of circular convolution for subband image coding. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 281–284, March 1992.

[24] S.-M. Phoong, C. W. Kim, and P. P. Vaidyanathan. A new class of two-channel biorthogonal filter banks and wavelet bases. *IEEE Trans. on Signal Processing*, 43(3):649–665, March 1995.

[25] O. Rioul and P. Duhamel. A Remez exchange algorithm for orthonormal wavelets. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 41(8):550–560, August 1994.

[26] M. J. T. Smith and T. P. Barnwell. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 34(3):434–441, June 1986.

[27] M. J. T. Smith and S. L. Eddins. Subband coding of images with octave band tree structures. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1382–1385, Dallas, TX, USA, 1987.

[28] M. J. T. Smith and S. L. Eddins. Analysis/synthesis techniques for subband image coding. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 38(8):1446–1456, August 1990.

[29] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, USA, 1996.

[30] P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial. *Proc. of IEEE*, 78(1):56–93, January 1990.

[31] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[32] P. P. Vaidyanathan and P.-Q. Hoang. Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36(1):81–94, January 1988.

[33] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

Chapter 4

Univariate Wavelet Systems

4.1 Introduction

Earlier, we introduced a number of characteristics that a good basis for representing functions might possess. First, a basis should be easy to describe (i.e., highly structured). It should have desirable localization properties in both time and frequency. It may be beneficial for a basis to be invariant under certain elementary operations such as shifts and dilations. One might like the basis functions to possess certain smoothness attributes (e.g., continuous, n-times differentiable) or to have certain moment properties. Some of the above criteria are in direct conflict with one another. Ultimately, the application at hand will determine the best choice of basis. A wavelet system corresponds to a particular choice of basis for representing functions. What makes a wavelet system special is the particular structure of its basis functions.

Often, it is beneficial to examine a function at different resolutions. In the case of a low resolution (i.e., coarse scale) representation of a function, many details are lost, but the general trends in function behavior are still apparent. At higher resolutions, more details of the original function are present. The notion of a multiresolution representation of a function is illustrated in Figure 4.1. In Figure 4.1(a), we have the original function. Then, we approximate this function at coarse, medium, and fine scales in Figures 4.1(b), (c), and (d), respectively. As we increase the resolution (i.e., move to finer scales), an improved approximation of the original function is obtained. Another example is given in Figure 4.2. In this case, the function corresponds to an image. In Figure 4.2(a), we have the original image. Then, the image is represented at coarse, medium, and fine scales in Figures 4.2(b), (c), and (d), respectively. Again, as we move to progressively higher resolutions (i.e., finer scales), we add more detail from the original function, and obtain a better approximation. Although this example considers a function that corresponds to an image, the multiresolution concept is useful for many classes of functions.

The idea of representing functions at different resolutions (or scales) leads to the notion of a multiresolution approximation, which we will formally introduce shortly. (As it turns out, wavelets are related to the missing detail in a coarse scale representation of a function that must be added when moving to a representation at a finer scale.)

4.2 Dyadic Wavelet Systems

With wavelet systems, we represent functions at different resolutions where each successive resolution differs in scale by some integer factor. In the case that this factor is two, we have what is called a **dyadic wavelet system**. In what follows, we will focus our attention on dyadic systems. This is the most commonly used type of wavelet system.

4.2.1 Multiresolution Approximations (MRAs)

One of the fundamental building blocks of wavelet systems is the multiresolution approximation (MRA). With a MRA, a function can be viewed at different resolutions (or scales). The formal definition of a MRA is given below.

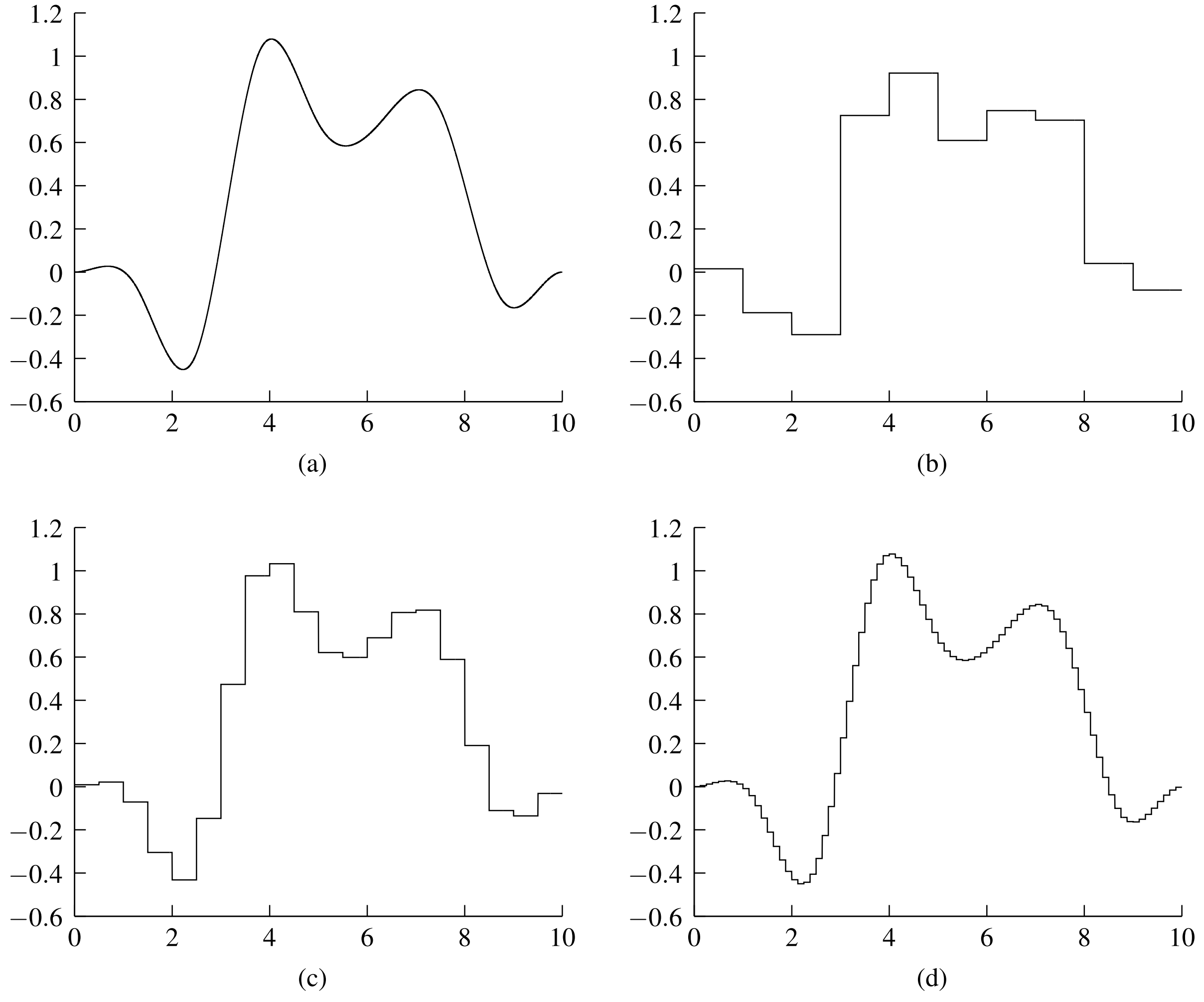


Figure 4.1: Multiresolution approximation of a function. (a) Original function. Approximation of the function at (b) coarse, (c) medium, and (d) fine scales.

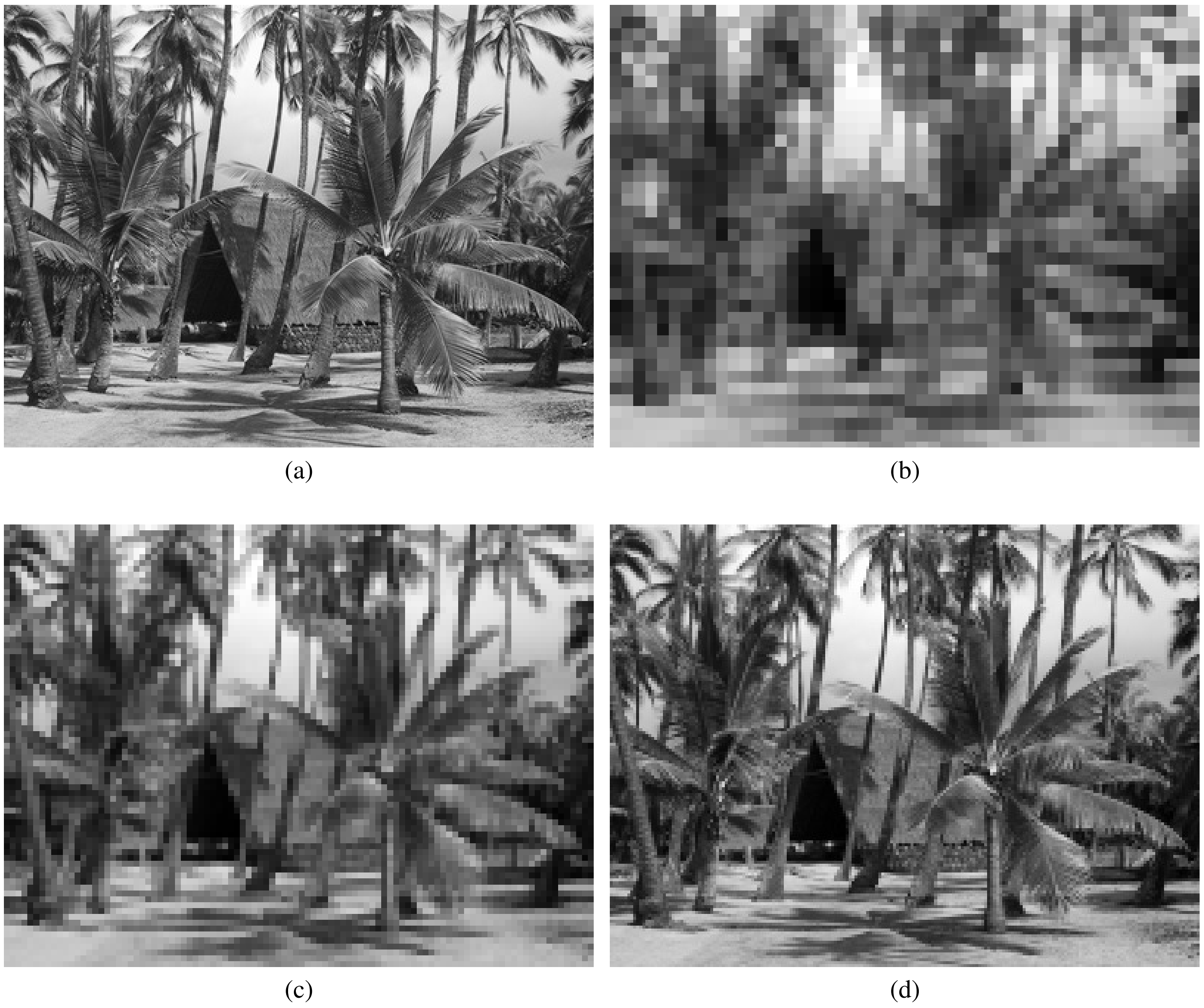


Figure 4.2: Multiresolution representation of an image. (a) Original image. Image represented at (b) coarse (c) medium, and (d) fine scales.

Definition 4.1 (Multiresolution approximation)**.** A sequence $\{V_p\}_{p\in\mathbb{Z}}$ of closed subspaces of $L^2(\mathbb{R})$ is said to be a (dyadic) **multiresolution approximation** (MRA) if the following properties hold:

1. for all $p \in \mathbb{Z}$, $V_p \subset V_{p-1}$ (nesting);
2. $\lim_{p\to\infty} V_p = \bigcap_{p\in\mathbb{Z}} V_p = \{0\}$ (separation);
3. $\lim_{p\to-\infty} V_p = \operatorname{clos}\left(\bigcup_{p\in\mathbb{Z}} V_p\right) = L^2(\mathbb{R})$ (density);
4. for all $p \in \mathbb{Z}$, $f(t) \in V_p \Leftrightarrow f(2t) \in V_{p-1}$ (scale invariance);
5. for all $k \in \mathbb{Z}$, $f(t) \in V_0 \Leftrightarrow f(t-k) \in V_0$ (shift invariance); and
6. there exists ϕ such that $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of V_0 (shift-invariant Riesz basis).

As a matter of terminology, we refer to the spaces V_p in the above definition as **approximation spaces** (or **scaling spaces**). In the above definition, we can see that a single prototype function ϕ is used to generate a (Riesz) basis of V_0. This prototype function is referred to as a **scaling function**.

It is worth noting that some of the properties of a MRA given in the above definition are redundant. That is, some of the properties can be inferred from the others. For example, the scale invariance and shift-invariant Riesz basis properties imply the shift invariance property. Also, the separation property can be deduced from the other properties of the MRA [6, p. 51, Definition 3.4].

The shift-invariance and scale-invariance properties together imply the relationship stated in the lemma below. This relationship is sometimes a useful one.

Lemma 4.1. *Let $\{V_p\}_{p\in\mathbb{Z}}$ be a MRA. Then,*

$$\text{for all } p,k \in \mathbb{Z},\ \ f(t) \in V_p \Leftrightarrow f(t-2^p k) \in V_p.$$

Proof. From the scale invariance property of the MRA (applied $|p|$ times), we have

$$f(t) \in V_p \Rightarrow g(t) = f(2^p t) \in V_0.$$

From the shift invariance property of the MRA, we have

$$g(t) \in V_0 \Rightarrow h_k(t) = g(t-k) \in V_0.$$

From the scale invariance property of the MRA (applied $|p|$ times), we have

$$h_k(t) \in V_0 \Rightarrow h_k(2^{-p}t) \in V_p. \tag{4.1}$$

From the definitions of g and h_k, we can write

$$\begin{aligned} h_k(t) &= g(t-k) \\ &= f(2^p[t-k]) \\ &= f(2^p t - 2^p k) \quad \text{and} \\ h_k(2^{-p}t) &= f(2^p[2^{-p}t] - 2^p k) \\ &= f(t-2^p k). \end{aligned}$$

Substituting this latter expression for $h_k(2^{-p}t)$ into (4.1), we conclude

$$f(t-2^p k) \in V_p.$$

Thus, we have shown that $f(t) \in V_p \Rightarrow f(t-2^p k) \in V_p$, for all $k \in \mathbb{Z}$. The implication in the other direction trivially follows. □

Let P_{V_p} denote the projection operator onto the space V_p. Then, the density property implies that

$$\text{for all } f \in L^2(\mathbb{R}), \quad \lim_{p \to -\infty} P_{V_p} f = f.$$

Similarly, the separation property implies that

$$\text{for all } f \in L^2(\mathbb{R}), \quad \lim_{p \to \infty} P_{V_p} f = 0.$$

Many sequences of spaces satisfy the nesting, separation, and density properties, yet have nothing to do with a MRA. The multiresolution nature of the MRA comes from the scale invariance property. All of the spaces in a MRA are scaled versions of V_0 (i.e., the other spaces are generated by dilation of the elements in V_0). For this reason, the scale invariance property is fundamentally important.

At this point, a few brief comments are in order regarding notation. Two different conventions are commonly employed for the indexing of approximation spaces. With the first convention, the scale of V_p becomes coarser (i.e., resolution decreases) as p increases. This is the convention followed herein. Some examples of other works employing this convention include [9, 17, 18, 20, 24, 25]. With the second convention, the scale of V_p becomes finer (i.e., resolution increases) as p increases. Some examples of works employing this convention include [3, 4, 5, 6, 13, 22]. Since two differing conventions are frequently employed, one must be mindful of the particular convention followed by any given author.

Having introduced the concept of a MRA, we now consider some examples. In what follows, we identify several specific examples of MRAs.

Example 4.1 (Piecewise constant approximations)**.** The simplest MRA is associated with piecewise constant approximations. In this case, the space V_0 is comprised of all functions $f \in L^2(\mathbb{R})$ such that f is constant on intervals of the form $[n, n+1)$, where $n \in \mathbb{Z}$. More generally, V_p is comprised of all functions $f \in L^2(\mathbb{R})$ such that f is constant over intervals of the form $[n2^p, (n+1)2^p)$, where $n \in \mathbb{Z}$.

Consider any arbitrary function $f \in V_0$. Since an integer shift of a function f that is constant over intervals of the form $[n, n+1)$ is also constant over intervals of the same form, the shift invariance property is satisfied. Consider any arbitrary function $f \in V_p$. Since f is constant on intervals of the form $[n2^p, (n+1)2^p)$, $f(2t)$ is constant on intervals of the form $[n2^{p-1}, (n+1)2^{p-1})$. Therefore, the scale invariance property is satisfied. Clearly, $V_p \subset V_{p-1}$, since functions constant on intervals of the form $[n2^p, (n+1)2^p)$ are also constant on intervals of the form $[n2^{p-1}, (n+1)2^{p-1})$. Hence, the nesting property is satisfied. One can show that an orthonormal basis of V_0 is given by $\{\phi(\cdot - k)\}_{k \in \mathbb{Z}}$, where

$$\phi(t) = \chi_{[0,1)}(t) = \begin{cases} 1 & \text{for } t \in [0,1) \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the shift-invariant Riesz basis property is satisfied. A plot of ϕ is given in Figure 4.3.

Since piecewise constant functions are dense in $L^2(\mathbb{R})$, the density property holds. The reader can confirm that the separation property of a MRA is also satisfied.

In some applications, it is desirable for approximations of smooth functions to themselves be smooth. Unfortunately, piecewise constant approximations are not smooth. So, if smooth approximations are required, this MRA is not particularly useful.

An example of approximating a function using different approximation spaces is illustrated in Figure 4.4.

Example 4.2 (Shannon approximations)**.** The Shannon approximation utilizes bandlimited functions in order to form its approximation spaces. The space V_p is comprised of the set of all $f \in L^2(\mathbb{R})$ such that $\operatorname{supp} \hat{f} \subset [-2^{-p}\pi, 2^{-p}\pi]$ (i.e., f is bandlimited to frequencies in the range $-2^{-p}\pi$ to $2^{-p}\pi$).

By using the Shannon sampling theorem (i.e., Theorem 2.51), one can show that an orthonormal basis of V_0 is given by $\{\phi(\cdot - k)\}_{k \in \mathbb{Z}}$, where

$$\phi(t) = \operatorname{sinc} \pi t.$$

Hence, the shift-invariant Riesz basis property is satisfied. A plot of ϕ is given in Figure 4.5.

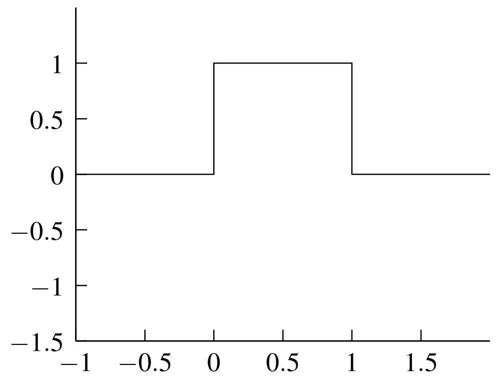


Figure 4.3: Scaling function for piecewise constant approximations.

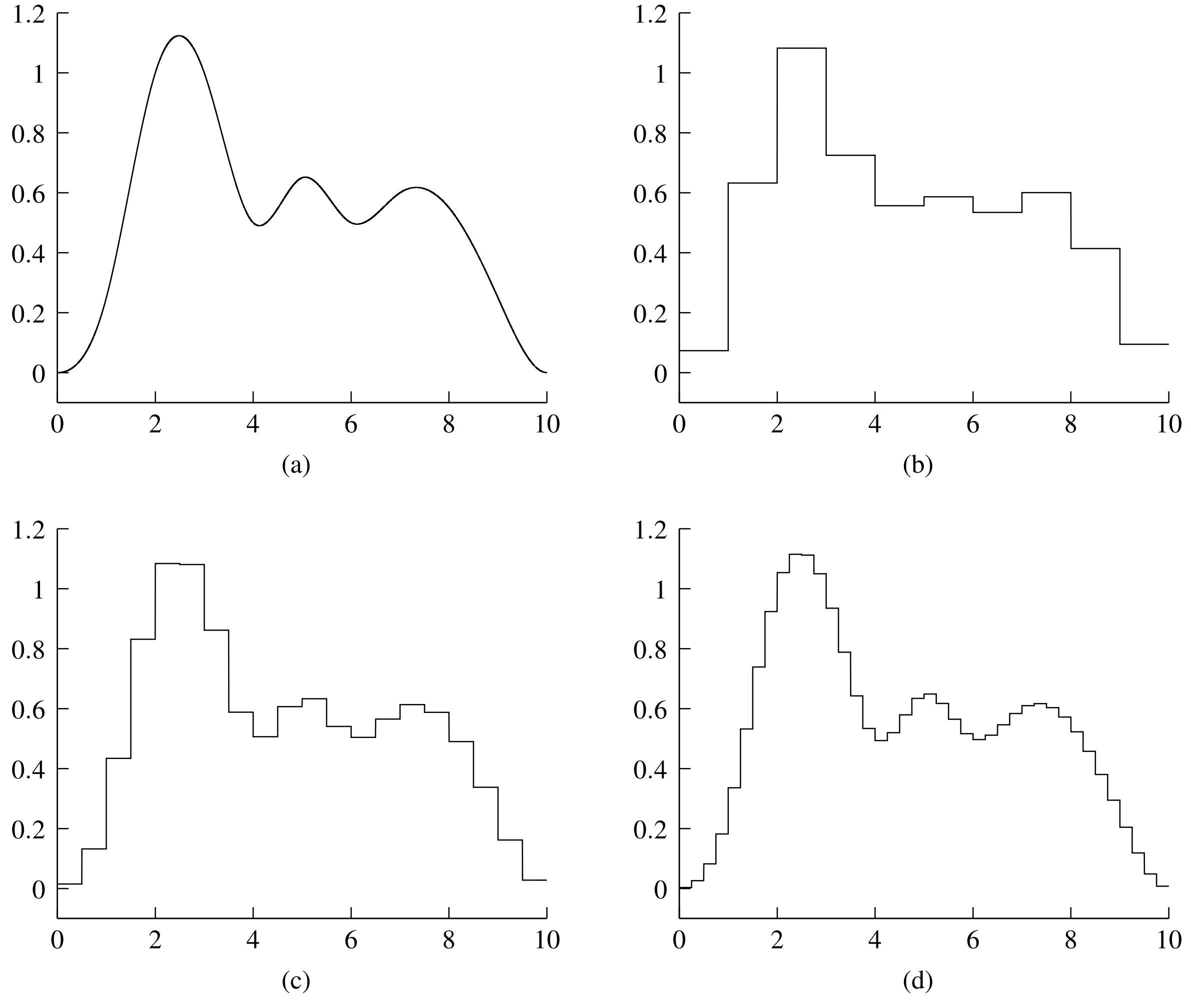


Figure 4.4: Piecewise constant approximation. (a) Original function. Projection of function onto the approximation spaces (b) V_0, (c) V_{-1}, and (d) V_{-2}.

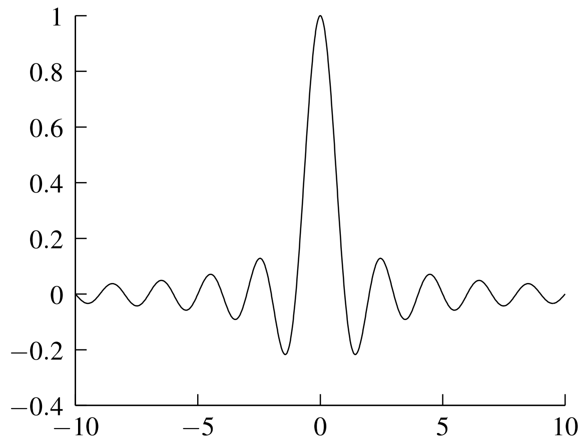


Figure 4.5: Scaling function for Shannon approximations.

From the dilation property of the Fourier transform, we have

$$f \in L^2(\mathbb{R}), \operatorname{supp}\hat{f} \subset [-2^{-p}\pi, 2^{-p}\pi] \quad \Leftrightarrow \quad f(2t) \in L^2(\mathbb{R}), \operatorname{supp}\widehat{f(2\cdot)} \subset [-2^{-(p-1)}\pi, 2^{-(p-1)}\pi].$$

We also have

$$f \in L^2(\mathbb{R}), \operatorname{supp}\hat{f} \subset [-2^{-p}\pi, 2^{-p}\pi] \Leftrightarrow f \in V_p$$
$$f \in L^2(\mathbb{R}), \operatorname{supp}\hat{f} \subset [-2^{-(p-1)}\pi, 2^{-(p-1)}\pi] \Leftrightarrow f \in V_{p-1}.$$

Combining the above sets of equivalences, we obtain

$$f(t) \in V_p \Leftrightarrow f(2t) \in V_{p-1}.$$

Thus, the scale invariance property is satisfied.

From the translation property of the Fourier transform, we have that $\operatorname{supp}\hat{f} = \operatorname{supp}\widehat{f(\cdot - k)}$ for all $k \in \mathbb{Z}$. So, $f \in V_0 \Leftrightarrow f(t-k) \in V_0$ for all $k \in \mathbb{Z}$. Therefore, the shift invariance property is satisfied.

Let P_{V_p} denote the projection operator onto V_p. Then, we have

$$\widehat{P_{V_p} f}(\omega) = \hat{f}(\omega)\chi_{[-2^{-p}\pi, 2^{-p}\pi]}(\omega).$$

The function $P_{V_p} f(t)$ decays as $O(|t|^{-1})$, although f itself might have compact support. For finite p, the space V_p contains only functions that are not compactly supported (aside from the trivial zero function).

Example 4.3 (Continuous piecewise linear approximations)**.** Consider a MRA associated with continuous piecewise linear approximations. The space V_0 is the set of all continuous functions $f \in L^2(\mathbb{R})$ such that f is linear on intervals of the form $[n, n+1)$, where $n \in \mathbb{Z}$. More generally, the space V_p is the set of all continuous functions $f \in L^2(\mathbb{R})$ such that f is linear on intervals of the form $[n2^p, (n+1)2^p)$.

Clearly, if a function f is linear on intervals of the form $[n2^p, (n+1)2^p)$ and continuous, then f is also linear on intervals of the form $[n2^{p-1}, (n+1)2^{p-1})$ and continuous. Therefore, the nesting property is satisfied.

One can show that a Riesz basis of V_0 is given by $\{\phi(\cdot - k)\}_{k \in \mathbb{Z}}$, where

$$\phi(t) = \begin{cases} 1 - |t| & \text{for } |t| < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the shift-invariant Riesz basis property is satisfied. (The above basis is not orthonormal, however.) A plot of ϕ is given in Figure 4.6.

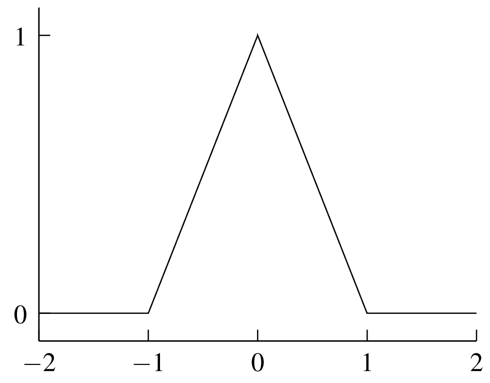


Figure 4.6: Scaling function for piecewise linear approximations.

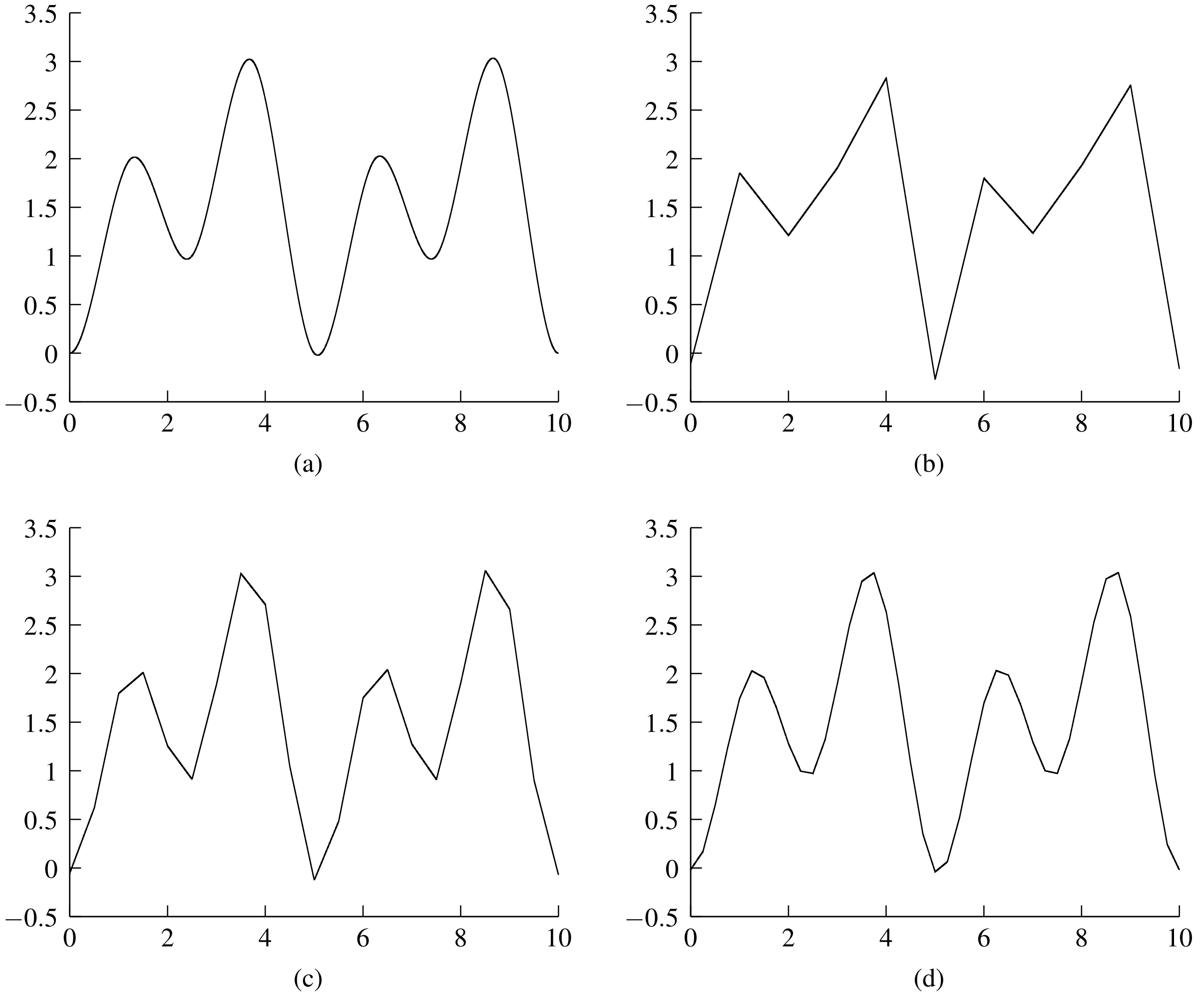


Figure 4.7: Continuous piecewise linear approximation. (a) Original function. Projection of function onto the approximation spaces (b) V_0, (c) V_{-1}, and (d) V_{-2}.

(The restriction that the functions in each approximation space be continuous is required. Without this constraint, there would not exist a single function whose integer shifts constitute a Riesz basis of V_0.)

An example of approximating a function using different approximation spaces is illustrated in Figure 4.7.

Example 4.4 (Cubic spline approximations)**.** Consider a MRA associated with cubic spline approximations. The space V_p is comprised of all functions $f \in L^2(\mathbb{R})$ such that f, f', and f'' are continuous, and f is piecewise cubic on intervals of the form $[n2^p, (n+1)2^p)$, where $n \in \mathbb{Z}$.

One can show that a Riesz basis of V_0 is given by $\{\phi(\cdot - k)\}_{k \in \mathbb{Z}}$, where

$$\phi(t) = \begin{cases} \frac{1}{6}t^3 + t^2 + 2t + \frac{4}{3} & \text{for } -2 \leq t \leq -1 \\ -\frac{1}{2}t^3 - t^2 + \frac{2}{3} & \text{for } -1 \leq t \leq 0 \\ \frac{1}{2}t^3 - t^2 + \frac{2}{3} & \text{for } 0 \leq t \leq 1 \\ -\frac{1}{6}t^3 + t^2 - 2t + \frac{4}{3} & \text{for } 1 \leq t \leq 2. \end{cases}$$

Hence, the shift-invariant Riesz basis property is satisfied. (The above basis is not orthonormal, however.) A plot of ϕ is provided in Figure 4.8.

It is left as an exercise to the reader to show that the remaining properties of a MRA are satisfied.

An example of approximating a function using different approximation spaces is illustrated in Figure 4.9.

From the piecewise constant and cubic spline approximations, one can see a pattern beginning to emerge. That is, approximating subspaces on regular meshes automatically satisfies the requirements of a MRA. For example, the requirements of a MRA are satisfied by a B-spline approximation of any order.

Example 4.5. Let V_p denote the space comprised of all functions $f \in L^2(\mathbb{R})$ such that $\operatorname{supp} f \subset [-2^{-p}, 2^{-p}]$. Determine whether $\{V_p\}_{p \in \mathbb{Z}}$ constitutes a MRA.

Solution. We need to determine whether $\{V_p\}_{p \in \mathbb{Z}}$ satisfies the properties of a MRA.

NESTING. Consider the nesting property. We have

$$V_p = \{f \in L^2(\mathbb{R}) : \operatorname{supp} f \subset [-2^{-p}, 2^{-p}]\} \quad \text{and}$$
$$V_{p-1} = \{f \in L^2(\mathbb{R}) : \operatorname{supp} f \subset [-2^{-p+1}, 2^{-p+1}]\}.$$

Since $[-2^{-p}, 2^{-p}] \subset [-2^{-p+1}, 2^{-p+1}]$, it is clear that any function in V_p is also in V_{p-1}. Therefore, the nesting property is satisfied.

SEPARATION AND DENSITY. Again, we have

$$V_p = \{f \in L^2(\mathbb{R}) : \operatorname{supp} f \subset [-2^{-p}, 2^{-p}]\}.$$

From this, it follows that

$$\lim_{p \to \infty} V_p = \{f \in L^2(\mathbb{R}) : \operatorname{supp} f \subset [0,0]\} = \{0\} \quad \text{and}$$
$$\lim_{p \to -\infty} V_p = \{f \in L^2(\mathbb{R}) : \operatorname{supp} f \subset \mathbb{R}\} = L^2(\mathbb{R}).$$

Therefore, the separation and density properties are satisfied.

SCALE INVARIANCE. Suppose $f \in V_p$. Then,

$$\operatorname{supp} f(t) \subset [-2^{-p}, 2^{-p}]$$

which implies

$$\operatorname{supp} f(2t) \subset [-2^{-p-1}, 2^{-p-1}].$$

Since $[-2^{-p-1}, 2^{-p-1}] \subset [-2^{-p+1}, 2^{-p+1}]$, $f(2t) \in V_{p-1}$. Thus, we have

$$f(t) \in V_p \Rightarrow f(2t) \in V_{p-1}.$$

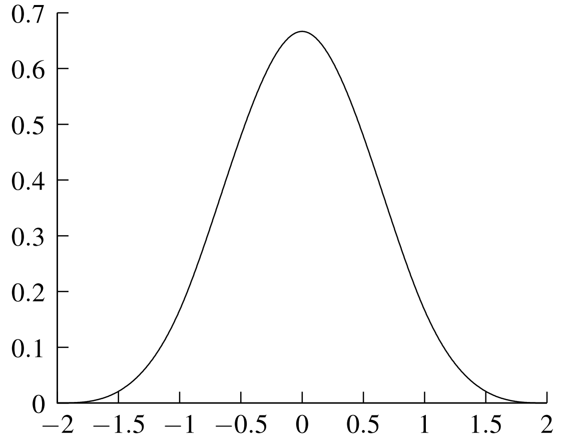


Figure 4.8: Scaling function for cubic spline approximations.

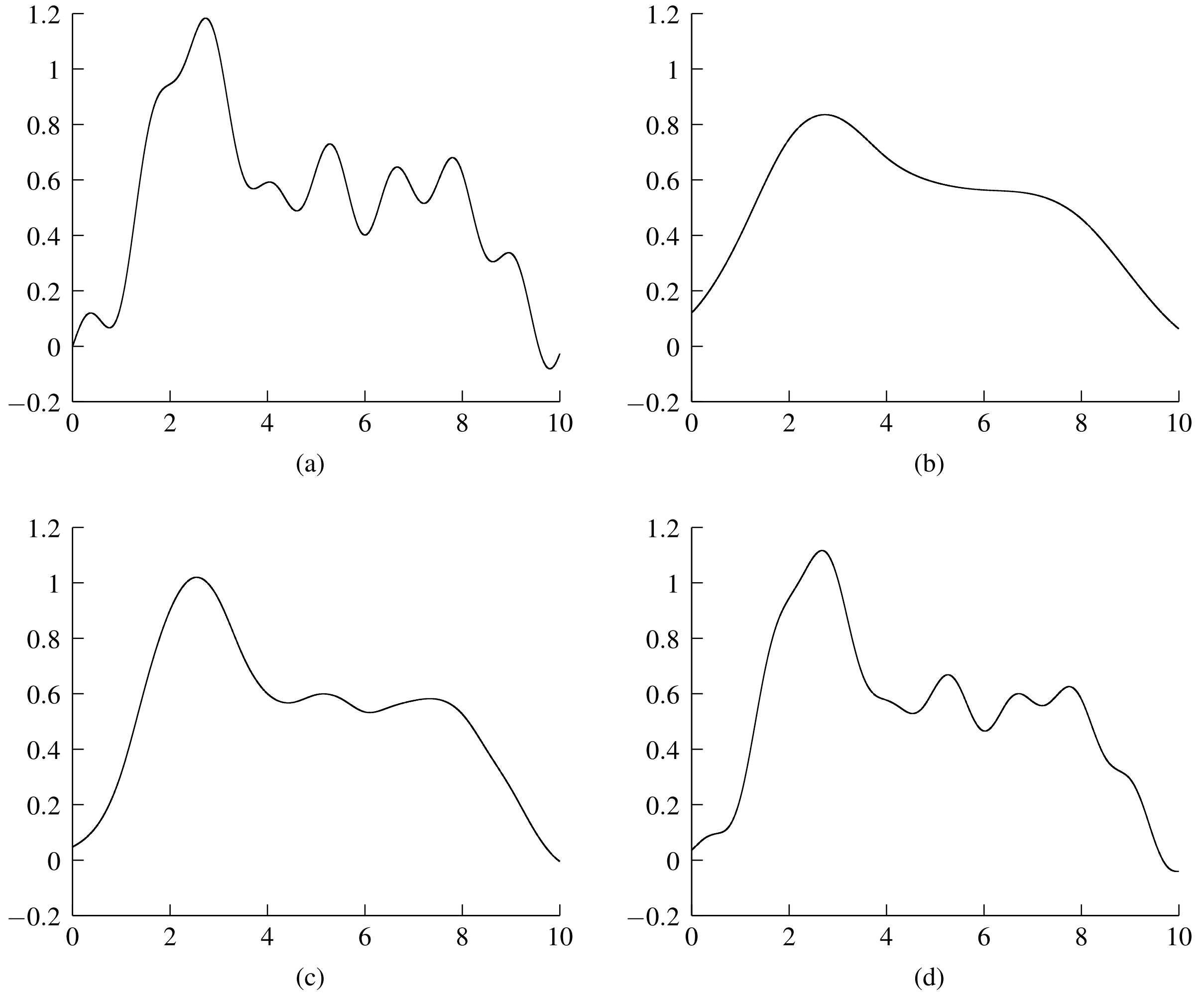


Figure 4.9: Cubic spline approximation. (a) Original function. Projection of function onto the approximation spaces (b) V_0, (c) V_{-1}, and (d) V_{-2}.

Suppose $f(2t) \in V_{p-1}$. Then,

$$\operatorname{supp} f(2t) \subset [-2^{-p+1}, 2^{-p+1}]$$

which implies

$$\operatorname{supp} f(t) \subset [-2^{-p+2}, 2^{-p+2}].$$

Since $[-2^{-p+2}, 2^{-p+2}] \not\subset [-2^{-p}, 2^{-p}]$, it is not necessarily true that $f(t) \in V_p$. Thus, we have

$$f(2t) \in V_{p-1} \not\Rightarrow f(t) \in V_p.$$

Therefore, the scale invariance property is not satisfied.

SHIFT INVARIANCE. Suppose that $f \in V_0$. Then, $\operatorname{supp} f \subset [-1,1]$. Further suppose that $\operatorname{supp} f = [-1,1]$. For $k \neq 0$, $\operatorname{supp} f(t-k) = [-1+k, 1+k] \not\subset [-1,1]$. So, $f(t-k)$ is not necessarily in V_0. Therefore, the shift invariance property is not satisfied.

SHIFT-INVARIANT RIESZ BASIS. Since the shift invariance property is not satisfied, there cannot exist a shift-invariant Riesz basis of V_0. (Clearly, a basis $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ of V_0 must be such that $\phi(\cdot - k) \in V_0$ for all $k \in \mathbb{Z}$. For any nonzero function $\phi \in V_0$, however, there exists some $k \in \mathbb{Z}$ for which $\phi(\cdot - k) \notin V_0$.) Therefore, the Riesz basis property is not satisfied.

From the results above, we conclude that the sequence of subspaces $\{V_p\}_{p\in\mathbb{Z}}$ does not satisfy the properties of a MRA. □

4.2.2 Existence of Riesz Basis

As we saw earlier, a MRA must have a shift-invariant Riesz basis. For this reason, one might wonder what conditions a function ϕ must satisfy in order for $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ to constitute a Riesz basis for its closed linear span. The answer to this question is given by the theorem below.

Theorem 4.1 (Condition for Riesz basis)**.** *A family $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of the space V_0 it generates (i.e., its closed linear span) if and only if there exist $A > 0$ and $B > 0$ such that*

$$\text{for } \omega \in [-\pi, \pi],\ A \leq \sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega - 2\pi k)\right|^2 \leq B \quad \text{a.e.}. \tag{4.2}$$

If such A and B do exist, they are the lower and upper Riesz bounds of $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$, respectively.

Proof. Any $f \in V_0$ can be decomposed as

$$f(t) = \sum_{n\in\mathbb{Z}} a[n]\theta(t-n) \tag{4.3}$$

(where $a \in l^2(\mathbb{Z})$). Taking the Fourier transform of both sides of the preceding equation yields

$$\begin{aligned} \hat{f}(\omega) &= \sum_{n\in\mathbb{Z}} a[n]\mathcal{F}\{\theta(\cdot - n)\} \\ &= \sum_{n\in\mathbb{Z}} a[n]e^{-jn\omega}\hat{\theta}(\omega). \end{aligned}$$

In other words, we have

$$\hat{f}(\omega) = \hat{a}(\omega)\hat{\theta}(\omega)$$

where $\hat{a}(\omega)$ is the 2π-periodic function

$$\hat{a}(\omega) = \sum_{n\in\mathbb{Z}} a[n]e^{-jn\omega}.$$

Using the Parseval identity, we can write

$$\begin{aligned}\|f\|^2 &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left|\hat{f}(\omega)\right|^2 d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left|\hat{a}(\omega)\hat{\theta}(\omega)\right|^2 d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}|\hat{a}(\omega)|^2\left|\hat{\theta}(\omega)\right|^2 d\omega.\end{aligned}$$

Now, we split this integral over $\mathbb{R}$ into intervals of length 2π to obtain

$$\|f\|^2 = \frac{1}{2\pi}\sum_{k\in\mathbb{Z}}\int_0^{2\pi}|\hat{a}(\omega-2\pi k)|^2\left|\hat{\theta}(\omega-2\pi k)\right|^2 d\omega.$$

Since $\hat{a}(\omega)$ is 2π-periodic we know that $\hat{a}(\omega)=\hat{a}(\omega-2\pi k)$. So, we have

$$\begin{aligned}\|f\|^2 &= \frac{1}{2\pi}\int_0^{2\pi}\sum_{k\in\mathbb{Z}}|\hat{a}(\omega)|^2\left|\hat{\theta}(\omega-2\pi k)\right|^2 d\omega \\ &= \frac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2\sum_{k\in\mathbb{Z}}\left|\hat{\theta}(\omega-2\pi k)\right|^2 d\omega. \qquad (4.4)\end{aligned}$$

(4.2) $\Rightarrow$ RIESZ BASIS. Now, we show that condition (4.2) implies a Riesz basis. By assumption, (4.2) is satisfied. So, replacing $\sum_{k\in\mathbb{Z}}\left|\hat{\theta}(\omega-2\pi k)\right|^2$ by its lower bound A yields the inequality

$$\|f\|^2 \geq \frac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 A d\omega \quad\Rightarrow\quad \tfrac{1}{A}\|f\|^2 \geq \tfrac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 d\omega.$$

Replacing the same summation by its upper bound B, we obtain

$$\|f\|^2 \leq \frac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 B d\omega \quad\Rightarrow\quad \tfrac{1}{B}\|f\|^2 \leq \tfrac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 d\omega.$$

Combining these inequalities, we obtain

$$\tfrac{1}{B}\|f\|^2 \leq \tfrac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 d\omega \leq \tfrac{1}{A}\|f\|^2. \qquad (4.5)$$

By the Parseval identity, however, we know that

$$\tfrac{1}{2\pi}\int_0^{2\pi}|\hat{a}(\omega)|^2 d\omega = \sum_{n\in\mathbb{Z}}|a[n]|^2.$$

So, we have

$$\tfrac{1}{B}\|f\|^2 \leq \sum_{n\in\mathbb{Z}}|a[n]|^2 \leq \tfrac{1}{A}\|f\|^2 \qquad (4.6)$$

(or equivalently, $A\sum_{n\in\mathbb{Z}}|a[n]|^2 \leq \|f\|^2 \leq B\sum_{n\in\mathbb{Z}}|a[n]|^2$). Thus, $\{\theta(\cdot-n)\}_{n\in\mathbb{Z}}$ satisfies the Riesz condition. The linear independence of $\{\theta(\cdot-n)\}_{n\in\mathbb{Z}}$ follows from (4.6) and (4.3). That is, if $f=0$, then $a[n]=0$ for all $n\in\mathbb{Z}$, which shows the linear independence of $\{\theta(\cdot-n)\}_{n\in\mathbb{Z}}$. Hence, (4.2) implies a Riesz basis.

RIESZ BASIS $\Rightarrow$ (4.2). Now, we show that a Riesz basis implies (4.2). Since $\{\theta(\cdot-n)\}_{n\in\mathbb{Z}}$ is a Riesz basis, (4.6) is valid for any $a[n]\in l^2(\mathbb{Z})$. Suppose now that (4.2) is violated. That is, $\sum_{k\in\mathbb{Z}}\left|\hat{\theta}(\omega-2k\pi)\right|^2$ does not satisfy (4.2) for almost all $\omega\in[-\pi,\pi]$. In this case, we can construct a 2π-periodic function $\hat{a}(\omega)$ whose support corresponds to frequencies where (4.2) does not hold. Next, we can deduce from (4.4) that (4.5) and (4.6) are not valid for $a[n]$. Thus, $\{\theta(\cdot-n)\}_{n\in\mathbb{Z}}$ cannot be a Riesz basis, and we have a contradiction. Hence, a Riesz basis implies that (4.2) must hold.
□

From the preceding theorem, we also trivially have the following result.

Theorem 4.2. *The family $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of the space that it generates if and only if*

$$\sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi k)\right|^2 = 1. \tag{4.7}$$

Proof. We observe that an orthonormal basis is simply a Riesz basis with both Riesz bounds equal to one. Thus, from Theorem 4.1, we have that $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of the space that it generates if and only if

$$1 \le \sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi k)\right|^2 \le 1$$

which is obviously equivalent to (4.7). □

From Theorem 4.1, for any given function θ, the quantity $\sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega-2\pi k)\right|^2$ determines whether $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ constitutes a Riesz basis. As it turns out, this quantity has an important interpretation in terms of θ. This relationship is further elucidated by the theorem below.

Theorem 4.3 (Autocorrelation sequences and shift-invariant Riesz bases)**.** *Let θ be a function with Fourier transform $\hat{\theta}$. Let $a[n]$ denote the autocorrelation sequence of θ. That is, $a[n]$ is given by*

$$a[n] = \int_{-\infty}^{\infty} \theta(t)\theta^*(t-n)dt. \tag{4.8}$$

Then, the (discrete-time) Fourier transform $\hat{a}(\omega)$ of $a[n]$ is the quantity

$$\hat{a}(\omega) = \sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega-2\pi k)\right|^2.$$

Proof. From (4.8), we can use the Parseval identity to write

$$\begin{aligned} a[n] &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\theta}(\omega)\left[\hat{\theta}(\omega)e^{-jn\omega}\right]^* d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\theta}(\omega)\hat{\theta}^*(\omega)e^{jn\omega}d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \left|\hat{\theta}(\omega)\right|^2 e^{jn\omega}d\omega. \end{aligned}$$

Now, we split the integral over $\mathbb{R}$ into intervals of length 2π to obtain

$$\begin{aligned} a[n] &= \frac{1}{2\pi}\sum_{k\in\mathbb{Z}}\int_0^{2\pi} \left|\hat{\theta}(\omega-2\pi k)\right|^2 e^{jn(\omega-2\pi k)}d\omega \\ &= \frac{1}{2\pi}\sum_{k\in\mathbb{Z}}\int_0^{2\pi} \left|\hat{\theta}(\omega-2\pi k)\right|^2 e^{jn\omega}d\omega. \end{aligned}$$

Reversing the order of the summation and integration, we have

$$a[n] = \frac{1}{2\pi}\int_0^{2\pi}\sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega-2\pi k)\right|^2 e^{jn\omega}d\omega.$$

Now, we observe that the expression on the right-hand side of the preceding equation is an inverse (discrete-time) Fourier transform integral. Thus, we have

$$a[n] = \frac{1}{2\pi}\int_0^{2\pi} \hat{a}(\omega)e^{jn\omega}d\omega$$

where

$$\hat{a}(\omega) = \sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega - 2\pi k)\right|^2.$$

In other words, $\sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega - 2\pi k)\right|^2$ is the (discrete-time) Fourier transform of the autocorrelation sequence a of θ. □

Thus, from the above theorem, we can see that the quantity $\sum_{k\in\mathbb{Z}} \left|\hat{\theta}(\omega - 2\pi k)\right|^2$ is the (discrete-time) Fourier transform of the autocorrelation sequence of θ. In passing, it is worthwhile to mention that the autocorrelation sequence (by virtue of its definition) satisfies $a[n] = a^*[-n]$. Thus, if θ is real valued, then $a[n]$ is an even sequence.

In light of the above result, we now determine whether a number of prototype functions generate shift-invariant Riesz bases. Some of the functions considered in what follows are taken from the MRA examples introduced earlier.

Example 4.6 (Haar scaling function)**.** Consider the scaling function

$$\phi(t) = \chi_{[0,1)}(t) = \begin{cases} 1 & \text{for } t \in [0,1) \\ 0 & \text{otherwise.} \end{cases}$$

Show that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis and determine the associated Riesz bounds.

Solution. We compute the autocorrelation sequence $a[k]$ of $\phi(t)$. We have

$$\begin{aligned} a[0] &= \int_{-\infty}^{\infty} \phi(t)\phi^*(t)dt \\ &= \int_{-\infty}^{\infty} (\chi_{[0,1)}(t))^2 dt \\ &= \int_0^1 dt \\ &= 1. \end{aligned}$$

Due to the form of ϕ (i.e., $\operatorname{supp}\phi \subset [0,1]$), $a[n] = 0$ for $n \neq 0$. So, we have

$$\hat{a}(\omega) = \sum_{n\in\mathbb{Z}} a[n]e^{-jn\omega} = 1.$$

Thus, $\hat{a}(\omega)$ is trivially bounded by

$$1 \leq \hat{a}(\omega) \leq 1.$$

Therefore, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis with both bounds equal to one. In other words, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis. (We could have arrived at the same conclusion by simply observing that $a[n] = \delta[n]$, which is the defining property of an orthonormal basis.) □

Example 4.7 (Shannon scaling function)**.** Consider the scaling function

$$\phi(t) = \operatorname{sinc} \pi t.$$

Show that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis and determine the corresponding Riesz bounds.

Solution. We need to compute the autocorrelation sequence $a[n]$ of $\phi(t)$. Using the Parseval identity, we have

$$\begin{aligned}a[n] &= \int_{-\infty}^{\infty} \phi(t)\phi^*(t-n)dt \\ &= \tfrac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\phi}(\omega)\left(e^{-jn\omega}\hat{\phi}(\omega)\right)^* d\omega \\ &= \tfrac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\phi}(\omega)e^{jn\omega}\hat{\phi}^*(\omega)d\omega \\ &= \tfrac{1}{2\pi}\int_{-\infty}^{\infty} \left(\operatorname{rect}\tfrac{\omega}{2\pi}\right)^2 e^{jn\omega}d\omega \\ &= \tfrac{1}{2\pi}\int_{-\pi}^{\pi} e^{jn\omega}d\omega.\end{aligned}$$

There are two cases to consider. First, consider the case that $n = 0$. We have

$$\begin{aligned}a[n] &= \tfrac{1}{2\pi}\int_{-\pi}^{\pi} d\omega \\ &= 1.\end{aligned}$$

Now, consider the case that $n \neq 0$. We have

$$\begin{aligned}a[n] &= \tfrac{1}{2\pi}\left[\tfrac{1}{jn}e^{jn\omega}\right]\Big|_{-\pi}^{\pi} \\ &= \tfrac{1}{j2\pi n}\left[e^{j\pi n} - e^{-j\pi n}\right] \\ &= \tfrac{1}{\pi n}\sin \pi n \\ &= 0.\end{aligned}$$

Combining the above two cases, we obtain

$$a[n] = \delta_n.$$

Clearly, $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is an orthonormal basis (for its closed linear span). Therefore, we trivially have that $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis with both Riesz bounds equal to one. □

Example 4.8 (Quadratic spline scaling function)**.** Consider the scaling function

$$\phi(t) = \begin{cases} \frac{1}{2}t^2 & \text{for } 0 \leq t < 1 \\ -t^2 + 3t - \frac{3}{2} & \text{for } 1 \leq t < 2 \\ \frac{1}{2}t^2 - 3t + \frac{9}{2} & \text{for } 2 \leq t < 3 \\ 0 & \text{otherwise.}\end{cases}$$

Show that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis and determine the associated Riesz bounds.

Solution. To begin, we compute the autocorrelation sequence $a[n]$ of $\phi(t)$. We have

$$\begin{aligned}a[0] &= \int_{-\infty}^{\infty} \phi^2(t)dt \\ &= \int_0^1 (\tfrac{1}{2}t^2)^2 dt + \int_1^2 (-t^2 + 3t - \tfrac{3}{2})^2 dt + \int_2^3 (\tfrac{1}{2}t^2 - 3t + \tfrac{9}{2})^2 dt \\ &= \tfrac{1}{20} + \tfrac{9}{20} + \tfrac{1}{20} \\ &= \tfrac{11}{20},\end{aligned}$$

$$\begin{aligned}
a[1] &= \int_{-\infty}^{\infty} \phi(t)\phi(t-1)dt \\
&= \int_1^2 (-t^2+3t-\tfrac{3}{2})(\tfrac{1}{2}(t-1)^2)dt + \int_2^3 (\tfrac{1}{2}t^2-3t+\tfrac{9}{2})(-(t-1)^2+3(t-1)-\tfrac{3}{2})dt \\
&= \tfrac{13}{120} + \tfrac{13}{120} \\
&= \tfrac{26}{120} \\
&= \tfrac{13}{60}, \quad \text{and}
\end{aligned}$$

$$\begin{aligned}
a[2] &= \int_{-\infty}^{\infty} \phi(t)\phi(t-2)dt \\
&= \int_2^3 (\tfrac{1}{2}t^2-3t+\tfrac{9}{2})(\tfrac{1}{2}(t-2)^2)dt \\
&= \tfrac{1}{120}.
\end{aligned}$$

Since $\operatorname{supp}\phi \subset [0,3]$, $a[n]=0$ for $n \geq 3$. Since $a[n]$ is even, we also have

$$a[-1] = \tfrac{13}{60}, \quad a[-2] = \tfrac{1}{120}, \quad \text{and} \quad a[n] = 0 \text{ for } n \leq -3.$$

Now, we compute the (discrete-time) Fourier transform of a to obtain

$$\begin{aligned}
\hat{a}(\omega) &= \sum_{n\in\mathbb{Z}} a[n]e^{-jn\omega} \\
&= \tfrac{11}{20} + \tfrac{13}{60}(e^{j\omega}+e^{-j\omega}) + \tfrac{1}{120}(e^{j2\omega}+e^{-j2\omega}) \\
&= \tfrac{11}{20} + \tfrac{13}{30}\cos\omega + \tfrac{1}{60}\cos 2\omega.
\end{aligned}$$

To determine the Riesz bounds, we need to find the bounds of $\hat{a}$. We have that

$$\tfrac{d}{d\omega}\hat{a}(\omega) = 0 \quad \Rightarrow \quad -\tfrac{13}{30}\sin\omega - \tfrac{1}{30}\sin 2\omega = 0 \quad \Rightarrow \quad \omega \in \{0, \pm\pi, \pm 2\pi, \ldots\}.$$

Thus, $\hat{a}(\omega)$ has a minimum of $\frac{2}{15}$ at $\omega = \pi$ and a maximum of 1 at $\omega = 0$. Thus, we have

$$\tfrac{2}{15} \leq \hat{a}(\omega) \leq 1.$$

Therefore, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis with lower and upper Riesz bounds of $\frac{2}{15}$ and 1, respectively. □

Example 4.9 (Cubic spline scaling function)**.** Consider the scaling function

$$\phi(t) = \begin{cases} \frac{1}{6}t^3+t^2+2t+\frac{4}{3} & \text{for } -2 \leq t \leq -1 \\ -\frac{1}{2}t^3-t^2+\frac{2}{3} & \text{for } -1 \leq t \leq 0 \\ \frac{1}{2}t^3-t^2+\frac{2}{3} & \text{for } 0 \leq t \leq 1 \\ -\frac{1}{6}t^3+t^2-2t+\frac{4}{3} & \text{for } 1 \leq t \leq 2 \end{cases}$$

Show that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis and determine the associated Riesz bounds.

Solution. We compute the autocorrelation sequence $a[n]$ of $\phi(t)$. We have

$$\begin{aligned}
a[0] &= \int_{-\infty}^{\infty} \phi^2(t)dt \\
&= \int_{-2}^{-1} (\tfrac{1}{6}t^3+t^2+2t+\tfrac{4}{3})^2 dt + \int_{-1}^{0} (-\tfrac{1}{2}t^3-t^2+\tfrac{2}{3})^2 dt \\
&\quad + \int_0^1 (\tfrac{1}{2}t^3-t^2+\tfrac{2}{3})^2 dt + \int_1^2 (-\tfrac{1}{6}t^3+t^2-2t+\tfrac{4}{3})^2 dt \\
&= \tfrac{1}{252} + \tfrac{33}{140} + \tfrac{33}{140} + \tfrac{1}{252} \\
&= \tfrac{151}{315},
\end{aligned}$$

$$\begin{aligned}
a[1] &= \int_{-\infty}^{\infty} \phi(t)\phi(t-1)dt \\
&= \int_{-1}^{0} (-\tfrac{1}{2}t^3 - t^2 + \tfrac{2}{3})(\tfrac{1}{6}(t-1)^3 + (t-1)^2 + 2(t-1) + \tfrac{4}{3})dt \\
&\quad + \int_{0}^{1} (\tfrac{1}{2}t^3 - t^2 + \tfrac{2}{3})(-\tfrac{1}{2}(t-1)^3 - (t-1)^2 + \tfrac{2}{3})dt \\
&\quad + \int_{1}^{2} (-\tfrac{1}{6}t^3 + t^2 - 2t + \tfrac{4}{3})(\tfrac{1}{2}(t-1)^3 - (t-1)^2 + \tfrac{2}{3})dt \\
&= \tfrac{43}{1680} + \tfrac{311}{1680} + \tfrac{43}{1680} \\
&= \tfrac{397}{1680},
\end{aligned}$$

$$\begin{aligned}
a[2] &= \int_{-\infty}^{\infty} \phi(t)\phi(t-2)dt \\
&= \int_{0}^{1} (\tfrac{1}{2}t^3 - t^2 + \tfrac{2}{3})(\tfrac{1}{6}(t-2)^3 + (t-2)^2 + 2(t-2) + \tfrac{4}{3})dt \\
&\quad + \int_{1}^{2} (-\tfrac{1}{6}t^3 + t^2 - 2t + \tfrac{4}{3})(-\tfrac{1}{2}(t-2)^3 - (t-2)^2 + \tfrac{2}{3})dt \\
&= \tfrac{1}{84} + \tfrac{1}{84} \\
&= \tfrac{1}{42}, \quad \text{and}
\end{aligned}$$

$$\begin{aligned}
a[3] &= \int_{-\infty}^{\infty} \phi(t)\phi(t-3)dt \\
&= \int_{1}^{2} (-\tfrac{1}{6}t^3 + t^2 - 2t + \tfrac{4}{3})(\tfrac{1}{6}(t-3)^3 + (t-3)^2 + 2(t-3) + \tfrac{4}{3})dt \\
&= \tfrac{1}{5040}.
\end{aligned}$$

Since $\operatorname{supp}\phi \subset [-2,2]$, $a[n] = 0$ for $n \geq 4$. Moreover, we can determine $a[n]$ for $n < 0$, by observing that $a[n]$ is even. This yields

$$a[-1] = \tfrac{397}{1680}, a[-2] = \tfrac{1}{42}, a[-3] = \tfrac{1}{5040}, \quad \text{and} \quad a[n] = 0 \;\text{ for } n \leq -4.$$

Computing the (discrete-time) Fourier transform of $a[n]$, we obtain

$$\begin{aligned}
\hat{a}(\omega) &= \sum_{n\in\mathbb{Z}} a[n]e^{-jn\omega} \\
&= \tfrac{151}{315} + \tfrac{397}{1680}(e^{j\omega} + e^{-j\omega}) + \tfrac{1}{42}(e^{j2\omega} + e^{-j2\omega}) + \tfrac{1}{5040}(e^{j3\omega} + e^{-j3\omega}) \\
&= \tfrac{151}{315} + \tfrac{397}{840}\cos\omega + \tfrac{1}{21}\cos 2\omega + \tfrac{1}{2520}\cos 3\omega.
\end{aligned}$$

One can show that $\hat{a}(\omega)$ has a minimum at $\omega = \pi$ given by

$$\hat{a}(\pi) = \tfrac{151}{315} - \tfrac{397}{840} + \tfrac{1}{21} - \tfrac{1}{2520} = \tfrac{17}{315}.$$

Likewise, one can show that $\hat{a}(\omega)$ has a maximum at $\omega = 0$ given by

$$\hat{a}(0) = \tfrac{151}{315} + \tfrac{397}{840} + \tfrac{1}{21} + \tfrac{1}{2520} = 1.$$

Thus, $\hat{a}(\omega)$ is bounded by

$$\tfrac{17}{315} \leq \hat{a}(\omega) \leq 1.$$

Therefore, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis with lower and upper Riesz bounds of $\frac{17}{315}$ and 1, respectively. □

4.2.3 Wavelet Spaces

As noted earlier, the MRA is one of the fundamental building blocks of wavelet systems. A MRA is simply a collection of subspaces called approximation (or scaling) spaces. These spaces, however, are not the only ones of interest in the context of wavelet systems. As we shall see, there is another set of spaces that are also very important.

Consider a MRA associated with the approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$. For each $p \in \mathbb{Z}$, since V_p is a proper subspace of V_{p-1}, there must be some space W_p such that W_p is the algebraic complement of V_p in V_{p-1} (by virtue of Theorem 2.9). That is, there must exist a space W_p such that

$$V_{p-1} = V_p \oplus W_p. \tag{4.9}$$

(Here, we are using the isomorphic form of the direct sum.) From this direct sum decomposition of V_{p-1}, it trivially follows that $V_p \cap W_p = \{0\}$. As a matter of terminology, we refer to W_p as a **wavelet space**. Thus, we can associate a sequence $\{W_p\}_{p\in\mathbb{Z}}$ of wavelet spaces with a MRA.

It follows from the structure of a MRA and the definition of the wavelet spaces that

$$\begin{gathered} V_k \cap V_l = V_l \quad \text{for } k < l, \\ W_k \cap W_l = \{0\} \quad \text{for } k \neq l, \quad \text{and} \\ V_k \cap W_l = \{0\} \quad \text{for } k \geq l. \end{gathered}$$

Through a trivial change of variable, we can rewrite (4.9) as

$$V_p = V_{p+1} \oplus W_{p+1}.$$

Applying this formula recursively several times yields

$$\begin{aligned} V_p &= V_{p+1} \oplus W_{p+1} \\ &= V_{p+2} \oplus W_{p+2} \oplus W_{p+1} \\ &= V_{p+3} \oplus W_{p+3} \oplus W_{p+2} \oplus W_{p+1}. \end{aligned}$$

Repeating this process ad infinitum, we obtain

$$V_p = \oplus_{k=p+1}^{\infty} W_k.$$

Taking the limit as $p \to -\infty$, we have

$$\lim_{p\to-\infty} V_p = \lim_{p\to-\infty} \bigoplus_{k=p+1}^{\infty} W_k.$$

From the density property of the MRA (i.e., $\lim_{p\to-\infty} V_p = L^2(\mathbb{R})$), we have

$$\operatorname{clos}\left(\bigoplus_{p\in\mathbb{Z}} W_p\right) = L^2(\mathbb{R}).$$

In other words, we have decomposed the space $L^2(\mathbb{R})$ into a sequence of mutually disjoint subspaces (i.e., the wavelet spaces). The basis functions for these wavelet spaces together form a Riesz basis for $L^2(\mathbb{R})$. Diagrammatically, we have decomposed $L^2(\mathbb{R})$ using the structure illustrated in Figure 4.10.

In general, we have that V_p and W_p are disjoint. Suppose now that, in addition, $V_p \perp W_p$. In other words, W_p is the orthogonal complement of V_p in V_{p-1}. In this case, not only is $V_{p-1} = V_p \oplus W_p$ but

$$V_{p-1} = V_p \overset{\perp}{\oplus} W_p,$$

$$\begin{tikzcd}
L^2(\mathbb{R}) \arrow[r] & \cdots V_{-2} \arrow[r] \arrow[dr] & V_{-1} \arrow[r] \arrow[dr] & V_0 \arrow[r] \arrow[dr] & V_1 \arrow[r] \arrow[dr] & V_2 \cdots \arrow[r] & \{0\} \\
& & W_{-1} & W_0 & W_1 & W_2 &
\end{tikzcd}$$

Figure 4.10: MRA decomposition of $L^2(\mathbb{R})$.

or equivalently

$$V_p = V_{p+1} \overset{\perp}{\oplus} W_{p+1}.$$

(Again, here we continue to use the isomorphic form of direct sum.) Recursive application of the preceding formula yields

$$\begin{aligned}
V_p &= V_{p+1} \overset{\perp}{\oplus} W_{p+1} \\
&= V_{p+2} \overset{\perp}{\oplus} W_{p+2} \overset{\perp}{\oplus} W_{p+1} \\
&= V_{p+3} \overset{\perp}{\oplus} W_{p+3} \overset{\perp}{\oplus} W_{p+2} \overset{\perp}{\oplus} W_{p+1}.
\end{aligned}$$

Continuing this process ad infinitum, we obtain

$$V_p = \overset{\perp}{\bigoplus}{}_{k=p+1}^{\infty} W_k.$$

Taking the limit as $p \to -\infty$, we have

$$\lim_{p \to -\infty} V_p = \lim_{p \to -\infty} \overset{\perp}{\bigoplus}{}_{k=p+1}^{\infty} W_k.$$

Using the density property of the MRA, we have

$$\operatorname{clos}\left(\overset{\perp}{\bigoplus}{}_{p \in \mathbb{Z}} W_p \right) = L^2(\mathbb{R}).$$

Thus, the space $L^2(\mathbb{R})$ has been decomposed into a sequence of mutually orthogonal subspaces (i.e., the wavelet spaces). The basis functions for these spaces collectively form an orthonormal basis for $L^2(\mathbb{R})$.

As we will see, depending on whether the wavelet subspaces are mutually orthogonal or only mutually disjoint, we obtain wavelet systems with distinct structures. The first case (where the wavelet spaces are mutually orthogonal) leads to what are called orthonormal and semiorthogonal wavelet systems. The second case leads to what are called biorthonormal wavelet systems.

From the earlier results, we have that a MRA is associated with a sequence $\{W_p\}_{p \in \mathbb{Z}}$ of wavelet spaces. It is now worthwhile to examine some of the properties possessed by this sequence of wavelet spaces. Several important properties are given by the theorem below.

Theorem 4.4 (Properties of wavelet spaces). *Let $\{W_p\}_{p \in \mathbb{Z}}$ denote the sequence of wavelet spaces associated with a MRA. The wavelet spaces are such that*

1. $\operatorname{clos}\left(\bigoplus_{p \in \mathbb{Z}} W_p \right) = L^2(\mathbb{R})$ *(density);*
2. *for all $p \in \mathbb{Z}$, $x(t) \in W_p \Leftrightarrow x(2t) \in W_{p-1}$ (scale invariance);*

3. *for all $k \in \mathbb{Z}$, $x(t) \in W_0 \Leftrightarrow x(t-k) \in W_0$ (shift invariance); and*

4. *there exists ψ such that $\{\psi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis of W_0 (shift-invariant Riesz basis).*

Proof. The density property essentially follows from the definition of $\{W_p\}_{p\in\mathbb{Z}}$ and properties of a MRA. The scale invariance and shift invariance properties follow from the corresponding properties of a MRA. The proof of the shift-invariant Riesz basis property is more involved and better deferred until later. □

From the above theorem, we see that a Riesz basis of the wavelet space W_0 is generated by the integer translates of a single prototype function ψ. As a matter of terminology, we refer to ψ as a **wavelet function**. Essentially, the wavelet function plays the same role for wavelet spaces as the scaling function does for approximation spaces.

At this point, a few comments of a pragmatic nature are in order. In practice, we often have a function represented in some approximation space, say V_ρ, and we want to find a representation in terms of the approximation space $V_{\rho'}$, where $\rho' > \rho$, and wavelet spaces W_p for $\rho < p \leq \rho'$. That is, we want to express the function in terms of its lower resolution representation plus the additional details necessary to obtain the original higher resolution version. In such a situation, we employ a decomposition of the form

$$\begin{aligned} V_\rho &= V_{\rho+1} \oplus W_{\rho+1} \\ &= V_{\rho+2} \oplus W_{\rho+2} \oplus W_{\rho+1} \\ &= V_{\rho'} \oplus \bigoplus_{k=\rho+1}^{\rho'} W_k. \end{aligned}$$

Notice that we are only using a finite number of approximation and wavelet spaces.

4.2.4 Bases of Scaling and Wavelet Spaces

Consider a MRA associated with approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$ and wavelet space sequence $\{W_p\}_{p\in\mathbb{Z}}$. Let $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ and $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}}$ denote the Riesz bases of V_0 and W_0, respectively. Suppose that ϕ and ψ are known. Then, it turns out that we can trivially find a basis for each of the other approximation and wavelet spaces.

To begin, we observe that the following result holds.

Theorem 4.5. *Suppose that the set $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis. Then, for each $p \in \mathbb{Z}$, the set $\{\theta_{p,k}\}_{k\in\mathbb{Z}}$ given by*

$$\theta_{p,k}(t) = 2^{-p/2}\theta(2^{-p}t - k)$$

is also a Riesz basis with the same Riesz bounds as $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$.

Proof. The proof makes use of Theorem 4.1 and is left as an exercise for the reader. □

Now, we consider the basis for each of the approximation spaces. The basis for each of these spaces can be determined using the theorem below.

Theorem 4.6 (Bases of approximation spaces). *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ and $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of V_0 with the dual basis $\{\tilde{\phi}(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, for each $p \in \mathbb{Z}$, the set $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$ given by*

$$\phi_{p,k}(t) = 2^{-p/2}\phi(2^{-p}t - k)$$

is a Riesz basis of V_p with the same Riesz bounds as $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ and with the dual basis $\{\tilde{\phi}_{p,k}\}_{k\in\mathbb{Z}}$ given by

$$\tilde{\phi}_{p,k}(t) = 2^{-p/2}\tilde{\phi}(2^{-p}t - k).$$

Proof. For any $f(t) \in V_p$, we have that $f(2^p t) \in V_0$. This follows from the scale invariance property of the MRA. Since $f(2^p t) \in V_0$, we can expand this function in terms of the basis of V_0. This gives us

$$\begin{aligned} f(2^p t) &= \sum_{n\in\mathbb{Z}} \left\langle f(2^p \cdot), \tilde{\phi}(\cdot - n) \right\rangle \phi(t-n) \\ &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} f(2^p \tau) \tilde{\phi}^*(\tau - n) d\tau \right] \phi(t-n). \end{aligned}$$

Now, we employ a change of variable. Let $t' = 2^p t$ so that $t = 2^{-p} t'$ and $dt = 2^{-p} dt'$. Also, let $\tau' = 2^p \tau$ so that $\tau = 2^{-p}\tau'$ and $d\tau = 2^{-p} d\tau'$. Applying the change of variable, we obtain

$$f(t') = \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} f(\tau') \tilde{\phi}^*(2^{-p}\tau' - n) 2^{-p} d\tau' \right] \phi(2^{-p} t' - n).$$

Dropping the primes and using straightforward algebraic manipulation, we have

$$\begin{aligned} f(t) &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} f(\tau) \tilde{\phi}^*(2^{-p}\tau - n) 2^{-p} d\tau \right] \phi(2^{-p} t - n) \\ &= \sum_{n\in\mathbb{Z}} 2^{-p} \left\langle f(\cdot), \tilde{\phi}(2^{-p} \cdot - n) \right\rangle \phi(2^{-p} t - n) \\ &= \sum_{n\in\mathbb{Z}} \left\langle f(\cdot), 2^{-p/2} \tilde{\phi}(2^{-p} \cdot - n) \right\rangle 2^{-p/2} \phi(2^{-p} t - n). \end{aligned}$$

Thus, we have shown that any $f(t) \in V_p$ has a representation of the form

$$f(t) = \sum_{n\in\mathbb{Z}} \left\langle f(\cdot), \tilde{\phi}_{p,n}(\cdot) \right\rangle \phi_{p,n}(t),$$

where

$$\begin{aligned} \tilde{\phi}_{p,k}(t) &= 2^{-p/2} \tilde{\phi}(2^{-p} t - k) \quad \text{and} \\ \phi_{p,k}(t) &= 2^{-p/2} \phi(2^{-p} t - k). \end{aligned}$$

We must now show that, for each $p \in \mathbb{Z}$, the family $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$ constitutes a Riesz basis. Observe that the set $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$ is of the same form considered by Theorem 4.5. Since $\{\phi_{0,k}(t)\}_{k\in\mathbb{Z}} = \{\phi(t-k)\}_{k\in\mathbb{Z}}$ is a Riesz basis for V_0, we can use Theorem 4.5 to conclude that $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$ is a Riesz basis for V_p with the same Riesz bounds as $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$. □

Now, we consider the basis of each of the wavelet spaces. The basis of each of these spaces can be determined using the theorem below.

Theorem 4.7 (Bases of wavelet spaces)**.** *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ with corresponding wavelet spaces $\{W_p\}_{p\in\mathbb{Z}}$ (i.e., $V_{p-1} = V_p \oplus W_p$) and $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of W_0 with the dual basis $\{\tilde{\psi}(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, for each $p \in \mathbb{Z}$, $\{\psi_{p,k}\}_{k\in\mathbb{Z}}$ given by*

$$\psi_{p,k}(t) = 2^{-p/2} \psi(2^{-p} t - k)$$

is a Riesz basis of W_p with the same Riesz bounds as $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}}$, and with a dual basis $\{\tilde{\psi}_{p,k}\}_{k\in\mathbb{Z}}$ given by

$$\tilde{\psi}_{p,k}(t) = 2^{-p/2} \tilde{\psi}(2^{-p} t - k).$$

Proof. The proof is essentially the same as the proof for Theorem 4.6 with the scaling function ϕ replaced by the wavelet function ψ. □

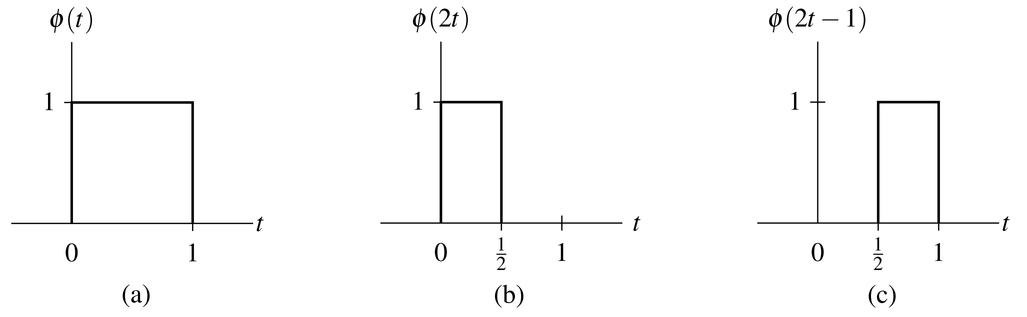


Figure 4.11: Example of a refinable function. (a) The Haar scaling function. The (b) first and (c) second terms in the refinement equation.

4.2.5 Scaling and Wavelet Equations

In the context of wavelet systems, we often encounter equations that relate a function to translated and dilated versions of itself. We more formally define this particular type of equation below.

Definition 4.2 (Refinement equation)**.** An equation of the form

$$\phi(t) = \sum_{n\in\mathbb{Z}} a[n]\phi(Mt-n), \tag{4.10}$$

where $M \in \mathbb{Z}$ and $M \geq 2$, is called an M-scale **refinement equation** (or **dilation equation**). The sequence $a[n]$ is referred to as a **refinement mask**. The solution of (4.10) is called an **M-refinable function** (or distribution). The **symbol** of a refinable function is the quantity $\frac{1}{M}\hat{a}(\omega/M)$.

Refinement equations play a crucial role in wavelet systems. Clearly, any dilation equation admits a trivial solution of $\phi(t) \equiv 0$. We are usually interested in nontrivial solutions, however. If a nontrivial solution does exist, the solution is not unique. If $\phi(t)$ is a solution, then $\alpha\phi(t)$ is also a solution, where α is a constant.

For the time being, we will only interest ourselves in 2-scale refinement equations and 2-refinable functions. A few examples of refinable functions are given below. Although the functions in these examples can be expressed in closed form, many refinable functions cannot be expressed in this way.

Example 4.10 (Haar scaling function)**.** Consider the function

$$\phi(t) = \chi_{[0,1)}(t).$$

One can confirm that ϕ satisfies the following refinement equation

$$\phi(t) = \phi(2t) + \phi(2t-1).$$

This refinement relationship is illustrated in Figure 4.11.

Example 4.11 (Linear B-spline scaling function)**.** Consider the function

$$\phi(t) = \begin{cases} 1-|t| & \text{for } |t| < 1 \\ 0 & \text{otherwise.} \end{cases}$$

One can confirm that ϕ satisfies the following refinement equation

$$\phi(t) = \tfrac{1}{2}\phi(2t+1) + \phi(2t) + \tfrac{1}{2}\phi(2t-1).$$

This refinement relationship is illustrated in Figure 4.12.

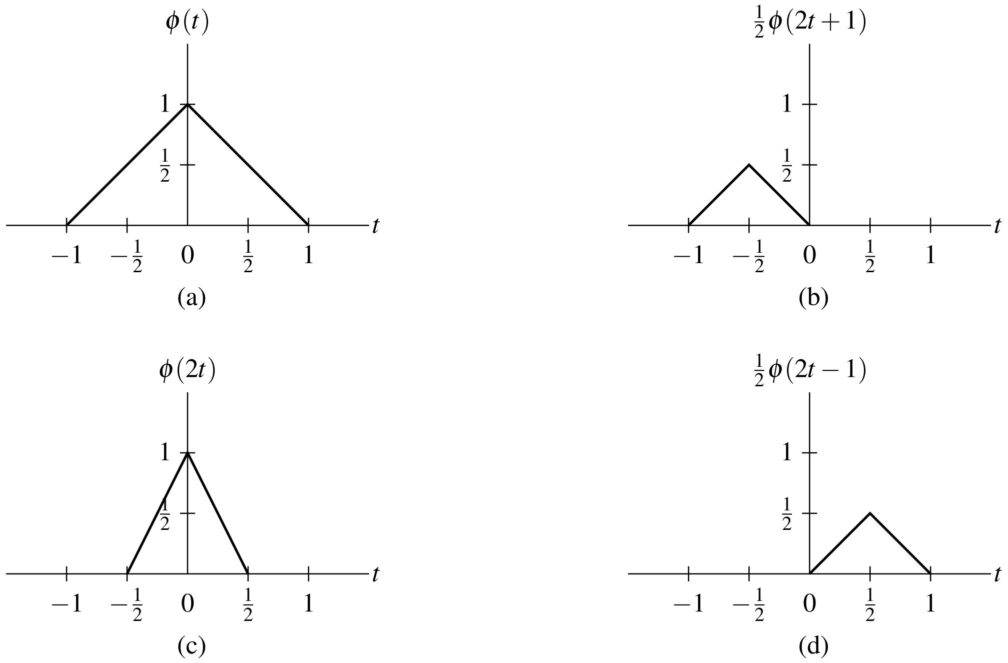


Figure 4.12: Linear B-spline scaling function. (a) scaling function The (a) first, (b) second, and (c) third terms in the refinement equation.

As it turns out, the scaling function ϕ of a MRA satisfies a refinement equation (i.e., ϕ is refinable). The particular form of the refinement relationship is given by the theorem below.

Theorem 4.8 (Scaling equation)**.** *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ and V_0 has the Riesz basis $\{\phi(\cdot-n)\}_{n\in\mathbb{Z}}$. Then, ϕ satisfies a refinement equation of the form*

$$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} c[n]\phi(2t-n), \tag{4.11a}$$

where

$$c[n] = \left\langle \phi(\cdot), \sqrt{2}\tilde{\phi}(2\cdot -n)\right\rangle. \tag{4.11b}$$

Proof. Trivially, we have that $\phi(t) \in V_0$. From the scale invariance property of the MRA, we know that $\phi(t) \in V_0 \Rightarrow \phi(t/2) \in V_1$. Since, from the nesting property of the MRA, $V_1 \subset V_0$, we can further deduce that $\phi(t/2) \in V_0$. Consequently, $\phi(t/2)$ can be expanded in terms of the basis of V_0 as

$$\begin{aligned}\phi(\tfrac{t}{2}) &= \sum_{n\in\mathbb{Z}} \left\langle \phi(\tfrac{1}{2}\cdot), \tilde{\phi}(\cdot-n)\right\rangle \phi(t-n)\\ &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \phi(\tfrac{\tau}{2})\tilde{\phi}^*(\tau-n)d\tau\right]\phi(t-n)\end{aligned}$$

which, by substitution, is equivalent to

$$\phi(t) = \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \phi(\tfrac{\tau}{2})\tilde{\phi}^*(\tau-n)d\tau\right]\phi(2t-n).$$

Now, we employ a change of variable. Let $\tau' = \frac{1}{2}\tau$ so that $\tau = 2\tau'$ and $d\tau = 2d\tau'$. Applying the change of variable and then performing straightforward algebraic manipulation, we obtain

$$\begin{aligned}
\phi(t) &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \phi(\tau')\tilde{\phi}^*(2\tau'-n)2d\tau'\right] \phi(2t-n) \\
&= 2\sum_{n\in\mathbb{Z}} \left\langle \phi(\cdot), \tilde{\phi}(2\cdot -n)\right\rangle \phi(2t-n) \\
&= \sqrt{2}\sum_{n\in\mathbb{Z}} \left\langle \phi(\cdot), \sqrt{2}\tilde{\phi}(2\cdot -n)\right\rangle \phi(2t-n) \\
&= \sqrt{2}\sum_{n\in\mathbb{Z}} c[n]\phi(2t-n).
\end{aligned}$$

□

The refinement mask in (4.11a) is the sequence $c'[n] = \sqrt{2}c[n]$. One might wonder why we write the scaling equation in terms of $c[n]$ instead of the refinement mask $c'[n]$ directly. This choice is somewhat arbitrary. Many authors, however, have adopted this convention. For this reason, this convention has been adopted herein. One must be mindful that more than one convention is used in the literature, however. Depending on which convention is employed, various formulae derived from the scaling equation may contain different constants.

Often, we are interested in the Fourier transform of the scaling equation. Sometimes, the Fourier transform representation provides us with additional insight into the behavior of the scaling function. The Fourier transform of the scaling equation is given by the theorem below.

Theorem 4.9 (Fourier transform of the scaling equation)**.** *Let ϕ be a scaling function with scaling equation coefficient sequence c. Then, $\hat{\phi}$ is given by*

$$\hat{\phi}(\omega) = \tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{2})\hat{\phi}(\tfrac{\omega}{2}) \tag{4.12}$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\phi}(\omega) = \hat{\phi}(0)\prod_{p=1}^{\infty} \frac{\hat{c}(2^{-p}\omega)}{\sqrt{2}}. \tag{4.13}$$

Proof. Taking the Fourier transform of both sides of the scaling equation (4.11a) yields

$$\begin{aligned}
\hat{\phi}(\omega) &= \tfrac{1}{2}(\sqrt{2})\sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega/2}\hat{\phi}(\tfrac{\omega}{2}) \\
&= \tfrac{1}{\sqrt{2}}\left[\sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega/2}\right]\hat{\phi}(\tfrac{\omega}{2}).
\end{aligned}$$

Let $\hat{c}(\omega) = \sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega}$ (i.e., $\hat{c}$ is the (discrete-time) Fourier transform of c). With this definition, we have

$$\hat{\phi}(\omega) = \tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{2})\hat{\phi}(\tfrac{\omega}{2}).$$

Recursively applying this formula, we have

$$\begin{aligned}
\hat{\phi}(\omega) &= \tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{2})\hat{\phi}(\tfrac{\omega}{2}) \\
&= \tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{2})\left[\tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{4})\hat{\phi}(\tfrac{\omega}{4})\right] \\
&= (\tfrac{1}{\sqrt{2}})^2\hat{c}(\tfrac{\omega}{2})\hat{c}(\tfrac{\omega}{4})\hat{\phi}(\tfrac{\omega}{4}) \\
&= (\tfrac{1}{\sqrt{2}})^2\hat{c}(\tfrac{\omega}{2})\hat{c}(\tfrac{\omega}{4})\left[\tfrac{1}{\sqrt{2}}\hat{c}(\tfrac{\omega}{8})\hat{\phi}(\tfrac{\omega}{8})\right] \\
&= (\tfrac{1}{\sqrt{2}})^3\hat{c}(\tfrac{\omega}{2})\hat{c}(\tfrac{\omega}{4})\hat{c}(\tfrac{\omega}{8})\hat{\phi}(\tfrac{\omega}{8}) \\
&= \left[\prod_{p=1}^{N} \tfrac{1}{\sqrt{2}}\hat{c}(2^{-p}\omega)\right]\hat{\phi}(2^{-N}\omega).
\end{aligned}$$

Taking the limit as $N \to \infty$, we obtain

$$\hat{\phi}(\omega) = \hat{\phi}(0) \prod_{p=1}^{\infty} \tfrac{1}{\sqrt{2}} \hat{c}(2^{-p}\omega).$$

□

Although the wavelet function does not satisfy a refinement equation, it can be expressed in terms of dilated and translated versions of the scaling function. This leads to what is known as the wavelet equation, given by the theorem below.

Theorem 4.10 (Wavelet equation)**.** *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ with wavelet subspaces $\{W_p\}_{p\in\mathbb{Z}}$, where V_0 has the Riesz basis $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ and W_0 has the Riesz basis $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, $\psi(t)$ can be expressed in terms of an equation of the form*

$$\psi(t) = \sqrt{2} \sum_{n\in\mathbb{Z}} d[n] \phi(2t - n)$$

where

$$d[n] = \left\langle \psi(\cdot), \sqrt{2}\tilde{\phi}(2\cdot - n) \right\rangle.$$

Proof. Trivially, we have that $\psi(t) \in W_0$. From the nesting property of the MRA, $W_0 \subset V_{-1} \Rightarrow \psi(t) \in V_{-1}$. From the scale invariance property of the MRA, $\psi(t) \in V_{-1} \Rightarrow \psi(t/2) \in V_0$. Consequently, $\psi(t/2)$ can be expanded in terms of the basis of V_0 as

$$\begin{aligned}\psi(\tfrac{1}{2}t) &= \sum_{n\in\mathbb{Z}} \left\langle \psi(\tfrac{1}{2}\cdot), \tilde{\phi}(\cdot - n) \right\rangle \phi(t-n) \\ &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \psi(\tfrac{\tau}{2}) \tilde{\phi}^*(\tau - n) d\tau \right] \phi(t-n).\end{aligned}$$

Now, we employ a change of variable. Let $t' = \frac{1}{2}t$ so that $t = 2t'$ and $dt = 2dt'$. Also, let $\tau' = \tau/2$ so that $\tau = 2\tau'$ and $d\tau = 2d\tau'$. Applying the change of variable, we obtain

$$\begin{aligned}\psi(t') &= \sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \psi(\tau') \tilde{\phi}^*(2\tau' - n) 2d\tau' \right] \phi(2t'-n) \\ &= 2\sum_{n\in\mathbb{Z}} \left[\int_{-\infty}^{\infty} \psi(\tau') \tilde{\phi}^*(2\tau' - n) d\tau' \right] \phi(2t'-n).\end{aligned}$$

Dropping the primes and simplifying, we have

$$\begin{aligned}\psi(t) &= 2\sum_{n\in\mathbb{Z}} \left\langle \psi(\cdot), \tilde{\phi}(2\cdot - n) \right\rangle \phi(2t-n) \\ &= \sqrt{2}\sum_{n\in\mathbb{Z}} \left\langle \psi(\cdot), \sqrt{2}\tilde{\phi}(2\cdot - n) \right\rangle \phi(2t-n) \\ &= \sqrt{2}\sum_{n\in\mathbb{Z}} d[n] \phi(2t-n).\end{aligned}$$

□

Note that the wavelet equation is not a refinement equation. That is, while the scaling function ϕ is refinable, the wavelet function ψ is not refinable.

Often, we are interested in the Fourier transform of the wavelet equation. This is given by the theorem below.

Theorem 4.11 (Fourier transform of wavelet equation)**.** *Let ϕ and ψ be the scaling and wavelet functions of a MRA. Suppose that ϕ has scaling equation coefficient sequence c and ψ has the wavelet equation coefficient sequence d. Then, $\hat{\psi}$ is given by*

$$\hat{\psi}(\omega) = \tfrac{1}{\sqrt{2}}\hat{d}(\tfrac{\omega}{2})\hat{\phi}(\tfrac{\omega}{2})$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\psi}(\omega) = \tfrac{1}{\sqrt{2}}\hat{\phi}(0)\hat{d}(\tfrac{\omega}{2})\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(2^{-p-1}\omega).$$

Proof. Taking the Fourier transform of both sides of the wavelet equation, we obtain

$$\begin{aligned}\hat{\psi}(\omega) &= \sqrt{2}\sum_{n\in\mathbb{Z}} d[n]\tfrac{1}{2}\hat{\phi}(\tfrac{\omega}{2})e^{-jn\omega/2} \\ &= \tfrac{1}{\sqrt{2}}\left[\sum_{n\in\mathbb{Z}} d[n]e^{-jn\omega/2}\right]\hat{\phi}(\tfrac{\omega}{2}).\end{aligned}$$

Let $\hat{d}(\omega) = \sum_{n\in\mathbb{Z}} d[n]e^{-jn\omega}$ (i.e., $\hat{d}$ is the discrete-time Fourier transform of d). With this definition, we can rewrite the above equation as

$$\hat{\psi}(\omega) = \tfrac{1}{\sqrt{2}}\hat{d}(\tfrac{\omega}{2})\hat{\phi}(\tfrac{\omega}{2}).$$

Using Theorem 4.9, we can re-express $\hat{\phi}(\omega/2)$ in terms of an infinite product, and then simplify to obtain

$$\begin{aligned}\hat{\psi}(\omega) &= \tfrac{1}{\sqrt{2}}\hat{d}(\tfrac{\omega}{2})\left[\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(2^{-p}\omega/2)\right]\hat{\phi}(0) \\ &= \tfrac{1}{\sqrt{2}}\hat{\phi}(0)\hat{d}(\tfrac{\omega}{2})\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(2^{-p-1}\omega).\end{aligned}$$

□

4.2.6 Generating MRAs from Scaling Functions

Rather than defining an MRA in terms of its approximation spaces, we can use the scaling function as the starting point for defining a MRA. To begin, we make an appropriate choice of scaling function ϕ. Then, from this scaling function, we can generate each of the approximation spaces of the MRA. This process is formalized by the theorem below.

Theorem 4.12. *Suppose that $\phi \in L^2(\mathbb{R})$ satisfies a refinement relation of the form*

$$\phi(t) = \sum_{k\in\mathbb{Z}} c[k]\phi(2t-k),$$

where $\sum_{k\in\mathbb{Z}} |c[k]|^2 < \infty$. Further, suppose that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis for the space that it generates. Define

$$\phi_{p,k}(t) = 2^{-p/2}\phi(2^{-p}t-k)$$

and let V_p be the space generated by $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$. Then, the sequence $\{V_p\}_{p\in\mathbb{Z}}$ of spaces constitutes a MRA.

Proof. NESTING. First, let us show that the nesting property is satisfied. In other words, we must prove that

$$\text{for all } p \in \mathbb{Z},\ x \in V_p \Rightarrow x \in V_{p-1}.$$

Suppose that $x(t) \in V_p$. Since $x(t) \in V_p$, we can expand $x(t)$ in terms of the basis of V_p as

$$\begin{aligned} x(t) &= \sum_{k \in \mathbb{Z}} a_k \phi_{p,k}(t) \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \phi(2^{-p}t - k). \end{aligned}$$

Using the refinement relation for $\phi(t)$, we can rewrite this equation as

$$\begin{aligned} x(t) &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \left[\sum_{l \in \mathbb{Z}} c[l] \phi(2[2^{-p}t - k] - l) \right] \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \sum_{l \in \mathbb{Z}} c[l] \phi(2^{-(p-1)}t - 2k - l). \end{aligned}$$

Now, we employ a change of variable. Let $l' = 2k + l$ so that $l = l' - 2k$. Applying the change of variable and dropping the primes, we have

$$\begin{aligned} x(t) &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \sum_{l \in \mathbb{Z}} c[l - 2k] \phi(2^{-(p-1)}t - l) \\ &= \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} 2^{-p/2} a_k c[l - 2k] \phi(2^{-(p-1)}t - l) \\ &= \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} 2^{1/2} a_k c[l - 2k] \left[2^{-(p-1)/2} \phi(2^{-(p-1)}t - l) \right] \\ &= \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} 2^{1/2} a_k c[l - 2k] \phi_{p-1,l}(t) \\ &= \sum_{l \in \mathbb{Z}} \left(\sum_{k \in \mathbb{Z}} 2^{1/2} a_k c[l - 2k] \right) \phi_{p-1,l}(t). \end{aligned}$$

Thus, we have expressed $x(t)$ in terms of the basis of V_{p-1}. This implies that $x(t) \in V_{p-1}$. Therefore, the nesting property is satisfied.

SHIFT INVARIANCE. Now, let us show that the shift invariance property is satisfied. In other words, we must show

$$\text{for all } n \in \mathbb{Z},\ x(t) \in V_0 \Leftrightarrow x(t - n) \in V_0.$$

Suppose that $x \in V_0$. Since $x \in V_0$, we can expand x in terms of the basis of V_0 as

$$x(t) = \sum_{k \in \mathbb{Z}} a_k \phi(t - k).$$

Substituting $t - n$ for t in the preceding equation, we obtain

$$x(t - n) = \sum_{k \in \mathbb{Z}} a_k \phi(t - n - k).$$

Now we employ a change of variable. Let $k' = n + k$ so that $k = k' - n$. Applying this change of variable and dropping the primes, we have

$$x(t - n) = \sum_{k \in \mathbb{Z}} a_{k-n} \phi(t - k).$$

Thus, we have expressed $x(t - n)$ in terms of the basis of V_0. Thus, $x(t - n) \in V_0$. Therefore, the shift invariance property holds.

SCALE INVARIANCE. Now, let us prove that the scale invariance property is satisfied. That is, we must show

$$\text{for all } p \in \mathbb{Z},\ x(t) \in V_p \Leftrightarrow x(2t) \in V_{p-1}.$$

First, we show that $x(t) \in V_p \Rightarrow x(2t) \in V_{p-1}$. Suppose that $x(t) \in V_p$. Since $x(t) \in V_p$, we can expand $x(t)$ in terms of the basis of V_p as

$$\begin{aligned} x(t) &= \sum_{k \in \mathbb{Z}} a_k \phi_{p,k}(t) \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \phi(2^{-p}t - k). \end{aligned}$$

Substituting $2t$ for t in this equation, we obtain

$$\begin{aligned} x(2t) &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \phi(2^{-p}[2t] - k) \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-p/2} \phi(2^{-(p-1)}t - k) \\ &= \sum_{k \in \mathbb{Z}} 2^{-1/2} a_k 2^{-(p-1)/2} \phi(2^{-(p-1)}t - k) \\ &= \sum_{k \in \mathbb{Z}} 2^{-1/2} a_k \phi_{p-1,k}(t). \end{aligned}$$

Thus, we have expressed $x(2t)$ in terms of the basis of V_{p-1}. This implies that $x(2t) \in V_{p-1}$.

Now, we show $x(2t) \in V_{p-1} \Rightarrow x(t) \in V_p$. Suppose that $x(2t) \in V_{p-1}$. Since $x(2t) \in V_{p-1}$, we can expand $x(2t)$ in terms of the basis of V_{p-1} as

$$\begin{aligned} x(2t) &= \sum_{k \in \mathbb{Z}} a_k \phi_{p-1,k}(t) \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-(p-1)/2} \phi(2^{-(p-1)}t - k). \end{aligned}$$

Substituting $t/2$ for t in this equation, we obtain

$$\begin{aligned} x(t) &= \sum_{k \in \mathbb{Z}} a_k 2^{-(p-1)/2} \phi(2^{-(p-1)}t/2 - k) \\ &= \sum_{k \in \mathbb{Z}} a_k 2^{-(p-1)/2} \phi(2^{-p}t - k) \\ &= \sum_{k \in \mathbb{Z}} 2^{1/2} a_k 2^{-p/2} \phi(2^{-p}t - k) \\ &= \sum_{k \in \mathbb{Z}} 2^{1/2} a_k \phi_{p,k}(t). \end{aligned}$$

Thus, we have expressed $x(t)$ in terms of the basis of V_p. This implies that $x(t) \in V_p$. Combining the above results, we have that the scale invariance property holds.

DENSITY AND SEPARATION. One can also show that the separation and density properties hold [9, Proposition 5.3.1, p. 141], [9, Proposition 5.3.2, p. 142]. The proof is somewhat technical and omitted here. (In the case of the density property, the assumption is made that $\hat{\phi}(\omega)$ is bounded for all ω and continuous near $\omega = 0$.)

□

4.2.7 Dual MRAs

Consider a MRA $\{V_p\}_{p \in \mathbb{Z}}$ with scaling function ϕ, wavelet space sequence $\{W_p\}_{p \in \mathbb{Z}}$, and the corresponding wavelet function ψ. In order to compute the expansion coefficients of an arbitrary function in terms of the basis of each of the approximation and/or wavelet spaces, we must take an inner product of the function with the appropriate dual basis functions. In this context, the functions $\tilde{\phi}$ and $\tilde{\psi}$ become quite important. So far, we have focused our attention primarily on ϕ and ψ and the spaces that they generate. One might wonder what structure (if any) is associated with the functions $\tilde{\phi}$ and $\tilde{\psi}$. An interesting result in this regard is given by the theorem below.

Theorem 4.13 (Dual MRAs)**.** *Let $\{V_p\}_{p\in\mathbb{Z}}$ be a MRA with scaling function ϕ, wavelet space sequence $\{W_p\}_{p\in\mathbb{Z}}$, and corresponding wavelet function ψ. Suppose that the dual Riesz bases of $\{\phi(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\psi(\cdot-k)\}_{k\in\mathbb{Z}}$ are given by $\{\tilde{\phi}(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\tilde{\psi}(\cdot-k)\}_{k\in\mathbb{Z}}$, respectively. Then, $\tilde{\phi}$ is also the scaling function of a MRA $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ with wavelet space sequence $\{\tilde{W}_p\}_{p\in\mathbb{Z}}$ and the corresponding wavelet function $\tilde{\psi}$.*

The above theorem is significant because it shows that MRAs occur in pairs. As a matter of terminology, $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ is said to be the **dual MRA** of $\{V_p\}_{p\in\mathbb{Z}}$. Furthermore, it follows that if $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ is the dual of $\{V_p\}_{p\in\mathbb{Z}}$ then $\{V_p\}_{p\in\mathbb{Z}}$ is also trivially the dual of $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$. In other words, this duality property is symmetric. As a matter of terminology, we refer to $\tilde{\phi}$ and $\tilde{\psi}$ as the **dual scaling function** and **dual wavelet function**, respectively.

An important relationship exists between the approximation and wavelet spaces of a MRA and its dual as stated by the theorem below.

Theorem 4.14. *Suppose that $\{V_p\}_{p\in\mathbb{Z}}$ and $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ are dual MRAs with corresponding wavelet space sequences $\{W_p\}_{p\in\mathbb{Z}}$ and $\{\tilde{W}_p\}_{p\in\mathbb{Z}}$, respectively. Then, we have*

$$\text{for all } p\in\mathbb{Z},\quad V_p\perp\tilde{W}_p \quad\text{and}\quad W_p\perp\tilde{V}_p.$$

Proof. Let $\{\phi_{p,k}\}_{p\in\mathbb{Z}}$ and $\{\tilde{\phi}_{p,k}\}_{p\in\mathbb{Z}}$ be Riesz bases of V_p and $\tilde{V}_p$ as defined previously. We define the following (possibly oblique) projection operators:

$$\begin{aligned}
P_p f &= \sum_{k\in\mathbb{Z}} \left\langle f,\tilde{\phi}_{p,k}\right\rangle \phi_{p,k},\\
\tilde{P}_p f &= \sum_{k\in\mathbb{Z}} \left\langle f,\phi_{p,k}\right\rangle \tilde{\phi}_{p,k},\\
Q_p f &= P_{p-1}f - P_p f, \quad\text{and}\\
\tilde{Q}_p f &= \tilde{P}_{p-1}f - \tilde{P}_p f.
\end{aligned}$$

(Note that the range spaces of Q_p and $\tilde{Q}_p$ are W_p and $\tilde{W}_p$, respectively.) In other words, P_p, $\tilde{P}_p$, Q_p, and $\tilde{Q}_p$ are projections onto V_p, $\tilde{V}_p$, W_p, and $\tilde{W}_p$, respectively.

First, we show that $W_p\perp\tilde{V}_p$ for all $p\in\mathbb{Z}$. Choose any arbitrary vector $f\in W_p$. Since $f\in W_p$, we have $Q_p f = f$, or equivalently

$$P_{p-1}f - P_p f = f.$$

Since $f\in W_p\subset V_{p-1}$, we also have

$$P_{p-1}f = f.$$

Combining the above two expressions for f, we have that $P_p f = 0$, or equivalently

$$\sum_{k\in\mathbb{Z}} \left\langle f,\tilde{\phi}_{p,k}\right\rangle \phi_{p,k} = 0.$$

Thus, $\left\langle f,\tilde{\phi}_{p,k}\right\rangle = 0$ for all $k\in\mathbb{Z}$. This implies that $f\perp\tilde{V}_p$. Furthermore, since f was chosen as any arbitrary vector in W_p, we also have that $W_p\perp\tilde{V}_p$.

Now, we show that $V_p\perp\tilde{W}_p$ for all $p\in\mathbb{Z}$. Choose any arbitrary vector $f\in\tilde{W}_p$. Since $f\in\tilde{W}_p$, we have $\tilde{Q}_p f = f$, or equivalently

$$\tilde{P}_{p-1}f - \tilde{P}_p f = f.$$

Since $f\in\tilde{W}_p\subset\tilde{V}_{p-1}$, we also have

$$\tilde{P}_{p-1}f = f.$$

Combining the above two expressions for f, we have that $\tilde{P}_p f = 0$, or equivalently

$$\sum_{k\in\mathbb{Z}} \left\langle f,\phi_{p,k}\right\rangle \tilde{\phi}_{p,k} = 0.$$

Thus, $\left\langle f,\phi_{p,k}\right\rangle = 0$ for all $k\in\mathbb{Z}$. This implies that $f\perp V_p$. Furthermore, since f was chosen as any arbitrary vector in $\tilde{W}_p$, we also have that $\tilde{W}_p\perp V_p$. □

In light of the above result, we observe that if the corresponding approximation and wavelet spaces associated with a MRA are orthogonal (i.e., for all $p \in \mathbb{Z}$, $V_p \perp W_p$), then the MRA is self dual.

4.2.8 Wavelet Systems

A wavelet system is simply a basis of $L^2(\mathbb{R})$ that is derived from a MRA. When constructing wavelet systems, we have a number of degrees of freedom available to us. In particular, we have some flexibility in the structure of approximation and wavelet spaces and the bases employed for these spaces. By exploiting this flexibility, we can obtain wavelet systems with differing types of structure. This leads to three different types of wavelet systems: orthonormal, semiorthogonal, and biorthonormal.

In what follows, we consider a MRA $\{V_p\}_{p\in\mathbb{Z}}$ with scaling function ϕ, wavelet space sequence $\{W_p\}_{p\in\mathbb{Z}}$, and corresponding wavelet function ψ. Let the dual Riesz bases of $\{\phi(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\psi(\cdot-k)\}_{k\in\mathbb{Z}}$ be given by $\{\tilde{\phi}(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\tilde{\psi}(\cdot-k)\}_{k\in\mathbb{Z}}$, respectively. Let the dual MRA of $\{V_p\}_{p\in\mathbb{Z}}$ be $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ with wavelet space sequence $\{\tilde{W}_p\}_{p\in\mathbb{Z}}$.

The most constrained type of wavelet system is an orthonormal wavelet system. With this type of system, the basis of each of the approximation and wavelet spaces is chosen to be orthonormal, and each wavelet space is chosen to be orthogonal to its corresponding approximation space. That is, we have

$$\{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is orthonormal}, \quad \{\psi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is orthonormal}, \quad \text{and}$$
$$\text{for each } p \in \mathbb{Z},\ V_p \perp W_p.$$

From this, it follows that

$$\tilde{\phi} = \phi, \quad \tilde{\psi} = \psi, \quad \{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \perp \{\psi(\cdot-k)\}_{k\in\mathbb{Z}},$$
$$\tilde{V}_p = V_p, \quad \tilde{W}_p = W_p, \quad \{W_p\}_{p\in\mathbb{Z}} \text{ is mutually orthogonal.}$$

Clearly, the MRA $\{V_p\}_{p\in\mathbb{Z}}$ is self dual. That is, there is only one (distinct) MRA associated with an orthonormal wavelet system.

Sometimes, it can be overly restrictive to require the use of an orthonormal basis for each of the approximation and wavelet spaces. Dropping this constraint leads to what is called a semiorthogonal wavelet system. With this type of system, we choose to use a Riesz basis for each of the approximation and wavelet spaces, and each wavelet space is chosen to be orthogonal to its corresponding approximation space. That is, we have

$$\{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is a Riesz basis}, \quad \{\psi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is a Riesz basis}, \quad \text{and}$$
$$\text{for each } p \in \mathbb{Z},\ V_p \perp W_p.$$

From this, it follows that

$$\{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ and } \{\tilde{\phi}(\cdot-k)\}_{k\in\mathbb{Z}} \text{ are dual Riesz bases,}$$
$$\{\psi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ and } \{\tilde{\psi}(\cdot-k)\}_{k\in\mathbb{Z}} \text{ are dual Riesz bases,}$$
$$\{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \perp \{\psi(\cdot-k)\}_{k\in\mathbb{Z}},$$
$$\{\tilde{\phi}(\cdot-k)\}_{k\in\mathbb{Z}} \perp \{\tilde{\psi}(\cdot-k)\}_{k\in\mathbb{Z}},$$
$$\tilde{V}_p = V_p, \quad \tilde{W}_p = W_p, \quad \text{and} \quad \{W_p\}_{p\in\mathbb{Z}} \text{ is mutually orthogonal.}$$

Clearly, the MRA $\{V_p\}_{p\in\mathbb{Z}}$ is self dual. Thus, there is only one (distinct) MRA associated with a semiorthogonal system.

Sometimes, even the requirement that the corresponding approximation and wavelet spaces be orthogonal is too restrictive. By dropping this constraint, we obtain what is called a biorthonormal wavelet system. That is, we have

$$\{\phi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is a Riesz basis}, \quad \{\psi(\cdot-k)\}_{k\in\mathbb{Z}} \text{ is a Riesz basis}, \quad \text{and}$$
$$\text{for each } p \in \mathbb{Z},\ V_p \perp \tilde{W}_p \text{ and } W_p \perp \tilde{V}_p.$$

From this, it follows that

$$\begin{gathered}
\{\phi(\cdot - k)\}_{k\in\mathbb{Z}} \text{ and } \{\tilde{\phi}(\cdot - k)\}_{k\in\mathbb{Z}} \text{ are dual Riesz bases,}\\
\{\psi(\cdot - k)\}_{k\in\mathbb{Z}} \text{ and } \{\tilde{\psi}(\cdot - k)\}_{k\in\mathbb{Z}} \text{ are dual Riesz bases,}\\
\{\phi(\cdot - k)\}_{k\in\mathbb{Z}} \perp \{\tilde{\psi}(\cdot - k)\}_{k\in\mathbb{Z}},\\
\{\psi(\cdot - k)\}_{k\in\mathbb{Z}} \perp \{\tilde{\phi}(\cdot - k)\}_{k\in\mathbb{Z}},\\
\{W_p\}_{p\in\mathbb{Z}} \text{ is mutually disjoint,} \quad \text{and}\\
\{\tilde{W}_p\}_{p\in\mathbb{Z}} \text{ is mutually disjoint.}
\end{gathered}$$

With a biorthonormal system, it is not necessarily true that $V_p \perp W_p$. Consequently, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ and $\{\psi(\cdot - k)\}_{k\in\mathbb{Z}}$ are not necessarily orthogonal. Moreover, the spaces in $\{W_p\}_{p\in\mathbb{Z}}$ are not necessarily mutually orthogonal. These spaces are only guaranteed to be mutually disjoint. The MRA $\{V_p\}_{p\in\mathbb{Z}}$ is no longer necessarily self dual. Thus, there are potentially two distinct MRAs associated with a biorthonormal wavelet system.

As one proceeds from orthonormal to semiorthogonal to biorthonormal systems, the amount of structure decreases (i.e., the degrees of freedom increase). Although orthonormal wavelets systems are often desirable, due to the orthonormal bases employed, the orthonormality constraint can often be overly restrictive. For this reason, some applications use biorthonormal (or semiorthogonal) wavelet systems.

4.2.9 Examples of Wavelet Systems

In this section, we introduce a number of wavelet systems.

Example 4.12 (Haar wavelet system)**.** One of the simplest examples of an orthonormal wavelet system is the Haar wavelet system. This system is associated with the MRA based on piecewise constant approximations, as introduced in Example 4.1.

The scaling function ϕ satisfies the refinement relationship

$$\phi(t) = \sqrt{2}\left(\tfrac{1}{\sqrt{2}}\phi(2t) + \tfrac{1}{\sqrt{2}}\phi(2t-1)\right).$$

The wavelet function ψ can be expressed in terms of the scaling function as

$$\psi(t) = \sqrt{2}\left(\tfrac{1}{\sqrt{2}}\phi(2t) - \tfrac{1}{\sqrt{2}}\phi(2t-1)\right).$$

Fortunately, ϕ and ψ can be expressed in closed form as

$$\phi(t) = \chi_{[0,1)}(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{and}$$

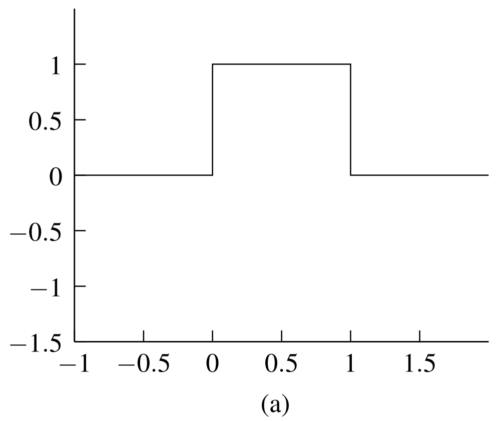
$$\psi(t) = \chi_{[0,1/2)}(t) - \chi_{[1/2,1)}(t) = \begin{cases} 1 & \text{if } 0 \le t < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \le t < 1 \\ 0 & \text{otherwise.} \end{cases}$$

These two functions are plotted in Figure 4.13. (Since the system is orthonormal, $\tilde{\phi} = \phi$ and $\tilde{\psi} = \psi$.)

Example 4.13 (Shannon wavelet system)**.** Another classic example of an orthonormal wavelet system is the Shannon wavelet system. This system is associated with the MRA introduced in Example 4.2, which is based on spaces of bandlimited functions.

The scaling function ϕ satisfies the refinement equation

$$\phi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} c_k \phi(2t-k)$$



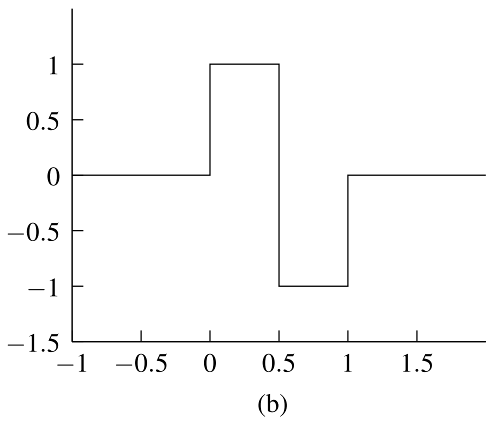


Figure 4.13: The Haar scaling and wavelet functions. The (a) scaling function and (b) wavelet function.

where

$$c_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ \frac{\sqrt{2}(-1)^{(k-1)/2}}{k\pi} & \text{odd } k \\ 0 & \text{even } k,\ k \neq 0. \end{cases}$$

The wavelet function ψ can be expressed in terms of the scaling function as

$$\psi(t) = \sqrt{2} \sum_{k \in \mathbb{Z}} d_k \phi(2t - k)$$

where $d_k = (-1)^k c_k$.

Fortunately, ϕ and ψ can be expressed in closed form as

$$\phi(t) = \operatorname{sinc} \pi t \quad \text{and}$$
$$\psi(t) = \left(\cos \tfrac{3\pi}{2} t\right) \operatorname{sinc} \tfrac{\pi}{2} t.$$

These two functions are plotted in Figure 4.14. (Since this system is orthonormal, $\tilde{\phi} = \phi$ and $\tilde{\psi} = \psi$.) In passing, we note that

$$\hat{\phi}(\omega) = \chi_{[-\pi,\pi]}(\omega) \quad \text{and} \quad \hat{\psi}(\omega) = \chi_{[-2\pi,-\pi)}(\omega) + \chi_{(\pi,2\pi]}(\omega).$$

Example 4.14 (Daubechies-2 wavelet system)**.** One famous example of a wavelet system is the Daubechies-2 wavelet system. This system is orthonormal.

The scaling function ϕ satisfies the refinement equation

$$\phi(t) = \sqrt{2} \sum_{k=-1}^{2} c_k \phi(2t - k)$$

where

$$c_{-1} = \tfrac{1+\sqrt{3}}{4\sqrt{2}}, \; c_0 = \tfrac{3+\sqrt{3}}{4\sqrt{2}}, \; c_1 = \tfrac{3-\sqrt{3}}{4\sqrt{2}}, \; \text{and} \; c_2 = \tfrac{1-\sqrt{3}}{4\sqrt{2}}.$$

The wavelet function ψ is given by

$$\psi(t) = \sqrt{2} \sum_{k=-1}^{2} d_k \phi(2t - k),$$

where $d_k = (-1)^{1-k} c_{1-k}$. The scaling and wavelet functions are plotted in Figures 4.15(a) and 4.15(b), respectively. (Since the system is orthonormal, $\tilde{\phi} = \phi$ and $\tilde{\psi} = \psi$.)

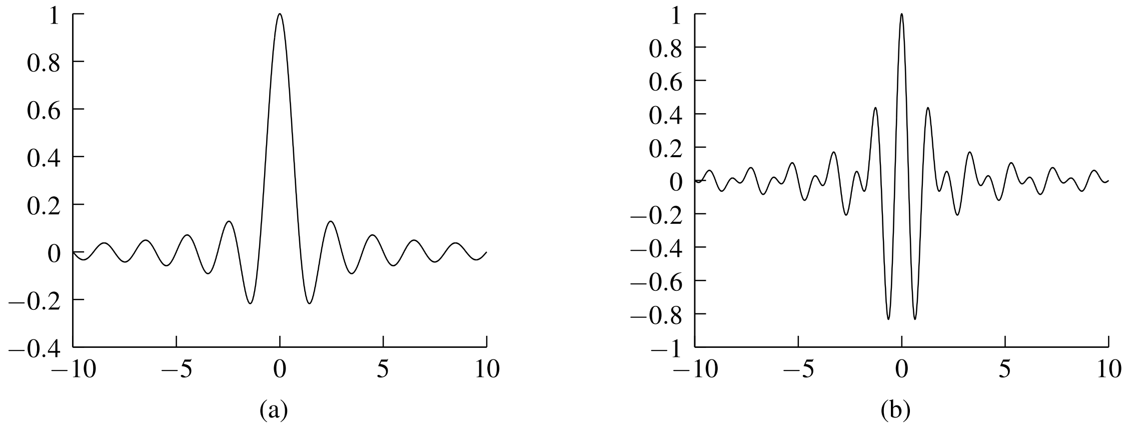


Figure 4.14: The Shannon scaling and wavelet functions. The (a) scaling function and (b) wavelet function.

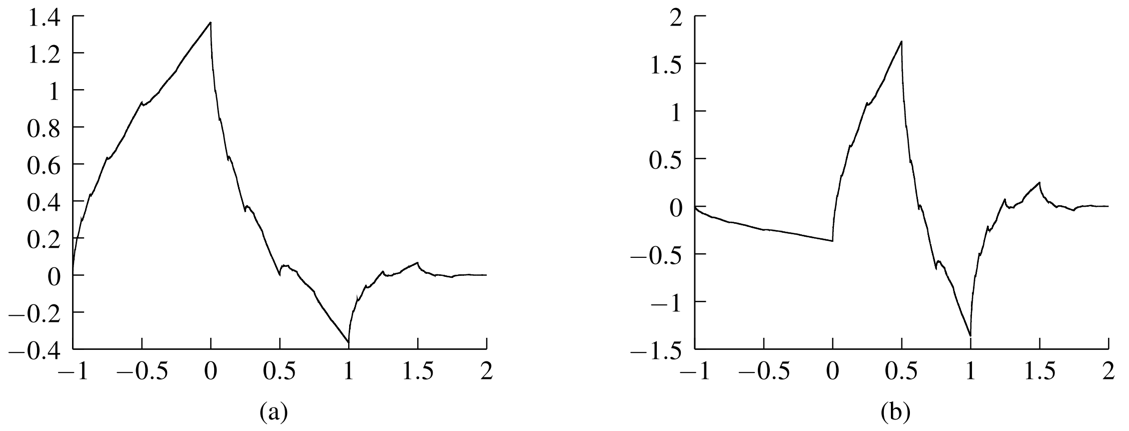


Figure 4.15: The Daubechies-2 scaling and wavelet functions. The (a) scaling function and (b) wavelet function.

Example 4.15 (Le Gall 5/3 wavelet system)**.** In this example, we introduce a biorthonormal wavelet system proposed in [12]. This particular wavelet system has proven to be extremely useful in signal coding applications (e.g., image compression).

This wavelet system is, in part, associated with the MRA introduced in Example 4.3. This is the synthesis MRA. The (primal) scaling function ϕ satisfies the refinement equation

$$\phi(t) = \sqrt{2}\sum_{k=-1}^{1} c_k \phi(2t-k)$$

where

$$c_{-1} = c_1 = \tfrac{1}{2\sqrt{2}}, \quad \text{and} \quad c_0 = \tfrac{1}{\sqrt{2}}.$$

The (primal) wavelet function ψ can be expressed in terms of the scaling function as

$$\psi(t) = \sqrt{2}\sum_{k=-1}^{3} d_k \phi(2t-k)$$

where

$$d_{-1} = d_3 = -\tfrac{1}{4\sqrt{2}}, \quad d_0 = d_2 = -\tfrac{2}{4\sqrt{2}}, \quad \text{and} \quad d_1 = \tfrac{6}{4\sqrt{2}}.$$

The above functions can be expressed in closed form as

$$\phi(t) = \begin{cases} 1-|t| & t \in [-1,1] \\ 0 & \text{otherwise} \end{cases} \quad \text{and}$$

$$\psi(t) = \begin{cases} \frac{3}{2} - 4\left|t-\frac{1}{2}\right| & t \in (0,1) \\ -\frac{3}{4} + \frac{1}{2}\left|t-\frac{1}{2}\right| & t \in (-1,0] \cup [1,2) \\ 0 & \text{otherwise.} \end{cases}$$

The functions ϕ and ψ are plotted in Figures 4.16(a) and (b), respectively.

The dual scaling function $\tilde{\phi}$ satisfies the refinement equation

$$\tilde{\phi}(t) = \sqrt{2}\sum_{k=-2}^{2} \tilde{c}_k \tilde{\phi}(2t-k)$$

where $\tilde{c}_k = (-1)^k d_{1-k}$. The dual wavelet function $\tilde{\psi}$ can be expressed as

$$\tilde{\psi}(t) = \sqrt{2}\sum_{k=0}^{2} \tilde{d}_k \tilde{\phi}(2t-k)$$

where $\tilde{d}_k = (-1)^{k+1} c_{1-k}$. Unfortunately, there is no closed form expression for $\tilde{\phi}$ and $\tilde{\psi}$. These functions are (approximately) plotted in Figures 4.16(c) and (d), respectively. In actual fact, these plots are somewhat misleading, as one can show that $\tilde{\phi}$ is infinite at every dyadic point [22, p. 248].

Example 4.16 (Cohen-Daubechies-Feauveau 9/7 wavelet system)**.** The Cohen-Daubechies-Feauveau (CDF) 9/7 wavelet system has found wide application in signal coding applications (e.g., the JPEG-2000 image compression standard [15] and FBI fingerprint compression standard [11]). This system is biorthonormal.

The (primal) scaling function ϕ satisfies the refinement equation

$$\phi(t) = \sqrt{2}\sum_{k=-3}^{3} c_k \phi(2t-k)$$

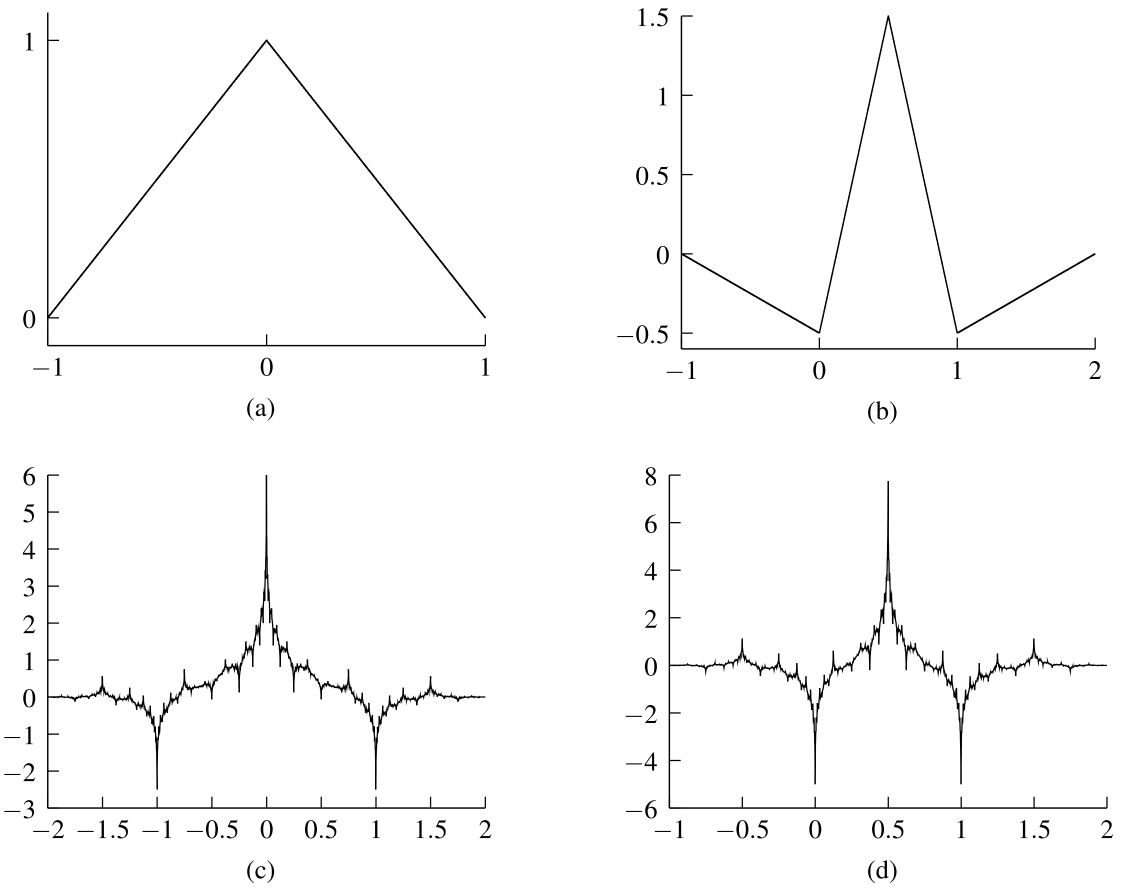


Figure 4.16: The Le Gall 5/3 scaling and wavelet functions. The (a) primal scaling function, (b) primal wavelet function, (c) dual scaling function, and (d) dual wavelet function.

where

$$c_{-3} = c_3 = \tfrac{1}{32\sqrt{2}\,x_1} \approx -0.06453888262894, \quad c_{-2} = c_2 = \tfrac{2x_1+1}{16\sqrt{2}\,x_1} \approx -0.04068941760956,$$
$$c_{-1} = c_1 = \tfrac{16x_1-1}{32\sqrt{2}\,x_1} \approx 0.41809227322221, \quad c_0 = \tfrac{6x_1-1}{8\sqrt{2}\,x_1} \approx 0.78848561640566,$$
$$x_1 = A + B - \tfrac{1}{6}, \quad A = \sqrt[3]{\tfrac{63-14\sqrt{15}}{1080\sqrt{15}}}, \quad \text{and} \quad B = -\sqrt[3]{\tfrac{63+14\sqrt{15}}{1080\sqrt{15}}}.$$

The (primal) wavelet function ψ is given by

$$\psi(t) = \sqrt{2} \sum_{k=-3}^{5} d_k \phi(2t-k)$$

where

$$d_{-3} = d_5 = -\tfrac{5}{32\sqrt{2}} x_1 \approx 0.0378284555069955,$$
$$d_{-2} = d_4 = \tfrac{5}{4\sqrt{2}} x_1 \operatorname{Re} x_2 \approx 0.0238494650193800,$$
$$d_{-1} = d_3 = -\tfrac{5}{8\sqrt{2}} x_1 (4\,|x_2|^2 + 4 \operatorname{Re} x_2 - 1) \approx -0.1106244044184234,$$
$$d_0 = d_2 = \tfrac{5}{4\sqrt{2}} x_1 (8\,|x_2|^2 - \operatorname{Re} x_2) \approx -0.3774028556126539,$$
$$d_1 = -\tfrac{5}{16\sqrt{2}} x_1 (48\,|x_2|^2 - 16 \operatorname{Re} x_2 + 3) \approx 0.8526986790094035, \quad \text{and}$$
$$x_2 = -\tfrac{1}{2}(A+B) - \tfrac{1}{6} + j\tfrac{\sqrt{3}}{2}(A-B).$$

The functions ϕ and ψ are plotted in Figures 4.17(a) and (b), respectively.

The dual scaling function $\tilde{\phi}$ satisfies the refinement equation

$$\tilde{\phi}(t) = \sqrt{2} \sum_{k=-4}^{4} \tilde{c}_k \tilde{\phi}(2t-k)$$

where $\tilde{c}_k = (-1)^k d_{1-k}$. The dual wavelet function $\tilde{\psi}$ is given by

$$\tilde{\psi}(t) = \sqrt{2} \sum_{k=-2}^{4} \tilde{d}_k \tilde{\phi}(2t-k)$$

where $\tilde{d}_k = (-1)^{k+1} c_{1-k}$. The functions $\tilde{\phi}$ and $\tilde{\psi}$ are plotted in Figures 4.17(c) and (d), respectively.

4.2.10 Relationship Between Wavelets and Filter Banks

Consider a wavelet system with MRA $\{V_p\}_{p\in\mathbb{Z}}$ and wavelet space sequence $\{W_p\}_{p\in\mathbb{Z}}$. Let $\{\phi_{p,k}\}_{k\in\mathbb{Z}}$ and $\{\psi_{p,k}\}_{k\in\mathbb{Z}}$ denote the bases of V_p and W_p, respectively. Suppose that we have a function $f \in V_p$. Since $f \in V_p$, f has an expansion in terms of the basis of V_p given by

$$f(t) = \sum_{n\in\mathbb{Z}} a_p[n] \phi_{p,n}(t). \tag{4.14}$$

Furthermore, as $V_p = V_{p+1} \oplus W_{p+1}$, we can also expand f in terms of the bases of V_{p+1} and W_{p+1} to obtain

$$f(t) = \sum_{n\in\mathbb{Z}} a_{p+1}[n] \phi_{p+1,n}(t) + \sum_{n\in\mathbb{Z}} b_{p+1}[n] \psi_{p+1,n}(t) \tag{4.15}$$

(i.e., f is the sum of a function in V_{p+1} and a function in W_{p+1}). Thus, we have two different representations of f. One might wonder if there exists a simple technique for computing (4.14) from (4.15) and vice versa. In other words, given $a_p[n]$, we would like to be able to determine $a_{p+1}[n]$ and $b_{p+1}[n]$; or given $a_{p+1}[n]$ and $b_{p+1}[n]$, we would like to be able to determine $a_p[n]$. Fortunately, there is a very elegant technique for accomplishing exactly this. This technique is known as the Mallat algorithm and is given by the theorem below.

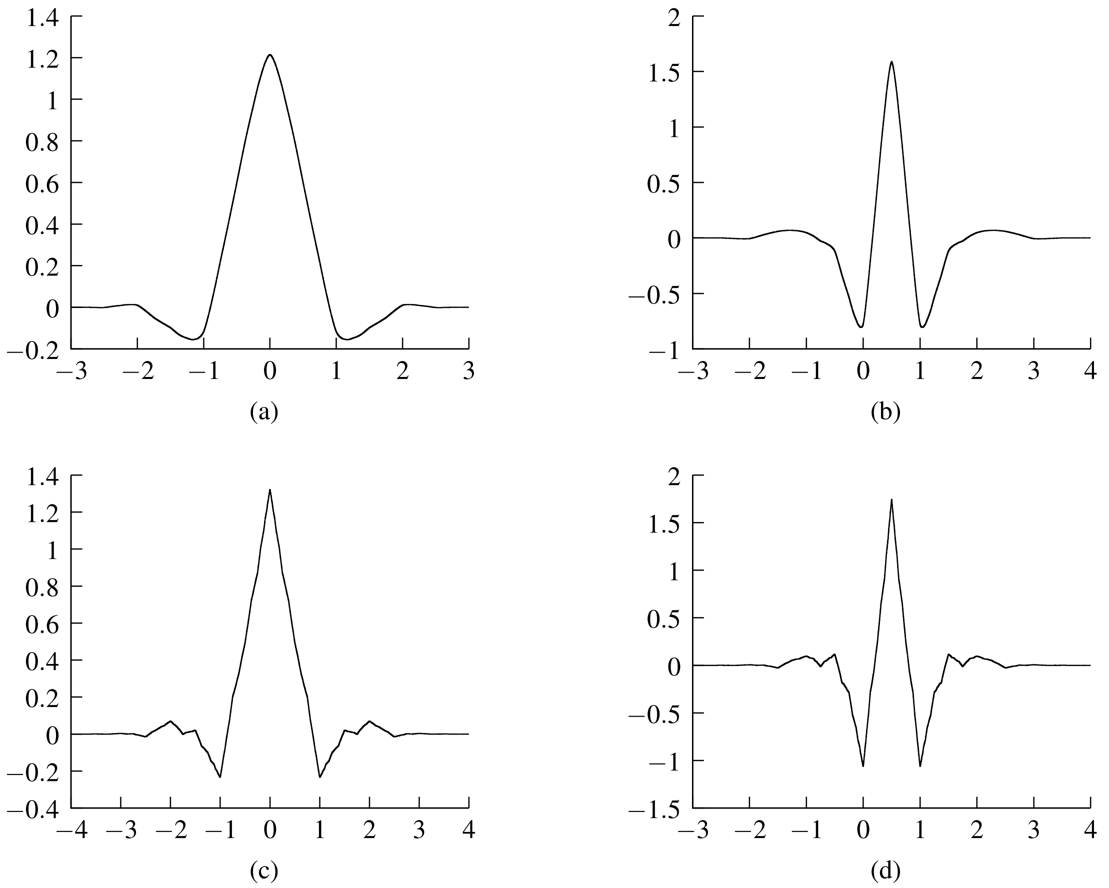


Figure 4.17: The CDF 9/7 scaling and wavelet functions. The (a) primal scaling function, (b) primal wavelet function, (c) dual scaling function, and (d) dual wavelet function.

Theorem 4.15 (Mallat algorithm)**.** *Consider a wavelet system with MRA $\{V_p\}_{p\in\mathbb{Z}}$, scaling function ϕ, wavelet space sequence $\{W_p\}_{p\in\mathbb{Z}}$, wavelet function ψ, dual scaling function $\tilde{\phi}$, and dual wavelet function $\tilde{\psi}$. Let the scaling equation coefficient sequences of ϕ and $\tilde{\phi}$ be denoted as c and $\tilde{c}$, respectively. Let the wavelet equation coefficient sequences of ψ and $\tilde{\psi}$ be denoted as d and $\tilde{d}$, respectively. Define the basis functions for the various spaces as before*

$$\begin{aligned}
\phi_{p,n}(t) &= 2^{-p/2}\phi(2^{-p}t-n), \\
\psi_{p,n}(t) &= 2^{-p/2}\psi(2^{-p}t-n), \\
\tilde{\phi}_{p,n}(t) &= 2^{-p/2}\tilde{\phi}(2^{-p}t-n), \quad \textit{and} \\
\tilde{\psi}_{p,n}(t) &= 2^{-p/2}\tilde{\psi}(2^{-p}t-n).
\end{aligned}$$

Any $f \in V_p$ can be represented in each of the following forms:

$$f = \sum_{n\in\mathbb{Z}} a_p[n]\phi_{p,n} \quad \textit{and} \tag{4.16}$$

$$f = \sum_{n\in\mathbb{Z}} a_{p+1}[n]\phi_{p+1,n} + \sum_{n\in\mathbb{Z}} b_{p+1}[n]\psi_{p+1,n}. \tag{4.17}$$

Given $a_p[n]$, we can compute the corresponding $a_{p+1}[n]$ and $b_{p+1}[n]$ as follows:

$$a_{p+1}[n] = (\downarrow 2)\,(a_p * h_0[n]) \quad \textit{and} \tag{4.18a}$$

$$b_{p+1}[n] = (\downarrow 2)\,(a_p * h_1[n]), \tag{4.18b}$$

where $h_0[n] = \tilde{c}^*[-n]$ and $h_1[n] = \tilde{d}^*[-n]$. Given $a_{p+1}[n]$ and $b_{p+1}[n]$, we can compute the corresponding $a_p[n]$ as follows:

$$a_p[n] = ((\uparrow 2)a_{p+1}[n]) * g_0[n] + ((\uparrow 2)b_{p+1}[n]) * g_1[n], \tag{4.19}$$

where $g_0[n] = c[n]$ and $g_1[n] = d[n]$.

Proof. ANALYSIS. To begin, let us consider the determination of $a_{p+1}[n]$ and $b_{p+1}[n]$ from $a_p[n]$.

First, we consider the calculation of $a_{p+1}[n]$. We have

$$\begin{aligned}
a_{p+1}[n] &= \left\langle f, \tilde{\phi}_{p+1,n} \right\rangle \\
&= \left\langle \sum_{k\in\mathbb{Z}} a_p[k]\phi_{p,k}, \tilde{\phi}_{p+1,n} \right\rangle \\
&= \sum_{k\in\mathbb{Z}} a_p[k] \left\langle \phi_{p,k}, \tilde{\phi}_{p+1,n} \right\rangle.
\end{aligned} \tag{4.20}$$

Now, we focus our attention on expressing $\tilde{\phi}_{p+1,n}$ in a form that will allow us to further simplify the above equation. We use the scaling equation for $\tilde{\phi}$ to write

$$\begin{aligned}
\tilde{\phi}_{p+1,n}(t) &= 2^{-(p+1)/2}\tilde{\phi}(2^{-(p+1)}t-n) \\
&= 2^{-(p+1)/2}\left(\sqrt{2}\sum_{l\in\mathbb{Z}} \tilde{c}[l]\tilde{\phi}(2[2^{-(p+1)}t-n]-l)\right) \\
&= 2^{-p/2}\sum_{l\in\mathbb{Z}} \tilde{c}[l]\tilde{\phi}(2^{-p}t-2n-l) \\
&= \sum_{l\in\mathbb{Z}} \tilde{c}[l]2^{-p/2}\tilde{\phi}(2^{-p}t-[2n+l]) \\
&= \sum_{l\in\mathbb{Z}} \tilde{c}[l]\tilde{\phi}_{p,2n+l}(t).
\end{aligned}$$

Substituting this expression for $\tilde{\phi}_{p+1,n}$ in (4.20), we obtain

$$\begin{aligned}
a_{p+1}[n] &= \sum_{k\in\mathbb{Z}} a_p[k] \left\langle \phi_{p,k}, \sum_{l\in\mathbb{Z}} \tilde{c}[l]\tilde{\phi}_{p,2n+l} \right\rangle \\
&= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} a_p[k]\tilde{c}^*[l] \left\langle \phi_{p,k}, \tilde{\phi}_{p,2n+l} \right\rangle \\
&= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} a_p[k]\tilde{c}^*[l]\delta[2n+l-k] \\
&= \sum_{k\in\mathbb{Z}} a_p[k]\tilde{c}^*[k-2n] \\
&= \sum_{k\in\mathbb{Z}} a_p[k]h_0[2n-k] \\
&= (\downarrow 2)\left(a_p * h_0[n]\right).
\end{aligned}$$

Thus, we have shown that (4.18a) is correct.

Next, we consider the computation of $b_{p+1}[n]$. We have

$$\begin{aligned}
b_{p+1}[n] &= \left\langle f, \tilde{\psi}_{p+1,n} \right\rangle \\
&= \left\langle \sum_{k\in\mathbb{Z}} a_p[k]\phi_{p,k}, \tilde{\psi}_{p+1,n} \right\rangle \\
&= \sum_{k\in\mathbb{Z}} a_p[k] \left\langle \phi_{p,k}, \tilde{\psi}_{p+1,n} \right\rangle.
\end{aligned} \tag{4.21}$$

Now, we focus our attention on finding an equivalent expression for $\tilde{\psi}_{p+1,n}$ that will allow us to simplify the above equation further. Using the wavelet equation for $\tilde{\psi}$, we can write

$$\begin{aligned}
\tilde{\psi}_{p+1,n}(t) &= 2^{-(p+1)/2}\tilde{\psi}(2^{-(p+1)}t-n) \\
&= 2^{-(p+1)/2}\left(\sqrt{2}\sum_{l\in\mathbb{Z}} \tilde{d}[l]\tilde{\phi}(2[2^{-(p+1)}t-n]-l)\right) \\
&= 2^{-p/2}\sum_{l\in\mathbb{Z}} \tilde{d}[l]\tilde{\phi}(2^{-p}t-2n-l) \\
&= \sum_{l\in\mathbb{Z}} \tilde{d}[l]2^{-p/2}\tilde{\phi}(2^{-p}t-[2n+l]) \\
&= \sum_{l\in\mathbb{Z}} \tilde{d}[l]\tilde{\phi}_{p,2n+l}(t).
\end{aligned}$$

Substituting the above expression for $\tilde{\psi}_{p+1,n}$ in (4.21), we obtain

$$\begin{aligned}
b_{p+1}[n] &= \sum_{k\in\mathbb{Z}} a_p[k] \left\langle \phi_{p,k}, \sum_{l\in\mathbb{Z}} \tilde{d}[l]\tilde{\phi}_{p,2n+l} \right\rangle \\
&= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} a_p[k]\tilde{d}^*[l] \left\langle \phi_{p,k}, \tilde{\phi}_{p,2n+l} \right\rangle \\
&= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} a_p[k]\tilde{d}^*[l]\delta[2n+l-k] \\
&= \sum_{k\in\mathbb{Z}} a_p[k]\tilde{d}^*[k-2n] \\
&= \sum_{k\in\mathbb{Z}} a_p[k]h_1[2n-k] \\
&= (\downarrow 2)\left(a_p * h_1[n]\right).
\end{aligned}$$

Thus, we have shown that (4.18b) is correct.

SYNTHESIS. Let us consider the computation of $a_p[n]$ from $a_{p+1}[n]$ and $b_{p+1}[n]$. We have

$$a_p[n] = \left\langle \sum_{k \in \mathbb{Z}} a_{p+1}[k] \phi_{p+1,k} + \sum_{k \in \mathbb{Z}} b_{p+1}[k] \psi_{p+1,k}, \tilde{\phi}_{p,n} \right\rangle. \tag{4.22}$$

Now, we find equivalent expressions for $\phi_{p+1,k}$ and $\psi_{p+1,k}$ that will allow us to simplify the above equation further. Using the scaling equation for ϕ, we can write

$$\begin{aligned}
\phi_{p+1,k}(t) &= 2^{-(p+1)/2} \phi(2^{-(p+1)} t - k) \\
&= 2^{-(p+1)/2} \left(\sqrt{2} \sum_{l \in \mathbb{Z}} c[l] \phi(2[2^{-(p+1)} t - k] - l) \right) \\
&= 2^{-p/2} \sum_{l \in \mathbb{Z}} c[l] \phi(2^{-p} t - 2k - l) \\
&= \sum_{l \in \mathbb{Z}} c[l] 2^{-p/2} \phi(2^{-p} t - [2k + l]) \\
&= \sum_{l \in \mathbb{Z}} c[l] \phi_{p,2k+l}.
\end{aligned}$$

Using the wavelet equation for ψ, we can write

$$\begin{aligned}
\psi_{p+1,k}(t) &= 2^{-(p+1)/2} \psi(2^{-(p+1)} t - k) \\
&= 2^{-(p+1)/2} \left(\sqrt{2} \sum_{l \in \mathbb{Z}} d[l] \phi(2[2^{-(p+1)} t - k] - l) \right) \\
&= 2^{-p/2} \sum_{l \in \mathbb{Z}} d[l] \phi(2^{-p} t - 2k - l) \\
&= \sum_{l \in \mathbb{Z}} d[l] 2^{-p/2} \phi(2^{-p} t - [2k + l]) \\
&= \sum_{l \in \mathbb{Z}} d[l] \phi_{p,2k+l}.
\end{aligned}$$

Substituting the expressions obtained above for $\phi_{p+1,k}$ and $\psi_{p+1,k}$ into (4.22) and simplifying, we obtain

$$\begin{aligned}
a_p[n] &= \left\langle \sum_{k \in \mathbb{Z}} a_{p+1}[k] \sum_{l \in \mathbb{Z}} c[l] \phi_{p,2k+l} + \sum_{k \in \mathbb{Z}} b_{p+1}[k] \sum_{l \in \mathbb{Z}} d[l] \phi_{p,2k+l}, \tilde{\phi}_{p,n} \right\rangle \\
&= \left\langle \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} \left(a_{p+1}[k] c[l] + b_{p+1}[k] d[l] \right) \phi_{p,2k+l}, \tilde{\phi}_{p,n} \right\rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} \left(a_{p+1}[k] c[l] + b_{p+1}[k] d[l] \right) \left\langle \phi_{p,2k+l}, \tilde{\phi}_{p,n} \right\rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} \left(a_{p+1}[k] c[l] + b_{p+1}[k] d[l] \right) \delta[2k + l - n] \\
&= \sum_{k \in \mathbb{Z}} \left(a_{p+1}[k] c[n - 2k] + b_{p+1}[k] d[n - 2k] \right) \\
&= \sum_{k \in \mathbb{Z}} a_{p+1}[k] c[n - 2k] + \sum_{k \in \mathbb{Z}} b_{p+1}[k] d[n - 2k] \\
&= \sum_{k \in \mathbb{Z}} a_{p+1}[k] g_0[n - 2k] + \sum_{k \in \mathbb{Z}} b_{p+1}[k] g_1[n - 2k] \\
&= ((\uparrow 2) a_{p+1}[n]) * g_0[n] + ((\uparrow 2) b_{p+1}[n]) * g_1[n].
\end{aligned}$$

Thus, we have shown that (4.19) holds. □

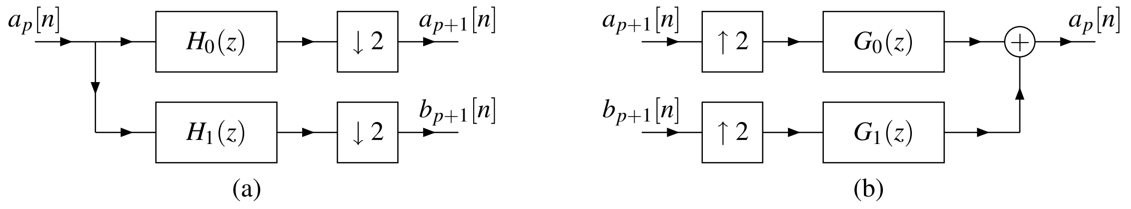


Figure 4.18: Computational structure associated with the Mallat algorithm.

Upon more careful inspection of (4.18), we see that this formula can be computed with the structure shown in Figure 4.18(a). Likewise, (4.19) can be computed with the structure shown in Figure 4.18(b). In other words, the Mallat algorithm is associated with a two-channel UMD filter bank. We use the analysis side of the filter bank to move from a representation of the form of (4.16) to (4.17), and the synthesis side to move from a representation of the form of (4.17) to (4.16). The Mallat algorithm is of great importance as it establishes a link between wavelet systems and filter banks.

4.2.11 Filter Banks and Wavelet Systems: Samples Versus Expansion Coefficients

A wavelet series is a representation scheme for continuous-time signals. As we have seen, however, a relationship exists between wavelet series and the discrete-time computations performed by a filter bank. In particular, this relationship between wavelet series and filter banks is given by the Mallat algorithm. As was shown earlier, the Mallat algorithm allows us to move between wavelet series representations that utilize basis functions from different combinations of approximation and wavelet spaces. This is accomplished with the filter bank structure from Figure 4.18.

When viewed from a wavelet perspective, the sequences processed by filter banks correspond to expansion coefficients of a wavelet series, not samples of a signal. This is an important distinction to make. Since sample values and expansion coefficients are fundamentally different quantities, this raises the issue of how to convert between them.

For example, suppose that we want to process a (continuous-time) signal x using a wavelet series representation. We need a way to determine an initial representation of x in terms of the basis $\{\phi_{p,n}\}_{n\in\mathbb{Z}}$ for some suitable approximation space V_p. That is, given $x \in V_p$, we need to find an expansion for x of the form

$$x = \sum_{n\in\mathbb{Z}} a_p[n]\phi_{p,n},$$

where the coefficient sequence a_p is to be determined. To determine a_p, we compute inner products. In particular, we have

$$a_p[n] = \langle x, \tilde{\phi}_{p,n} \rangle,$$

where $\{\tilde{\phi}_{p,n}\}_{n\in\mathbb{Z}}$ is the dual basis of $\{\phi_{p,n}\}_{n\in\mathbb{Z}}$. Essentially, we are projecting x onto the space V_p, where the projection operator being employed may or may not be orthogonal.

Some practical issues arise in the computation of the above inner products. Since we are performing processing in discrete time, we usually do not explicitly know x (i.e., the original continuous-time signal before sampling). We only have samples of x. Of course, we could reconstruct x from its samples under the assumption that x is bandlimited and was sampled at a sufficiently high rate to avoid aliasing (i.e., we can use sinc interpolation). The regeneration of x from its samples, however, is a rather painful process, with the above inner-product computation also being quite tedious.

In addition to the problem of how to choose an initial expansion-coefficient sequence, we face one further problem. After processing is completed, we probably want the final output of our discrete-time system to correspond to sample values, not expansion coefficients. Essentially, as a final processing step, we need to perform a conversion from expansion coefficients to their corresponding sample values.

Since converting between expansion coefficients and sample values is a tedious process and can often lead to many other practical problems, we would prefer to avoid such conversions if possible. This leads to the frequent use of a very mathematically inelegant but often highly effective solution. Namely, in many practical applications, we instead choose a_p as a sampled version of x (i.e., $a_p[n] = x(nT)$ for some choice of sampling period T). In other words, we use samples of x as expansion coefficients and vice versa. It is important to understand, however, that strictly speaking this is wrong, since it is not necessarily true that $a_p[n] = x(nT)$. In theory, we should apply a prefiltering operation to convert sample values into expansion coefficients (before processing is started) and a postfiltering operation to convert expansion coefficients into sample values (after processing is complete).

Although the sample values and expansion coefficients generally tend not to be equivalent, in some circumstances, they can be. We explore this situation in what follows. Before proceeding further, however, we need to introduce the notion of an interpolating function.

Definition 4.3 (Interpolating function)**.** A function f defined on $\mathbb{R}$ is said to be **interpolating** if $f(n) = \delta[n]$ for all $n \in \mathbb{Z}$.

Some examples of interpolating functions include $\chi_{[0,1)}$ (i.e., the Haar scaling function) and $\operatorname{sinc}(\pi\cdot)$ (i.e., the Shannon scaling function).

Now, consider a wavelet series expansion associated with the (primal) scaling function ϕ. If ϕ is interpolating, we can show that $a_p[n]$ equals $x(nT)$ to within a scale factor. Suppose that x is represented in terms of the basis for V_p as

$$\begin{aligned} x(t) &= \sum_{k\in\mathbb{Z}} a_p[k]\phi_{p,k}(t) \\ &= \sum_{k\in\mathbb{Z}} a_p[k]2^{-p/2}\phi(2^{-p}t-k). \end{aligned}$$

If we evaluate $x(t)$ at points of the form $t = 2^p n$ where $n \in \mathbb{Z}$ (i.e., periodic sampling with period 2^p), we obtain

$$\begin{aligned} x(2^p n) &= \sum_{k\in\mathbb{Z}} a_p[k]2^{-p/2}\phi(2^{-p}[2^p n]-k) \\ &= \sum_{k\in\mathbb{Z}} a_p[k]2^{-p/2}\phi(n-k). \end{aligned}$$

We now employ a change of variable. Let $k' = n-k$ so $k = n-k'$. Applying the change of variable and dropping the primes, we obtain

$$\begin{aligned} x(2^p n) &= \sum_{k\in\mathbb{Z}} a_p[n-k]2^{-p/2}\phi(k) \\ &= 2^{-p/2}a_p[n]. \end{aligned}$$

(We used the fact that ϕ is interpolating in the last simplification step.) Thus, if ϕ is interpolating, a_p is given (up to a scale factor) by the samples of x sampled with period 2^p, with the scale factor being one when $p = 0$. Unfortunately, most ϕ are not interpolating. So, this equivalence between sample values and expansion coefficients does not usually hold.

Whether it is advisable to use sample values as expansion coefficients and vice versa is open to debate. Mathematical purists would probably cringe at the idea of doing so. In many applications, however, this approach (although technically incorrect) does not cause any serious problems and has the important benefit of greatly simplifying things. This said, however, it is important to understand that expansion coefficients and sample values are not necessarily the same. Therefore, in some situations, treating these two different entities as the same may cause problems. For this reason, one needs to exercise good judgement in this regard.

Although the above sample-value versus expansion-coefficient issue is frequently ignored by authors, this is not always so. The interested reader can find this matter discussed, for example, in [22, pp. 232–233].

4.2.12 Characterizing Biorthonormal and Orthonormal Wavelet Systems

Often, we are interested in biorthonormal and orthonormal wavelet systems. For this reason, it is useful to characterize biorthonormal and orthonormal properties in terms of scaling and wavelet equation coefficient sequences.

Let $\{\phi_{p,k}\}_{p\in\mathbb{Z}}$, $\{\tilde{\phi}_{p,k}\}_{p\in\mathbb{Z}}$, $\{\psi_{p,k}\}_{p\in\mathbb{Z}}$, and $\{\tilde{\psi}_{p,k}\}_{p\in\mathbb{Z}}$ denote the Riesz bases of V_p, $\tilde{V}_p$, W_p, $\tilde{W}_p$, respectively, where $\phi_{p,k}$, $\tilde{\phi}_{p,k}$, $\psi_{p,k}$, $\tilde{\psi}_{p,k}$ are as defined in earlier sections.

A biorthonormal system is such that

$$\langle \phi_{p,k}, \tilde{\phi}_{p,l} \rangle = \delta[k-l], \tag{4.23a}$$

$$\langle \psi_{p,k}, \tilde{\psi}_{p,l} \rangle = \delta[k-l], \tag{4.23b}$$

$$\langle \phi_{p,k}, \tilde{\psi}_{p,l} \rangle = 0, \quad \text{and} \tag{4.23c}$$

$$\langle \psi_{p,k}, \tilde{\phi}_{p,l} \rangle = 0. \tag{4.23d}$$

An orthonormal system is such that

$$\begin{gathered} \langle \phi_{p,k}, \phi_{p,l} \rangle = \delta[k-l], \\ \langle \psi_{p,k}, \psi_{p,l} \rangle = \delta[k-l], \quad \text{and} \\ \langle \phi_{p,k}, \psi_{p,l} \rangle = 0. \end{gathered}$$

In what follows, we will encounter several integrals of the form

$$2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q \in \mathbb{Z}} \lambda_1[q] \phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r \in \mathbb{Z}} \lambda_2^*[r] \tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt.$$

So, before we continue, we will derive a simplified expression for the above integral. We have

$$\begin{aligned} &2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q \in \mathbb{Z}} \lambda_1[q] \phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r \in \mathbb{Z}} \lambda_2^*[r] \tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\ =&2^{-p+1} \int_{-\infty}^{\infty} \left[\sum_{q \in \mathbb{Z}} \lambda_1[q] \phi(2^{-p+1}t-2k-q) \right] \left[\sum_{r \in \mathbb{Z}} \lambda_2^*[r] \tilde{\phi}^*(2^{-p+1}t-2l-r) \right] dt \\ =&2^{-p+1} \sum_{q \in \mathbb{Z}} \sum_{r \in \mathbb{Z}} \lambda_1[q] \lambda_2^*[r] \int_{-\infty}^{\infty} \phi(2^{-p+1}t-2k-q) \tilde{\phi}^*(2^{-p+1}t-2l-r) dt. \end{aligned} \tag{4.24}$$

Now, we momentarily shift our focus to the computation of the integral appearing on the last line above. We employ a change of variable. Let $t' = 2^{-p+1}t - 2k - q$ so that $t = 2^{p-1}t' + 2^p k + 2^{p-1}q$ and $dt = 2^{p-1}dt'$. Applying the change of variable and dropping the primes, we have

$$\begin{aligned} \int_{-\infty}^{\infty} \phi(2^{-p+1}t-2k-q) \tilde{\phi}^*(2^{-p+1}t-2l-r) dt &= \int_{-\infty}^{\infty} \phi(t) \tilde{\phi}^*(2^{-p+1}[2^{p-1}t+2^p k+2^{p-1}q]-2l-r) 2^{p-1} dt \\ &= 2^{p-1} \int_{-\infty}^{\infty} \phi(t) \tilde{\phi}^*(t+2k+q-2l-r) dt \\ &= 2^{p-1} \langle \phi(\cdot), \tilde{\phi}(\cdot - [2l-2k+r-q]) \rangle \\ &= 2^{p-1} \delta[2l-2k+r-q]. \end{aligned}$$

Substituting this result into (4.24), we have

$$\begin{aligned} &2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q \in \mathbb{Z}} \lambda_1[q] \phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r \in \mathbb{Z}} \lambda_2^*[r] \tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\ =&2^{-p+1} \sum_{q \in \mathbb{Z}} \sum_{r \in \mathbb{Z}} \lambda_1[q] \lambda_2^*[r] 2^{p-1} \delta[2l-2k+r-q] \\ =& \sum_{q \in \mathbb{Z}} \sum_{r \in \mathbb{Z}} \lambda_1[q] \lambda_2^*[r] \delta[2l-2k+r-q] \\ =& \sum_{q \in \mathbb{Z}} \lambda_1[q] \lambda_2^*[q+2k-2l] \\ =& \langle \lambda_1[\cdot], \lambda_2[\cdot - 2(l-k)] \rangle. \end{aligned} \tag{4.25}$$

Consider the inner product $\langle \phi_{p,k}, \tilde{\phi}_{p,l} \rangle$ in (4.23a). Using (4.25), we can write

$$\begin{aligned}
\langle \phi_{p,k}, \tilde{\phi}_{p,l} \rangle &= \int_{-\infty}^{\infty} 2^{-p/2} \phi(2^{-p}t-k) 2^{-p/2} \tilde{\phi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \phi(2^{-p}t-k)\tilde{\phi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q\in\mathbb{Z}} c[q]\phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r\in\mathbb{Z}} \tilde{c}^*[r]\tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\
&= \langle c[\cdot], \tilde{c}[\cdot - 2(l-k)] \rangle .
\end{aligned}$$

Using this result and (4.23a), we can deduce

$$\langle c[\cdot], \tilde{c}[\cdot - 2n] \rangle = \delta[n]. \tag{4.26}$$

Consider the inner product $\langle \phi_{p,k}, \tilde{\psi}_{p,l} \rangle$ in (4.23c). Using (4.25), we can write

$$\begin{aligned}
\langle \phi_{p,k}, \tilde{\psi}_{p,l} \rangle &= \int_{-\infty}^{\infty} 2^{-p/2} \phi(2^{-p}t-k) 2^{-p/2} \tilde{\psi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \phi(2^{-p}t-k)\tilde{\psi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q\in\mathbb{Z}} c[q]\phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r\in\mathbb{Z}} \tilde{d}^*[r]\tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\
&= \langle c[\cdot], \tilde{d}[\cdot - 2(l-k)] \rangle .
\end{aligned}$$

Using this result and (4.23c), we can deduce

$$\langle c[\cdot], \tilde{d}[\cdot - 2n] \rangle = 0. \tag{4.27}$$

Consider the inner product $\langle \psi_{p,k}, \tilde{\psi}_{p,l} \rangle$ in (4.23b). Using (4.25), we can write

$$\begin{aligned}
\langle \psi_{p,k}, \tilde{\psi}_{p,l} \rangle &= \int_{-\infty}^{\infty} 2^{-p/2} \psi(2^{-p}t-k) 2^{-p/2} \tilde{\psi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \psi(2^{-p}t-k)\tilde{\psi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q\in\mathbb{Z}} d[q]\phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r\in\mathbb{Z}} \tilde{d}^*[r]\tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\
&= \langle d[\cdot], \tilde{d}[\cdot - 2(l-k)] \rangle .
\end{aligned}$$

Using this result and (4.23b), we can conclude

$$\langle d[\cdot], \tilde{d}[\cdot - 2n] \rangle = \delta[n]. \tag{4.28}$$

Consider the inner product $\langle \psi_{p,k}, \tilde{\phi}_{p,l} \rangle$ in (4.23d). Using (4.25), we can write

$$\begin{aligned}
\langle \psi_{p,k}, \tilde{\phi}_{p,l} \rangle &= \int_{-\infty}^{\infty} 2^{-p/2} \psi(2^{-p}t-k) 2^{-p/2} \tilde{\phi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \phi(2^{-p}t-k)\tilde{\psi}^*(2^{-p}t-l)dt \\
&= 2^{-p} \int_{-\infty}^{\infty} \left[2^{1/2} \sum_{q\in\mathbb{Z}} d[q]\phi(2[2^{-p}t-k]-q) \right] \left[2^{1/2} \sum_{r\in\mathbb{Z}} \tilde{c}^*[r]\tilde{\phi}^*(2[2^{-p}t-l]-r) \right] dt \\
&= \langle d[\cdot], \tilde{c}[\cdot - 2(l-k)] \rangle .
\end{aligned}$$

Using this result and (4.23d), we can conclude

$$\langle d[\cdot], \tilde{c}[\cdot - 2n]\rangle = 0. \tag{4.29}$$

From Section 4.2.10, we know that a wavelet system is associated with a two-channel UMD filter bank. We can relate c, $\tilde{c}$, d, and $\tilde{d}$ to the analysis and synthesis filter impulse responses h_0, h_1, g_0, and g_1. Rewriting (4.26), (4.27), (4.28), and (4.29) in terms of h_0, h_1, g_0, and g_1, we obtain

$$\begin{gathered}\langle g_0[\cdot], h_0^*[2n-\cdot]\rangle = \delta[n],\\ \langle g_0[\cdot], h_1^*[2n-\cdot]\rangle = 0,\\ \langle g_1[\cdot], h_1^*[2n-\cdot]\rangle = \delta[n], \quad \text{and}\\ \langle g_1[\cdot], h_0^*[2n-\cdot]\rangle = 0,\end{gathered}$$

or equivalently

$$\langle g_k[\cdot], h_l^*[2n-\cdot]\rangle = \delta[k-l]\delta[n]. \tag{4.30}$$

The astute reader will recognize (4.30) as the time-domain condition for a two-channel UMD filter bank with the shift-free PR property. (See Theorem 3.14 on page 145 or (3.32).)

If the wavelet system is orthonormal, we have $\tilde{c} = c$ and $\tilde{d} = d$. In this case, (4.26), (4.27), (4.28), and (4.29) simplify to

$$\langle g_k[\cdot], g_l[\cdot - 2n]\rangle = \delta[k-l]\delta[n], \quad h_k[n] = g_k^*[-n].$$

Again, the astute reader will recognize this as the condition for an orthonormal filter bank. (See Theorem 3.15 on page 145.)

Now, we consider some of the above conditions expressed in the Fourier domain. First, we need the result given below.

Lemma 4.2. *For any two sequences f and g, we have*

$$\langle f[\cdot], g[\cdot - 2n]\rangle = \alpha\delta[n] \;\Leftrightarrow\; \hat{f}(\omega)\hat{g}^*(\omega) + \hat{f}(\omega+\pi)\hat{g}^*(\omega+\pi) = 2\alpha, \quad \textit{where } \alpha \in \mathbb{C}.$$

Proof. Define $v[n] = g^*[-n]$. Using this definition, we can write

$$\begin{aligned} & \langle f[\cdot], g[\cdot - 2n]\rangle = \alpha\delta[n]\\ \Leftrightarrow\quad & \sum_{k\in\mathbb{Z}} f[k]g^*[k-2n] = \alpha\delta[n]\\ \Leftrightarrow\quad & \sum_{k\in\mathbb{Z}} f[k]v[2n-k] = \alpha\delta[n]\\ \Leftrightarrow\quad & (\downarrow 2)(f * v[n]) = \alpha\delta[n].\end{aligned}$$

Taking the Fourier transform of both sides of the preceding equation and using the fact that $\hat{v}(\omega) = \hat{g}^*(\omega)$, we obtain

$$\begin{aligned} \Leftrightarrow\quad & (\downarrow 2)(\hat{f}(\omega)\hat{v}(\omega)) = \alpha\\ \Leftrightarrow\quad & \tfrac{1}{2}\hat{f}(\omega)\hat{v}(\omega) + \tfrac{1}{2}\hat{f}(\omega+\pi)\hat{v}(\omega+\pi) = \alpha\\ \Leftrightarrow\quad & \hat{f}(\omega)\hat{v}(\omega) + \hat{f}(\omega+\pi)\hat{v}(\omega+\pi) = 2\alpha\\ \Leftrightarrow\quad & \hat{f}(\omega)\hat{g}^*(\omega) + \hat{f}(\omega+\pi)\hat{g}^*(\omega+\pi) = 2\alpha.\end{aligned}$$

□

Using the above lemma, we can equivalently express (4.26), (4.27), (4.28), and (4.29) as

$$\hat{c}(\omega)\hat{\tilde{c}}^*(\omega)+\hat{c}(\omega+\pi)\hat{\tilde{c}}^*(\omega+\pi)=2, \tag{4.31a}$$

$$\hat{c}(\omega)\hat{\tilde{d}}^*(\omega)+\hat{c}(\omega+\pi)\hat{\tilde{d}}^*(\omega+\pi)=0, \tag{4.31b}$$

$$\hat{d}(\omega)\hat{\tilde{d}}^*(\omega)+\hat{d}(\omega+\pi)\hat{\tilde{d}}^*(\omega+\pi)=2, \quad \text{and} \tag{4.31c}$$

$$\hat{d}(\omega)\hat{\tilde{c}}^*(\omega)+\hat{d}(\omega+\pi)\hat{\tilde{c}}^*(\omega+\pi)=0. \tag{4.31d}$$

Rewriting these equations in terms of analysis and synthesis filter impulse responses, we have

$$\begin{aligned}
\hat{g}_0(\omega)\hat{h}_0(\omega)+\hat{g}_0(\omega+\pi)\hat{h}_0(\omega+\pi)&=2,\\
\hat{g}_0(\omega)\hat{h}_1(\omega)+\hat{g}_0(\omega+\pi)\hat{h}_1(\omega+\pi)&=0,\\
\hat{g}_1(\omega)\hat{h}_1(\omega)+\hat{g}_1(\omega+\pi)\hat{h}_1(\omega+\pi)&=2, \quad \text{and}\\
\hat{g}_1(\omega)\hat{h}_0(\omega)+\hat{g}_1(\omega+\pi)\hat{h}_0(\omega+\pi)&=0.
\end{aligned}$$

These equations, however, are nothing more than a restatement of the biorthonormality (i.e., shift-free PR) condition for a UMD filter bank, obtained by evaluating $\boldsymbol{H}_{\mathsf{m}}(z)\boldsymbol{G}_{\mathsf{m}}(z)=2\boldsymbol{I}$ for $z=e^{j\omega}$.

If the wavelet system is orthonormal, $\tilde{c}=c$ and $\tilde{d}=d$, and (4.31) simplifies to

$$|\hat{c}(\omega)|^2+|\hat{c}(\omega+\pi)|^2=2, \tag{4.33a}$$

$$\hat{c}(\omega)\hat{d}^*(\omega)+\hat{c}(\omega+\pi)\hat{d}^*(\omega+\pi)=0, \quad \text{and} \tag{4.33b}$$

$$\left|\hat{d}(\omega)\right|^2+\left|\hat{d}(\omega+\pi)\right|^2=2. \tag{4.33c}$$

Rewriting (4.33) in terms of analysis and synthesis filter impulse responses, we have

$$|\hat{g}_0(\omega)|^2+|\hat{g}_0(\omega+\pi)|^2=2, \tag{4.34a}$$

$$\hat{g}_0(\omega)\hat{g}_1^*(\omega)+\hat{g}_0(\omega+\pi)\hat{g}_1^*(\omega+\pi)=0, \quad \text{and} \tag{4.34b}$$

$$|\hat{g}_1(\omega)|^2+|\hat{g}_1(\omega+\pi)|^2=2. \tag{4.34c}$$

These equations, however, are simply a restatement of the (shift-free PR) orthonormality condition for a UMD filter bank, obtained by evaluating $\boldsymbol{G}^T_{\mathsf{m}_*}(z^{-1})\boldsymbol{G}_{\mathsf{m}}(z)=2\boldsymbol{I}$ for $z=e^{j\omega}$.

From the above results, we see that orthonormal wavelet systems are associated with orthonormal filter banks. Similarly, biorthonormal wavelet systems are associated with biorthonormal filter banks.

4.2.13 Properties of Scaling and Wavelet Functions

Since scaling and wavelet functions play an important role in characterizing wavelet systems, it is beneficial to examine some of the properties of such functions. First, we introduce some properties of scaling functions as given below.

Theorem 4.16 (Properties of scaling function)**.** *Suppose that ϕ is a compactly supported function with zeroth moment $\mu_0 \neq 0$ and stable integer shifts (i.e., $\{\phi(\cdot-k)\}_{k\in\mathbb{Z}}$ is a Riesz basis), and ϕ satisfies a refinement equation with mask $\sqrt{2}c[n]$. Then, we have*

$$\tfrac{1}{\mu_0}\sum_{k\in\mathbb{Z}}\phi(t-k)=1, \tag{4.35}$$

$$\hat{\phi}(2\pi k)=0 \text{ for all } k\in\mathbb{Z}\setminus\{0\}, \tag{4.36}$$

$$\hat{c}(0)=\sqrt{2}, \quad \text{and} \tag{4.37}$$

$$\hat{c}(\pi)=0. \tag{4.38}$$

Proof. Since $\phi \in L^2$ with compact support, we also have that $\phi \in L^1$. So, by the Riemann-Lebesgue lemma, $\hat{\phi}$ is continuous and decays to zero at infinity. As c has finite support, $\hat{c}$ is continuous. Since $\hat{\phi}$ and $\hat{c}$ are both continuous, (4.12) must hold at every point. For $\omega = 0$, we have

$$\hat{\phi}(0) = \tfrac{1}{\sqrt{2}}\hat{c}(0)\hat{\phi}(0).$$

Since, by assumption, $\hat{\phi}(0) \neq 0$, we have $\hat{c}(0) = \sqrt{2}$. Thus, (4.37) is satisfied. Furthermore, by the periodicity of $\hat{c}$, we have

$$\hat{c}(2\pi k) = \sqrt{2} \quad \text{for all } k \in \mathbb{Z}.$$

So, by substitution into the Fourier transform of the scaling equation, we have

$$\hat{\phi}(4\pi k) = \tfrac{1}{\sqrt{2}}\hat{c}(2\pi k)\hat{\phi}(2\pi k) = \hat{\phi}(2\pi k)$$

or more generally

$$\hat{\phi}(2^n \pi k) = \hat{\phi}(2\pi k) \quad n \in \mathbb{N}.$$

Since $\left|\hat{\phi}(\omega)\right| \to 0$ as $|\omega| \to \infty$, we have $\hat{\phi}(2\pi k) = 0$ for all $k \in \mathbb{Z} \setminus \{0\}$. Thus, (4.36) holds. Furthermore, by considering the Fourier transform of ϕ, we can conclude from (4.36) that (4.38) holds. (See the proof of Theorem 4.40 for more details.)

Using the Poisson summation formula and (4.36), we can write

$$\begin{aligned}\sum_{k\in\mathbb{Z}} \phi(t-k) &= \sum_{k\in\mathbb{Z}} \hat{\phi}(2\pi k) e^{j2\pi kt} \\ &= \hat{\phi}(0) \\ &= \mu_0.\end{aligned}$$

Thus, (4.35) holds. □

The results of the above theorem are quite interesting. In particular, (4.35) is a quite remarkable property of scaling functions. Regardless of the "complexity" in the appearance of a scaling function ϕ, the integer shifts of ϕ always sum to a constant (in particular, they sum to the zeroth moment of ϕ).

Theorem 4.17 (Sum of dyadic samples of scaling function)**.** *Let $\phi(t)$ be a scaling function with zeroth moment μ_0. Then,*

$$\sum_{k\in\mathbb{Z}} \phi(k/2^P) = 2^P \mu_0,$$

where $P \in \mathbb{Z}$ and $P \geq 0$.

Proof. Define the sequence

$$a[n] = \sum_{k\in\mathbb{Z}} \phi(k/2^n).$$

We can expand $\phi(k/2^n)$ using the refinement equation for $\phi(t)$ to obtain

$$\begin{aligned}a[n] &= \sum_{k\in\mathbb{Z}} \left(\sqrt{2} \sum_{l\in\mathbb{Z}} c[l] \phi(2[k/2^n] - l) \right) \\ &= \sqrt{2} \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} c[l] \phi(k/2^{n-1} - l) \\ &= \sqrt{2} \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} c[l] \phi\left(\frac{k - 2^{n-1}l}{2^{n-1}}\right) \\ &= \sqrt{2} \sum_{l\in\mathbb{Z}} c[l] \sum_{k\in\mathbb{Z}} \phi\left(\frac{k - 2^{n-1}l}{2^{n-1}}\right).\end{aligned}$$

Now, we employ a change of variable. Let $k' = k - 2^{n-1}l$ so that $k = k' + 2^{n-1}l$. Applying the change of variable and dropping the primes, we obtain

$$\begin{aligned} a[n] &= \sqrt{2}\sum_{l\in\mathbb{Z}} c[l] \sum_{k\in\mathbb{Z}} \phi(k/2^{n-1}) \\ &= \sqrt{2}\sum_{l\in\mathbb{Z}} c[l]a[n-1] \\ &= \sqrt{2}a[n-1]\sum_{l\in\mathbb{Z}} c[l]. \end{aligned}$$

Since $\sum_{l\in\mathbb{Z}} c[l] = \sqrt{2}$, we can simplify the preceding equation, yielding

$$\begin{aligned} a[n] &= \sqrt{2}a[n-1]\sqrt{2} \\ &= 2a[n-1]. \end{aligned}$$

So, we have

$$a[n] - 2a[n-1] = 0.$$

We can use the unilateral $\mathcal{Z}$ transform to solve this difference equation. Let $A(z)$ denote the $\mathcal{Z}$ transform of $a[n]$. Taking the $\mathcal{Z}$ transform of both sides of the above equation, we have

$$zA(z) - za[0] - 2A(z) = 0$$

which implies

$$A(z) = \frac{a[0]}{1-2z^{-1}}.$$

Taking the inverse $\mathcal{Z}$ transform of $A(z)$, we obtain

$$a[n] = 2^n a[0] \quad \text{for } n \geq 0. \tag{4.39}$$

Now, we must determine $a[0]$. From the definition of $a[n]$, we observe that $a[n]/2^n$ is a Riemann sum whose limit as $n \to \infty$ is the definition of a Riemann integral. Thus, we have

$$\lim_{n\to\infty} \frac{a[n]}{2^n} = \int_{-\infty}^{\infty} \phi(t)dt. \tag{4.40}$$

From (4.39), we can also write

$$\lim_{n\to\infty} \frac{a[n]}{2^n} = \lim_{n\to\infty} \frac{2^n a[0]}{2^n} = a[0]. \tag{4.41}$$

Combining (4.40) and (4.41), we have

$$a[0] = \int_{-\infty}^{\infty} \phi(t)dt.$$

So, (4.39) simplifies to

$$a[n] = 2^n \mu_0.$$

From definition of $a[n]$, we have

$$\sum_{k\in\mathbb{Z}} \phi(k/2^n) = 2^n \mu_0.$$

This is precisely the result that we were required to prove. □

Theorem 4.18. *If $\phi, \tilde{\phi} \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ are biorthogonal scaling functions, then*

$$\mu_0 = 1/(\tilde{\mu}_0^*),$$

where μ_0 and $\tilde{\mu}_0$ are the zeroth moments of ϕ and $\tilde{\phi}$, respectively.

Proof. Since ϕ satisfies a partition of unity relationship, we can expand the constant one in a series to obtain

$$\begin{aligned}1 &= \sum_{k\in\mathbb{Z}} \left\langle 1, \tilde{\phi}(\cdot - k)\right\rangle \phi(t-k) \\ &= \sum_{k\in\mathbb{Z}} \left(\int_{-\infty}^{\infty} \tilde{\phi}^*(\tau - k) d\tau\right) \phi(t-k) \\ &= \sum_{k\in\mathbb{Z}} \tilde{\mu}_0^* \phi(t-k) \\ &= \tilde{\mu}_0^* \sum_{k\in\mathbb{Z}} \phi(t-k) \\ &= \tilde{\mu}_0^* \mu_0.\end{aligned}$$

Thus, we have that $\mu_0 \tilde{\mu}_0^* = 1$, or equivalently, $\mu_0 = 1/(\tilde{\mu}_0^*)$. □

Since a scalar multiple of a solution to a refinement equation is also a solution, there is some freedom in how we choose to normalize the scaling function of a MRA (i.e., how we select the zeroth moment of the function). The above theorem is significant because it shows what normalization of a dual scaling function is associated with a particular normalization of a primal scaling function.

Theorem 4.19. *Let ψ be a wavelet function. Then, the zeroth moment of ψ is zero (i.e., $\int_{-\infty}^{\infty} \psi(t) dt = 0$). The wavelet equation coefficient sequence d is such that $\hat{d}(0) = 0$.*

Proof. First, we consider the quantity $\hat{d}(0)$. By evaluating (4.31d) at $\omega = 0$, we obtain

$$\hat{d}(0)\hat{c}^*(0) + \hat{d}(\pi)\hat{c}^*(\pi) = 0.$$

Since $\hat{c}(0) = \sqrt{2}$ and $\hat{c}(\pi) = 0$ (from (4.37) and (4.38)), the preceding equation can be simplified to $\sqrt{2}\hat{d}(0) = 0$. Thus, we conclude that $\hat{d}(0) = 0$.

We know that ψ can be expressed in terms of the scaling function as

$$\psi(t) = \sqrt{2} \sum_{n\in\mathbb{Z}} d[n] \phi(2t - n).$$

Integrating both sides of the preceding equation (with respect to t) over $\mathbb{R}$, we obtain

$$\begin{aligned}\int_{-\infty}^{\infty} \psi(t) dt &= \int_{-\infty}^{\infty} \sqrt{2} \sum_{n\in\mathbb{Z}} d[n] \phi(2t-n) dt \\ &= \sqrt{2} \sum_{n\in\mathbb{Z}} d[n] \int_{-\infty}^{\infty} \phi(2t-n) dt \\ &= \sqrt{2} \left[\sum_{n\in\mathbb{Z}} d[n]\right] \int_{-\infty}^{\infty} \phi(2t-n) dt \\ &= \sqrt{2} \hat{d}(0) \int_{-\infty}^{\infty} \phi(2t-n) dt \\ &= 0.\end{aligned}$$

□

Theorem 4.20 (Relationship between continuous and discrete moments)**.** *Let ϕ be a scaling function with scaling equation coefficient sequence c. Let ψ be a corresponding wavelet function with wavelet equation coefficient sequence d. Denote the kth moments of ϕ and ψ as μ_k and ν_k, respectively. Denote the kth moments of c and d as m_k and n_k, respectively. Then, we have*

$$\mu_k = \frac{1}{2^{1/2}(2^k-1)} \sum_{q=0}^{k-1} \binom{k}{q} m_{k-q}\mu_q \text{ for } k \geq 1 \quad \text{and} \tag{4.42}$$

$$\nu_k = 2^{-k-1/2} \sum_{q=0}^{k} \binom{k}{q} n_{k-q}\mu_q \text{ for } k \geq 0. \tag{4.43}$$

(Note: The notation $\binom{n}{k}$ is as defined in (E.1).)

Proof. We begin with the Fourier transform of the scaling equation which can be rewritten as

$$\hat{\phi}(2\omega) = 2^{-1/2}\hat{c}(\omega)\hat{\phi}(\omega).$$

Differentiating k times (using the Leibniz rule), we obtain

$$2^k\hat{\phi}^{(k)}(2\omega) = 2^{-1/2} \sum_{q=0}^{k} \binom{k}{q} \hat{c}^{(k-q)}(\omega)\hat{\phi}^{(q)}(\omega).$$

Substituting $\omega = 0$ into the above equation yields

$$2^k\hat{\phi}^{(k)}(0) = 2^{-1/2} \sum_{q=0}^{k} \binom{k}{q} \hat{c}^{(k-q)}(0)\hat{\phi}^{(q)}(0).$$

From the moment properties of the Fourier transform, however, this is equivalent to

$$2^k\mu_k = 2^{-1/2} \sum_{q=0}^{k} \binom{k}{q} m_{k-q}\mu_q.$$

Rearranging, we have

$$2^k\mu_k = 2^{-1/2} \sum_{q=0}^{k-1} \binom{k}{q} m_{k-q}\mu_q + 2^{-1/2}m_0\mu_k$$

$$\Rightarrow \quad (2^k-1)\mu_k = 2^{-1/2} \sum_{q=0}^{k-1} \binom{k}{q} m_{k-q}\mu_q.$$

Solving for μ_k in the above equation yields (4.42).

The proof of (4.43) is obtained in a similar fashion as above, except we start from the Fourier transform of the wavelet equation. The details of the proof are left as an exercise for the reader. □

The above result is quite significant from a practical perspective. Often scaling and wavelet functions cannot be expressed in closed form. In spite of this, however, we would often like to know the moments of these functions. The above theorem provides a means to calculate the moments of these functions from their corresponding coefficient sequences (from the scaling and wavelet equations) without ever needing to explicitly compute the functions themselves. Below, we give an example illustrating how the above theorem can be applied.

Example 4.17. Consider the Daubechies-2 scaling function ϕ with scaling equation coefficient sequence c, where

$$c_0 = \tfrac{1+\sqrt{3}}{4\sqrt{2}}, \; c_1 = \tfrac{3+\sqrt{3}}{4\sqrt{2}}, \; c_2 = \tfrac{3-\sqrt{3}}{4\sqrt{2}}, \text{ and } c_3 = \tfrac{1-\sqrt{3}}{4\sqrt{2}}.$$

(Note that, here, ϕ is normalized so $\operatorname{supp}\phi \subset [0,3]$.) Compute the first four moments of c. Suppose that ϕ is normalized such that $\mu_0 = 1$. Compute the next three moments of ϕ.

Solution. We begin by computing moments of c. From the properties of scaling function, we know (without doing any computation) that

$$m_0 = \hat{c}(0) = \sqrt{2}.$$

Using the definition of moments, we have

$$\begin{aligned} m_1 &= \sum_{k\in\mathbb{Z}} kc_k = c_1 + 2c_2 + 3c_3 \\ &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} + 2\left(\tfrac{3-\sqrt{3}}{4\sqrt{2}}\right) + 3\left(\tfrac{1-\sqrt{3}}{4\sqrt{2}}\right) = \tfrac{3+\sqrt{3}+6-2\sqrt{3}+3-3\sqrt{3}}{4\sqrt{2}} = \tfrac{12-4\sqrt{3}}{4\sqrt{2}} \\ &= \tfrac{3-\sqrt{3}}{\sqrt{2}}, \end{aligned}$$

$$\begin{aligned} m_2 &= \sum_{k\in\mathbb{Z}} k^2 c_k = c_1 + 4c_2 + 9c_3 \\ &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} + 4\left(\tfrac{3-\sqrt{3}}{4\sqrt{2}}\right) + 9\left(\tfrac{1-\sqrt{3}}{4\sqrt{2}}\right) = \tfrac{3+\sqrt{3}+12-4\sqrt{3}+9-9\sqrt{3}}{4\sqrt{2}} = \tfrac{24-12\sqrt{3}}{4\sqrt{2}} \\ &= \tfrac{6-3\sqrt{3}}{\sqrt{2}}, \quad \text{and} \end{aligned}$$

$$\begin{aligned} m_3 &= \sum_{k\in\mathbb{Z}} k^3 c_k = c_1 + 8c_2 + 27c_3 \\ &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} + 8\left(\tfrac{3-\sqrt{3}}{4\sqrt{2}}\right) + 27\left(\tfrac{1-\sqrt{3}}{4\sqrt{2}}\right) = \tfrac{3+\sqrt{3}+24-8\sqrt{3}+27-27\sqrt{3}}{4\sqrt{2}} = \tfrac{54-34\sqrt{3}}{4\sqrt{2}} \\ &= \tfrac{27-17\sqrt{3}}{2\sqrt{2}}. \end{aligned}$$

Using (4.42), we proceed to compute moments of ϕ. We have

$$\begin{aligned} \mu_1 &= \tfrac{1}{\sqrt{2}(2-1)} \sum_{q=0}^{0} \binom{1}{q} m_{1-q}\mu_q = \tfrac{1}{\sqrt{2}} \left[\binom{1}{0} m_1\mu_0\right] \\ &= \tfrac{1}{\sqrt{2}} \left(\binom{1}{0} (\tfrac{3-\sqrt{3}}{\sqrt{2}})(1)\right) = \tfrac{1}{\sqrt{2}} \left(\tfrac{3-\sqrt{3}}{\sqrt{2}}\right) \\ &= \tfrac{3-\sqrt{3}}{2}. \end{aligned}$$

$$\begin{aligned} \mu_2 &= \tfrac{1}{\sqrt{2}(2^2-1)} \sum_{q=0}^{1} \binom{2}{q} m_{2-q}\mu_q = \tfrac{1}{3\sqrt{2}} \left[\binom{2}{0} m_2\mu_0 + \binom{2}{1} m_1\mu_1\right] \\ &= \tfrac{1}{3\sqrt{2}} \left(\binom{2}{0} (\tfrac{6-3\sqrt{3}}{\sqrt{2}})(1) + \binom{2}{1} (\tfrac{3-\sqrt{3}}{\sqrt{2}})(\tfrac{3-\sqrt{3}}{2})\right) = \tfrac{1}{3\sqrt{2}} \left(\tfrac{6-3\sqrt{3}}{\sqrt{2}} + 2(\tfrac{3-\sqrt{3}}{\sqrt{2}})(\tfrac{3-\sqrt{3}}{2})\right) \\ &= \tfrac{6-3\sqrt{3}}{2}, \quad \text{and} \end{aligned}$$

$$\begin{aligned} \mu_3 &= \tfrac{1}{\sqrt{2}(2^3-1)} \sum_{q=0}^{2} \binom{3}{q} m_{3-q}\mu_q \\ &= \tfrac{1}{7\sqrt{2}} \left(\binom{3}{0} m_3\mu_0 + \binom{3}{1} m_2\mu_1 + \binom{3}{2} m_1\mu_2\right) \\ &= \tfrac{1}{7\sqrt{2}} \left((1)(\tfrac{27-17\sqrt{3}}{2\sqrt{2}})(1) + (3)(\tfrac{6-3\sqrt{3}}{\sqrt{2}})(\tfrac{3-\sqrt{3}}{2}) + (3)(\tfrac{3-\sqrt{3}}{\sqrt{2}})(\tfrac{6-3\sqrt{3}}{2})\right) \\ &= \tfrac{1}{7\sqrt{2}} \left(\tfrac{27-17\sqrt{3}}{2\sqrt{2}} + \tfrac{(18-9\sqrt{3})(3-\sqrt{3})}{2\sqrt{2}} + \tfrac{(9-3\sqrt{3})(6-3\sqrt{3})}{2\sqrt{2}}\right) \\ &= \tfrac{1}{28} \left(27 - 17\sqrt{3} + 54 - 18\sqrt{3} - 27\sqrt{3} + 27 + 54 - 27\sqrt{3} - 18\sqrt{3} + 27\right) \\ &= \tfrac{189-107\sqrt{3}}{28}. \end{aligned}$$

□

Often, we are interested in the specific case when certain moments of the wavelet and/or scaling function are zero. The below theorem is useful in this regard.

Theorem 4.21 (Equivalence of vanishing continuous and discrete moments)**.** *Let ϕ be a scaling function with scaling equation coefficient sequence c. Let ψ be the corresponding wavelet function with wavelet equation coefficient sequence d. Denote the kth moments of ϕ and ψ as μ_k and ν_k, respectively. Denote the kth moments of c and d as m_k and n_k, respectively. Then, we have that*

$$m_k = 0 \textit{ for } k = 1,2,\ldots,\eta-1 \Leftrightarrow \mu_k = 0 \textit{ for } k = 1,2,\ldots,\eta-1; \quad \textit{and}$$
$$n_k = 0 \textit{ for } k = 0,1,\ldots,\eta-1 \Leftrightarrow \nu_k = 0 \textit{ for } k = 0,1,\ldots,\eta-1.$$

Proof. The proof is by induction and is left as an exercise for the reader. □

4.2.14 Orthogonalization of Bases

Suppose that we have a wavelet system where the bases of the approximation and wavelet spaces are only Riesz bases, but not orthonormal. In some cases, it might be desirable to employ orthonormal bases for these spaces. Fortunately, a shift invariant Riesz basis can always be orthonormalized. This can be accomplished using the result of the theorem below.

Theorem 4.22 (Orthonormalization of basis)**.** *Let $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ be a Riesz basis of the space V. Then, $\{\theta^\perp(\cdot - n)\}_{n\in\mathbb{Z}}$ is an orthonormal basis of V, where*

$$\widehat{\theta^\perp}(\omega) = \frac{\hat\theta(\omega)}{\left[\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi k)\right|^2\right]^{1/2}}.$$

Proof. Suppose that $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis of the space that it generates. Then, by virtue of Theorem 4.1, $\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi k)\right|^2 > 0$ for all ω. Since this quantity is strictly positive, the Fourier transform

$$\hat\theta^\perp(\omega) = \frac{\hat\theta(\omega)}{\left(\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi k)\right|^2\right)^{1/2}}$$

is well defined. Using Theorem 4.2, we can test for orthonormality. We have

$$\begin{aligned}
\sum_{l\in\mathbb{Z}}\left|\hat\theta^\perp(\omega+2\pi l)\right|^2 &= \sum_{l\in\mathbb{Z}}\left|\frac{\hat\theta(\omega+2\pi k)}{\left[\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi l+2\pi k)\right|^2\right]^{1/2}}\right|^2 \\
&= \sum_{l\in\mathbb{Z}}\frac{\left|\hat\theta(\omega+2\pi l)\right|^2}{\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi l+2\pi k)\right|^2} \\
&= \sum_{l\in\mathbb{Z}}\frac{\left|\hat\theta(\omega+2\pi l)\right|^2}{\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi k)\right|^2} \\
&= \frac{\sum_{l\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi l)\right|^2}{\sum_{k\in\mathbb{Z}}\left|\hat\theta(\omega+2\pi k)\right|^2} \\
&= 1.
\end{aligned}$$

Thus, $\{\theta^\perp(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of its closed linear span.

Now, it remains to be shown that $\operatorname{clos}\left(\operatorname{span}\{\theta^\perp(\cdot - k)\}_{k\in\mathbb{Z}}\right) = V$. It is sufficient to show that $\theta^\perp \in V$ and θ has an expansion in terms of $\{\theta^\perp(\cdot - k)\}_{k\in\mathbb{Z}}$.

Define

$$\hat{g}(\omega) = \left[\sum_{n\in\mathbb{Z}} \left| \hat{\theta}(\omega + 2\pi n) \right|^2 \right]^{-1/2}.$$

Since $\hat{g}$ is a 2π-periodic function, it can be expressed in terms of the Fourier series

$$\hat{g}(\omega) = \sum_{k\in\mathbb{Z}} g_k e^{jk\omega}.$$

From the convolution property of the Fourier transform, we have

$$\theta^\perp(t) = \theta * g(t).$$

Taking the inverse Fourier transform of $\hat{g}(\omega)$, we have

$$g(t) = \sum_{k\in\mathbb{Z}} g_k \delta(t+k).$$

Combining the preceding two equations, we obtain

$$\begin{aligned}
\theta^\perp(t) &= \theta * g(t) \\
&= \int_{-\infty}^{\infty} \theta(\tau) g(t-\tau) d\tau \\
&= \int_{-\infty}^{\infty} \theta(\tau) \sum_{k\in\mathbb{Z}} g_k \delta(t-\tau+k) d\tau \\
&= \sum_{k\in\mathbb{Z}} g_k \int_{-\infty}^{\infty} \theta(\tau)\delta(t-\tau+k) d\tau \\
&= \sum_{k\in\mathbb{Z}} g_k \theta(t+k).
\end{aligned}$$

Thus, $\theta^\perp$ can be represented in terms of the basis $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ of V. So, $\theta^\perp \in V$.

Consider the quantity $\sum_{k\in\mathbb{Z}} \left| \left\langle \theta(\cdot), \theta^\perp(\cdot - k) \right\rangle \right|^2$. We have

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} \left| \left\langle \theta(\cdot), \theta^\perp(\cdot - k) \right\rangle \right|^2 &= \sum_{k\in\mathbb{Z}} \left| \tfrac{1}{2\pi} \left\langle \hat{\theta}(\cdot), e^{-jk\cdot} \widehat{\theta^\perp}(\cdot) \right\rangle \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \left| \tfrac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\theta}(\omega) e^{-jk\omega} \widehat{\theta^\perp}^*(\omega) d\omega \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \left| \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\theta}(\omega) \frac{\hat{\theta}^*(\omega)}{\left[\sum_{n\in\mathbb{Z}} \left| \hat{\theta}(\omega+2\pi n) \right|^2 \right]^{1/2}} e^{-jk\omega} d\omega \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \left| \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\left| \hat{\theta}(\omega) \right|^2}{\left[\sum_{n\in\mathbb{Z}} \left| \hat{\theta}(\omega+2\pi n) \right|^2 \right]^{1/2}} e^{-jk\omega} d\omega \right|^2.
\end{aligned}$$

We split the integral over $\mathbb{R}$ into intervals of length 2π to obtain

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} \left|\left\langle \theta(\cdot), \theta^{\perp}(\cdot - k)\right\rangle\right|^2 &= \sum_{k\in\mathbb{Z}} \left| \tfrac{1}{2\pi} \sum_{l\in\mathbb{Z}} \int_0^{2\pi} \frac{\left|\hat{\theta}(\omega+2\pi l)\right|^2}{\left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2}} e^{-jk\omega} d\omega \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \left| \tfrac{1}{2\pi} \int_0^{2\pi} \frac{\sum_{l\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi l)\right|^2}{\left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2}} e^{-jk\omega} d\omega \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \left| \tfrac{1}{2\pi} \int_0^{2\pi} \left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2} e^{-jk\omega} d\omega \right|^2 \\
&= \sum_{k\in\mathbb{Z}} \tfrac{1}{2\pi} \left| \int_0^{2\pi} \left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2} e^{-jk\omega} d\omega \right|^2 .
\end{aligned}$$

We recognize the kth term in the right-hand side summation as $\frac{1}{2\pi}$ times the kth Fourier series coefficient of the 2π-periodic function $\left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2}$. By the Parseval identity, we have

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} \left|\left\langle \theta(\cdot), \theta^{\perp}(\cdot - k)\right\rangle\right|^2 &= \tfrac{1}{2\pi} \left\| \left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2} \right\|_{L^2[0,2\pi]}^2 \\
&= \tfrac{1}{2\pi} \left\langle \left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2}, \left[\sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2\right]^{1/2} \right\rangle_{L^2[0,2\pi]} \\
&= \tfrac{1}{2\pi} \int_0^{2\pi} \sum_{n\in\mathbb{Z}} \left|\hat{\theta}(\omega+2\pi n)\right|^2 d\omega \\
&= \tfrac{1}{2\pi} \sum_{n\in\mathbb{Z}} \int_0^{2\pi} \left|\hat{\theta}(\omega+2\pi n)\right|^2 d\omega \\
&= \tfrac{1}{2\pi} \sum_{n\in\mathbb{Z}} \int_0^{2\pi} \hat{\theta}(\omega+2\pi n)\hat{\theta}^*(\omega+2\pi n) d\omega \\
&= \tfrac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\theta}(\omega)\hat{\theta}^*(\omega) d\omega \\
&= \tfrac{1}{2\pi} \left\|\hat{\theta}\right\|^2 \\
&= \|\theta\|^2 .
\end{aligned}$$

Thus, we have

$$\sum_{k\in\mathbb{Z}} \left|\left\langle \theta(\cdot), \theta^{\perp}(\cdot - k)\right\rangle\right|^2 = \|\theta\|^2 .$$

By the Bessel inequality, this implies that θ lies in the closed linear span of $\{\theta^{\perp}(\cdot - k)\}_{k\in\mathbb{Z}}$. Therefore, the closed linear span of $\{\theta^{\perp}(\cdot - k)\}_{k\in\mathbb{Z}}$ is V. □

Although one can always construct an orthonormal basis using the result of the above theorem, an orthonormal basis may not always be desirable. For example, even if θ is compactly supported, there is no guarantee that $\theta^{\perp}$ will also be compactly supported. Consequently, if an application requires compactly-supported basis functions, one might need to be content with a Riesz basis that is not orthonormal.

4.2.15 Support of Scaling and Wavelet Functions

One might wonder if there is any simple way to characterize the support of a refinable function. This is, in fact, possible. As it turns out, there is a relationship between the support of a refinable function and the support of its refinement mask. The relationship is introduced by the theorem below.

Theorem 4.23 (Support of refinable functions with finite masks)**.** *Suppose that we have a refinement equation of the form*

$$\phi(t) = \sum_{n\in\mathbb{Z}} c[n]\phi(2t-n) \tag{4.44}$$

where

$$c[n] = 0 \text{ for } n < N_1 \text{ or } n > N_2$$

and $\sum_{n=N_1}^{N_2} c[n] = 1$. *Then, the solution to the refinement equation is a function (or distribution) with support in* $[N_1, N_2]$.

Proof. The difficult part of this theorem to prove is that the solution ϕ must have compact support. So, we simply assert that ϕ must have compact support as shown in [8, p. 497, Lemma 3.1]. Suppose that

$$\operatorname{supp}\phi(t) \subset [A_1, A_2]$$

which implies

$$\operatorname{supp}\phi(2t-k) \subset \left[\frac{A_1+k}{2}, \frac{A_2+k}{2}\right], \; k \in \mathbb{Z}.$$

Consider the right-hand side of the refinement equation (4.44). The right-hand side of (4.44) is formed by summing shifted versions of $\phi(2t)$. Thus, the lower and upper bounds of the support interval of the right-hand side will be determined by the leftmost and rightmost shifted versions of $\phi(2t)$, respectively. The leftmost shifted version of $\phi(2t)$, given by $\phi(2t-N_1)$ has support on an interval with the lower bound $\frac{A_1+N_1}{2}$. Similarly, the rightmost shifted version of $\phi(2t)$, given by $\phi(2t-N_2)$, has its support contained on an interval with the upper bound $\frac{A_2+N_2}{2}$. Thus, the support of the right-hand side of (4.44) is contained in $[\frac{A_1+N_1}{2}, \frac{A_2+N_2}{2}]$. Since the support of the left- and right-hand sides of the equation must be equal, we have

$$A_1 = \frac{A_1+N_1}{2} \quad \Rightarrow \quad A_1 = N_1 \quad \text{and}$$
$$A_2 = \frac{A_2+N_2}{2} \quad \Rightarrow \quad A_2 = N_2.$$

Thus, we have $\operatorname{supp}\phi \subset [N_1, N_2]$. □

From the above theorem, we see that a refinement equation with a finite mask must have a solution with compact support. The converse of this statement, however, is not true. This is illustrated by the example below.

Example 4.18 (Compactly supported refinable function with infinite mask)**.** Consider the refinement equation

$$\phi(t) = \sum_{n\in\mathbb{Z}} c[n]\phi(2t-n)$$

where $c[n] = \mathcal{Z}^{-1}C(z)$ and

$$C(z) = 1 + \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2} + \sum_{k=0}^{\infty}\left(-\frac{1}{2}\right)^k \frac{3}{8}z^{-(k+3)}.$$

One can show [23] that the solution $\phi(t)$ to the above refinement equation is given by

$$\phi(t) = 2\chi_{[0,1)} + \chi_{[1,2]} = \begin{cases} 2 & \text{if } 0 \leq t < 1 \\ 1 & \text{if } 1 \leq t \leq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, $\operatorname{supp}\phi \subset [0,2]$. Thus, we can see that a refinement equation with an infinite mask can, in some cases, have a solution with compact support.

Theorem 4.24 (Compactly supported scaling and wavelet functions)**.** *Let ϕ be a scaling function with scaling equation coefficient sequence c. Let ψ be the corresponding wavelet function with wavelet equation coefficient sequence d. If $c[n] = 0$ whenever $n < N_1$ or $n > N_2$, then*

$$\operatorname{supp}\phi \subset [N_1, N_2].$$

If, in addition, $d[n] = 0$ for $n < M_1$ and $n > M_2$, then

$$\operatorname{supp}\psi \subset \left[\frac{N_1+M_1}{2}, \frac{N_2+M_2}{2}\right].$$

Proof. From Theorem 4.23, we trivially have that

$$\operatorname{supp}\phi(t) \subset [N_1, N_2].$$

Consider the wavelet equation

$$\psi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} d[k]\phi(2t-k)$$

Let $[A_1, A_2]$ denote the support of the right-hand side of the equation. Since the support of the left- and right-hand sides of the equation must be equal, we have $\operatorname{supp}\psi(t) \subset [A_1, A_2]$. Now, we must determine A_1 and A_2. Since $\operatorname{supp}\phi(t) \subset [N_1, N_2]$, we have

$$\operatorname{supp}\phi(2t-k) \subset \left[\frac{N_1+k}{2}, \frac{N_2+k}{2}\right]. \tag{4.45}$$

We will use this fact in what follows.

We have that A_1 is determined by the lower support bound of the leftmost-shifted function $\phi(2t-k)$ where $k = M_1$. Thus, from (4.45), we can write $A_1 = \frac{N_1+M_1}{2}$. Similarly, A_2 is determined by the upper support bound of the rightmost-shifted $\phi(2t-k)$ where $k = M_2$. Thus, from (4.45), we can write $A_2 = \frac{N_2+M_2}{2}$. Combining these results, we have shown

$$\operatorname{supp}\psi(t) \subset [A_1, A_2] = \left[\frac{N_1+M_1}{2}, \frac{N_2+M_2}{2}\right].$$

□

Below, we provide some examples illustrating how Theorem 4.24 can be applied.

Example 4.19. Consider a wavelet system with scaling function ϕ, wavelet function ψ, dual scaling function $\tilde{\phi}$, and dual wavelet function $\tilde{\psi}$. Let c and $\tilde{c}$ denote the scaling equation coefficient sequences of ϕ and $\tilde{\phi}$, respectively. Let d and $\tilde{d}$ denote the wavelet equation coefficient sequences of ψ and $\tilde{\psi}$, respectively. Suppose that c, d, $\tilde{c}$, and $\tilde{d}$ are as follows:

$$c_{-1} = c_0 = \tfrac{1}{\sqrt{2}},$$
$$d_{-5} = -d_4 = \tfrac{3}{128\sqrt{2}}, \quad d_{-4} = -d_3 = \tfrac{3}{128\sqrt{2}}, \quad d_{-3} = -d_2 = -\tfrac{22}{128\sqrt{2}},$$
$$d_{-2} = -d_1 = -\tfrac{22}{128\sqrt{2}}, \quad d_{-1} = -d_0 = \tfrac{1}{\sqrt{2}},$$
$$\tilde{c}_n = (-1)^n d_{n-1}, \quad \text{and}$$
$$\tilde{d}_n = (-1)^{n-1} c_{n-1}.$$

Determine the support of ϕ, ψ, $\tilde{\phi}$, and $\tilde{\psi}$.

Solution. We use the result of Theorem 4.24. The sequence c_n is zero for $n < -1$ or $n > 0$. Therefore, we conclude

$$\operatorname{supp}\phi \subset [-1,0].$$

The sequence d_n is zero for $n < -5$ or $n > 4$. Therefore, we conclude

$$\operatorname{supp}\psi \subset [\tfrac{-1-5}{2}, \tfrac{0+4}{2}] = [-3,2].$$

The sequence $\tilde{c}_n$ is zero for $n < -4$ or $n > 5$. Therefore, we conclude

$$\operatorname{supp}\tilde{\phi} \subset [-4,5].$$

The sequence $\tilde{d}_n$ is zero for $n < 0$ or $n > 1$. Therefore, we conclude

$$\operatorname{supp}\tilde{\psi} \subset [\tfrac{-4+0}{2}, \tfrac{5+1}{2}] = [-2,3].$$

□

Example 4.20. Consider the Daubechies-2, Le-Gall-5/3, and CDF-9/7 wavelet systems introduced earlier. For each of these systems, find the support of the scaling and wavelet functions.

Solution. (a) Consider the Daubechies-2 wavelet system. We use the result of Theorem 4.24. We have that $c_n = 0$ for $n < -1$ or $n > 2$. So, we conclude

$$\operatorname{supp}\phi \subset [-1,2].$$

We have that $d_n = 0$ for $n < -1$ or $n > 2$. So, we conclude

$$\operatorname{supp}\psi \subset [\tfrac{(-1)+(-1)}{2}, \tfrac{2+2}{2}] = [-1,2].$$

The above support intervals are consistent with the plots of ϕ, ψ, $\tilde{\phi}$, and $\tilde{\psi}$ shown in Figure 4.15.

(b) Consider the Le Gall 5/3 wavelet system. We have that $c_n = 0$ for $n < -1$ or $n > 1$. Therefore, we conclude

$$\operatorname{supp}\phi \subset [-1,1].$$

We have $d_n = 0$ for $n < -1$ or $n > 3$. Thus, we have

$$\operatorname{supp}\psi \subset \left[\tfrac{(-1)+(-1)}{2}, \tfrac{1+3}{2}\right] = [-1,2].$$

We have $\tilde{c}_n = 0$ for $n < -2$ or $n > 2$. So, we have

$$\operatorname{supp}\tilde{\phi} \subset [-2,2].$$

We have $\tilde{d}_n = 0$ for $n < 0$ or $n > 2$. So, we have

$$\operatorname{supp}\tilde{\psi} \subset \left[\tfrac{-2+0}{2}, \tfrac{2+2}{2}\right] = [-1,2].$$

The above support intervals are consistent with the plots of ϕ, ψ, $\tilde{\phi}$, and $\tilde{\psi}$ shown in Figure 4.16.

(c) Consider the CDF 9/7 wavelet system. We have that $c_n = 0$ for $n < -3$ or $n > 3$. Therefore, we conclude

$$\operatorname{supp}\phi \subset [-3,3].$$

We have $d_n = 0$ for $n < -3$ or $n > 5$. Thus, we have

$$\operatorname{supp}\psi \subset \left[\tfrac{(-3)+(-3)}{2}, \tfrac{3+5}{2}\right] = [-3,4].$$

We have $\tilde{c}_n = 0$ for $n < -4$ or $n > 4$. So, we have

$$\operatorname{supp}\tilde{\phi} \subset [-4,4].$$

We have $\tilde{d}_n = 0$ for $n < -2$ or $n > 4$. So, we have

$$\operatorname{supp}\tilde{\psi} \subset \left[\tfrac{-4+(-2)}{2}, \tfrac{4+4}{2}\right] = [-3,4].$$

The above support intervals are consistent with the plots of ϕ, ψ, $\tilde{\phi}$, and $\tilde{\psi}$ shown in Figure 4.17.

□

We can also say something about the value of a scaling function at the end points of its support interval. In particular, we have the result below.

Theorem 4.25. *Let ϕ be a scaling function with scaling equation coefficient sequence c, where $c[n] = 0$ for $c < N_1$ or $c > N_2$. If $c[N_1] \neq \frac{1}{\sqrt{2}}$, then $\phi(N_1) = 0$. If $c[N_2] \neq \frac{1}{\sqrt{2}}$, then $\phi(N_2) = 0$.*

Proof. Using the scaling equation, we can evaluate ϕ at the integer points (i.e., for $n \in \mathbb{Z}$) yielding

$$\phi(n) = \sqrt{2} \sum_{k=N_1}^{N_2} c[k]\phi(2n-k).$$

Now, we employ a change of variable. Let $k' = 2n - k$ so that $k = 2n - k'$. Applying the change of variable and dropping the primes, we obtain

$$\phi(n) = \sqrt{2} \sum_{k=2n-N_2}^{2n-N_1} c[2n-k]\phi(k). \tag{4.46}$$

Using (4.46) and the fact that $\phi(k) = 0$ for $k \notin [N_1, N_2]$, we can write

$$\begin{aligned}\phi(N_1) &= \sqrt{2} \sum_{k=2N_1-N_2}^{N_1} c[2N_1-k]\phi(k)\\ &= \sqrt{2}c[N_1]\phi(N_1).\end{aligned}$$

If $\sqrt{2}c[N_1] \neq 1$ (or equivalently, $c[N_1] \neq \frac{1}{\sqrt{2}}$), then the preceding equation implies that $\phi(N_1) = 0$. Also, using (4.46) and the fact that $\phi(k) = 0$ for $k \notin [N_1, N_2]$, we can write

$$\begin{aligned}\phi(N_2) &= \sqrt{2} \sum_{k=N_2}^{2N_2-N_1} c[2N_2-k]\phi(k)\\ &= \sqrt{2}c[N_2]\phi(N_2).\end{aligned}$$

If $\sqrt{2}c[N_2] \neq 1$ (or equivalently, $c[N_2] \neq \frac{1}{\sqrt{2}}$), then the preceding equation implies that $\phi(N_2) = 0$. □

4.2.16 Order of Approximation

Polynomials are a useful tool for the approximation of many classes of functions. In many practical applications, polynomials often prove to be a suitable model for signals. Since polynomials are often a useful tool for approximation, one might be interested in knowing how well polynomials can be approximated by the bases associated with a wavelet system.

As we have already seen, any scaling function satisfies a partition of unity relationship. In other words, the integer shifts of a scaling function can be used to reproduce any constant function. In some cases, however, it so happens that scaling functions can exactly represent polynomials of higher orders. The polynomial approximation properties of a scaling function are related to the number of vanishing moments of the corresponding dual wavelet function.

Suppose, for a moment, that the integer shifts of a scaling function could reproduce polynomials of order η. In many situations, this can be quite advantageous. Why is this so? Consider the representation of a function that is well approximated by a polynomial of order η, such as a piecewise smooth function. Since the scaling function can exactly reproduce polynomials of order η, the wavelet coefficients will essentially be zero over all regions where the function in question is well approximated by a polynomial of order η. In this way, very sparse representations can be obtained for piecewise smooth functions (i.e., representations with relatively few nonzero expansion coefficients). This has major advantages in a broad spectrum of applications.

In order to characterize the approximation properties of scaling functions, we will need an important result from approximation theory which is given below.

Theorem 4.26 (Strang-Fix condition)**.** *Let ϕ be a scaling function with the scaling equation coefficient sequence c. Further, let $\hat{\phi}$ and $\hat{c}$ denote the Fourier transforms of ϕ and c, respectively. If $\hat{c}(\omega)$ has a pth order zero at π, then $\hat{\phi}(\omega)$ must have a pth order zero at all frequencies of the form $\omega = 2\pi k$, where $k \in \mathbb{Z} \setminus \{0\}$.*

Proof. Consider the quantity $\hat{\phi}(2\pi k)$ for any arbitrary $k \in \mathbb{Z} \setminus \{0\}$. We can express k in the form of $k = 2^n l$, where $n, l \in \mathbb{Z}$, $n \geq 0$, and l is odd. Now, we apply formula (4.12) recursively n times to obtain

$$\begin{aligned}
\hat{\phi}(\omega) &= \tfrac{1}{\sqrt{2}}\hat{c}(\omega/2)\hat{\phi}(\omega/2) \\
&= \left[\prod_{q=1}^{n+1} \tfrac{1}{\sqrt{2}}\hat{c}(2^{-q}\omega)\right]\hat{\phi}(2^{-(n+1)}\omega) \\
&= \tfrac{1}{\sqrt{2}}\hat{c}(2^{-(n+1)}\omega)\left[\prod_{q=1}^{n} \tfrac{1}{\sqrt{2}}\hat{c}(2^{-q}\omega)\right]\hat{\phi}(2^{-n-1}\omega) \\
&= \tfrac{1}{\sqrt{2}}\hat{c}(2^{-n-1}\omega)\left[\prod_{q=1}^{n} \tfrac{1}{\sqrt{2}}\hat{c}(2^{-q}\omega)\right]\hat{\phi}(2^{-n-1}\omega).
\end{aligned}$$

Now, consider the factor $\hat{c}(2^{-n-1}\omega)$ above. We are evaluating $\hat{c}(\omega')$ at $\omega' = 2^{-n-1}\omega = 2^{-n-1}2\pi k = 2^{-n-1}2\pi 2^n l = \pi l$, where l is odd. In other words, we are evaluating $\hat{c}(\omega')$ at an odd integer multiple of π. By assumption, however, $\hat{c}(\omega)$ has a pth order zero at $\omega = \pi$. Therefore, by periodicity, $\hat{c}(\omega)$ also has a pth order zero at all odd integer multiples of π. Thus, $c(2^{-n-1}\omega)$ has a pth order zero at $\omega = \pi$. Consequently, $\hat{\phi}(\omega)$ has a pth order zero at $\omega = 2\pi k$. □

Now, we are in a position to study the approximation properties of scaling functions more closely. In fact, we can characterize the approximation properties of scaling functions as given by the theorem below.

Theorem 4.27 (Order of approximation)**.** *Let ϕ be a scaling function with the scaling equation coefficient sequence c. Let $\tilde{\psi}$ denote the corresponding dual wavelet function with the wavelet equation coefficient sequence $\tilde{d}$. Suppose that $\hat{\phi}$ and $\hat{\tilde{\psi}}$ are $\eta - 1$ times differentiable. Then, the following statements are equivalent:*

1. *ϕ has approximation order η;*
2. *for any $0 \leq k < \eta$, $q_k(t) = \sum_{n \in \mathbb{Z}} n^k \phi(t-n)$ is a kth degree polynomial;*
3. *$\hat{c}(\omega)$ has a ηth order zero at $\omega = \pi$;*
4. *$\tilde{\psi}$ has η vanishing moments (i.e., $\tilde{\psi}$ has all of its moments of order less than η vanish);*
5. *$\hat{\tilde{d}}(\omega)$ has a ηth order zero at $\omega = 0$.*

Proof. $4 \Leftrightarrow 5$. We observe that 5 is equivalent to $\tilde{d}$ having η vanishing moments (from the moment properties of the Fourier transform). From Theorem 4.21, this is equivalent to $\tilde{\psi}$ having η vanishing moments.

$3 \Leftrightarrow 5$. From the biorthogonality condition (4.31a), we have $\hat{c}(\omega)\hat{\tilde{c}}^*(\omega) + \hat{c}(\omega+\pi)\hat{\tilde{c}}^*(\omega+\pi) = 2$. Letting $\omega = 0$, this becomes

$$\hat{c}(0)\hat{\tilde{c}}^*(0) + \hat{c}(\pi)\hat{\tilde{c}}^*(\pi) = 2.$$

From this, we can conclude that $\hat{c}(\pi) = 0 \Rightarrow \hat{c}(0) \neq 0$. From the biorthogonality condition (4.31c), we have $\hat{d}(\omega)\hat{\tilde{d}}^*(\omega) + \hat{d}(\omega+\pi)\hat{\tilde{d}}^*(\omega+\pi) = 2$. Letting $\omega = 0$, we have

$$\hat{d}(0)\hat{\tilde{d}}^*(0) + \hat{d}(\pi)\hat{\tilde{d}}^*(\pi) = 2.$$

From this, we can deduce that $\hat{\tilde{d}}(0) = 0 \Rightarrow \hat{\tilde{d}}(\pi) \neq 0$. From the biorthogonality condition (4.31b), we have

$$\hat{c}(\omega)\hat{\tilde{d}}^*(\omega) + \hat{c}(\omega+\pi)\hat{\tilde{d}}^*(\omega+\pi) = 0. \tag{4.47}$$

Letting $\omega = 0$, this becomes

$$\hat{c}(0)\hat{\tilde{d}}^*(0) + \hat{c}(\pi)\hat{\tilde{d}}^*(\pi) = 0.$$

In combination with the above results, we can deduce that $\hat{c}(\pi) = 0 \Rightarrow \hat{\tilde{d}}(0) = 0$ and $\hat{\tilde{d}}(0) = 0 \Rightarrow \hat{c}(\pi) = 0$. This shows the equivalence for $\eta = 1$. To show the equivalence holds for higher order η, we differentiate (4.47) and use a similar argument as above.

$3 \Rightarrow 2$. Consider the function $f(t) = t^p\phi(t)$, where $\hat{f}(\omega) = j^p\hat{\phi}^{(p)}(\omega)$. Using the Poisson summation formula, we can write

$$\sum_{n\in\mathbb{Z}} f(t+n) = \sum_{n\in\mathbb{Z}} \hat{f}(2\pi n)e^{j2\pi nt}$$

which gives us

$$\sum_{n\in\mathbb{Z}} (t+n)^p\phi(t+n) = \sum_{n\in\mathbb{Z}} j^p\hat{\phi}^{(p)}(2\pi n)e^{j2\pi nt}. \tag{4.48}$$

From the Strang-Fix condition and moment properties of the Fourier transform, we can simplify the preceding equation to obtain

$$\mu_p = \sum_{l\in\mathbb{Z}} (t-l)^p\phi(t-l) \quad \text{for } 0 \leq p < \eta.$$

Now, let us consider $q_0(t)$. We trivially have that

$$q_0(t) = \mu_0. \tag{4.49}$$

Now let us consider $q_p(t)$ for $p > 0$. For $1 \le p < \eta$, we have

$$\begin{aligned}
\mu_p &= \sum_{n\in\mathbb{Z}} (t-n)^p \phi(t-n) \\
&= \sum_{n\in\mathbb{Z}} \sum_{k=0}^{p} \binom{p}{k} t^{p-k} (-n)^k \phi(t-n) \\
&= \sum_{n\in\mathbb{Z}} \sum_{k=0}^{p} (-1)^k \binom{p}{k} t^{p-k} n^k \phi(t-n) \\
&= \sum_{n\in\mathbb{Z}} \left(\sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} n^k + (-1)^p n^p \right) \phi(t-n) \\
&= \sum_{n\in\mathbb{Z}} \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} n^k \phi(t-n) + \sum_{n\in\mathbb{Z}} (-1)^p n^p \phi(t-n) \\
&= \sum_{k=0}^{p-1} \sum_{n\in\mathbb{Z}} (-1)^k \binom{p}{k} t^{p-k} n^k \phi(t-n) + (-1)^p \sum_{n\in\mathbb{Z}} n^p \phi(t-n) \\
&= \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} \sum_{n\in\mathbb{Z}} n^k \phi(t-n) + (-1)^p \sum_{n\in\mathbb{Z}} n^p \phi(t-n) \\
&= \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} q_k(t) + (-1)^p q_p(t).
\end{aligned}$$

Rearranging, we have

$$\mu_p - \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} q_k(t) = (-1)^p q_p(t).$$

Rearranging further, we obtain

$$\begin{aligned}
q_p(t) &= (-1)^p \mu_p - (-1)^p \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} t^{p-k} q_k(t) \\
&= (-1)^p \mu_p - \sum_{k=0}^{p-1} (-1)^{p+k} \binom{p}{k} t^{p-k} q_k(t). \qquad (4.50)
\end{aligned}$$

Combining (4.49) and (4.50), we have

$$q_p(t) = \begin{cases} (-1)^p \mu_p - \sum_{k=0}^{p-1} (-1)^{p+k} \binom{p}{k} t^{p-k} q_k(t) & \text{for } 1 \le p < \eta \\ \mu_0 & \text{for } p = 0. \end{cases} \qquad (4.51)$$

Thus, we have a recursive formula for computing $q_p(t)$. Observe that, for $0 \le p < \eta$, $q_p(t)$ is a pth degree polynomial. For example, computing $q_p(t)$ for the first few p, we have

$$q_0(t) = \mu_0,$$

$$\begin{aligned}
q_1(t) &= (-1)^1 \mu_1 - \sum_{k=0}^{0} (-1)^{1+k} \binom{1}{k} t^{1-k} q_k(t) \\
&= -\mu_1 - [-t q_0(t)] \\
&= \mu_0 t - \mu_1, \quad \text{and}
\end{aligned}$$

$$\begin{aligned}
q_2(t) &= (-1)^2\mu_2 - \sum_{k=0}^{1}(-1)^{2+k}\binom{2}{k}t^{2-k}q_k(t) \\
&= \mu_2 - (t^2q_0(t) - 2tq_1(t)) \\
&= \mu_2 - t^2q_0(t) + 2tq_1(t) \\
&= \mu_2 - \mu_0t^2 + 2t(-\mu_1 + \mu_0t) \\
&= \mu_0t^2 - 2\mu_1t + \mu_2.
\end{aligned}$$

□

From (4.51), we can derive an expression for t^p in terms of ϕ for $p < \eta$. For example, one can show that

$$t^p = \sum_{n\in\mathbb{Z}} a_p[n]\phi(t-n), \tag{4.52a}$$

where

$$a_p[n] = \begin{cases} \frac{1}{\mu_0} & \text{for } p = 0 \\ \frac{1}{\mu_0}n + \frac{\mu_1}{\mu_0^2} & \text{for } p = 1 \\ \frac{1}{\mu_0}n^2 + \frac{2\mu_1}{\mu_0^2}n + \frac{-\mu_2\mu_0+2\mu_1^2}{\mu_0^3} & \text{for } p = 2. \end{cases} \tag{4.52b}$$

Since we are clearly interested in the number of zeros that $\hat{c}(\omega)$ has at $\omega = \pi$, it is convenient to have a method to quickly determine this quantity. To this end, we will often find the result below to be helpful.

Theorem 4.28 (Sum rule)**.** *Let c be a sequence. Then, $\hat{c}(\omega)$ has a ηth order zero at $\omega = \pi$ if and only if*

$$\sum_{n\in\mathbb{Z}}(-1)^n n^k c[n] = 0 \textit{ for } k = 0,1,\ldots,\eta-1.$$

This is often referred to as the ***sum rule*** *of order η.*

Proof. The Fourier transform $\hat{c}$ of c is given by $\hat{c}(\omega) = \sum_{n\in\mathbb{Z}} c[n]e^{-j\omega n}$. Differentiating this equation k times, we obtain

$$\begin{aligned}
\hat{c}^{(k)}(\omega) &= \sum_{n\in\mathbb{Z}}(-jn)^k c[n]e^{-j\omega n} \\
&= (-j)^k \sum_{n\in\mathbb{Z}} n^k c[n]e^{-j\omega n}.
\end{aligned}$$

Evaluating $\hat{c}^{(k)}(\omega)$ at $\omega = \pi$, we have

$$\hat{c}^{(k)}(\pi) = (-j)^k \sum_{n\in\mathbb{Z}}(-1)^n n^k c[n].$$

Now, we observe that $\hat{c}(\omega)$ having a ηth order zero at $\omega = \pi$ is (by definition) equivalent to $\hat{c}^{(k)}(\pi) = 0$ for $k = 0,1,\ldots,\eta-1$. Thus, we obtain the condition stated in the theorem by observing that

$$c^{(k)}(\pi) = 0 \quad \Leftrightarrow \quad (-j)^k \sum_{n\in\mathbb{Z}}(-1)^n n^k c[n] = 0 \quad \Leftrightarrow \quad \sum_{n\in\mathbb{Z}}(-1)^n n^k c[n] = 0.$$

□

Using the above theorem, we can determine the approximation order of a scaling function relatively easily. This is illustrated by the example below.

Example 4.21. Let ϕ and $\tilde{\phi}$ denote the scaling and dual scaling functions of a wavelet system, respectively. Suppose that ϕ and $\tilde{\phi}$ have scaling equation coefficient sequences c and $\tilde{c}$, respectively, where

$$c_{-2} = c_3 = -\tfrac{1}{8\sqrt{2}}, \quad c_{-1} = c_2 = \tfrac{1}{8\sqrt{2}}, \quad c_0 = c_1 = \tfrac{1}{\sqrt{2}}, \quad \text{and}$$
$$\tilde{c}_{-1} = \tilde{c}_0 = \tfrac{1}{2\sqrt{2}}.$$

Determine the approximation order of each of the scaling functions.

Solution. We employ the sum rule to determine the approximation order. First, we consider the scaling function ϕ. We check whether c satisfies a sum rule of order one. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n c_n &= (1)(-\tfrac{1}{8\sqrt{2}}) + (-1)(\tfrac{1}{8\sqrt{2}}) + (1)(\tfrac{1}{\sqrt{2}}) + (-1)(\tfrac{1}{\sqrt{2}}) + (1)(\tfrac{1}{8\sqrt{2}}) + (-1)(-\tfrac{1}{8\sqrt{2}})\\ &= -\tfrac{1}{8\sqrt{2}} - \tfrac{1}{8\sqrt{2}} + \tfrac{1}{\sqrt{2}} - \tfrac{1}{\sqrt{2}} + \tfrac{1}{8\sqrt{2}} + \tfrac{1}{8\sqrt{2}}\\ &= 0.\end{aligned}$$

We check whether c satisfies a sum rule of order two. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n n c_n &= (1)(-2)(-\tfrac{1}{8\sqrt{2}}) + (-1)(-1)(\tfrac{1}{8\sqrt{2}}) + (-1)(1)(\tfrac{1}{\sqrt{2}}) + (1)(2)(\tfrac{1}{8\sqrt{2}}) + (-1)(3)(-\tfrac{1}{8\sqrt{2}})\\ &= \tfrac{2}{8\sqrt{2}} + \tfrac{1}{8\sqrt{2}} - \tfrac{1}{\sqrt{2}} + \tfrac{2}{8\sqrt{2}} + \tfrac{3}{8\sqrt{2}}\\ &= 0.\end{aligned}$$

We check whether c satisfies a sum rule of order three. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n n^2 c_n &= (1)(-2)^2(-\tfrac{1}{8\sqrt{2}}) + (-1)(-1)^2(\tfrac{1}{8\sqrt{2}}) + (-1)(1)^2(\tfrac{1}{\sqrt{2}}) + (1)(2)^2(\tfrac{1}{8\sqrt{2}}) + (-1)(3)^2(-\tfrac{1}{8\sqrt{2}})\\ &= -\tfrac{4}{8\sqrt{2}} - \tfrac{1}{8\sqrt{2}} - \tfrac{1}{\sqrt{2}} + \tfrac{4}{8\sqrt{2}} + \tfrac{9}{8\sqrt{2}}\\ &= 0.\end{aligned}$$

We check whether c satisfies a sum rule of order four. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n n^3 c_n &= (1)(-2)^3(-\tfrac{1}{8\sqrt{2}}) + (-1)(-1)^3(\tfrac{1}{8\sqrt{2}}) + (-1)(1)^3(\tfrac{1}{\sqrt{2}}) + (1)(2)^3(\tfrac{1}{8\sqrt{2}}) + (-1)(3)^3(-\tfrac{1}{8\sqrt{2}})\\ &= \tfrac{1}{\sqrt{2}} + \tfrac{1}{8\sqrt{2}} - \tfrac{1}{\sqrt{2}} + \tfrac{1}{\sqrt{2}} + \tfrac{27}{8\sqrt{2}}\\ &= \tfrac{9}{2\sqrt{2}}\\ &\neq 0.\end{aligned}$$

Thus, c satisfies a third-order sum rule. Therefore, the approximation order of ϕ is three. That is, ϕ can be used to represent polynomials of degree two or less (i.e., quadratic polynomials).

Second, we consider the dual scaling function $\tilde{\phi}$. We check whether $\tilde{c}$ satisfies a sum rule of order one. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n \tilde{c}_n &= (-1)(\tfrac{1}{2\sqrt{2}}) + (1)(\tfrac{1}{2\sqrt{2}})\\ &= 0.\end{aligned}$$

We check whether $\tilde{c}$ satisfies a sum rule of order two. We have

$$\begin{aligned}\sum_{n\in\mathbb{Z}}(-1)^n n\tilde{c}_n &= (-1)(-1)(\tfrac{1}{2\sqrt{2}})\\ &\neq 0.\end{aligned}$$

Thus, $\tilde{c}$ satisfies a first-order sum rule. The approximation order of $\tilde{\phi}$ is one. That is, $\tilde{\phi}$ can be used to represent constant polynomial functions.

□

Using our knowledge of the approximation properties of scaling functions, we can represent a given polynomial in terms of the integer shifts of a given scaling function. Below, we provide some examples to demonstrate how this can be done.

Example 4.22. The Daubechies-2 scaling function ϕ has the scaling equation coefficient sequence c given by

$$c_{-1} = \tfrac{1+\sqrt{3}}{4\sqrt{2}}, \; c_0 = \tfrac{3+\sqrt{3}}{4\sqrt{2}}, \; c_1 = \tfrac{3-\sqrt{3}}{4\sqrt{2}}, \text{ and } c_2 = \tfrac{1-\sqrt{3}}{4\sqrt{2}}.$$

One can show that $\hat{c}$ has a second-order zero at π. Suppose that ϕ has been normalized such that its first two moments are given by

$$\mu_0 = 1 \quad \text{and} \quad \mu_1 = \tfrac{1-\sqrt{3}}{2}.$$

(Note: These moments are for $\operatorname{supp}\phi \subset [-1,2]$.) Using the integer shifts of ϕ, represent the function $f(t) = 3t-1$ on the interval $[0,2]$ (with as few terms as possible).

Solution. Since $\hat{c}$ has a second-order zero at π, the integer shifts of ϕ can represent linear (and constant) functions exactly. From the support of c, we can infer that $\operatorname{supp}\phi \subset [-1,2]$. Furthermore, since $c_{-1} \neq \frac{1}{\sqrt{2}}$ and $c_2 \neq \frac{1}{\sqrt{2}}$, $\phi(-1) = 0$ and $\phi(2) = 0$. Since $\operatorname{supp}\phi(t-k) \subset [-1+k, 2+k]$ and $\phi(-1) = \phi(2) = 0$, $\phi(t-k)$ has partial support in $[0,2]$ for $k \in \{-1,0,1,2\}$. Thus, we seek an expansion of the form

$$f(t) = \sum_{k=-1}^{2} a_k \phi(t-k) \quad \text{for } t \in [0,2].$$

Using (4.52b), we have that the coefficient sequence a is given by

$$\begin{aligned} a_n &= 3\left(\tfrac{1}{\mu_0}n + \tfrac{\mu_1}{\mu_0^2}\right) - \left(\tfrac{1}{\mu_0}\right) \\ &= 3\left(n + \tfrac{1-\sqrt{3}}{2}\right) - 1 \\ &= \tfrac{6n+3-3\sqrt{3}-2}{2} \\ &= \tfrac{6n+1-3\sqrt{3}}{2}. \end{aligned}$$

In other words, we have that

$$f(t) = \tfrac{-5-3\sqrt{3}}{2}\phi(t+1) + \tfrac{1-3\sqrt{3}}{2}\phi(t) + \tfrac{7-3\sqrt{3}}{2}\phi(t-1) + \tfrac{13-3\sqrt{3}}{2}\phi(t-2),$$

for $t \in [0,2]$. A plot of the resulting function is given in Figure 4.19. □

Example 4.23. Consider the scaling function ϕ with the scaling coefficient sequence c_k, where

$$(c_{-2}, c_{-1}, c_0, c_1, c_2, c_3) = \left(-\tfrac{1}{8\sqrt{2}}, \tfrac{1}{8\sqrt{2}}, \tfrac{8}{8\sqrt{2}}, \tfrac{8}{8\sqrt{2}}, \tfrac{1}{8\sqrt{2}}, -\tfrac{1}{8\sqrt{2}}\right)$$

and the zeroth moment of ϕ is one. One can show that $\hat{c}$ has a third-order zero at π. Represent $f(t) = 10t^2$ on the interval $I = [-2,2]$ using integer shifts of ϕ.

Solution. Since $\hat{c}$ has a third-order zero at π, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ can locally represent quadratic functions exactly. From the support of c, we can infer that $\operatorname{supp}\phi \subset [-2,3]$. Furthermore, since $c_{-2} \neq \frac{1}{\sqrt{2}}$ and $c_3 \neq \frac{1}{\sqrt{2}}$, $\phi(-2) = 0$ and $\phi(3) = 0$. Since $\operatorname{supp}\phi(t-k) \subset [-2+k, 3+k]$ and $\phi(-2) = \phi(3) = 0$, $\phi(t-k)$ has partial support in $[-2,2]$ for $k \in \{-4,-3,-2,-1,0,1,2,3\}$. Thus, we seek an expansion of the form

$$f(t) = \sum_{k=-4}^{3} a_k \phi(t-k) \quad \text{for } t \in [-2,2].$$

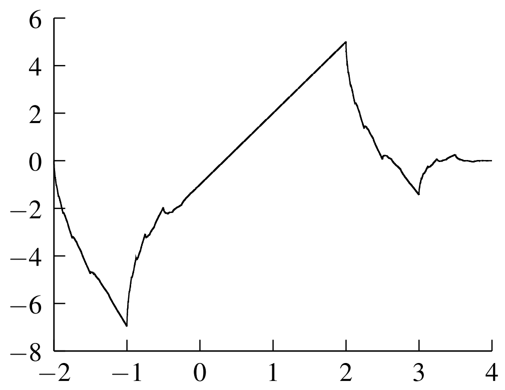


Figure 4.19: Representing linear function on interval using Daubechies-2 scaling function.

Let m_k and μ_k denote the kth moments of c and ϕ, respectively. Using (4.52b), we have that the coefficient sequence a is given by

$$a_n = 10\left(\tfrac{1}{\mu_0}n^2 + \tfrac{2\mu_1}{\mu_0^2}n + \tfrac{-\mu_2\mu_0+2\mu_1^2}{\mu_0^3}\right).$$

Now, we compute the necessary moments m_k. We have

$$\begin{aligned}
m_0 &= \sqrt{2},\\
m_1 &= \sum_{k\in\mathbb{Z}} kc_k = (-2)\left(-\tfrac{1}{8\sqrt{2}}\right) + (-1)\left(\tfrac{1}{8\sqrt{2}}\right) + (1)\left(\tfrac{8}{8\sqrt{2}}\right) + (2)\left(\tfrac{1}{8\sqrt{2}}\right) + (3)\left(-\tfrac{1}{8\sqrt{2}}\right)\\
&= \tfrac{2-1+8+2-3}{8\sqrt{2}} = \tfrac{8}{8\sqrt{2}}\\
&= \tfrac{1}{\sqrt{2}}, \quad \text{and}\\
m_2 &= \sum_{k\in\mathbb{Z}} k^2c_k = (-2)^2\left(-\tfrac{1}{8\sqrt{2}}\right) + (-1)^2\left(\tfrac{1}{8\sqrt{2}}\right) + (1)^2\left(\tfrac{8}{8\sqrt{2}}\right) + (2)^2\left(\tfrac{1}{8\sqrt{2}}\right) + (3)^2\left(-\tfrac{1}{8\sqrt{2}}\right)\\
&= \tfrac{-4+1+8+4-9}{8\sqrt{2}}\\
&= 0.
\end{aligned}$$

Next, we compute the necessary moments μ_k. We have

$$\begin{aligned}
\mu_1 &= \tfrac{1}{\sqrt{2}(2-1)}\sum_{q=0}^{0}\binom{1}{q}m_{1-q}\mu_q = \tfrac{1}{\sqrt{2}}\binom{1}{0}m_1\mu_0\\
&= \tfrac{1}{\sqrt{2}}(\tfrac{1}{\sqrt{2}})(1)\\
&= \tfrac{1}{2}, \quad \text{and}\\
\mu_2 &= \tfrac{1}{\sqrt{2}(2^2-1)}\sum_{q=0}^{1}\binom{2}{q}m_{2-q}\mu_q = \tfrac{1}{3\sqrt{2}}\left[\binom{2}{0}m_2\mu_0 + \binom{2}{1}m_1\mu_1\right]\\
&= \tfrac{1}{3\sqrt{2}}\left[-\tfrac{1}{2\sqrt{2}} + 2\left(\tfrac{1}{\sqrt{2}}\right)\left(\tfrac{1}{2}\right)\right]\\
&= \tfrac{1}{6}.
\end{aligned}$$

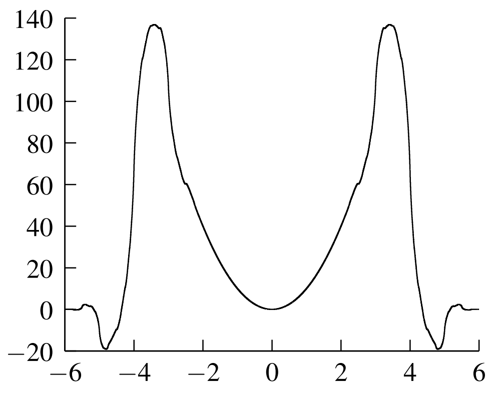


Figure 4.20: Representing quadratic function on interval using scaling function.

Combining the above results, we have

$$\begin{aligned} a_n &= 10\left(n^2+n+(-\tfrac{1}{6})+2(\tfrac{1}{2})^2\right) \\ &= 10\left(n^2+n+\tfrac{1}{3}\right) \\ &= 10\left(\tfrac{3n^2+3n+1}{3}\right). \end{aligned}$$

Thus, we have that

$$f(t) = \tfrac{370}{3}\phi(t+4)+\tfrac{190}{3}\phi(t+3)+\tfrac{70}{3}\phi(t+2)+\tfrac{10}{3}\phi(t+1)+\tfrac{10}{3}\phi(t)+\tfrac{70}{3}\phi(t-1)+\tfrac{190}{3}\phi(t-2)+\tfrac{370}{3}\phi(t-3),$$

for $t \in [-2,2]$. A plot of the resulting approximation is shown in Figure 4.20. □

4.2.17 Determination of Scaling and Wavelet Functions

Most of the time, it is not possible to derive a closed-form expression for a solution to a refinement equation. For this reason, we usually need to compute such solutions numerically. In what follows, we look at a few methods for determining solutions to refinement equations.

4.2.17.1 Spectral Method

Conceptually, one of the simplest methods for computing the scaling function is to use the infinite product formula for the Fourier transform of the scaling function. Recall that we can write

$$\hat{\phi}(\omega) = \hat{\phi}(0)\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2^p).$$

So, in principle, we can approximate the above infinite product by its first k factors. Then, we can take the inverse Fourier transform of the result in order to find an approximation to $\phi(t)$. While this approach works, it is neither very exact nor particularly fast. This leads us to consider other methods for determining the scaling function.

4.2.17.2 Eigenmethod

Another technique for solving refinement equations determines the solution at dyadic points. First, we solve an eigenproblem to find the solution at integers. Then, we employ the scaling equation to compute the solution at half integers, quarter integers, and so on. This approach is both exact and fast. It can be used to evaluate the solution at any dyadic point.

Consider the scaling equation

$$\phi(t) = \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi(2t - k).$$

Without loss of generality, we assume

$$c_n = 0 \text{ for } n < 0 \text{ or } n > N, \; c_0 \neq 0, \quad \text{and} \quad c_N \neq 0.$$

(If ϕ has support on a different interval, we can always shift it so it has support on $[0,N]$.) The form of c_n implies $\operatorname{supp}\phi(t) \subset [0,N]$.

We evaluate the scaling equation at integer n to obtain

$$\phi(n) = \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi(2n - k).$$

Now, we employ a change of variable. Let $k' = 2n - k$ so that $k = 2n - k'$. Applying this change of variable and dropping the primes, we obtain

$$\phi(n) = \sqrt{2} \sum_{k \in \mathbb{Z}} c_{2n-k} \phi(k).$$

Since $\operatorname{supp}\phi \subset [0,N]$, $\phi(k)$ can only be nonzero at the integers $k \in \{0,1,\ldots,N\}$, and we have

$$\phi(n) = \sqrt{2} \sum_{k=0}^{N} c_{2n-k} \phi(k).$$

Writing the preceding equation for $n = 0,1,\ldots,N$ in matrix form, we have

$$\underbrace{\begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \vdots \\ \phi(N-1) \\ \phi(N) \end{bmatrix}}_{\boldsymbol{\phi}} = \underbrace{\left(\sqrt{2} \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ c_2 & c_1 & c_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ c_4 & c_3 & c_2 & c_1 & c_0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & c_N & c_{N-1} & c_{N-2} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & c_N \end{bmatrix} \right)}_{\boldsymbol{M}} \underbrace{\begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \vdots \\ \phi(N-1) \\ \phi(N) \end{bmatrix}}_{\boldsymbol{\phi}} \tag{4.53}$$

which can be written more compactly as

$$\boldsymbol{\phi} = \boldsymbol{M}\boldsymbol{\phi},$$

where $[\boldsymbol{M}]_{p,q} = \sqrt{2} c_{2p-q}$. Finding a solution to this equation is equivalent to solving the eigenproblem

$$\boldsymbol{M}\boldsymbol{v} = \lambda \boldsymbol{v},$$

where $\lambda = 1$. We must find an eigenvector $\boldsymbol{v}$ of $\boldsymbol{M}$ with a corresponding eigenvalue λ of one. If no such eigenvector exists, then the refinement equation has no solution. Suppose that such a solution does exist. Then, the eigenvector $\boldsymbol{v}$ gives us $\boldsymbol{\phi}$. From $\boldsymbol{\phi}$, we trivially obtain $\phi(t)$ at the integers.

Now, let us return to the refinement equation. From the refinement equation, we can evaluate ϕ at a point $n/2^p$ as follows

$$\begin{aligned} \phi(n/2^p) &= \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi(2[n/2^p] - k) \\ &= \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi\left(\frac{n - 2^{p-1}k}{2^{p-1}} \right). \end{aligned} \tag{4.54}$$

Letting $p = 1$ in (4.54), we have

$$\phi(n/2) = \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi(n-k).$$

The right-hand side of the preceding equation depends on $\phi(t)$ evaluated only at integers. The left hand side gives us $\phi(t)$ at any half integer. So, we can calculate $\phi(t)$ at all of the half integers. Letting $p = 2$ in (4.54), we have

$$\phi(n/4) = \sqrt{2} \sum_{k \in \mathbb{Z}} c_k \phi\left(\frac{n-2k}{2}\right).$$

The right-hand side of the preceding equation depends only on $\phi(t)$ evaluated at half integers (which are known quantities). The left-hand side gives us $\phi(t)$ at quarter integers. So, we can calculate $\phi(t)$ at all of the quarter integers. We can repeat this process ad infinitum to compute $\phi(t)$ at all dyadic points.

Observe that if a solution to a refinement equation exists, the solution is not unique. If $\boldsymbol{v}$ is a solution, so too is $a\boldsymbol{v}$ for any $a \in \mathbb{C}$. The solution of a refinement equation is only determined up to a scale factor. Also, the possibility exists that the eigenvalue of one may be repeated. In this case, we can obtain solutions for $\boldsymbol{v}$ that are not related by a scale factor. For example, the refinement equation associated with the Haar scaling function has a repeated eigenvalue of one. (We obtain the distinct eigenvectors $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$ and $\begin{bmatrix} 0 & 1 \end{bmatrix}^T$, leading to the solutions $\phi = \chi_{[0,1)}$ and $\phi = \chi_{(0,1]}$.)

We can also make another interesting observation. Recall that c_0 and c_N are assumed to be nonzero. From (4.53), we have $\phi(0) = \sqrt{2}c_0\phi(0)$ and $\phi(N) = \sqrt{2}c_N\phi(N)$. This implies, however, that $\phi(0) = 0$ if $c_0 \neq 1/\sqrt{2}$ and $\phi(N) = 0$ if $c_N \neq 1/\sqrt{2}$.

If $\phi(0)$ and $\phi(N)$ known to be zero, we can reduce the number of equations/unknowns in (4.53). This process yields

$$\boldsymbol{\phi}' = \boldsymbol{M}'\boldsymbol{\phi}',$$

where

$$\boldsymbol{\phi}' = \begin{bmatrix} \phi(1) & \phi(2) & \cdots & \phi(N-1) \end{bmatrix}^T \quad \text{and}$$

$$\boldsymbol{M}' = \sqrt{2} \begin{bmatrix} c_1 & c_0 & 0 & 0 & \cdots & 0 & 0 \\ c_3 & c_2 & c_1 & c_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & c_N & c_{N-1} \end{bmatrix}.$$

In a similar manner, we can compute the derivatives of ϕ provided that they exist. We begin by differentiating the scaling equation to obtain

$$(D\phi)(t) = 2\sqrt{2} \sum_{k \in \mathbb{Z}} c[k](D\phi)(2t-k).$$

We evaluate $D\phi(t)$ at the integers and write the resulting equations in matrix form to obtain

$$\boldsymbol{\phi}' = 2\boldsymbol{M}\boldsymbol{\phi}'$$

where

$$\boldsymbol{\phi}' = \begin{bmatrix} \phi'(0) & \phi'(1) & \cdots & \phi'(N) \end{bmatrix}^T.$$

Now, we observe that $\boldsymbol{\phi}'$ is an eigenvector of $\boldsymbol{M}'$ corresponding to an eigenvalue of $\frac{1}{2}$. Having determined $\phi'(t)$ at the integers, we can then compute $\phi'(t)$ at the half integers, quarter integers, and so on. One can show [16, p. 37] that the correct normalization of the nth derivative is given by

$$\sum_{k \in \mathbb{Z}} k^n D^n \phi(k) = (-1)^n n! \mu_0.$$

Below, we provide some examples to demonstrate the use of the eigenmethod approach for solving refinement equations.

Example 4.24. Consider the refinement equation

$$\phi(t) = \sqrt{2}\sum_{k=0}^{2} c_k \phi(2t-k)$$

where

$$c_0 = \tfrac{1}{2\sqrt{2}}, \quad c_1 = \tfrac{1}{\sqrt{2}}, \quad \text{and} \quad c_2 = \tfrac{1}{2\sqrt{2}}.$$

(Note that ϕ is a translated version of the primal scaling function from the Le Gall 5/3 wavelet system of Example 4.15.) Determine whether or not a solution to this refinement equation exists. If the equation does have a solution, let ϕ_0 denote that particular normalization of the solution with a zeroth moment of unity, and evaluate ϕ_0 at all half integers (i.e., $\frac{1}{2}\mathbb{Z}$).

Solution. We have the eigenproblem

$$\boldsymbol{\phi} = \boldsymbol{M}\boldsymbol{\phi}$$

where

$$\begin{aligned}\boldsymbol{M} &= \sqrt{2}\begin{bmatrix} c_0 & 0 & 0 \\ c_2 & c_1 & c_0 \\ 0 & 0 & c_2 \end{bmatrix} \\ &= \sqrt{2}\begin{bmatrix} \frac{1}{2\sqrt{2}} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ 0 & 0 & \frac{1}{2\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \end{bmatrix}, \quad \text{and}\end{aligned}$$

$$\boldsymbol{\phi} = \begin{bmatrix} \phi(0) & \phi(1) & \phi(2) \end{bmatrix}^T.$$

Solving this eigenproblem yields

$$\boldsymbol{\phi} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T.$$

Or alternatively, we can observe that $c_0, c_2 \neq 1/\sqrt{2}$, which implies that $\phi(0) = \phi(2) = 0$. This allows us to reduce the eigenproblem to the simpler one given by

$$\boldsymbol{M}'\boldsymbol{\phi}' = \boldsymbol{\phi}'$$

where

$$\boldsymbol{M}' = [1] \quad \text{and} \quad \boldsymbol{\phi}' = [\phi(1)].$$

By inspection, we have

$$\boldsymbol{\phi}' = 1.$$

Observe that this is the desired normalization of the solution sought (since, the value of $\boldsymbol{\phi}'$ and (4.35) together imply that ϕ has a zeroth moment equal to one). Thus, we have

$$\phi(1) = 1 \quad \text{and} \quad \phi(t) = 0 \ \text{for}\ t \in \mathbb{Z} \setminus \{1\}.$$

We can then use the scaling equation to determine $\phi(t)$ at the half integer points. In particular, from (4.54) with $p = 1$, we have

$$\begin{aligned}\phi(\tfrac{1}{2}) &= \sqrt{2}\sum_{k\in\mathbb{Z}} c_k\phi(1-k)\\ &= \sqrt{2}[c_0\phi(1)]\\ &= \sqrt{2}[\tfrac{1}{2\sqrt{2}}(1)]\\ &= \tfrac{1}{2} \quad \text{and}\end{aligned}$$

$$\begin{aligned}\phi(\tfrac{3}{2}) &= \sqrt{2}\sum_{k\in\mathbb{Z}} c_k\phi(3-k)\\ &= \sqrt{2}[c_2\phi(1)]\\ &= \sqrt{2}[\tfrac{1}{2\sqrt{2}}(1)]\\ &= \tfrac{1}{2}.\end{aligned}$$

So, we have

$$\phi(\tfrac{1}{2}) = \phi(\tfrac{3}{2}) = \tfrac{1}{2}.$$

□

Example 4.25. Consider the refinement equation

$$\phi(t) = \sqrt{2}\sum_{k=0}^{3} c_k\phi(2t-k)$$

where

$$c_0 = \tfrac{1+\sqrt{3}}{4\sqrt{2}},\ c_1 = \tfrac{3+\sqrt{3}}{4\sqrt{2}},\ c_2 = \tfrac{3-\sqrt{3}}{4\sqrt{2}},\ \text{and}\ c_3 = \tfrac{1-\sqrt{3}}{4\sqrt{2}}.$$

(Note that ϕ is the scaling function of the Daubechies-2 wavelet system of Example 4.14.) Determine whether or not a solution to this refinement equation exists. If the equation does have a solution, let ϕ_0 denote the particular normalization of the solution with a zeroth moment equal to unity, and evaluate ϕ_0 at all half integers (i.e., $\frac{1}{2}\mathbb{Z}$).

Solution. Since $c_0 \neq \frac{1}{\sqrt{2}}$ and $c_3 \neq \frac{1}{\sqrt{2}}$, we have $\phi(0) = \phi(3) = 0$. Consequently, we can reduce this problem to the simpler one of finding $\phi(1)$ and $\phi(2)$. We have the eigenproblem

$$\boldsymbol{\phi} = \boldsymbol{M}\boldsymbol{\phi}$$

where

$$\begin{aligned}\boldsymbol{M} &= \sqrt{2}\begin{bmatrix} c_1 & c_0\\ c_3 & c_2\end{bmatrix}\\ &= \sqrt{2}\begin{bmatrix} \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{1+\sqrt{3}}{4\sqrt{2}}\\ \frac{1-\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}}\end{bmatrix}\\ &= \begin{bmatrix} \frac{3+\sqrt{3}}{4} & \frac{1+\sqrt{3}}{4}\\ \frac{1-\sqrt{3}}{4} & \frac{3-\sqrt{3}}{4}\end{bmatrix} \quad \text{and}\end{aligned}$$

$$\boldsymbol{\phi} = \begin{bmatrix}\phi(1) & \phi(2)\end{bmatrix}^T.$$

Solving this eigenproblem (e.g., by using Maple), we obtain

$$\boldsymbol{\phi} = \begin{bmatrix} \frac{1+\sqrt{3}}{2} & \frac{1-\sqrt{3}}{2} \end{bmatrix}^T.$$

Now, we observe that $\boldsymbol{\phi}$ corresponds to the desired normalization of the solution sought (due to (4.35) with $t=0$). Thus, we have

$$\phi(1) = \tfrac{1+\sqrt{3}}{2} \approx 1.366025 \quad \text{and} \quad \phi(2) = \tfrac{1-\sqrt{3}}{2} \approx -0.366025.$$

Now, using (4.54) with $p=1$, we evaluate ϕ at the half integers to obtain

$$\begin{aligned} \phi(\tfrac{1}{2}) &= \sqrt{2}\sum_{k\in\mathbb{Z}} c_k\phi(1-k) \\ &= \sqrt{2}[c_0\phi(1)] \\ &= \sqrt{2}[\tfrac{1+\sqrt{3}}{4\sqrt{2}}(\tfrac{1+\sqrt{3}}{2})] \\ &= \tfrac{(1+\sqrt{3})^2}{8} \\ &= \tfrac{1+2\sqrt{3}+3}{8} \\ &= \tfrac{4+2\sqrt{3}}{8} \\ &= \tfrac{2+\sqrt{3}}{4} \\ &\approx 0.93301, \end{aligned}$$

$$\begin{aligned} \phi(\tfrac{3}{2}) &= \sqrt{2}\sum_{k\in\mathbb{Z}} c_k\phi(3-k) \\ &= \sqrt{2}[c_1\phi(2)+c_2\phi(1)] \\ &= \sqrt{2}[\tfrac{3+\sqrt{3}}{4\sqrt{2}}(\tfrac{1-\sqrt{3}}{2})+\tfrac{3-\sqrt{3}}{4\sqrt{2}}(\tfrac{1+\sqrt{3}}{2})] \\ &= \tfrac{(3+\sqrt{3})(1-\sqrt{3})+(3-\sqrt{3})(1+\sqrt{3})}{8} \\ &= \tfrac{3-3\sqrt{3}+\sqrt{3}-3+3+3\sqrt{3}-\sqrt{3}-3}{8} \\ &= 0, \quad \text{and} \end{aligned}$$

$$\begin{aligned} \phi(\tfrac{5}{2}) &= \sqrt{2}\sum_{k\in\mathbb{Z}} c_k\phi(5-k) \\ &= \sqrt{2}[c_3\phi(2)] \\ &= \sqrt{2}[\tfrac{1-\sqrt{3}}{4\sqrt{2}}(\tfrac{1-\sqrt{3}}{2})] \\ &= \tfrac{(1-\sqrt{3})^2}{8} \\ &= \tfrac{1-2\sqrt{3}+3}{8} \\ &= \tfrac{4-2\sqrt{3}}{8} \\ &= \tfrac{2-\sqrt{3}}{4} \\ &\approx 0.066987. \end{aligned}$$

So, we have

$$\phi(\tfrac{1}{2}) = \tfrac{2+\sqrt{3}}{4}, \quad \phi(\tfrac{3}{2}) = 0, \quad \text{and} \quad \phi(\tfrac{5}{2}) = \tfrac{2-\sqrt{3}}{4}.$$

□

4.2.17.3 Cascade Algorithm

Another approach to determining the scaling function involves repeated application of a filter bank. This method is known as the cascade algorithm. This algorithm is fast. It is not exact but often yields a good approximation. This approach is formalized by the theorem below.

Theorem 4.29 (Cascade algorithm)**.** *Suppose that we have a refinement equation*

$$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} c[n]\phi(2t-n). \tag{4.55}$$

Define the iterative process

$$\phi^{(k+1)}(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} c[n]\phi^{(k)}(2t-n) \tag{4.56}$$

where we choose $\phi^{(0)}(t)$ such that

$$\int_{-\infty}^{\infty}\phi^{(0)}(t)dt = \mu_0 \neq 0.$$

If this iterative process converges to a fixed point, this fixed point is a solution to (4.55) normalized such that $\int_{-\infty}^{\infty}\phi(t)dt = \mu_0$.

Proof. Taking the Fourier transform of both sides of (4.56), we obtain

$$\begin{aligned}
\hat{\phi}^{(k+1)}(\omega) &= \tfrac{1}{2}\sqrt{2}\sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega/2}\hat{\phi}^{(k)}(\omega/2) \\
&= \tfrac{1}{\sqrt{2}}\sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega/2}\hat{\phi}^{(k)}(\omega/2) \\
&= \tfrac{1}{\sqrt{2}}\hat{c}(\omega/2)\hat{\phi}^{(k)}(\omega/2)
\end{aligned}$$

where $\hat{c}(\omega) = \sum_{n\in\mathbb{Z}} c[n]e^{-jn\omega}$ (i.e., $\hat{c}$ is the discrete-time Fourier transform of c). Equivalently, we can write

$$\hat{\phi}^{(k)}(\omega) = \tfrac{1}{\sqrt{2}}\hat{c}(\omega/2)\hat{\phi}^{(k-1)}(\omega/2).$$

Recursively applying this formula $(k-1)$ times, we obtain

$$\begin{aligned}
\hat{\phi}^{(k)}(\omega) &= \tfrac{1}{\sqrt{2}}\hat{c}(\omega/2)\hat{\phi}^{(k-1)}(\omega/2) \\
&= \left(\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2)\right)\left(\tfrac{1}{\sqrt{2}}\hat{c}(\omega/4)\right)\hat{\phi}^{(k-2)}(\omega/4) \\
&= \left(\prod_{p=1}^{k}\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2^p)\right)\hat{\phi}^{(0)}(\omega/2^k).
\end{aligned}$$

Taking the limit as $k\to\infty$, we have

$$\begin{aligned}
\hat{\phi}^{(\infty)}(\omega) &= \lim_{k\to\infty}\hat{\phi}^{(k)}(\omega) \\
&= \lim_{k\to\infty}\left(\prod_{p=1}^{k}\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2^p)\right)\hat{\phi}^{(0)}(\omega/2^k) \\
&= \left(\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2^p)\right)\hat{\phi}^{(0)}(0) \\
&= \left(\prod_{p=1}^{\infty}\tfrac{1}{\sqrt{2}}\hat{c}(\omega/2^p)\right)\hat{\phi}(0) \\
&= \hat{\phi}(\omega).
\end{aligned}$$

Taking the inverse Fourier transform, we obtain

$$\phi^{(\infty)}(t) = \phi(t).$$

Therefore, if the stated iterative process converges to a fixed point, this fixed point is a solution to (4.55). □

In practice, we usually choose $\phi^{(0)} = \chi_{[0,1)}$. With such a choice, we then have that $\phi^{(k)}$ is of the form

$$\phi^{(k)}(t) = \sum_{n\in\mathbb{Z}} a^{(k)}[n]\chi_{[n2^{-k},(n+1)2^{-k})}(t).$$

That is, $\phi^{(k)}$ is piecewise constant on intervals of the form $[n2^{-k},(n+1)2^{-k})$, where $n \in \mathbb{Z}$. Furthermore, one can show that the sequence $a^{(k)}$ is given by

$$a^{(k)}[n] = \begin{cases} \left(\left[(\uparrow 2)a^{(k-1)}\right] * (\sqrt{2}c)\right)[n] & \text{for } k \geq 1 \\ \delta[n] & \text{for } k = 0. \end{cases}$$

The above algorithm can be implemented very conveniently in software. We need only compute the sequence $a^{(\kappa)}$ for some sufficiently large κ. From the resulting sequence, we can then trivially deduce $\phi^{(\kappa)}$.

Below, we provide some examples demonstrating how the cascade algorithm can be used to solve refinement equations.

Example 4.26 (Linear B-spline scaling function)**.** Consider the refinement equation

$$\phi(t) = \tfrac{1}{2}\phi(2t) + \phi(2t-1) + \tfrac{1}{2}\phi(2t-2).$$

This refinement equation is associated with the linear B-spline scaling function from Example 4.11 (to within a normalizing shift). Using three iterations of the cascade algorithm, find an approximate solution to the refinement equation having a zeroth moment of one.

Solution. Applying the cascade algorithm, starting with $\phi^{(0)} = \chi_{[0,1)}$, we obtain

$$\begin{gathered} a^{(0)}[0] = 1, \\ (a^{(1)}[0], a^{(1)}[1], a^{(1)}[2]) = \left(\tfrac{1}{2}, 1, \tfrac{1}{2}\right), \\ (a^{(2)}[0], a^{(2)}[1], \ldots, a^{(2)}[6]) = \left(\tfrac{1}{4}, \tfrac{1}{2}, \tfrac{3}{4}, 1, \tfrac{3}{4}, \tfrac{1}{2}, \tfrac{1}{4}\right), \\ (a^{(3)}[0], a^{(3)}[1], \ldots, a^{(3)}[14]) = \left(\tfrac{1}{8}, \tfrac{1}{4}, \tfrac{3}{8}, \tfrac{1}{2}, \tfrac{5}{8}, \tfrac{3}{4}, \tfrac{7}{8}, 1, \tfrac{7}{8}, \tfrac{3}{4}, \tfrac{5}{8}, \tfrac{1}{2}, \tfrac{3}{8}, \tfrac{1}{4}, \tfrac{1}{8}\right), \end{gathered}$$

and so on. The results of the first few iterations are illustrated in Figure 4.21. As we would expect, as we iterate, the approximate solution appears to be converging to the linear B-spline scaling function (up to a shift), shown earlier in Figure 4.12(a) on page 221.

□

Example 4.27. Consider the refinement equation $\phi(t) = \sum_{k\in\mathbb{Z}} c_k\phi(2t-n)$, where

$$(c_0, c_1, c_2, c_3, c_4, c_5) = \left(-\tfrac{1}{8}, \tfrac{1}{8}, \tfrac{8}{8}, \tfrac{8}{8}, \tfrac{1}{8}, -\tfrac{1}{8}\right) \quad \text{and} \quad c_n = 0 \text{ for } n \notin \{0,1,\ldots,5\}.$$

Let $\phi^{(n)}(t)$ denote the approximate solution to the above equation obtained after n iterations of the cascade algorithm, starting from the function $\phi^{(0)}(t) = \chi_{[0,1)}(t)$. Calculate $\phi^{(1)}(t)$ and $\phi^{(2)}(t)$.

Solution. Applying the cascade algorithm, starting with $\phi^{(0)} = \chi_{[0,1)}$, we obtain

$$\begin{gathered} a^{(0)}[0] = 1, \\ (a^{(1)}[0], a^{(1)}[1], \ldots, a^{(1)}[5]) = \left(-\tfrac{1}{8}, \tfrac{1}{8}, \tfrac{8}{8}, \tfrac{8}{8}, \tfrac{1}{8}, -\tfrac{1}{8}\right), \\ (a^{(2)}[0], a^{(2)}[1], \ldots, a^{(2)}[15]) = \left(\tfrac{1}{64}, -\tfrac{1}{64}, -\tfrac{9}{64}, -\tfrac{7}{64}, -\tfrac{1}{64}, \tfrac{17}{64}, \tfrac{57}{64}, \tfrac{71}{64}, \tfrac{71}{64}, \tfrac{57}{64}, \tfrac{17}{64}, -\tfrac{1}{64}, -\tfrac{7}{64}, -\tfrac{9}{64}, -\tfrac{1}{64}, \tfrac{1}{64}\right). \end{gathered}$$

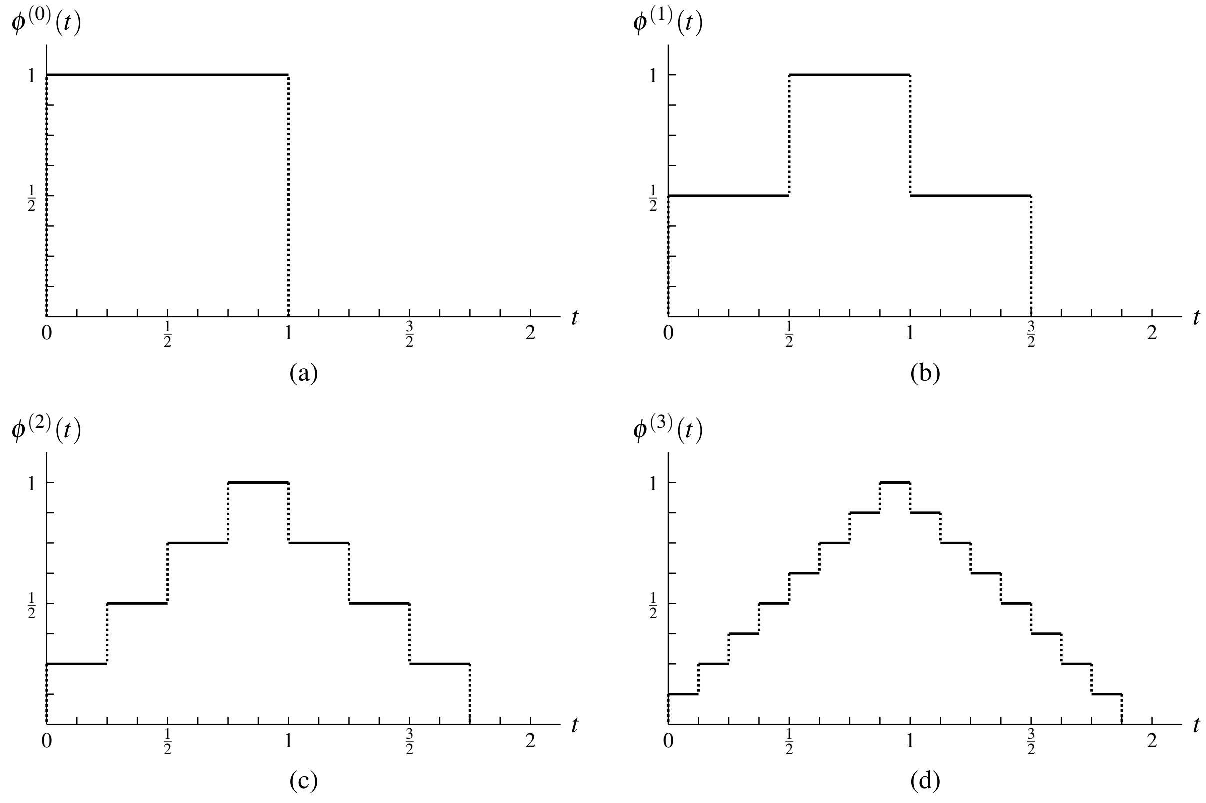


Figure 4.21: Cascade algorithm applied to B-spline scaling function.

Note that

$$\begin{aligned}
a^{(2)}[0] &= ((\uparrow 2)a^{(1)} * c)[0] = (-\tfrac{1}{8})(-\tfrac{1}{8}) = \tfrac{1}{64},\\
a^{(2)}[1] &= ((\uparrow 2)a^{(1)} * c)[1] = (-\tfrac{1}{8})(\tfrac{1}{8}) + (0)(-\tfrac{1}{8}) = -\tfrac{1}{64},\\
a^{(2)}[2] &= ((\uparrow 2)a^{(1)} * c)[2] = (-\tfrac{1}{8})(\tfrac{8}{8}) + (0)(\tfrac{1}{8}) + (\tfrac{1}{8})(-\tfrac{1}{8}) = -\tfrac{9}{64},\\
a^{(2)}[3] &= ((\uparrow 2)a^{(1)} * c)[3] = (-\tfrac{1}{8})(\tfrac{8}{8}) + (0)(\tfrac{8}{8}) + (\tfrac{1}{8})(\tfrac{1}{8}) + (0)(-\tfrac{1}{8}) = -\tfrac{7}{64},\\
a^{(2)}[4] &= ((\uparrow 2)a^{(1)} * c)[4] = (-\tfrac{1}{8})(\tfrac{1}{8}) + (0)(\tfrac{8}{8}) + (\tfrac{1}{8})(\tfrac{8}{8}) + (0)(\tfrac{1}{8}) + (\tfrac{8}{8})(-\tfrac{1}{8}) = -\tfrac{1}{64},\\
a^{(2)}[5] &= ((\uparrow 2)a^{(1)} * c)[5] = (-\tfrac{1}{8})(-\tfrac{1}{8}) + (0)(\tfrac{1}{8}) + (\tfrac{1}{8})(\tfrac{8}{8}) + (0)(\tfrac{8}{8}) + (\tfrac{8}{8})(\tfrac{1}{8}) + (0)(-\tfrac{1}{8}) = \tfrac{17}{64},\\
a^{(2)}[6] &= ((\uparrow 2)a^{(1)} * c)[6] = (0)(-\tfrac{1}{8}) + (\tfrac{1}{8})(\tfrac{1}{8}) + (0)(\tfrac{8}{8}) + (\tfrac{8}{8})(\tfrac{8}{8}) + (0)(\tfrac{1}{8}) + (\tfrac{8}{8})(-\tfrac{1}{8}) = \tfrac{57}{64}, \quad \text{and}\\
a^{(2)}[7] &= ((\uparrow 2)a^{(1)} * c)[7] = (\tfrac{1}{8})(-\tfrac{1}{8}) + (0)(\tfrac{1}{8}) + (\tfrac{8}{8})(\tfrac{8}{8}) + (0)(\tfrac{8}{8}) + (\tfrac{8}{8})(\tfrac{1}{8}) + (0)(-\tfrac{1}{8}) = \tfrac{71}{64}.
\end{aligned}$$

Since $a^{(1)}$ and c are symmetric, $a^{(2)}$ is symmetric. Because $a^{(2)}$ is symmetric, the remaining nonzero elements of this sequence can be trivially deduced. Thus, $\phi^{(0)}$, $\phi^{(1)}$, and $\phi^{(2)}$ have the graphs shown in Figures 4.22(a), (b), and (c), respectively. □

4.3 *M*-adic Wavelet Systems

Multiresolution representations of functions are often extremely useful. Sometimes, however, we may not wish to restrict ourselves to multiresolution representations that are based on a scale factor of two. Although in some applications, a factor of two might be a very natural choice, in others a more appropriate choice may exist. In what follows, we will explore the generalization of wavelet systems to arbitrary integer scale factors (greater than or equal to two). As we shall see, many results in the more general case are very similar to those in the dyadic case.

4.3.1 MRAs

In the more general M-adic case, we need to modify the definition of a MRA. For an M-adic wavelet system, a MRA is as defined previously except that the scale invariance property becomes:

for all $p \in \mathbb{Z}$, $f \in V_p \Leftrightarrow f(M\cdot) \in V_{p-1}$, where $M \in \mathbb{Z}$ and $M \geq 2$.

One can show that the scale invariance and shift invariance properties together imply that

$$\text{for all } p \in \mathbb{Z}, \quad f \in V_p \Leftrightarrow f(\cdot - M^p k) \in V_p.$$

In what follows, we present some examples of MRAs.

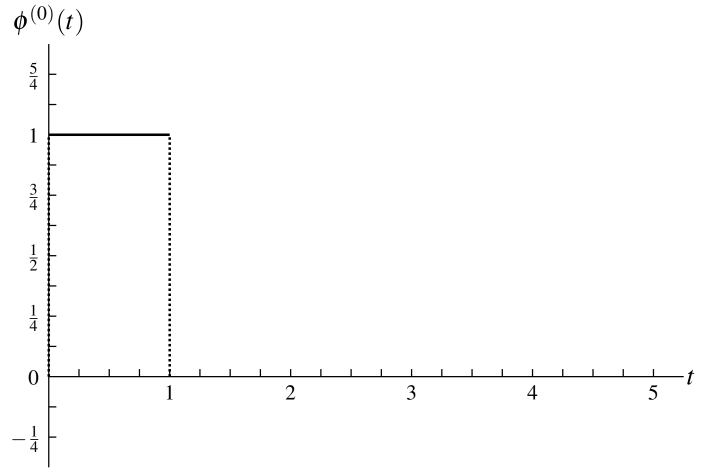
Example 4.28 (Piecewise constant approximations)**.** The idea of piecewise constant approximations can be generalized from the dyadic case in Example 4.1 to arbitrary dilation factors. The space V_0 is comprised of all functions $f \in L^2(\mathbb{R})$ such that f is constant on intervals of the form $[n, n+1)$, where $n \in \mathbb{Z}$. More generally, V_p is comprised of all functions $f \in L^2(\mathbb{R})$ such that f is constant over intervals of the form $[nM^p, (n+1)M^p)$, where $n \in \mathbb{Z}$. One can show that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis of V_0, where $\phi = \chi_{[0,1)}$. In fact, one can show that $\{V_p\}_{p\in\mathbb{Z}}$ constitutes a MRA.

Example 4.29 (Bandlimited approximations)**.** Consider the sequence $\{V_p\}_{p\in\mathbb{Z}}$ of subspaces of $L^2(\mathbb{R})$ given by

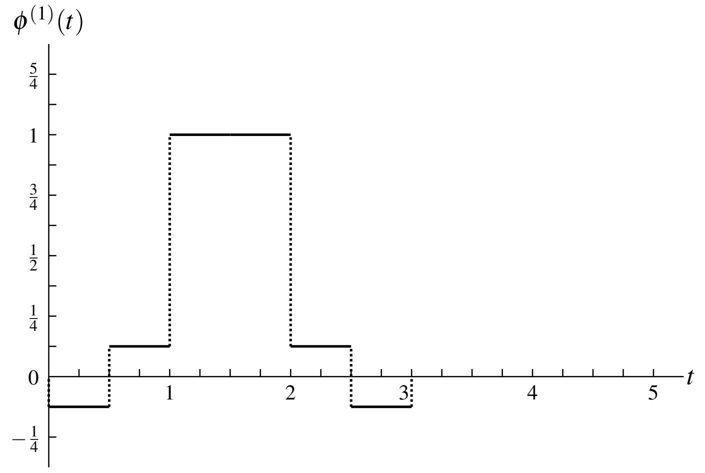
$$V_p = \{f : \operatorname{supp} \hat{f} \subset [-M^{-p}\pi, M^{-p}\pi]\},$$

where $M \in \mathbb{Z}$ and $M \geq 2$. The above sequence of subspaces constitutes a MRA. (In the case that $M = 2$, we have the Shannon MRA from Example 4.2.) One can show that a Riesz basis of V_0 is given by $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$, where

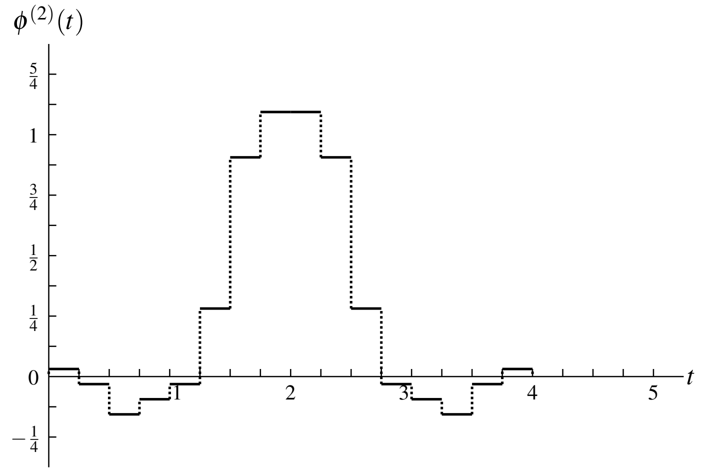
$$\phi(t) = \operatorname{sinc} \pi t.$$



(a)



(b)



(c)

Figure 4.22: Results of the cascade algorithm.

$$\begin{tikzcd}
L^2(\mathbb{R}) & \cdots & V_{-2} \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_{-1} \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_0 \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_1 & \cdots & \{0\} \\
 & & & W_{1,-1} & W_{1,0} & W_{1,1} & & \\
 & & & W_{2,-1} & W_{2,0} & W_{2,1} & & \\
 & & & \vdots & \vdots & \vdots & & \\
 & & & W_{M-1,-1} & W_{M-1,0} & W_{M-1,1} & &
\end{tikzcd}$$

Figure 4.23: MRA decomposition of $L^2(\mathbb{R})$.

From the dilation property of the Fourier transform, we have

$$\operatorname{supp}\hat{f} \subset [-M^{-p}\pi, M^{-p}\pi] \Leftrightarrow \operatorname{supp}\widehat{f(M\cdot)} \subset [-M^{-(p-1)}\pi, M^{-(p-1)}\pi].$$

From the definition of $\{V_p\}_{p\in\mathbb{Z}}$, we have

$$\operatorname{supp}\hat{f} \subset [-M^{-p}\pi, M^{-p}\pi] \Leftrightarrow f \in V_p \quad \text{and}$$
$$\operatorname{supp}\hat{f} \subset [-M^{-(p-1)}\pi, M^{-(p-1)}\pi] \Leftrightarrow f \in V_{p-1}.$$

Combining the above equivalences, we conclude that $f \in V_p \Leftrightarrow f(M\cdot) \in V_{p-1}$. Thus, the scale invariance property holds. From the translation property of the Fourier transform, we have

$$\operatorname{supp}\hat{f} = \operatorname{supp}\widehat{f(\cdot - n)} \text{ for all } n \in \mathbb{Z}.$$

From this, it follows that $f \in V_0 \Leftrightarrow f(\cdot - n) \in V_0$ for all $n \in \mathbb{Z}$. Thus, the shift invariance property holds. We leave it as an exercise for the reader to confirm that the other properties of a MRA also hold.

4.3.2 Wavelet Spaces

Consider a MRA with the approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$. Since V_p is a proper subspace of V_{p-1} there must be some space U_p which is the algebraic complement of V_p in V_{p-1}. In other words, we have

$$V_{p-1} = V_p \oplus U_p.$$

Unfortunately, in the case of M-adic MRAs where $M > 2$, it is not possible for the integer shifts of a single function to form a basis for U_0. In fact, one can show that we need the integer shifts of $M-1$ functions. Thus, we need to further partition the space U_p into the $M-1$ subspaces $\{W_{q,p}\}_{q\in\{1,2,\ldots,M-1\}}$. That is, we have

$$U_p = \bigoplus_{q=1}^{M-1} W_{q,p}.$$

This leads to the relationship

$$V_{p-1} = V_p \oplus \left(\bigoplus_{k=1}^{M-1} W_{k,p} \right).$$

We refer to the $W_{q,p}$ spaces as wavelet spaces. Notice, however, that we now have $M-1$ wavelet spaces at each level in the MRA. This leads to the decomposition of $L^2(\mathbb{R})$ illustrated in Figure 4.23.

One can show that the various spaces have the following properties:

$$V_k \cap V_l = V_l \quad \text{for } k < l$$
$$W_{q,p} \cap W_{l,k} = \{0\} \quad \text{for } (q,p) \neq (l,k)\text{; and}$$
$$V_k \cap W_{q,p} = \{0\} \quad \text{for } k \geq p.$$

Moreover, the wavelet spaces have the additional properties given by the theorem below.

Theorem 4.30 (Properties of wavelet spaces (M-adic case))**.** *Let $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$ denote the wavelet spaces associated with a MRA. Then, these spaces are such that*

1. $\operatorname{clos}\left(\bigoplus_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}} W_{q,p}\right) = L^2(\mathbb{R})$ *(density);*
2. *for each $q \in \{1,2,\ldots,M-1\}$, $f \in W_{q,p} \Leftrightarrow f(M\cdot) \in W_{q,p-1}$ for all $p \in \mathbb{Z}$ (scale invariance);*
3. *for each $q \in \{1,2,\ldots,M-1\}$, $f \in W_{q,0} \Leftrightarrow f(\cdot - n) \in W_{q,0}$ for all $n \in \mathbb{Z}$ (shift invariance); and*
4. *for each $q \in \{1,2,\ldots,M-1\}$, there exists ψ_q such that $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of $W_{q,0}$ (shift-invariant Riesz basis).*

The basis of each wavelet space $W_{q,0}$ is generated by the integer shifts of a single function ψ_q. We refer to ψ_q as a wavelet function. Notice that, in the M-adic case, we have $M-1$ wavelet functions (since there are $M-1$ wavelet spaces at each level in the MRA).

4.3.3 Bases of Scaling and Wavelet Spaces

Consider a MRA associated with the approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$. Let $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ denote the Riesz basis of V_0. For each $q \in \{1,2,\ldots,M-1\}$, let $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}}$ denote the Riesz basis of $W_{q,0}$. Suppose that ϕ and $\{\psi_q\}_{q\in\{1,2,\ldots,M-1\}}$ are known. Then, just like in the dyadic case, we can trivially determine a basis for each of the other approximation and wavelet spaces. In the M-adic case, we have the more general result below.

Theorem 4.31 (Bases of approximation spaces)**.** *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ and $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of V_0 with the dual basis $\{\tilde{\phi}(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, for each $p \in \mathbb{Z}$, the set $\{\phi_{p,n}\}_{n\in\mathbb{Z}}$ given by*

$$\phi_{p,n}(t) = M^{-p/2}\phi(M^{-p}t - n)$$

is a Riesz basis of V_p with the same Riesz bounds as $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ and the dual basis $\{\tilde{\phi}_{p,n}\}_{n\in\mathbb{Z}}$ given by

$$\tilde{\phi}_{p,n}(t) = M^{-p/2}\tilde{\phi}(M^{-p}t - n).$$

Proof. The proof is analogous to that of Theorem 4.6. □

Now, we consider the bases the wavelet spaces. The basis of each of these spaces can be determined using the theorem below.

Theorem 4.32 (Bases of wavelet spaces)**.** *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ with the corresponding wavelet spaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$ and $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of $W_{q,0}$ with the dual basis $\{\tilde{\psi}_q(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, for each $p \in \mathbb{Z}$, $\{\psi_{q,p,n}\}_{n\in\mathbb{Z}}$ given by*

$$\psi_{q,p,n}(t) = M^{-p/2}\psi_q(M^{-p}t - n)$$

is a Riesz basis of $W_{q,p}$ with the same Riesz bounds as $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}}$ and dual basis $\{\tilde{\psi}_{q,p,n}\}_{n\in\mathbb{Z}}$ given by

$$\tilde{\psi}_{q,p,n}(t) = M^{-p/2}\tilde{\psi}_q(M^{-p}t - n).$$

Proof. The proof is essentially the same as the proof for Theorem 4.31 with the scaling function ϕ replaced by the wavelet function ψ_q. □

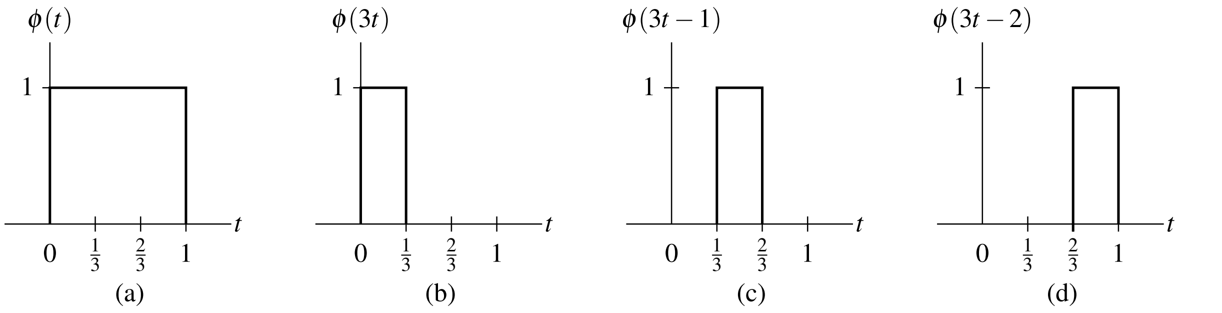


Figure 4.24: Example of a 3-adic refinable function. (a) Refinable function. (b,c,d) Components of refinable function.

4.3.4 Scaling and Wavelet Equations

Here, we introduce some examples of M-refinable functions where $M > 2$.

Example 4.30. Consider the function

$$\phi(t) = \chi_{[0,1)}(t).$$

One can confirm that ϕ satisfies a refinement relationship for any choice of dilation factor M (where $M \geq 2$). In particular, we have

$$\phi(t) = \sum_{k=0}^{M-1} \phi(Mt-k).$$

This relationship is illustrated in Figure 4.24 for the case of $M = 3$.

Example 4.31. Consider the function

$$\phi(t) = \begin{cases} t & \text{for } 0 \leq t < 1 \\ 2-t & \text{for } 1 \leq t < 2 \\ 0 & \text{otherwise.} \end{cases}$$

One can confirm that ϕ is 3-refinable. In particular, we have

$$\phi(t) = \sum_{k=0}^{4} c[k]\phi(3t-k),$$

where c is given by

$$(c[0], c[1], \ldots, c[4]) = \left(\tfrac{1}{3}, \tfrac{2}{3}, 1, \tfrac{2}{3}, \tfrac{1}{3}\right).$$

(We already saw that ϕ is also 2-refinable. In passing, we note that most 2-refinable functions are not 3-refinable, and vice versa. One can show, however, that B-spline functions are both 2- and 3-refinable [19, Section 4.5]. We leave it as an exercise for the reader to confirm that ϕ is also refinable for other choices of dilation factor.)

Just like in the 2-scale case, in the M-scale case, the scaling function ϕ is refinable. In the M-scale case, however, ϕ is M-refinable. This leads to the scaling equation as given by the theorem below.

Theorem 4.33 (Scaling equation (M-adic case)). *Suppose that we have an M-adic MRA $\{V_p\}_{p\in\mathbb{Z}}$ and V_0 has the Riesz basis $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, ϕ satisfies a refinement equation of the form*

$$\phi(t) = M^{1/2} \sum_{n\in\mathbb{Z}} c[n]\phi(Mt-n) \tag{4.57a}$$

where

$$c[n] = \left\langle \phi, M^{1/2}\tilde{\phi}(M\cdot - n)\right\rangle. \tag{4.57b}$$

Proof. The proof is analogous to that of the dyadic case. □

Often, it is convenient to work with the Fourier transform of the scaling equation, which is given by the following theorem.

Theorem 4.34 (Fourier transform of scaling equation (M-adic case)). *Let ϕ be an M-adic scaling function with scaling equation coefficient sequence c. Then, $\hat{\phi}$ is given by*

$$\hat{\phi}(\omega) = M^{-1/2}\hat{c}(M^{-1}\omega)\hat{\phi}(M^{-1}\omega)$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\phi}(\omega) = \hat{\phi}(0)\prod_{p=1}^{\infty} M^{-1/2}\hat{c}(M^{-p}\omega).$$

Proof. The proof is analogous to that of the dyadic case. □

Although the wavelet functions are not refinable, they can be represented in terms of dilated and translated versions of the scaling function. For each wavelet function, this leads to a wavelet equation, as given by the theorem below.

Theorem 4.35 (Wavelet equation (M-adic case)). *Suppose that we have a MRA $\{V_p\}_{p\in\mathbb{Z}}$ with corresponding wavelet subspaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$, where V_0 has the Riesz basis $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ and $W_{q,0}$ has the Riesz basis $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}}$. Then, ψ_q can be expressed in terms of an equation of the form*

$$\psi_q(t) = M^{1/2}\sum_{n\in\mathbb{Z}} d_q[n]\phi(Mt - n)$$

where

$$d_q[n] = \left\langle \psi_q, M^{1/2}\tilde{\phi}(M\cdot - n)\right\rangle.$$

Proof. The proof is analogous to that of the dyadic case. □

Often, the Fourier transform of a wavelet equation is useful to know. It is given by the theorem below.

Theorem 4.36 (Fourier transform of the wavelet equation (M-adic case)). *Let ϕ and $\{\psi_q\}_{q\in\{1,2,\ldots,M-1\}}$ be the scaling and wavelet functions of a MRA. Suppose that ϕ has the scaling equation coefficient sequence c and ψ_q has the wavelet equation coefficient sequence d_q. Then, $\hat{\psi}_q$ is given by*

$$\hat{\psi}_q(\omega) = M^{-1/2}\hat{d}_q(M^{-1}\omega)\hat{\phi}(M^{-1}\omega)$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\psi}_q(\omega) = M^{-1/2}\hat{\phi}(0)\hat{d}_q(M^{-1}\omega)\prod_{p=1}^{\infty} M^{-1/2}\hat{c}(M^{-p-1}\omega).$$

Proof. The proof is analogous to that of the dyadic case. □

4.3.5 Dual MRAs

Just like in the dyadic case, each MRA is associated with a dual MRA. This fact is formalized in the following theorem.

Theorem 4.37 (Dual MRAs)**.** *Let $\{V_p\}_{p\in\mathbb{Z}}$ be a MRA with scaling function ϕ, with corresponding wavelet spaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$, and corresponding wavelet function ψ_q. Suppose that the dual Riesz bases of $\{\phi(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\psi_q(\cdot-k)\}_{k\in\mathbb{Z}}$ are given by $\{\tilde{\phi}(\cdot-k)\}_{k\in\mathbb{Z}}$ and $\{\tilde{\psi}_q(\cdot-k)\}_{k\in\mathbb{Z}}$, respectively. Then, $\tilde{\phi}$ is also the scaling function of a MRA $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ with corresponding wavelet spaces $\{\tilde{W}_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$ and the corresponding wavelet functions $\{\tilde{\psi}_q\}_{q\in\{1,2,\ldots,M-1\}}$.*

Proof. The proof is left as an exercise for the reader. □

An important relationship between a MRA and its dual is given by the theorem below.

Theorem 4.38. *Suppose that $\{V_p\}_{p\in\mathbb{Z}}$ and $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ are dual MRAs with corresponding wavelet spaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$ and $\{\tilde{W}_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$, respectively. Then, we have*

$$\text{for all } (q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z},\quad V_p\perp\tilde{W}_{q,p}\ \text{ and }\ W_{q,p}\perp\tilde{V}_p.$$

Proof. The proof follows from that of the dyadic case, by observing that $W_{q,p}$ is contained in the algebraic complement of V_p in V_{p-1} and $\tilde{W}_{q,p}$ is contained in the algebraic complement of $\tilde{V}_p$ in $\tilde{V}_{p-1}$. □

4.3.6 Wavelet Systems

Like in the dyadic case, we have three basic types of wavelet systems: orthonormal, semiorthogonal, and biorthogonal. The definitions are analogous to the dyadic case.

4.3.7 Examples of Wavelet Systems

In what follows, we provide some examples of M-adic wavelet systems (where $M>2$).

Example 4.32 (Bandlimited approximations)**.** In this example, we introduce a family of wavelet systems based on spaces of bandlimited functions. This is a straightforward generalization of the Shannon wavelet system considered in Example 4.13.

Let us construct a MRA $\{V_p\}_{p\in\mathbb{Z}}$ as follows. For each $p\in\mathbb{Z}$, we let V_p be the set of all functions f (in $L^2(\mathbb{R})$) such that

$$\operatorname{supp}\hat{f}\subset[-M^{-p}\pi,M^{-p}\pi]$$

(i.e., V_p is the set of all functions bandlimited to $[-M^{-p}\pi,M^{-p}\pi]$). Denote the corresponding wavelet spaces as $\{W_{q,p}\}_{(q,p)\in\{\}\times\mathbb{Z}}$. Then, for each (q,p), one can choose $W_{q,p}$ as the set of all functions f such that

$$\operatorname{supp}\hat{f}\subset[-(q+1)M^{-(p-1)}\pi,-qM^{-(p-1)}\pi]\cup[qM^{-(p-1)}\pi,(q+1)M^{-(p-1)}\pi].$$

The scaling function ϕ is given by

$$\phi(t)=\operatorname{sinc}\pi t$$

and for each $q\in\{1,2,\ldots,M-1\}$, the wavelet function ψ_q is given by

$$\psi_q(t)=\cos\left(\tfrac{(2q-1)\pi t}{2}\right)\operatorname{sinc}\left(\tfrac{\pi t}{2}\right).$$

One can readily show that the Fourier transforms of the above functions are given by

$$\hat{\phi}(\omega)=\chi_{[-\pi,\pi]}(\omega)\quad\text{and}$$
$$\hat{\psi}_q(\omega)=\chi_{[-(q+1)\pi,-q\pi]}(\omega)+\chi_{[q\pi,(q+1)\pi]}(\omega).$$

Example 4.33 (Belogay-Wang wavelet system)**.** In this example, we introduce the 3-adic Belogay-Wang wavelet system proposed in [2]. This particular system is orthonormal with symmetric basis functions.

The scaling function ϕ satisfies the refinement equation

$$\phi(t) = \sum_{k=0}^{5} c[k]\phi(3t-k)$$

where c is given by

$$(c[0], c[1], \ldots, c[5]) = \left(\tfrac{2-\sqrt{6}}{4}, \tfrac{1}{2}, \tfrac{2+\sqrt{6}}{4}, \tfrac{2+\sqrt{6}}{4}, \tfrac{1}{2}, \tfrac{2-\sqrt{6}}{4}\right).$$

Furthermore, it can be shown that ϕ is continuous. The wavelet functions ψ_1 and ψ_2 can be expressed in terms of the scaling function as

$$\psi_1(t) = \sum_{k=0}^{5} d_1[k]\phi(3t-k) \quad \text{and}$$
$$\psi_2(t) = \sum_{k=0}^{2} d_2[k]\phi(3t-k),$$

where d_1 and d_2 are given by [1]

$$(d_1[0], d_1[1], \ldots, d_1[5]) = \left(\tfrac{2-\sqrt{6}}{4}, \tfrac{1}{2}, \tfrac{2+\sqrt{6}}{4}, -\tfrac{2+\sqrt{6}}{4}, -\tfrac{1}{2}, -\tfrac{2-\sqrt{6}}{4}\right) \quad \text{and}$$
$$(d_2[0], d_2[1], d_2[2]) = \left(\tfrac{1}{2}, -1, \tfrac{1}{2}\right).$$

The scaling and wavelet functions are plotted in Figure 4.25. (Since the system is orthonormal, $\tilde{\phi} = \phi$, $\tilde{\psi}_1 = \psi_1$, and $\tilde{\psi}_2 = \psi_2$.)

Observe that this system is orthonormal with basis functions that are real, symmetric, compactly supported, and of "nontrivial" length. As was noted earlier, a wavelet system with all of these properties is not possible in the dyadic case.

4.3.8 Relationship Between Wavelet Systems and Filter Banks

Just like in the 2-scale case, there is also a relationship between wavelet systems and filter banks in the M-scale case (where $M > 2$). In particular, we have the result below.

Theorem 4.39 (Mallat algorithm)**.** *Consider a wavelet system with approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$, scaling function ϕ, wavelet spaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}}$, wavelet functions $\{\psi_q\}_{q\in\{1,2,\ldots,M-1\}}$, dual scaling function $\tilde{\phi}$, and dual wavelet functions $\{\tilde{\psi}_q\}_{q\in\{1,2,\ldots,M-1\}}$. Let the scaling equation coefficient sequences of ϕ and $\tilde{\phi}$ be denoted as c and $\tilde{c}$, respectively. Let the wavelet equation coefficient sequences of ψ_q and $\tilde{\psi}_q$ be denoted as d_q and $\tilde{d}_q$, respectively. Define the basis functions for the various spaces as before:*

$$\phi_{p,n}(t) = M^{-p/2}\phi(M^{-p}t-n),$$
$$\psi_{q,p,n}(t) = M^{-p/2}\psi_q(M^{-p}t-n),$$
$$\tilde{\phi}_{p,n}(t) = M^{-p/2}\tilde{\phi}(M^{-p}t-n), \quad \textit{and}$$
$$\tilde{\psi}_{q,p,n}(t) = M^{-p/2}\tilde{\psi}_q(M^{-p}t-n).$$

Any $f \in V_p$ can be represented in each of the following forms:

$$f = \sum_{n\in\mathbb{Z}} a_p[n]\phi_{p,n} \quad \textit{and} \tag{4.59}$$

$$f = \sum_{n\in\mathbb{Z}} a_{p+1}[n]\phi_{p+1,n} + \sum_{q=1}^{M-1}\sum_{n\in\mathbb{Z}} b_{q,p+1}[n]\psi_{q,p+1,n}. \tag{4.60}$$

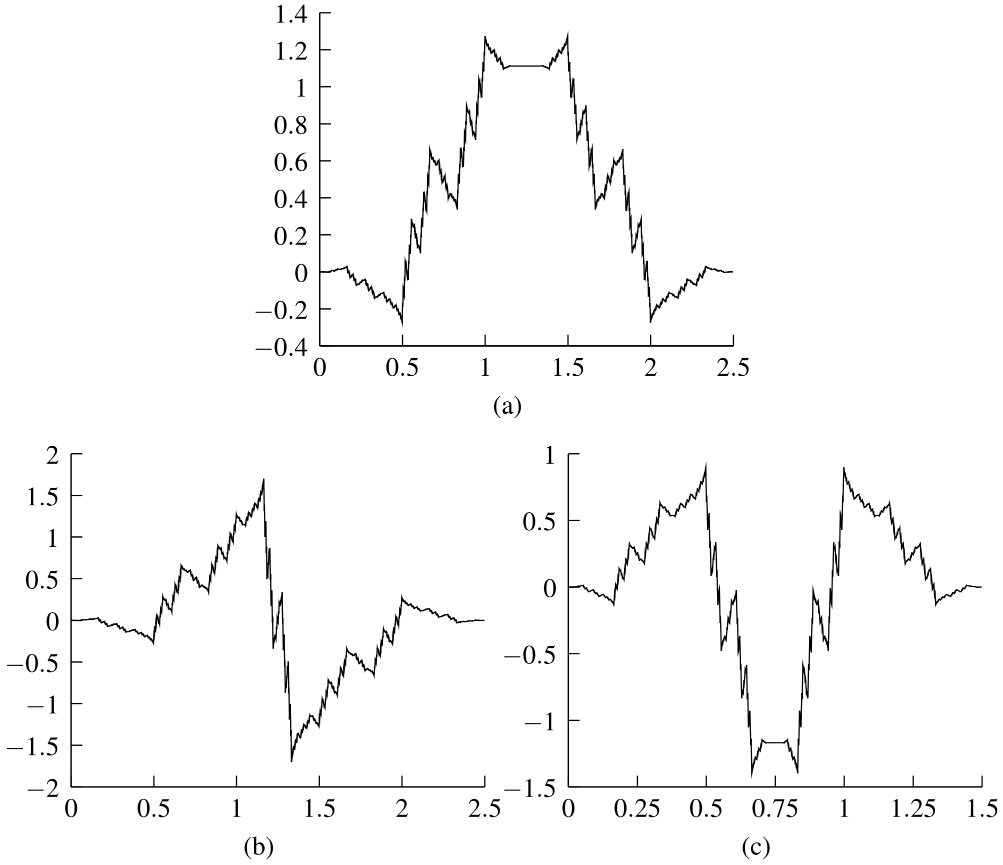


Figure 4.25: Belogay-Wang scaling and wavelet functions. (a) Scaling function and (b) and (c) wavelet functions.

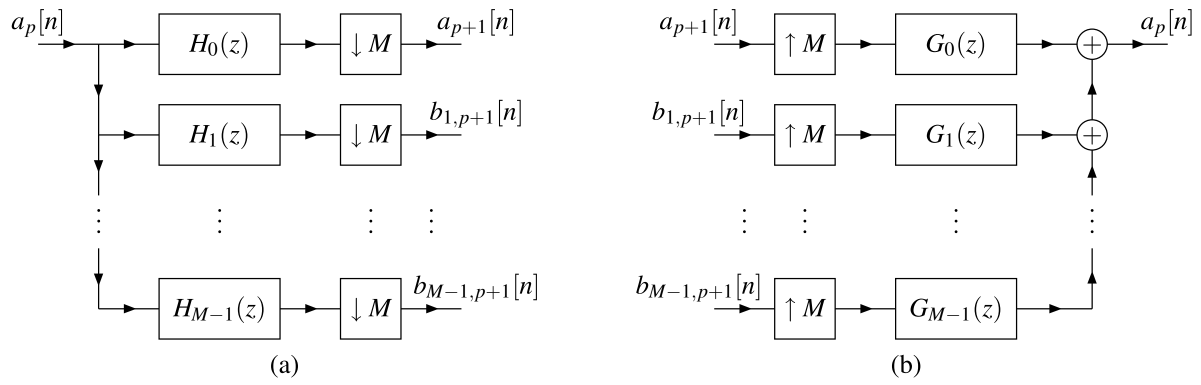


Figure 4.26: Computational structure associated with the Mallat algorithm (M-adic case).

Given a_p, we can compute the corresponding a_{p+1} and $b_{q,p+1}$ as follows:

$$a_{p+1} = (\downarrow M)(a_p * h_0) \quad \text{and} \tag{4.61a}$$
$$b_{q,p+1} = (\downarrow M)(a_p * h_q), \tag{4.61b}$$

where $h_0[n] = \tilde{c}^*[-n]$ and $h_q[n] = \tilde{d}_q^*[-n]$ for $q \in \{1,2,\ldots,M-1\}$. Given a_{p+1} and $b_{q,p+1}$, we can compute the corresponding a_p as follows:

$$a_p = ((\uparrow M)a_{p+1}) * g_0 + \sum_{q=1}^{M-1} \left((\uparrow M)b_{q,p+1}\right) * g_q, \tag{4.62}$$

where $g_0[n] = c[n]$ and $g_q[n] = d_q[n]$ for $q \in \{1,2,\ldots,M-1\}$.

From the above theorem, we can see that the process of moving between representations of functions at different resolutions is accomplished by UMD filter banks. In particular, the Mallat algorithm is associated with the analysis and synthesis sides of an M-channel UMD filter bank as shown in Figure 4.26.

4.3.9 Characterizing Biorthonormal and Orthonormal Wavelet Systems

From Theorem 4.38, we have that

$$\left\langle \phi_{p,k}, \tilde{\phi}_{p,l} \right\rangle = \delta[k-l], \tag{4.63a}$$
$$\left\langle \psi_{q,p,k}, \tilde{\psi}_{q,p,l} \right\rangle = \delta[k-l], \tag{4.63b}$$
$$\left\langle \phi_{p,k}, \tilde{\psi}_{q,p,l} \right\rangle = 0, \quad \text{and} \tag{4.63c}$$
$$\left\langle \psi_{q,p,k}, \tilde{\phi}_{p,l} \right\rangle = 0. \tag{4.63d}$$

In a similar fashion as in the dyadic case, we can show that these relationships imply

$$\langle c[\cdot], \tilde{c}[\cdot - Mn] \rangle = \delta[n], \tag{4.64a}$$
$$\left\langle d_q[\cdot], \tilde{d}_q[\cdot - Mn] \right\rangle = \delta[n], \tag{4.64b}$$
$$\left\langle c[\cdot], \tilde{d}_q[\cdot - Mn] \right\rangle = 0, \quad \text{and} \tag{4.64c}$$
$$\left\langle d_q[\cdot], \tilde{c}[\cdot - Mn] \right\rangle = 0. \tag{4.64d}$$

We would like to rewrite the above relationships in the Fourier domain. To this end, the result below will be helpful.

Lemma 4.3. *For any two sequences f and g, and $\alpha \in \mathbb{C}$, we have*

$$\langle f[\cdot], g[\cdot - Mn] \rangle = \alpha \delta[n] \quad \Leftrightarrow \quad \sum_{l=0}^{M-1} \hat{f}(\omega - \tfrac{2\pi l}{M}) \hat{g}^*(\omega - \tfrac{2\pi l}{M}) = M\alpha.$$

Proof. Define $v[n] = g^*[-n]$. Using this definition, we can write

$$\begin{aligned}
&\langle f[\cdot], g[\cdot - Mn]\rangle = \alpha\delta[n] \\
\Leftrightarrow\quad &\sum_{k\in\mathbb{Z}} f[k]g^*[k-Mn] = \alpha\delta[n] \\
\Leftrightarrow\quad &\sum_{k\in\mathbb{Z}} f[k]v[Mn-k] = \alpha\delta[n] \\
\Leftrightarrow\quad &((\downarrow M)(f * v))[n] = \alpha\delta[n].
\end{aligned}$$

Taking the Fourier transform of both sides of the preceding equation and using the fact that $\hat{v}(\omega) = \hat{g}^*(\omega)$, we obtain

$$\begin{aligned}
\Leftrightarrow\quad &(\downarrow M)(\hat{f}(\omega)\hat{v}(\omega)) = \alpha \\
\Leftrightarrow\quad &\tfrac{1}{M}\sum_{l=0}^{M-1} \hat{f}(\omega - \tfrac{2\pi l}{M})\hat{v}(\omega - \tfrac{2\pi l}{M}) = \alpha \\
\Leftrightarrow\quad &\sum_{l=0}^{M-1} \hat{f}(\omega - \tfrac{2\pi l}{M})\hat{g}^*(\omega - \tfrac{2\pi l}{M}) = M\alpha.
\end{aligned}$$

□

Using Lemma 4.3, we can equivalently rewrite (4.64) as

$$\sum_{l=0}^{M-1} \hat{c}\left(\omega - \tfrac{2\pi l}{M}\right)\hat{\tilde{c}}^*\left(\omega - \tfrac{2\pi l}{M}\right) = M, \tag{4.65a}$$

$$\sum_{l=0}^{M-1} \hat{d}_q\left(\omega - \tfrac{2\pi l}{M}\right)\hat{\tilde{d}}_q^*\left(\omega - \tfrac{2\pi l}{M}\right) = M, \tag{4.65b}$$

$$\sum_{l=0}^{M-1} \hat{c}\left(\omega - \tfrac{2\pi l}{M}\right)\hat{\tilde{d}}_q^*\left(\omega - \tfrac{2\pi l}{M}\right) = 0, \quad \text{and} \tag{4.65c}$$

$$\sum_{l=0}^{M-1} \hat{d}_q\left(\omega - \tfrac{2\pi l}{M}\right)\hat{\tilde{c}}^*\left(\omega - \tfrac{2\pi l}{M}\right) = 0. \tag{4.65d}$$

In the case that the system is orthonormal, we have that $c = \tilde{c}$ and $d_q = \tilde{d}_q$, and the above equations become

$$\sum_{l=0}^{M-1} \left|\hat{c}\left(\omega - \tfrac{2\pi l}{M}\right)\right|^2 = M, \tag{4.66a}$$

$$\sum_{l=0}^{M-1} \left|\hat{d}_q\left(\omega - \tfrac{2\pi l}{M}\right)\right|^2 = M, \quad \text{and} \tag{4.66b}$$

$$\sum_{l=0}^{M-1} \hat{c}\left(\omega - \tfrac{2\pi l}{M}\right)\hat{d}_q^*\left(\omega - \tfrac{2\pi l}{M}\right) = 0. \tag{4.66c}$$

4.3.10 Properties of Scaling and Wavelet Functions

Since scaling and wavelet functions play an important role in characterizing wavelet systems, it is beneficial to examine some of the properties of such functions. Some of these properties are introduced by the theorems below.

Theorem 4.40 (Properties of scaling function)**.** *Suppose that ϕ is a compactly supported function with zeroth moment $\mu_0 \neq 0$ and stable integer shifts (i.e., $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis), and ϕ satisfies an M-adic scaling equation with* *coefficient sequence $c[n]$. Then, we have*

$$\sum_{k\in\mathbb{Z}} \phi(t-k) = \mu_0, \tag{4.67}$$

$$\hat{\phi}(2\pi k) = 0 \textit{ for all } k \in \mathbb{Z}\setminus\{0\}, \tag{4.68}$$

$$\hat{c}(0) = M^{1/2}, \quad \textit{and} \tag{4.69}$$

$$\hat{c}\left(\tfrac{2\pi k}{M}\right) = 0 \quad \textit{for } k \in \{1,2,\ldots,M-1\}. \tag{4.70}$$

Proof. In a similar manner as in the dyadic case, we can readily show that (4.69), (4.68), and (4.67) hold.

Now, we consider (4.70). From the Fourier transform of the scaling equation, we can write

$$\hat{\phi}(M\omega) = M^{-1/2}\hat{c}(\omega)\hat{\phi}(\omega). \tag{4.71}$$

Let $N(\phi) \triangleq \{\omega \in \mathbb{R} : \hat{\phi}(\omega+2\pi k) = 0 \text{ for all } k \in \mathbb{Z}\}$. Since ϕ is compactly supported and $\phi \in L^2(\mathbb{R})$, we have that $\phi \in L^1(\mathbb{R})$. Thus, by the Riemann-Lebesgue lemma, $\hat{\phi}$ is continuous. Since $\hat{\phi}$ is continuous, $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ being a Riesz basis is equivalent to $N(\phi)$ being the empty set. (Since $\hat{\phi}$ is continuous, the Riesz basis condition $0 < A \le \sum_{k\in\mathbb{Z}} \left|\hat{\phi}(\omega+2\pi k)\right| \le B < \infty$ for ω almost everywhere is violated if and only if there exists some ω such that $\hat{\phi}(\omega+2\pi k) = 0$ for all $k \in \mathbb{Z}$.) Let $\omega \in \{\frac{2\pi k}{M} : k \in \{1,2,\ldots,M-1\}\}$. There must exist γ of the form $\gamma = 2\pi\beta + \omega$ with $\beta \in \mathbb{Z}$ such that $\hat{\phi}(\gamma) \neq 0$. Since $\hat{\phi}(\gamma) \neq 0$, we can rearrange (4.71) to obtain

$$\hat{c}(\gamma) = M^{1/2}\hat{\phi}(M\gamma)/\hat{\phi}(\gamma).$$

Now, we consider the quantity $M\gamma$ more closely. We have

$$\begin{aligned} M\gamma &= M(2\pi\beta+\omega) \\ &= 2\pi M\beta + M(\tfrac{2\pi k}{M}) \\ &= 2\pi M\beta + 2\pi k \\ &= 2\pi(M\beta + k) \\ &\in 2\pi\mathbb{Z}\setminus\{0\}. \end{aligned}$$

In other words, $M\gamma$ is a nonzero integer multiple of 2π. Consequently, from (4.68), we have that $\hat{\phi}(M\gamma) = 0$ which implies that $\hat{c}(\gamma) = 0$. So, we have

$$\begin{aligned} 0 &= \hat{c}(\gamma) \\ &= \hat{c}(2\pi\beta + \tfrac{2\pi k}{M}) \\ &= \hat{c}(\tfrac{2\pi k}{M}). \end{aligned}$$

Thus, (4.70) holds. □

Theorem 4.41 (Zeroth moment of wavelet function)**.** *Let ψ be a wavelet function. Then, the zeroth moment of ψ is zero (i.e., $\int_{-\infty}^{\infty}\psi(t)dt = 0$). The wavelet equation coefficient sequence d is such that $\hat{d}(0) = 0$.*

Proof. Consider (4.65d) for $\omega = 2\pi(M-1)/M$. We have

$$\sum_{l=0}^{M-1} \hat{d}\left(\tfrac{2\pi l}{M}\right)\hat{c}^*\left(\tfrac{2\pi l}{M}\right) = 0.$$

From (4.70) and (4.69), we have that only the term for $l = 0$ in the above summation is nonzero. In particular, we have

$$M^{1/2}\hat{d}(0) = 0.$$

This implies, however, that $\hat{d}(0) = 0$. Evaluating the Fourier transform of the wavelet equation at $\omega = 0$, we obtain

$$\begin{aligned}\hat{\psi}(0) &= M^{-1/2}\hat{d}(0)\hat{\phi}(0)\\ &= 0.\end{aligned}$$

Thus, from the moment property of the Fourier transform, the zeroth moment of ψ vanishes. □

Theorem 4.42 (Relationship between continuous and discrete moments)**.** *Let ϕ be an M-adic scaling function with scaling equation coefficient sequence c. Let ψ be a corresponding wavelet function with wavelet equation coefficient sequence d. Denote the kth moments of ϕ and ψ as μ_k and ν_k, respectively. Denote the kth moments of c and d as m_k and n_k, respectively. Then, we have*

$$\mu_k = \frac{1}{M^{1/2}(M^k-1)} \sum_{q=0}^{k-1} \binom{k}{q} m_{k-q}\mu_q \text{ for } k \geq 1 \quad \text{and} \tag{4.72}$$

$$\nu_k = M^{-k-1/2} \sum_{q=0}^{k} \binom{k}{q} n_{k-q}\mu_q \text{ for } k \geq 0. \tag{4.73}$$

4.3.11 Support of Scaling and Wavelet Functions

One might wonder if there is any simple way to characterize the support of scaling and wavelet functions. This is, in fact, possible. As it turns out, there is a relationship between the support of scaling and wavelet functions and their corresponding scaling and wavelet sequences. This relationship is introduced by the theorem below.

Theorem 4.43 (Support of scaling and wavelet functions)**.** *Let ϕ be an M-adic scaling function with scaling coefficient sequence c. Let ψ be a corresponding wavelet function with wavelet equation coefficient sequence d. If $c[n] = 0$ whenever $n < n_0$ or $n > n_1$, then*

$$\operatorname{supp}\phi \subset [\tfrac{n_0}{M-1}, \tfrac{n_1}{M-1}]$$

If, in addition, $d[n] = 0$ whenever $n < w_0$ or $n > w_1$, then

$$\operatorname{supp}\psi \subset [\tfrac{n_0+(M-1)w_0}{M(M-1)}, \tfrac{n_1+(M-1)w_1}{M(M-1)}].$$

Proof. The difficult part of the proof is to show that ϕ must have compact support. So, we simply assert this to be the case, and proceed based on this fact. Suppose that

$$\operatorname{supp}\phi \subset [a_0, a_1].$$

Then, we have

$$\operatorname{supp}\phi(M\cdot -k) \subset [\tfrac{a_0+k}{M}, \tfrac{a_1+k}{M}].$$

From the scaling equation, we have

$$\operatorname{supp}\phi = \operatorname{supp}\left[\sum_{k=n_0}^{n_1} c[k]\phi(Mt-k)\right].$$

Since ϕ is compactly supported, the lower support bound of the right-hand side is given by the lower support bound of the leftmost shifted version of $\phi(Mt)$. Similarly, the upper support bound of the right-hand side is given by the upper support bound of the rightmost shifted version of $\phi(Mt)$. Thus, we have

$$\operatorname{supp}\phi \subset [\tfrac{a_0+n_0}{M}, \tfrac{a_1+n_1}{M}].$$

Equating this expression for $\operatorname{supp}\phi$ with our initial one from above, we obtain

$$a_0 = \tfrac{a_0+n_0}{M} \Rightarrow a_0 = \tfrac{n_0}{M-1} \quad \text{and}$$
$$a_1 = \tfrac{a_1+n_1}{M} \Rightarrow a_1 = \tfrac{n_1}{M-1}.$$

Thus, the support of ϕ is as stated in the theorem. Using the previous result, we have

$$\begin{aligned}\operatorname{supp}\phi(M\cdot-k) &\subset \left[\frac{\frac{n_0}{M-1}+k}{M}, \frac{\frac{n_1}{M-1}+k}{M}\right] \\ &= \left[\tfrac{n_0+(M-1)k}{M(M-1)}, \tfrac{n_1+(M-1)k}{M(M-1)}\right].\end{aligned}$$

Using an argument similar to above, we obtain the lower and upper support bounds of ψ stated in the theorem. □

Below, we provide an example showing how the preceding theorem (i.e., Theorem 4.43) can be used to determine the support of scaling and wavelet functions.

Example 4.34. Consider the Belogay-Wang wavelet system introduced earlier in Example 4.33. Let ϕ denote the scaling function. Let ψ_1 and ψ_2 denote the two wavelet functions. Let c denote the scaling equation coefficient sequence of ϕ, and let d_1 and d_2 denote the wavelet equation coefficient sequences of ψ_1 and ψ_2, respectively. Find the support of ϕ, ψ_1, and ψ_2.

Solution. We use the result of Theorem 4.43 in what follows. Since $c[n] = 0$ whenever $n < 0$ or $n > 5$, we have

$$\operatorname{supp}\phi \subset \left[\tfrac{0}{3-1}, \tfrac{5}{3-1}\right] = [0, \tfrac{5}{2}].$$

Since $d_1[n] = 0$ whenever $n < 0$ or $n > 5$, we have

$$\operatorname{supp}\psi_1 \subset \left[\tfrac{0+(3-1)0}{3(3-1)}, \tfrac{5+(3-1)5}{3(3-1)}\right] = \left[0, \tfrac{15}{6}\right] = \left[0, \tfrac{5}{2}\right].$$

Since $d_2[n] = 0$ whenever $n < 0$ or $n > 2$, we have

$$\operatorname{supp}\psi_2 \subset \left[\tfrac{0+(3-1)0}{3(3-1)}, \tfrac{5+(3-1)2}{3(3-1)}\right] = \left[0, \tfrac{9}{6}\right] = \left[0, \tfrac{3}{2}\right].$$

One can see that these results are consistent with the plots of ϕ, ψ_1, and ψ_2 given earlier in Figures 4.25(a), 4.25(b), and 4.25(c), respectively. □

4.3.12 Order of Approximation

To begin, we introduce a generalization of the sum rule introduced in Theorem 4.28.

Theorem 4.44 (Sum rule). *For a sequence c (defined on $\mathbb{Z}$), $\hat{c}$ has a ηth order zero at $\frac{2\pi l}{M}$ for all $l \in \{1,2,\ldots,M-1\}$ if and only if*

$$\sum_{n\in\mathbb{Z}} W_M^{ln} n^k c[n] = 0 \text{ for all } (l,k) \in \{1,2,\ldots,M-1\} \times \{0,1,\ldots,\eta-1\},$$

where $W_M = e^{-j2\pi/M}$. The above condition is sometimes referred to as a sum rule of order η,

Proof. Differentiating $\hat{c}$ k times, we obtain

$$\hat{c}^{(k)}(\omega) = (-j)^k \sum_{n\in\mathbb{Z}} n^k c[n] e^{-jn\omega}.$$

Letting $\omega = \frac{2\pi l}{M}$, we have

$$\begin{aligned}\hat{c}^{(k)}(\tfrac{2\pi l}{M}) &= (-j)^k \sum_{n\in\mathbb{Z}} n^k c[n] e^{-j2\pi nl/M} \\ &= (-j)^k \sum_{n\in\mathbb{Z}} W_M^{ln} n^k c[n].\end{aligned}$$

Thus, $\hat{c}$ has a ηth order zero at $\frac{2\pi l}{M}$ if and only if the condition stated in the theorem holds. □

In order to characterize the approximation properties of scaling functions, we will need the important result from approximation theory given in the theorem below.

Theorem 4.45 (Strang-Fix condition)**.** *Let ϕ be an M-adic scaling function with the scaling equation coefficient sequence c. If $\hat{c}(\omega)$ has a pth order zero at $\frac{2\pi k}{M}$ for each $k \in \{1,2,\ldots,M-1\}$, then $\hat{\phi}(\omega)$ must have a pth order zero at all frequencies of the form $\omega = 2\pi k$, where $k \in \mathbb{Z} \setminus \{0\}$.*

Proof. Consider the quantity $\hat{\phi}(2\pi k)$ for any arbitrary $k \in \mathbb{Z} \setminus \{0\}$. We can express k in the form of $k = M^n l$, where $n, l \in \mathbb{Z}$, $n \geq 0$, and $M \not| l$. Furthermore, we can (uniquely) decompose l as $l = M\alpha + \beta$, where $\alpha \in \mathbb{Z}$ and $\beta \in \{1,2,\ldots,M-1\}$. Now, we apply formula (4.12) recursively n times to obtain

$$\begin{aligned}
\hat{\phi}(\omega) &= \tfrac{1}{\sqrt{M}}\hat{c}(\omega/M)\hat{\phi}(\omega/M) \\
&= \left[\prod_{q=1}^{n+1} \tfrac{1}{\sqrt{M}}\hat{c}(M^{-q}\omega)\right]\hat{\phi}(M^{-(n+1)}\omega) \\
&= \tfrac{1}{\sqrt{M}}\hat{c}(M^{-(n+1)}\omega)\left[\prod_{q=1}^{n} \tfrac{1}{\sqrt{M}}\hat{c}(M^{-q}\omega)\right]\hat{\phi}(M^{-n-1}\omega) \\
&= \tfrac{1}{\sqrt{M}}\hat{c}(M^{-n-1}\omega)\left[\prod_{q=1}^{n} \tfrac{1}{\sqrt{M}}\hat{c}(M^{-q}\omega)\right]\hat{\phi}(M^{-n-1}\omega).
\end{aligned}$$

Now, we use the preceding equation to evaluate $\hat{\phi}(\omega)$ at $\omega = 2\pi k = 2\pi M^n(M\alpha+\beta)$. Consider the factor $\hat{c}(M^{-n-1}\omega)$ above. Using the 2π-periodicity of $\hat{c}$, we can write

$$\begin{aligned}
\hat{c}(M^{-n-1}\omega) &= \hat{c}(M^{-n-1}2\pi M^n(M\alpha+\beta)) \\
&= \hat{c}(2\pi M^{-1}(M\alpha+\beta)) \\
&= \hat{c}(2\pi\alpha + \tfrac{2\pi\beta}{M}) \\
&= \hat{c}(\tfrac{2\pi\beta}{M}).
\end{aligned}$$

So, in effect, we have

$$\phi(2\pi k) = M^{-1/2}\hat{c}(\tfrac{2\pi\beta}{M})\left[\prod_{q=1}^{n} \tfrac{\sqrt{M}}{\sqrt{M}}\right]\hat{\phi}(M^{-n-1}2\pi k).$$

In other words, we are evaluating $\hat{c}$ at $\frac{2\pi\beta}{M}$ for $\beta \in \{1,2,\ldots,M-1\}$. By assumption, however, $\hat{c}$ has a pth order zero at $\frac{2\pi\beta}{M}$. Consequently, $\hat{\phi}(\omega)$ has a pth order zero at $2\pi k$. □

Now, we are in a position to consider the approximation properties of scaling functions more closely. The approximation properties of scaling functions are summarized by the following theorem.

Theorem 4.46 (Order of approximation)**.** *Let ϕ be an M-adic scaling function with the scaling equation coefficient sequence c. Let $\{\tilde{\psi}_q\}_{q\in\{1,2,\ldots,M-1\}}$ denote the corresponding dual wavelet functions with the respective wavelet equation coefficient sequences $\{\tilde{d}_q\}_{q\in\{1,2,\ldots,M-1\}}$. Suppose that $\hat{\phi}$ and $\hat{\tilde{\psi}}_q$ are $\eta-1$ times differentiable. Then, the following statements are equivalent:*

1. *ϕ has approximation order η;*
2. *linear combinations of $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ can locally reproduce polynomials of degree less than η;*
3. *$\hat{c}$ has a ηth order zero at $\frac{2\pi k}{M}$ for all $k \in \{1,2,\ldots,M-1\}$;*
4. *for each $q \in \{1,2,\ldots,M-1\}$, $\tilde{\psi}_q$ has all of its moments of order less than η vanish (i.e., $\tilde{\psi}_q$ has η vanishing moments);*

5. *for each* $q \in \{1, 2, \ldots, M-1\}$, $\hat{\hat{d}}_q$ *has a* η*th order zero at* 0.

Using the above theorem, we can determine the approximation order of a scaling function relatively easily. This is illustrated by the example below.

Example 4.35 (Belogay-Wang scaling function)**.** Let ϕ denote the 3-adic Belogay-Wang scaling function (introduced earlier) with the scaling equation coefficient sequence c. The nonzero coefficients of c are given by

$$(c[0], c[1], \ldots, c[5]) = \left(\tfrac{2-\sqrt{6}}{4}, \tfrac{1}{2}, \tfrac{2+\sqrt{6}}{4}, \tfrac{2+\sqrt{6}}{4}, \tfrac{1}{2}, \tfrac{2-\sqrt{6}}{4}\right).$$

Determine the approximation accuracy of ϕ.

Solution. Let $W = e^{-j2\pi/3}$. First, we consider a sum rule of order one. We have

$$\begin{aligned}\sum_{n=0}^{5} W^n c[n] &= W^0 c[0] + W^1 c[1] + W^2 c[2] + W^3 c[3] + W^4 c[4] + W^5 c[5] \\ &= \tfrac{2-\sqrt{6}}{4} + \tfrac{1}{2} e^{-j2\pi/3} + \tfrac{2+\sqrt{6}}{4} e^{-j4\pi/3} + \tfrac{2+\sqrt{6}}{4} + \tfrac{1}{2} e^{-j2\pi/3} + \tfrac{2-\sqrt{6}}{4} e^{-j4\pi/3} \\ &= 1 + e^{-j2\pi/3} + e^{-j4\pi/3} \\ &= 0 \quad \text{and}\end{aligned}$$

$$\begin{aligned}\sum_{n=0}^{5} W^{2n} c[n] &= W^0 c[0] + W^2 c[1] + W^4 c[2] + W^6 c[3] + W^8 c[4] + W^{10} c[5] \\ &= \tfrac{2-\sqrt{6}}{4} + \tfrac{1}{2} e^{-j4\pi/3} + \tfrac{2+\sqrt{6}}{4} e^{-j2\pi/3} + \tfrac{2+\sqrt{6}}{4} + \tfrac{1}{2} e^{-j4\pi/3} + \tfrac{2-\sqrt{6}}{4} e^{-j2\pi/3} \\ &= 1 + e^{-j2\pi/3} + e^{-j4\pi/3} \\ &= 0.\end{aligned}$$

Now, we consider a sum rule of order two. We have

$$\begin{aligned}\sum_{n=0}^{5} W^n n c[n] &= W(1)c[1] + W^2(2)c[2] + W^3(3)c[3] + W^4(4)c[4] + W^5(5)c[5] \\ &= \tfrac{1}{2} e^{-j2\pi/3} + 2\left(\tfrac{2+\sqrt{6}}{4}\right) e^{-j4\pi/3} + 3\left(\tfrac{2+\sqrt{6}}{4}\right) + 4(\tfrac{1}{2}) e^{-j2\pi/3} + 5\left(\tfrac{2-\sqrt{6}}{4}\right) e^{-j4\pi/3} \\ &= \tfrac{5}{2} e^{-j2\pi/3} + \tfrac{14-3\sqrt{6}}{4} e^{-j4\pi/3} + \tfrac{6+3\sqrt{6}}{4} \\ &\approx 1.2557 - j0.7250 \\ &\neq 0.\end{aligned}$$

Therefore, ϕ satisfies a sum rule of order one. Thus, ϕ can locally reproduce constant functions. □

4.3.13 Determination of Scaling and Wavelet Functions

The approaches for solving refinement equations introduced earlier in the 2-scale case can be easily extended to the M-scale case. For example, the cascade algorithm can be easily generalized to the M-scale case by replacing the constant 2 with M in various places in formulas.

4.4 Additional Reading

For additional information on wavelets to supplement the contents of this chapter, the reader can refer to [3, 4, 5, 6, 7, 9, 10, 13, 14, 16, 17, 18, 21, 22, 24, 25].

4.5 Problems

4.1 For each function θ given below, determine whether $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis (for its closed linear span). If $\{\theta(\cdot - k)\}_{k\in\mathbb{Z}}$ is a Riesz basis, indicate its Riesz bounds.
(a) $\theta(t) = \chi_{[0,2)}(t)$;
(b) $\theta(t) = \chi_{[0,1/2)}(t) - \chi_{[1/2,1)}(t)$;
(c) $\theta(t) = \begin{cases} 1-|t| & 0 \le |t| < 1 \\ 0 & \text{otherwise;} \end{cases}$
(d) $\theta(t) = \left(\cos \frac{3\pi}{2}t\right) \operatorname{sinc} \frac{\pi}{2}t$;
(e) $\theta(t) = \begin{cases} 1-t & |t| \le 1 \\ 0 & \text{otherwise.} \end{cases}$

4.2 Let $\{V_p\}_{p\in\mathbb{Z}}$ be a MRA and let $\{W_p\}_{p\in\mathbb{Z}}$ denote the corresponding sequence of wavelet spaces. Suppose that $f \in V_0$ and $g \in V_{-1}$.
(a) Show that it is not necessarily true that $f - g \in W_0$.
(b) Determine a necessary and sufficient condition on f and g for $f - g$ to be in W_0.

4.3 Show that the wavelet spaces $\{W_p\}_{p\in\mathbb{Z}}$ associated with a MRA satisfy the:
(a) scale invariance property (i.e., for all $p \in \mathbb{Z}$, $f(t) \in W_p \Leftrightarrow f(2t) \in W_{p-1}$).
(b) shift invariance property (i.e., $f(t) \in W_0 \Leftrightarrow f(t-k) \in W_0$, for all $k \in \mathbb{Z}$).

4.4 Define the function

$$f(t) = \begin{cases} t^2+1 & \text{for } -1 \le t < 2 \\ 0 & \text{otherwise.} \end{cases}$$

Let $\{V_p\}_{p\in\mathbb{Z}}$ be the MRA generated by the Haar scaling function $\phi = \chi_{[0,1)}$, where $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis. Find the orthogonal projection of f onto V_0.

4.5 Show that the Gaussian function $\phi(t) = e^{-t^2}$ cannot be the scaling function of a MRA. [Hint: Assume that $\phi(t)$ satisfies a refinement equation, and then show that this leads to a contradiction by considering the Fourier transform of both sides of the refinement equation.] [Note: Table E.1 might be helpful in determining $\hat{\phi}$.]

4.6 Let $\phi_1(t)$ and $\phi_2(t)$ be solutions to the refinement equations with coefficient sequences $c_1[n]$ and $c_2[n]$, respectively.
(a) Show that $\phi(t) = \phi_1 * \phi_2(t)$ satisfies a refinement equation.
(b) Show that $\phi(t) = (\phi_1\phi_2)(t)$ does not generally satisfy a refinement equation. State a condition on ϕ_1 and ϕ_2 that is sufficient for ϕ to be refinable.
(c) Show that the autocorrelation of ϕ_1 satisfies a refinement equation. [Note: The autocorrelation r of a function f is defined as $r(t) = \int_{\mathbb{R}} f(\tau)f(\tau - t)d\tau$.]

4.7 Let ϕ be a scaling function satisfying the refinement relationship

$$\phi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} c[k]\phi(2t-k).$$

(a) Show that ϕ is symmetric if and only if the coefficient sequence c is symmetric.
(b) Show that ϕ is antisymmetric if and only if the coefficient sequence c is antisymmetric.
[Hint: For both parts of this problem, use the Fourier domain.]

4.8 Let ϕ be a solution to the refinement equation

$$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} c[n]\phi(2t-n),$$

where $\phi \in L^1(\mathbb{R})$ and the zeroth moment of ϕ does not vanish. By integrating both sides of the refinement equation, show that

$$\sum_{n\in\mathbb{Z}} c[n] = \sqrt{2}.$$

4.9 Let ϕ be a function in $L^2(\mathbb{R})$ satisfying the refinement relation

$$\phi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} c[k]\phi(2t-k)$$

and also satisfying the interpolation condition given by

$$\phi(n) = \delta[n] \quad \text{for all } n \in \mathbb{Z}.$$

(a) Show that $c[n] = \frac{1}{\sqrt{2}}\delta[n]$ for even n.
(b) Show that $\hat{c}(\omega) + \hat{c}(\omega+\pi) = \sqrt{2}$.

4.10 Show that if ϕ is an orthogonal scaling function, then

$$\sum_{k\in\mathbb{Z}} \phi(t-k) = 1.$$

4.11 Let ϕ denote the solution to the refinement equation

$$\phi(t) = \sum_{k\in\mathbb{Z}} c_k\phi(2t-k),$$

satisfying $\int_{-\infty}^{\infty}\phi(t) = 1$. For each refinement mask c_n given below, use the eigenvalue method to find the exact value of $\phi(t)$ at $t = \frac{1}{2}$.
(a) $c_0 = 1$, $c_1 = 1$, and $c_n = 0$ for $n < 0 \cup n > 1$.
(b) $c_{-1} = \frac{1}{2}$, $c_0 = 1$, $c_1 = \frac{1}{2}$, and $c_n = 0$ for $n < -1 \cup n > 1$.
(c) $c_0 = \frac{1+\sqrt{3}}{4}$, $c_1 = \frac{3+\sqrt{3}}{4}$, $c_2 = \frac{3-\sqrt{3}}{4}$, $c_3 = \frac{1-\sqrt{3}}{4}$, and $c_n = 0$ for $n < 0 \cup n > 3$.

4.12 Implement the eigenvalue method for solving refinement equations in software. Then, for each of the sequences $c[k]$ given below, use the resulting software to plot scaling function ϕ associated with the refinement equation

$$\phi(t) = \sum_{k\in\mathbb{Z}} c_k\phi(2t-k).$$

(a) $c_0 = 1$, $c_1 = 1$, and $c_n = 0$ for $n < 0 \cup n > 1$.
(b) $c_{-1} = \frac{1}{2}$, $c_0 = 1$, $c_1 = \frac{1}{2}$, and $c_n = 0$ for $n < -1 \cup n > 1$.
(c) $c_0 = \frac{1+\sqrt{3}}{4}$, $c_1 = \frac{3+\sqrt{3}}{4}$, $c_2 = \frac{3-\sqrt{3}}{4}$, $c_3 = \frac{1-\sqrt{3}}{4}$, and $c_n = 0$ for $n < 0 \cup n > 3$.

4.13 Suppose that we have a MRA with a scaling function ϕ and dual wavelet function $\tilde{\psi}$, where $\tilde{\psi}$ has η vanishing moments. Let μ_k denote the kth moment of ϕ. For each of the cases below, represent the polynomial $f(t)$ on the interval I using an expression of the form (with the fewest possible number of terms)

$$f(t) = \sum_{n=n_0}^{n_1} a_n\phi(t-n).$$

(a) $\operatorname{supp}\phi \subset [-2,3]$, $\eta = 3$, $f(t) = 1 - 3t + 2t^2$, $I = [0,3]$.
(b) $\operatorname{supp}\phi \subset [0,3]$, $\eta = 2$, $f(t) = 5t + 2$, $I = [3,6]$.

4.14 Let ϕ be a scaling function with a corresponding scaling equation coefficient sequence $c[n]$. Let μ_k denote the kth moment of ϕ. Suppose that $\mu_k = 0$ for $1, 2, \ldots, \eta$. Show that, for $p < \eta$,

$$\sum_{k \in \mathbb{Z}} k^p \phi(k) = \int_{-\infty}^{\infty} t^p \phi(t) dt.$$

4.15 Let ϕ be a scaling function that is associated with a halfband filter. Determine an expression for $\phi(t)$ at integer values of t.

4.16 Let ϕ be a scaling function with scaling coefficient sequence c. Let m_k and μ_k denote the kth moments of c and ϕ, respectively. Suppose that ϕ is normalized such that $\mu_0 = 1$. For each of the cases below, determine the quantities m_0, m_1, m_2, m_3, μ_1, μ_2, and μ_3.
(a) $c_0 = \frac{1}{\sqrt{2}}$, $c_1 = \frac{1}{8\sqrt{2}}$, $c_2 = -\frac{1}{8\sqrt{2}}$, $c_k = 0$ for $k \geq 3$, and $c_k = c_{-1-k}$.
(b) $c_0 = \frac{1}{\sqrt{2}}$, $c_1 = \frac{1}{2\sqrt{2}}$, $c_k = 0$ for $k \geq 2$, and $c_k = c_{-k}$.

4.17 Let ϕ denote a scaling function with the refinement mask c_n. For each of the coefficient sequences given below, determine the approximation order of ϕ.
(a) $c_0 = \frac{1}{2}$, $c_1 = \frac{1}{2}$, $c_n = 0$ for $n \notin \{0, 1\}$;
(b) $c_{-1} = \frac{1}{4}$, $c_0 = \frac{1}{2}$, $c_1 = \frac{1}{4}$, $c_n = 0$ for $n \notin \{-1, 0, 1\}$;
(c) $c_{-3} = -\frac{1}{16}$, $c_{-2} = 0$, $c_{-1} = \frac{9}{16}$, $c_0 = 1$, $c_1 = \frac{9}{16}$, $c_2 = 0$, $c_3 = -\frac{1}{16}$, $c_n = 0$ for $n \notin \{-3, -2, -1, 0, 1, 2, 3\}$;
(d) $c_0 = \frac{1+\sqrt{3}}{8}$, $c_1 = \frac{3+\sqrt{3}}{8}$, $c_2 = \frac{3-\sqrt{3}}{8}$, $c_3 = \frac{1-\sqrt{3}}{8}$, $c_n = 0$ for $n \notin \{0, 1, 2, 3\}$.
(e) $c_{-3} = -\frac{1}{16}$, $c_{-1} = \frac{9}{16}$, $c_0 = 1$, $c_1 = \frac{9}{16}$, $c_3 = -\frac{1}{16}$, $c_n = 0$ otherwise.
(f) $(c_{-4}, c_{-3}, c_{-2}, c_{-1}, c_0, c_1, c_2, c_3, c_4) = \left(\frac{1}{32}, 0, -\frac{1}{4}, \frac{1}{2}, \frac{23}{16}, \frac{1}{2}, -\frac{1}{4}, 0, \frac{1}{32}\right)$, $c_n = 0$ otherwise.
(g) $(c_0, c_1, \ldots, c_{12}) = \frac{1}{256}(-1, 0, 18, -16, -63, 144, 348, 144, -63, -16, 18, 0, -1)$ and $c_n = 0$ otherwise.
(h) $(c_{-4}, c_{-3}, \ldots, c_5) = \frac{1}{128}(3, -3, -22, 22, 128, 128, 22, -22, -3, 3)$ and $c_n = 0$ otherwise.
(i) $(c_{-3}, c_{-2}, \ldots, c_2) = \frac{1}{8}(-1, 1, 8, 8, 1, -1)$ and $c_n = 0$ otherwise.

4.18 Let ϕ be a continuous function with fast decay that satisfies refinement equation

$$\phi(t) = \sum_{n \in \mathbb{Z}} c[n] \phi(2t - n).$$

Show that

$$\sum_{n \in \mathbb{Z}} c[2n] = \sum_{n \in \mathbb{Z}} c[2n+1] = 1 \quad \Rightarrow \quad \sum_{n \in \mathbb{Z}} \phi(t - n) = \mu_0,$$

where μ_0 is a nonzero constant.

4.19 Let ϕ be a continuous and differentiable function that satisfies the refinement equation

$$\phi(t) = \sum_{k \in \mathbb{Z}} c[k] \phi(2t - k),$$

where $\sum_{k \in \mathbb{Z}} c[2k] = \sum_{k \in \mathbb{Z}} c[2k+1] = 1$. Let ϕ' denote the first derivative of ϕ. Show that $\phi'(t)$ is refinable (i.e., satisfies a refinement equation).

4.20 Let ϕ and ψ denote the scaling and wavelet functions of a MRA, respectively. Let c and d denote the coefficient sequences of the scaling and wavelet equations, respectively. Let $\tilde{\phi}$ and $\tilde{\psi}$ denote the corresponding dual scaling and wavelet functions, respectively. Let $\tilde{c}$ and $\tilde{d}$ denote the coefficient sequences of the dual scaling and wavelet equations, respectively. For each of the cases below, find $\tilde{c}$ and $\tilde{d}$.
(a) $c_0 = \frac{1}{\sqrt{2}}$, $c_1 = \frac{1}{\sqrt{2}}$, $c_n = 0$ for $n \notin \{0, 1\}$; and $d_{-2} = \frac{1}{\sqrt{2}}$, $d_{-1} = -\frac{1}{\sqrt{2}}$, $d_n = 0$ for $n \notin \{-2, -1\}$;
(b) $c_0 = -\frac{1}{4\sqrt{2}}$, $c_1 = \frac{1}{2\sqrt{2}}$, $c_2 = \frac{3}{2\sqrt{2}}$, $c_3 = \frac{1}{2\sqrt{2}}$, $c_4 = -\frac{1}{4\sqrt{2}}$, $c_n = 0$ for $n \notin \{0, 1, 2, 3, 4\}$; and $d_0 = \frac{1}{2\sqrt{2}}$, $d_1 = -\frac{1}{\sqrt{2}}$, $d_2 = \frac{1}{2\sqrt{2}}$, $d_n = 0$ for $n \notin \{0, 1, 2\}$;

4.21 Let ϕ be an M-adic refinable function with compact support and nonvanishing zeroth moment μ_0, where

$$\phi(t) = \sqrt{M} \sum_{k \in \mathbb{Z}} c_k \phi(Mt - k) \quad (\text{where } M \in \mathbb{N}).$$

(a) Show that $\hat{c}(0) = \sqrt{M}$.
(b) One can show that $\hat{c}(2\pi k/M) = 0$ for $k = 1, 2, \ldots, M-1$. Using this fact, prove that ϕ must satisfy $\hat{\phi}(2\pi k/M) = \mu_0 \delta[k]$ for all $k \in \mathbb{Z}$.

4.22 Develop a software implementation of the cascade algorithm that can be used to solve a refinement equation of the following form:

$$\phi(t) = \sum_{n \in \mathbb{Z}} c[n] \phi(2t - n).$$

Then, for the case of each sequence c given below, use this software to plot the solution ϕ to the preceding equation having a zeroth moment of one. Use at least eight iterations of the cascade algorithm in order to ensure a reasonably accurate result. [Note: In each plot, ensure that the horizontal and vertical axes are both correctly labelled.]
(a) $(c_0, c_1, c_2, c_3) = \left(\frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}\right)$;
(b) $(c_0, c_1, c_2) = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$;
(c) $(c_0, c_1, c_2, c_3) = \left(\frac{1+\sqrt{3}}{4}, \frac{3+\sqrt{3}}{4}, \frac{3-\sqrt{3}}{4}, \frac{1-\sqrt{3}}{4}\right)$; and
(d) $c_0 = \frac{1+\sqrt{10}+\sqrt{5+2\sqrt{10}}}{16}$, $c_1 = \frac{5+\sqrt{10}+3\sqrt{5+2\sqrt{10}}}{16}$, $c_2 = \frac{5-\sqrt{10}+\sqrt{5+2\sqrt{10}}}{8}$, $c_3 = \frac{5-\sqrt{10}-\sqrt{5+2\sqrt{10}}}{8}$, $c_4 = \frac{5+\sqrt{10}-3\sqrt{5+2\sqrt{10}}}{16}$, and $c_5 = \frac{1+\sqrt{10}-\sqrt{5+2\sqrt{10}}}{16}$.

4.23 Let $\{V_p\}_{p \in \mathbb{Z}}$ be a MRA. Let $\hat{V}_p$ denote the space of Fourier transforms of functions in V_p. Show that the scale invariance property

$$f(t) \in V_p \Leftrightarrow f(2t) \in V_{p-1}$$

can be restated as

$$\hat{f}(\omega) \in \hat{V}_p \Leftrightarrow \hat{f}(2\omega) \in \hat{V}_{p+1}.$$

4.24 Let ϕ and ψ respectively denote the scaling and wavelet functions of a MRA. Let c and d denote the coefficient sequences of the corresponding scaling and wavelet equations. For each of the cases given below, find $\operatorname{supp}\phi$ and $\operatorname{supp}\psi$.
(a) $c_0 = \frac{1}{2}$, $c_1 = \frac{1}{2}$, $c_n = 0$ for $n \notin \{0,1\}$,
$d_0 = \frac{1}{2}$, $d_1 = -\frac{1}{2}$, $d_n = 0$ for $n \notin \{0,1\}$;
(b) $c_{-2} = -\frac{1}{8}$, $c_{-1} = \frac{1}{4}$, $c_0 = \frac{3}{4}$, $c_1 = \frac{1}{4}$, $c_2 = -\frac{1}{8}$, $c_n = 0$ for $n \notin \{-2,-1,0,1,2\}$,
$d_0 = -\frac{1}{2}$, $d_1 = 1$, $d_2 = -\frac{1}{2}$, $d_n = 0$ for $n \notin \{0,1,2\}$;
(c) $c_0 = \frac{1+\sqrt{3}}{4}$, $c_1 = \frac{3-\sqrt{3}}{4}$, $c_2 = \frac{3+\sqrt{3}}{4}$, $c_3 = \frac{1+\sqrt{3}}{4}$, $c_n = 0$ for $n \notin \{0,1,2,3\}$,
$d_0 = \frac{1+\sqrt{3}}{4}$, $d_1 = -\frac{3-\sqrt{3}}{4}$, $d_2 = \frac{3+\sqrt{3}}{4}$, $d_3 = -\frac{1+\sqrt{3}}{4}$, $d_n = 0$ for $n \notin \{0,1,2,3\}$.
(d) $(c_{-5}, c_{-4}, \ldots, c_5) = \frac{1}{128\sqrt{2}}(1,2,-7,0,70,124,70,0,-7,2,1)$ and $c_n = 0$ otherwise,
$(d_{-1}, d_0, \ldots, d_3) = -\frac{1}{4\sqrt{2}}(1,2,-6,2,1)$ and $d_n = 0$ otherwise.
(e) $(c_{-2}, c_{-1}, \ldots, c_2) = -\frac{1}{4\sqrt{2}}(1,-2,-6,-2,1)$ and $c_n = 0$ otherwise,
$(d_{-4}, d_{-3}, \ldots, d_6) = -\frac{1}{128\sqrt{2}}(1,-2,-7,0,70,-124,70,0,-7,-2,1)$ and $d_n = 0$ otherwise.

4.25 Let ϕ_1 and ϕ_2 be M-refinable functions with refinement masks c_1 and c_2, respectively. Show that $\phi = \phi_1 * \phi_2$ is also M-refinable.

4.6 Bibliography

[1] E. Belogay. Personal communication, July 2005.

[2] E. Belogay and Y. Wang. Compactly supported orthogonal symmetric scaling functions. *Applied and Computational Harmonic Analysis*, 7(2):137–150, September 1999.

[3] A. Boggess and F. J. Narcowich. *A First Course in Wavelets with Fourier Analysis*. Prentice Hall, Upper Saddle River, NJ, USA, 2001.

[4] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Upper Saddle River, NJ, USA, 1998.

[5] C. K. Chui. *An Introduction to Wavelets*. Academic Press, San Diego, CA, USA, 1992.

[6] C. K. Chui. *Wavelets: A Mathematical Tool for Signal Analysis*. SIAM, Philadelphia, PA, USA, 1997.

[7] M. A. Coffey and D. M. Etter. Multiresolution analysis on bounded domains for the design of biorthogonal wavelet bases. *IEEE Trans. on Signal Processing*, 50(3):509–519, 2002.

[8] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45:485–560, 1992.

[9] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

[10] L. Debnath. *Wavelet Transform and Their Applications*. Birkhauser, Boston, MA, USA, 2002.

[11] Federal Bureau of Investigation, Washington, DC, USA. *WSQ Gray-Scale Fingerprint Image Compression Specification, IAFIS-IC-0110(V3)*, December 1997.

[12] D. Le Gall and A. Tabatabai. Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 761–764, New York, NY, USA, April 1988.

[13] J. C. Goswami and A. K. Chan. *Fundamentals of Wavelets: Theory, Algorithms and Applications*. John Wiley & Sons, New York, NY, USA, 1999.

[14] D. Hong, J. Wang, and R. Gardner. *Real Analysis with an Introduction to Wavelets and Applications*. Elsevier, San Francisco, CA, USA, 2005.

[15] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.

[16] F. Keinert. *Wavelets and Multiwavelets*. Chapman & Hall/CRC, Boca Raton, FL, USA, 2004.

[17] A. K. Louis, P. Maab, and A. Rieder. *Wavelets: Theory and Applications*. Wiley, New York, NY, USA, 1998.

[18] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, USA, 2nd edition, 1999.

[19] D. Malone. Solutions to dilation equations. Technical report, University of Dublin, Ireland, 2001.

[20] A. Mertins. *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. John Wiley & Sons, Toronto, 2000.

[21] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, USA, 1996.

[22] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, USA, 1996.

[23] G. Strang, V. Strela, and D.-X. Zhou. Compactly supported refinable functions with infinite masks. Preprint, 1998.

[24] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[25] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A.K. Peters, Wellesley, MA, 1994.

Part III

Multidimensional Filter Banks and Multivariate Wavelets

Chapter 5

Multidimensional Systems Preliminaries

5.1 Introduction

Before we can study multidimensional multirate filter banks and multivariate wavelet systems, we must first introduce some mathematical preliminaries related to multidimensional signal processing.

5.2 Notation

To begin, we introduce some basic notation and terminology that is useful in the context of multidimensional signal processing.

Definition 5.1 (Multi-index)**.** A **multi-index** is an ordered n-tuple of (nonnegative) integers (i.e., an element of $(\mathbb{Z}^*)^n$).

Let $k = (k_1, k_2, \ldots, k_n)$ be a multi-index. The **length** (or size) of the multi-index k, denoted $|k|$, is defined as $|k| = \sum_{l=1}^{n} k_l$.

For two vectors $z = \begin{bmatrix} z_0 & z_1 & \cdots & z_{D-1} \end{bmatrix}^T$ and $n = \begin{bmatrix} n_0 & n_1 & \cdots & n_{D-1} \end{bmatrix}^T$, we define the quantity z^n as

$$z^n = \prod_{k=0}^{D-1} z_k^{n_k}.$$

Note that the quantity z^n is a scalar. Let z be a D-dimensional column vector and let M be a $D \times D$ matrix, where m_k denotes the kth column of M (i.e., $M = \begin{bmatrix} m_0 & m_1 & \cdots & m_{D-1} \end{bmatrix}$). Then, we define the quantity z^M as

$$z^M = \begin{bmatrix} z^{m_0} & z^{m_1} & \cdots & z^{m_{D-1}} \end{bmatrix}^T.$$

Note that the quantity z^M is a D-dimensional column vector. With the above definitions, one can show that, for any D-dimensional column vectors k, n, u, v, and w, any $D \times D$ matrices A, M, and L, and any scalar α, the following relationships hold:

$$\begin{aligned} w^{u+v} &= w^u w^v, \\ (\alpha k)^n &= \alpha^{|n|} k^n, \\ \left(z^M\right)^n &= z^{Mn}, && (5.1) \\ (z^L)^M &= z^{LM}, \\ u^n v^n &= (u \circ v)^n, \quad \text{and} && (5.2) \\ u^A \circ v^A &= (u \circ v)^A. && (5.3) \end{aligned}$$

Let M be a $D\times D$ matrix, where m_k denotes the kth column of M (i.e., $M = \begin{bmatrix} m_0 & m_1 & \cdots & m_{D-1}\end{bmatrix}$) and let ω be a D-dimensional column vector. Then, we define the quantity $e_M(\omega)$ as

$$e_M(\omega) = \begin{bmatrix} e^{j\omega^T m_0} & e^{j\omega^T m_1} & \ldots & e^{j\omega^T m_{D-1}}\end{bmatrix}^T.$$

(Note that our definition of $e_M(\omega)$ differs slightly from that used in [8].) In addition, we define the quantity W_M^k as

$$W_M^k = e_M(-j2\pi k) = \begin{bmatrix} e^{-j2\pi k^T m_0} & e^{-j2\pi k^T m_1} & \ldots & e^{-j2\pi k^T m_{D-1}}\end{bmatrix}^T.$$

Define $\omega = \begin{bmatrix} \omega_0 & \omega_1 & \cdots & \omega_{D-1}\end{bmatrix}^T$. From the above definition of $e_M(\omega)$, one can show that the following relationships hold:

$$e_I(\omega) = \begin{bmatrix} e^{j\omega_0} & e^{j\omega_1} & \ldots & e^{j\omega_{D-1}}\end{bmatrix}^T, \tag{5.4}$$

$$e_M(\omega)^n = e^{j\omega^T Mn}, \tag{5.5}$$

$$e_A(\omega)^B = e_{AB}(\omega), \tag{5.6}$$

$$e_A(\omega) = e_I(A^T\omega), \tag{5.7}$$

$$\cos\omega^T k = \tfrac{1}{2}\left(e_I(\omega)^k + e_I(\omega)^{-k}\right), \quad \text{and}$$

$$\sin\omega^T k = \tfrac{1}{2j}\left(e_I(\omega)^k - e_I(\omega)^{-k}\right).$$

Definition 5.2 (Polynomial)**.** A (multivariate) **polynomial** is a function or sequence f of the form

$$f(\lambda) = \sum_{k\in(\mathbb{Z}^*)^D} a[k]\lambda^k,$$

where only finitely many of the $\{a[k]\}_{k\in(\mathbb{Z}^*)^D}$ are nonzero. We call $\{a[k]\}_{k\in(\mathbb{Z}^*)^D}$ the **coefficients** of the polynomial. The polynomial with all coefficients equal to zero (i.e., $f\equiv 0$) is called the **zero polynomial**. A polynomial with exactly one nonzero coefficient is called a **monomial**. Similarly, a polynomial with exactly two terms is called a **binomial**. For $f\not\equiv 0$, we define the **degree** of f, denoted $\deg f$, as

$$\deg f = \max\{|k| : k\in(\mathbb{Z}^*)^D, a[k]\neq 0\}$$

(i.e., the maximum of the coefficient index sizes of all nonzero terms). We define $\deg 0 = -\infty$.

Example 5.1. Let f be the polynomial function defined on $\mathbb{R}^2$ given by $f(t) = f(\begin{bmatrix} t_0 & t_1\end{bmatrix}^T) = 3t_0t_1 + 4t_0 - t_1 - 5t_0^5t_1^3$. Then, $\deg f = 8$. Let f be the polynomial sequence defined on $\mathbb{Z}^2$ given by $f[n] = f[\begin{bmatrix} n_0 & n_1\end{bmatrix}^T] = 3n_0n_1 - 4n_0^2n_1 + 2n_0^2n_1^5$. Then, $\deg f = 7$.

Definition 5.3 (Moment)**.** The kth **moment** of a function f defined on $\mathbb{R}^D$ is given by

$$\mu_k = \int_{\mathbb{R}^D} t^k f(t)dt$$

where $k\in(\mathbb{Z}^*)^D$ (i.e., the elements of k are all nonnegative integers). The **order** of the moment k is given by $|k|$ (i.e., the sum of the elements of k).

From the above definition, we can observe that a function has exactly one moment of order zero (namely, the one associated with $k=0$). Similarly, a function has exactly D (distinct) first order moments.

Definition 5.4 (Periodicity)**.** A sequence f defined on $\mathbb{Z}^D$ is said to be **periodic** if there exists some $D\times D$ nonsingular integer matrix M such that

$$f[n] = f[n+Mk] \quad \text{for all } k\in\mathbb{Z}^D.$$

We refer to M as a **periodicity matrix**. The columns of M are referred to as **periodicity vectors**.

One can show that the number of samples in a single period of a M-periodic function is given by $|\det M|$. The periodicity matrix is not unique.

5.3 Derivatives, Gradients, and Jacobians

Sometimes, we are interested in the derivatives of multivariate functions. Since such functions have more than one independent variable, we need to deal with partial derivatives. For this reason, it is convenient to define a compact notation for indicating a particular partial derivative of a multivariate function. In particular, we define

$$D_t^n f(t) = \frac{\partial^{|n|}}{\partial t_0^{n_0} \partial t_1^{n_1} \cdots \partial t_{D-1}^{n_{D-1}}}.$$

Thus, D_t^n denotes a $|n|$-th order partial derivative of f with respect to t. In the degenerate (1-D) case, we simply have that $D_t^n = \frac{d^n}{dt^n}$.

The **gradient** of the function f, denoted as ∇f, is defined as

$$\nabla_{\boldsymbol{t}}(f(\boldsymbol{t})) = \begin{bmatrix} \frac{\partial f}{\partial t_1} & \frac{\partial f}{\partial t_2} & \cdots & \frac{\partial f}{\partial t_n} \end{bmatrix}$$

where $t = \begin{bmatrix} t_1 & t_2 & \cdots & t_n \end{bmatrix}^T$. Note that the gradient is defined to be a row vector (as opposed to a column vector).

The **Jacobian** of the vector function f, denoted as Jf, is defined as

$$J(f(t)) = \frac{\partial(f_1, f_2, \ldots, f_n)}{\partial(t_1, t_2, \ldots, t_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial t_1} & \frac{\partial f_1}{\partial t_2} & \cdots & \frac{\partial f_1}{\partial t_n} \\ \frac{\partial f_2}{\partial t_1} & \frac{\partial f_2}{\partial t_2} & \cdots & \frac{\partial f_2}{\partial t_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial t_1} & \frac{\partial f_n}{\partial t_2} & \cdots & \frac{\partial f_n}{\partial t_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_n \end{bmatrix}$$

where $f = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}^T$ and $t = \begin{bmatrix} t_1 & t_2 & \cdots & t_n \end{bmatrix}^T$. The Jacobian is also sometimes denoted as $\frac{df}{dt}$.

5.4 Integration

Often, it is necessary to perform a change of variable involving a variable of integration. In the 1-D case, we have

$$\int_R f(x)dx = \int_{R'} f(x(u))x'(u)du$$

where $R = [a, b]$, $R' = [\alpha, \beta]$, $x(\alpha) = a$, and $x(\beta) = b$. In the D-dimensional case, we have

$$\int_R f(x)dx = \int_{R'} f(x(u)) \left|\det[(Jx)(u)]\right| du.$$

5.5 Lattices

In the one-dimensional case, periodic sampling is easily characterized by a single scalar value, namely the sampling period. In the multidimensional case, sampling is considerably more complicated. Sampling is characterized using lattices. A lattice is essentially a regularly spaced array of points. More formally, we have the following definition:

Definition 5.5 (Lattice)**.** Let $\{v_k\}_{k=1}^D$ be a set of linearly independent real vectors in $\mathbb{R}^D$. The set of all points $\sum_{k=1}^D a_k v_k$, where $\{a_k\}_{k=1}^D$ are integers, is called a (point) **lattice**.

Figure 5.1 shows four commonly used 2-D lattices, namely the integer, quincunx, hexagonal, and rectangular lattices.

If every point of the lattice $\mathcal{L}$ is also a point of the lattice $\mathcal{M}$, we say that $\mathcal{L}$ is a **sublattice** of $\mathcal{M}$.

Every lattice contains a set of linearly independent points $\{v_1, v_2, \ldots v_n\}$ such that every other point can be expressed as a linear combination of the form $\sum_{k=1}^n a_k v_k$. Thus, $\{v_1, v_2, \ldots, v_n\}$ form a basis for the lattice. We can place the basis vectors in a matrix $L = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$, yielding what is called a **generating matrix** L of the lattice $\mathcal{L}$. Each $x \in \mathcal{L}$ can be written as $x = Lk$ for some $k \in \mathbb{Z}^D$.

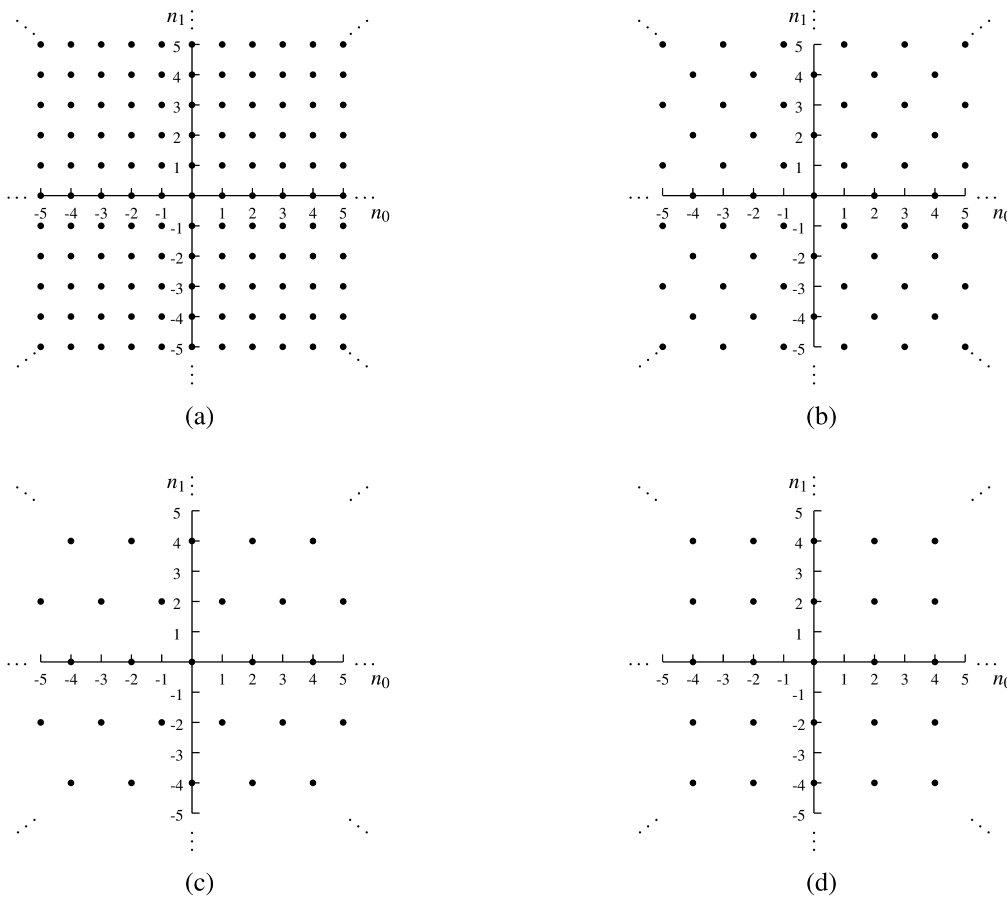


Figure 5.1: Examples of 2-D lattices. (a) Integer lattice. (b) Quincunx lattice. (c) Hexagonal lattice. (d) Rectangular lattice.

We sometimes also refer to a generating matrix as a **sampling matrix**, since a generating matrix defines a lattice on which a function can be sampled.

Since a generating matrix uniquely specifies a lattice, it is often convenient to identify a lattice by way of a generating matrix. We denote the lattice $\mathcal{L}$ generated by matrix L as either $\operatorname{LAT}(L)$ or simply $L\mathbb{Z}^D$.

It is important to note that a lattice does not have a unique generating matrix. In fact, one can show that if L is a generating matrix of $\mathcal{L}$, then for any unimodular integer matrix U, LU is also a generating matrix of $\mathcal{L}$. Postmultiplying the original generating matrix L by U corresponds to a (unimodular) change of basis. Every generating matrix of a lattice is of this form. While the generating matrix is not unique, the determinant of all generating matrices is the same up to sign. This fact motivates the definition given below.

Definition 5.6 (Determinant of lattice)**.** For a lattice $\mathcal{L}$ with a generating matrix L, we define the **determinant** of $\mathcal{L}$ as

$$\det \mathcal{L} = |\det L|.$$

The determinant of a lattice has an important geometric interpretation. In particular, the ratio of the density of points in the integer lattice to the density of points in the lattice $\mathcal{L}$ is given by $\det \mathcal{L}$.

Since we are often interested in expressing the determinant of a lattice in terms of its generating matrix, we introduce a new notational convention which allows us to do this more easily. In particular, we define $\mathcal{J}(M)$ as the determinant of the lattice generated by M. That is, we have

$$\mathcal{J}(M) = |\det M|.$$

If there exists a generating matrix for a lattice $\mathcal{L}$ that is diagonal, then the lattice $\mathcal{L}$ is said to be **separable**.

Example 5.2 (Integer lattice)**.** The integer lattice $\mathcal{L} = \mathbb{Z}^2$, shown in Figure 5.1(a), is separable with generating matrix $L = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and $\det \mathcal{L} = |\det L| = 1$.

Example 5.3 (Quincunx lattice)**.** The quincunx lattice $\mathcal{L}$, shown in Figure 5.1(b), is nonseparable with generating matrix $L = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$ and $\det \mathcal{L} = |\det L| = 2$.

Example 5.4 (Hexagonal lattice)**.** The hexagonal lattice $\mathcal{L}$, shown in Figure 5.1(c), is nonseparable with generating matrix $L = \left[\begin{smallmatrix} 2 & 1 \\ 0 & 2 \end{smallmatrix}\right]$ and $\det \mathcal{L} = |\det L| = 4$.

Example 5.5 (Rectangular lattice)**.** The rectangular lattice $\mathcal{L}$, shown in Figure 5.1(d), is separable with generating matrix $L = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ and $\det \mathcal{L} = |\det L| = 4$.

5.5.1 Cosets and Fundamental Domains

Definition 5.7 (Coset and coset group)**.** Let D be either a lattice contained in $\mathbb{R}^n$ or $\mathbb{R}^n$ itself, and let $\mathcal{L}$ be a lattice contained in D. With each $p \in D$, we can associate a translated version $p + \mathcal{L}$ of the lattice $\mathcal{L}$, called a **coset**. The set of all such cosets is referred to as the **coset group** of $\mathcal{L}$ with respect to D and is denoted as $D/\mathcal{L}$.

Let $\mathcal{L} \subset M$. Then, $x, y \in M$ are in the same coset of $\mathcal{L}$ with respect to D if and only if $x - y \in \mathcal{L}$.

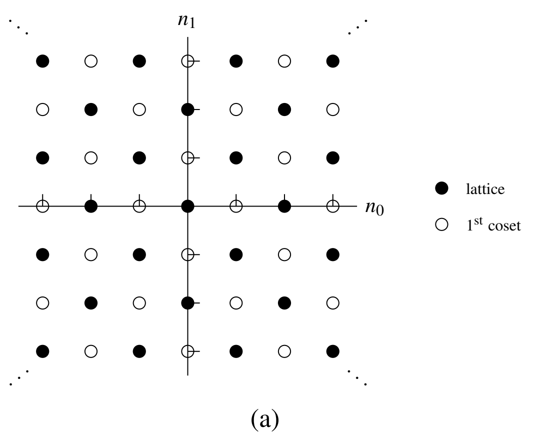
Figure 5.2 shows the coset groups of several lattices with respect to the lattice $\mathbb{Z}^2$ (i.e., the 2-D integer lattice).

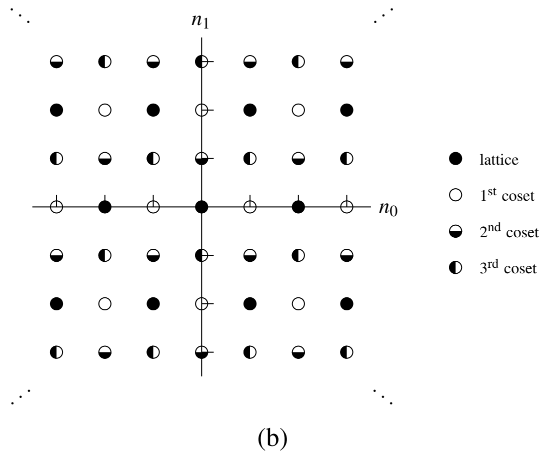
Definition 5.8 (Fundamental domain)**.** Let D be either a lattice contained in $\mathbb{R}^n$ or $\mathbb{R}^n$ itself, let P be an arbitrary subset of D, and let $\mathcal{L}$ be a lattice contained in D. The set P is said to be a **fundamental domain** of the lattice $\mathcal{L}$ in D if P intersects each coset of $D/\mathcal{L}$ in exactly one point.

A lattice $\mathcal{L}$ can be used to partition its embedding space into fundamental domains. Fundamental domains are particularly important in the context of $\mathcal{L}$-periodic functions, as knowing a $\mathcal{L}$-periodic function on one fundamental domain is equivalent to knowing the function over the entire domain of its definition.

Definition 5.9 (Voronoi cell)**.** For a lattice $\mathcal{L}$, the **Voronoi cell** of $\mathcal{L}$ is the set S of all points (in $\mathbb{R}^n$) for which there is no closer lattice point than the origin. That is, S is defined as

$$S = \{x \in \mathbb{R}^n : d(x, 0) \leq d(x, p) \text{ for all } p \in \mathcal{L}\}.$$





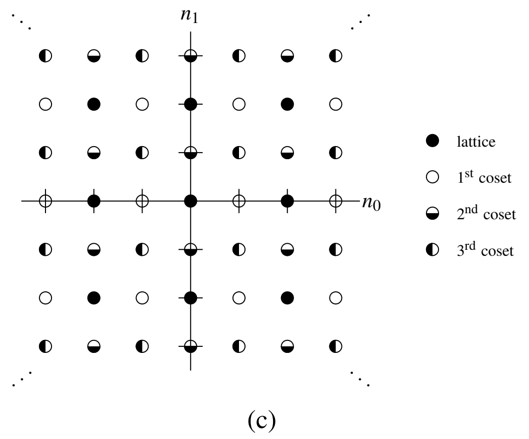


Figure 5.2: Cosets of various lattices. Cosets of the (a) quincunx, (b) hexagonal, and (c) rectangular lattices.

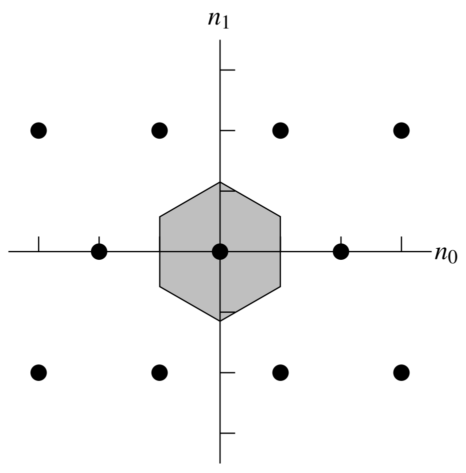


Figure 5.3: Voronoi cell

One fundamental domain that is often of interest is the fundamental parallelepiped as defined below.

Definition 5.10 (Fundamental parallelepiped)**.** For a lattice with generating matrix M, we define the **fundamental parallelepiped** (FPD) as

$$\operatorname{FPD}(M) = \{y : y = Mx, x \in [0,1)^D\}.$$

For a lattice $\mathcal{L}$ with generating matrix M, we have that $\det(M\mathcal{L})$ is the volume of $\operatorname{FPD}(M)$. We denote the set of all integer vectors inside $\operatorname{FPD}(M)$ as $\mathcal{N}(M)$. That is, we define

$$\mathcal{N}(M) = \operatorname{FPD}(M) \cap \mathbb{Z}^D.$$

The number of elements in $\mathcal{N}(M)$ is $\mathcal{J}(M)$.

Sometimes we are also interested in another parallelepiped region related to the FPD called the symmetric parallelepiped as defined below.

Definition 5.11 (Symmetric parallelepiped)**.** For a lattice $\mathcal{L}$ with generating matrix M, we define the **symmetric parallelepiped** (SPD) as

$$\operatorname{SPD}(M) = \{y : y = Mx, x \in [-1,1)^D\}.$$

Notice that the SPD is simply a shifted and scaled version of the FPD. That is, $\operatorname{SPD}(M) = \operatorname{FPD}(2M) - M1$.

The index of the sublattice $\mathcal{L}$ of $\mathcal{M}$ is the number of cosets in $\mathcal{M}$ with respect to $\mathcal{L}$. For example, the index of the quincunx lattice in $\mathbb{Z}^2$ is two.

Let $\mathcal{L}$ be a sublattice of lattice $\mathcal{M}$. Then, the index of $\mathcal{L}$ in $\mathcal{M}$ is $\det\mathcal{L}/\det\mathcal{M}$. This quantity indicates how much more densely lattice points are packed per unit volume in the original lattice $\mathcal{M}$ relative to the sublattice $\mathcal{L}$.

Let $\mathcal{L}$ be a lattice in $\mathbb{R}^D$ and let M be a $D \times D$ matrix. Then, $\det(A\mathcal{L}) = |\det A| \det\mathcal{L}$.

5.5.2 Reciprocal Lattices

Definition 5.12 (Reciprocal lattice)**.** Let $\mathcal{L}$ be a lattice with a basis $\{v_k\}_{k=1}^{D-1}$. Since $\{v_k\}_{k=1}^{D-1}$ is linearly independent, there must exist another set $\{u_k\}_{k=1}^{D-1}$ such that $\{v_k\}_{k=1}^{D-1}$ and $\{u_k\}_{k=1}^{D-1}$ are biorthonormal (i.e., $\langle u_k, v_l\rangle = \delta[k-l]$). The lattice with basis $\{u_k\}_{k=1}^{D-1}$ is called the **reciprocal lattice** of $\mathcal{L}$, and is denoted as $\mathcal{L}^*$. (The reciprocal lattice is also sometimes referred to as the dual lattice or polar lattice.)

The reciprocal lattice $\mathcal{L}^*$ of the lattice $\mathcal{L}$ is independent of the choice of basis for $\mathcal{L}$. This follows from the result of the following lemma.

Lemma 5.1. *The reciprocal lattice $\mathcal{L}^*$ of $\mathcal{L}$ consists of all vectors y such that $\langle y, x\rangle \in \mathbb{Z}$ for all $x \in \mathcal{L}$.*

From the definition of the reciprocal lattice, it follows that $(\mathcal{L}^*)^* = \mathcal{L}$.

Also, one can show that

$$\det\mathcal{L}\det(\mathcal{L}^*) = 1.$$

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be lattices. If $\mathcal{L}_1 \subset \mathcal{L}_2$ then $\mathcal{L}_2^* \subset \mathcal{L}_1^*$. In other words, set inclusions are reversed by reciprocation.

5.5.3 Miscellany

Example 5.6 (Quincunx sampling)**.** One generating matrix for the quincunx lattice is given by

$$M = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

We have

$$\begin{aligned}
M^2 &= 2I, \\
|\det M| &= 2, \\
M^T &= M, \\
M^{-1} = M^{-T} &= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}, \\
\mathcal{N}(M) = \mathcal{N}(M^T) &= \left\{ \begin{bmatrix} 0 & 0 \end{bmatrix}^T, \begin{bmatrix} 1 & 0 \end{bmatrix}^T \right\}, \\
2\pi M^{-T} &= \begin{bmatrix} \pi & \pi \\ \pi & -\pi \end{bmatrix}.
\end{aligned}$$

The aliasing frequency is given by

$$\begin{bmatrix} \pi & \pi \end{bmatrix}^T.$$

Example 5.7 (Rectangular sampling)**.** One generating matrix of the rectangular lattice is

$$M = 2I = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

We have

$$\begin{aligned}
|\det M| &= 4, \\
M^T &= M, \\
M^{-1} = M^{-T} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}, \\
\mathcal{N}(M) = \mathcal{N}(M^T) &= \left\{ \begin{bmatrix} 0 & 0 \end{bmatrix}^T, \begin{bmatrix} 1 & 0 \end{bmatrix}^T, \begin{bmatrix} 0 & 1 \end{bmatrix}^T, \begin{bmatrix} 1 & 1 \end{bmatrix}^T \right\}, \\
2\pi M^{-T} &= \begin{bmatrix} \pi & 0 \\ 0 & \pi \end{bmatrix}.
\end{aligned}$$

The aliasing frequencies are given by

$$\left\{ \begin{bmatrix} \pi & 0 \end{bmatrix}^T, \begin{bmatrix} 0 & \pi \end{bmatrix}^T, \begin{bmatrix} \pi & \pi \end{bmatrix}^T \right\}.$$

5.6 Fourier Analysis

In the following sections, we very briefly state several definitions and theorems that are of interest in the context of multidimensional signal processing.

Definition 5.13 (Convolution)**.** Let f and g be sequences. The **convolution** of f and g, denoted $f * g$, is defined as

$$(f * g)[n] = \sum_{k \in \mathbb{Z}^D} f[k] g[n-k].$$

Definition 5.14 ($\mathcal{Z}$ transform)**.** For a sequence x defined on $\mathbb{Z}^D$, we define the $\mathcal{Z}$ **transform** X of x as

$$X(z) = \sum_{n \in \mathbb{Z}^D} x[n] z^{-n}.$$

One can show that

$$y[n] = x[-n] \Leftrightarrow Y(z) = X(z^{-I}).$$

5.6.1 Continuous-Time Fourier Transform for $L^1(\mathbb{R}^d)$

Definition 5.15 (Continuous-time Fourier transform)**.** For a function $f \in L^1(\mathbb{R}^d)$, the (continuous-time) **Fourier transform** of f, denoted $\hat{f}$ or $\mathcal{F}(f)$, is defined as

$$(\mathcal{F}f)(\omega) = \hat{f}(\omega) = \int_{\mathbb{R}^d} f(t) e^{-j\omega^T t} dt,$$

for $\omega \in \mathbb{R}^d$.

The Fourier transform is invertible. The inverse Fourier transform is computed as per the theorem given below.

Theorem 5.1 (Inverse continuous-time Fourier transform)**.** *If $f \in L^1(\mathbb{R}^d)$ and $\hat{f} \in L^1(\mathbb{R}^d)$, then the inverse Fourier transform of $\hat{f}$ is given by*

$$(\mathcal{F}^{-1}f)(t) = f(t) = (2\pi)^{-D} \int_{\mathbb{R}^D} \hat{f}(\omega) e^{j\omega^T t} d\omega.$$

The Fourier transform has a number of important properties. Some of these properties are given below.

Theorem 5.2 (Fourier transform properties)**.** *The Fourier transform has the following properties:*

1. *for any nonsingular real matrix M, $\widehat{f(M\cdot)}(\omega) = \frac{1}{|\det M|} \hat{f}(M^{-T}\omega)$ (dilation);*
2. *for any $d \in \mathbb{R}^D$, $\widehat{f(\cdot - d)}(\omega) = e^{-j\omega^T d} \hat{f}(\omega)$ (translation).*

Theorem 5.3 (Time-domain differentiation)**.** *Let f be a continuous function in $L^1(\mathbb{R}^d)$ with a continuous, absolutely integrable nth partial derivative for any n satisfying $|n| \le k$. Then, there exists a constant C so that $|\hat{f}(\omega)| \le \frac{C}{(1+\|\omega\|)^k}$, and for each such n, we have*

$$\widehat{D^n_{(\cdot)} f(\cdot)}(\omega) = (j\omega)^n \hat{f}(\omega).$$

Theorem 5.4 (Fourier-domain differentiation)**.** *If $(1 + \|t\|)^k f(t) \in L^1(\mathbb{R}^d)$ for a positive integer k, then the Fourier transform of f has k continuous derivatives. The partial derivatives of $\hat{f}$ are given by*

$$D^n_\omega \hat{f}(\omega) = (-j)^{|n|} \widehat{t^n f}(\omega).$$

Theorem 5.5 (Moments)**.** *Let μ_k denote the kth moment of f. Then, we have*

$$\mu_k = \int_{\mathbb{R}^D} t^k f(t) dt = j^{|k|} (D^k_{(\cdot)} \hat{f}(\cdot))(0).$$

Theorem 5.6 (Riemann-Lebesgue lemma)**.** *If $f \in L^1(\mathbb{R}^d)$, then $\hat{f}$ is continuous, and*

$$\lim_{\|\omega\| \to \infty} \hat{f}(\omega) = 0.$$

5.6.2 Continuous-Time Fourier Transform for $L^2(\mathbb{R}^d)$

Definition 5.16 (Fourier transform)**.** Let $f \in L^2(\mathbb{R}^d)$. The Fourier transform of f, denoted $\hat{f}$, is defined as

$$\hat{f}(t) = \lim_{r \to \infty} \int_{\|t\| < r} f(t) e^{-j\omega^T t} dt.$$

Theorem 5.7 (Fourier inversion formula)**.** *If $f \in L^2(\mathbb{R}^d)$, then the inverse (continuous-time) Fourier transform of $\hat{f}$ is given by*

$$f(t) = \lim_{r \to \infty} (2\pi)^{-d} \int_{\|\omega\| < r} \hat{f}(\omega) e^{j\omega^T t} d\omega.$$

5.6.3 Discrete-Time Fourier Transform

Definition 5.17 (Discrete-time Fourier transform)**.** For a sequence f defined on $\mathbb{Z}^D$, the (discrete-time) **Fourier transform** of f, denoted $\hat{f}$ or $\mathcal{F}f$, is the complex-valued function defined on $\mathbb{R}^D$ given by

$$(\mathcal{F}f)(\omega) = \hat{f}(\omega) = \sum_{n \in \mathbb{Z}^D} f[n] e^{-j\omega^T n}.$$

(Note that $\hat{f}(\omega)$ is $2\pi I$ periodic.)

The inverse of the Fourier transform is computed as given by the theorem below.

Theorem 5.8 (Inverse discrete-time Fourier transform)**.** *The inverse Fourier transform of $\hat{f}$ is given by*

$$f[n] = (2\pi)^{-D} \int_R \hat{f}(\omega) e^{j\omega^T n} d\omega,$$

where R is one period of $\hat{f}$ (e.g., $R = [-\pi, \pi)^D$).

5.6.4 Continuous-Time Fourier Series

Theorem 5.9 (Continuous-time Fourier series)**.** *An N-periodic function f defined on $\mathbb{R}^D$ can be represented in the form*

$$f(t) = \sum_{k \in \mathbb{Z}^D} c[k] e^{j2\pi k^T N^{-1} t}, \tag{5.8}$$

where c is given by

$$c[k] = |\det N|^{-1} \int_R f(t) e^{-j2\pi k^T N^{-1} t} dt,$$

and R denotes a single period of f (e.g., $R = \mathrm{FPD}(N)$). Such a representation is known as a (continuous-time) ***Fourier series****.*

5.6.5 Discrete-Time Fourier Series

Theorem 5.10 (Discrete-time Fourier series)**.** *An N-periodic sequence f defined on $\mathbb{Z}^d$ can be represented in the form*

$$f[n] = |\det N|^{-1} \sum_{k \in S_1} a[k] e^{j2\pi k^T N^{-1} n},$$

where $S_1 = \mathcal{N}(N^T)$. Such a representation is known as a (discrete-time) ***Fourier series****. The sequence a is given by*

$$a[n] = \sum_{k \in S_0} f[k] e^{-j2\pi k^T N^{-T} n},$$

where $S_0 = \mathcal{N}(N)$. (Note that the sequence a is N^T periodic. (Also, note that S_0 and S_1 constitute one period of f and a, respectively).

The discrete-time Fourier series is commonly referred to as the discrete Fourier transform (DFT), although strictly speaking this is not an integral transform.

5.6.6 Miscellany

Theorem 5.11 (Relationship between $\mathcal{Z}$ and Fourier transforms)**.** *Let x be a sequence with $\mathcal{Z}$ transform X. If the region of convergence of X includes the unit hypersphere (i.e., $e_I(\omega)$ for all $\omega \in \mathbb{R}^d$), then the (discrete-time) Fourier transform $\hat{x}$ of x is given by*

$$\hat{x}(\omega) = X\left(e_I(\omega)\right).$$

Theorem 5.12 (Poisson summation formula)**.** *We have*

$$\sum_{k \in \mathbb{Z}^D} f(t-k) = \sum_{k \in \mathbb{Z}^D} \hat{f}(2\pi k) e^{j2\pi k^T t}.$$

5.7 Additional Reading

Some additional references for material related to this chapter include: multidimensional Fourier analysis [5], lattices [2], multidimensional signal processing [1, 4], two-dimensional signal processing [6], and symmetry in multidimensional signals [3].

5.8 Problems

5.1 Show that $e_M(M^{-T}\omega) = e_I(\omega)$.

5.2 Show that, for any $x, y \in \mathbb{Z}^D$, $x - y \in \operatorname{LAT}(M)$ if and only if x and y belong to the same coset of $\operatorname{LAT}(M)$.

5.3 Let h be a symmetric/antisymmetric sequence defined on $\mathbb{Z}^d$. Such a sequence is of the form $h[n] = sh[2c-n]$, where $c \in \frac{1}{2}\mathbb{Z}^d$ is the center of symmetry and $s \in \{-1, 1\}$.
(a) Show that $\hat{h}$ can be expressed as

$$\hat{h}(\omega) = \begin{cases} e^{-j\omega^T c} \displaystyle\sum_{n \in \mathbb{Z}^d} h[n] \cos\left(\omega^T(n-c)\right) & \text{if } s = 1 \\ -je^{-j\omega^T c} \displaystyle\sum_{n \in \mathbb{Z}^d} h[n] \sin\left(\omega^T(n-c)\right) & \text{if } s = -1. \end{cases}$$

(b) Let H denote the $\mathcal{Z}$ transform of h. Show that

$$H(z) = \tfrac{1}{2} z^{-c} \left[\sum_{k \in \mathbb{Z}^d} h[k] \left(z^{c-k} + sz^{-(c-k)} \right) \right].$$

5.4 Show that $\Delta^n \hat{a}(\omega) = (-j)^{|n|} \sum_{k \in \mathbb{Z}^d} a_k k^n e^{-jk^T\omega}$.

5.5 Show that $(\alpha k)^n = \alpha^{|n|} k^n$, where $\alpha \in \mathbb{C}$ and $k, n \in \mathbb{Z}^d$.

5.6 Show that $w^{u+v} = w^u w^v$, where $u, v, w \in \mathbb{Z}^d$.

5.7 Let $\omega, k \in \mathbb{Z}^d$. Show that
(a) $\cos \omega^T k = \frac{1}{2}\left(e_I(\omega)^k + e_I(\omega)^{-k}\right)$; and
(b) $\sin \omega^T k = \frac{1}{2}\left(e_I(\omega)^k - e_I(\omega)^{-k}\right)$.

5.8 Let A and B denote $d \times d$ matrices. Let z denote a d-dimensional column vector. Show that $(z^A)^B = z^{AB}$.

5.9 Let M be a $d \times d$ matrix and let z be a d-dimensional column vector. Show that $(z^M)^n = z^{Mn}$.

5.9 Bibliography

[1] N. K. Bose. Multidimensional systems and signal processing: Good theory for good practice. *IEEE Circuits and Systems Magazine*, pages 20–41, 2007.

[2] J. W. S. Cassels. *An Introduction to the Geometry of Numbers*. Springer-Verlag, Berlin, Germany, 1959.

[3] S. Coulombe and E. Dubois. Linear phase and symmetries for multidimensional FIR filters over lattices. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 45(4):473–481, April 1998.

[4] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1984.

[5] C. L. Epstein. *Introduction to the Mathematics of Medical Imaging*. Prentice Hall, Upper Saddle River, NJ, USA, 2003.

[6] J. S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[7] P. P. Vaidyanathan. Fundamentals of multidimensional multirate digital signal processing. *Sadhana*, 15(3):157–176, November 1990.

[8] E. Viscito and J. P. Allebach. The analysis and design of multidimensional FIR perfect reconstruction filter banks for arbitrary sampling lattices. *IEEE Trans. on Circuits and Systems*, 38(1):29–41, January 1991.

Chapter 6

Multidimensional Multirate Filter Banks

6.1 Introduction

In this chapter, we consider the generalization of multirate filter banks from the one-dimensional case to the multidimensional case. Multidimensional multirate filter banks play an important role in the context of multivariate wavelet systems. The relationship between multidimensional filter banks and multivariate wavelet systems is established later in Chapter 7.

6.2 Multirate Systems

To begin, we consider how some of the multirate systems concepts generalize from the one-dimensional case to the multidimensional case. In multidimensional multirate systems, more than one sampling lattice is employed, with each having its own associated sampling density. Often, in practice, the need arises to either increase or decrease the sampling density. This is achieved through processes known as upsampling and downsampling, which we introduce next.

6.2.1 Downsampling

One of the basic operations in multirate systems is that of decreasing the sampling density. This operation is known as downsampling and is performed by a processing element known as a downsampler.

Definition 6.1 (Downsampling)**.** Let M denote a $D \times D$ sampling matrix. The M-fold **downsampler**, shown in Figure 6.1, takes an input sequence x and produces the output sequence y, where

$$y[n] = (\downarrow M)x[n] = x[Mn].$$

In simple terms, the downsampling operation keeps samples on the lattice generated by M and discards all other samples.

Frequently, it is advantageous to work with the $\mathbb{Z}$-domain representation of sequences. For this reason, one might wonder how the downsampling operation is characterized in the $\mathbb{Z}$-domain. The answer to this question is given by the theorem below.

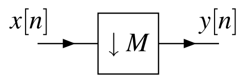


Figure 6.1: M-fold downsampler.

Theorem 6.1 (Downsampling in $\mathcal{Z}$ domain)**.** *Suppose that $y = (\downarrow M)x$. Let X and Y denote the $\mathcal{Z}$ transforms of x and y, respectively. Then, Y, which we abbreviate as $(\downarrow M)X(z)$, is given by*

$$Y(z) = \tfrac{1}{|\det M|} \sum_{c \in \mathcal{N}(M^T)} X\left(W_{M^{-1}}^{c} \circ z^{M^{-1}}\right).$$

Proof. The $\mathcal{Z}$ transform of $y[n]$ can be written as

$$\begin{aligned} Y(z) &= \sum_{n \in \mathbb{Z}^D} y[n] z^{-n} \\ &= \sum_{n \in \mathbb{Z}^D} x[Mn] z^{-n}. \end{aligned}$$

Now, we define the sequence

$$v[n] = \begin{cases} x[n] & n \in M\mathbb{Z}^D \\ 0 & \text{otherwise.} \end{cases}$$

Since $v[Mn] = x[Mn]$ for all $n \in \mathbb{Z}^D$, we have

$$Y(z) = \sum_{n \in \mathbb{Z}^D} v[Mn] z^{-n}.$$

Now, we employ a change of variable. Let $n' = Mn$. So, $n = M^{-1}n'$. Applying the change of variable and dropping the primes, we obtain

$$Y(z) = \sum_{n \in M\mathbb{Z}^D} v[n] z^{-M^{-1}n}$$

Since $v[n] = 0$ for $n \notin M\mathbb{Z}^D$

$$\begin{aligned} Y(z) &= \sum_{n \in \mathbb{Z}^D} v[n] z^{-M^{-1}n} \\ &= \sum_{n \in \mathbb{Z}^D} v[n] z^{M^{-1}(-n)} \\ &= \sum_{n \in \mathbb{Z}^D} v[n] (z^{M^{-1}})^{(-n)} \\ &= V(z^{M^{-1}}). \end{aligned}$$

To complete the proof, we express $V(z)$ in terms of $X(z)$. We observe that $v[n]$ can be expressed as

$$v[n] = s_M[n] x[n]$$

where

$$s_M[n] = \begin{cases} 1 & n \in M\mathbb{Z}^D \\ 0 & \text{otherwise.} \end{cases}$$

In what follows, let m_l' denote the lth column of M^{-1} and let n_l denote the lth element of n. Since s_M is M-periodic, it can be expressed in terms of a Fourier series as

$$\begin{aligned}
s_M[n] &= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} e^{j2\pi k^T M^{-1} n} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} e^{-j2\pi k^T M^{-1}(-n)} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} e^{-j2\pi k^T (-m'_0 n_0 - m'_1 n_1 - \cdots - m'_{D-1} n_{D-1})} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} e^{-j2\pi k^T m'_0 (-n_0)} e^{-j2\pi k^T m'_1 (-n_1)} \cdots e^{-j2\pi k^T m'_{D-1} (-n_{D-1})} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \left(\begin{bmatrix} e^{-j2\pi k^T m'_0} & e^{-j2\pi k^T m'_1} & \cdots & e^{-j2\pi k^T m'_{D-1}} \end{bmatrix}^T \right)^{-n} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} (W_{M^{-1}}^k)^{-n}.
\end{aligned}$$

From the definition of v and using the identity (5.2), we have

$$\begin{aligned}
V(z) &= \mathcal{Z}\{s_M[n]x[n]\} \\
&= \sum_{n \in \mathbb{Z}^D} \left(\tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} (W_{M^{-1}}^k)^{-n} x[n] \right) z^{-n} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \sum_{n \in \mathbb{Z}^D} (W_{M^{-1}}^k)^{-n} x[n] z^{-n} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \sum_{n \in \mathbb{Z}^D} x[n] (W_{M^{-1}}^k \circ z)^{-n} \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X(W_{M^{-1}}^k \circ z).
\end{aligned}$$

Combining the preceding equation with the earlier one for Y in terms of V, we obtain

$$\begin{aligned}
Y(z) &= V(z^{M^{-1}}) \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X(W_{M^{-1}}^{k_l} \circ z^{M^{-1}}).
\end{aligned}$$

□

The above $\mathcal{Z}$-domain characterization of downsampling leads to the following characterization of downsampling in the Fourier domain.

Theorem 6.2 (Downsampling in Fourier domain)**.** *Let $y = (\downarrow M)x$. Then, we have*

$$\hat{y}(\omega) = \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \hat{x}(M^{-T}\omega - 2\pi M^{-T} k) = \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \hat{x}\left(M^{-T}[\omega - 2\pi k]\right).$$

Proof. Let X and Y denote the $\mathcal{Z}$ transforms of x and y, respectively. To obtain $\hat{y}(\omega)$, we evaluate Y on the unit hypersphere and use (5.6) to obtain

$$\begin{aligned}
\hat{y}(\omega) &= Y(e_I(\omega)) \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X(W_{M^{-1}}^k \circ [e_I(\omega)]^{M^{-1}}) \\
&= \tfrac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X(W_{M^{-1}}^k \circ e_{M^{-1}}(\omega)).
\end{aligned}$$

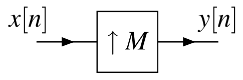


Figure 6.2: M-fold upsampler.

Letting m'_l denote the lth column of M^{-1}, we can write

$$\hat{y}(\omega) = \frac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X\left(\begin{bmatrix} e^{-j2\pi k^T m'_0} \\ e^{-j2\pi k^T m'_1} \\ \vdots \\ e^{-j2\pi k^T m'_{D-1}} \end{bmatrix} \circ \begin{bmatrix} e^{j\omega^T m'_0} \\ e^{j\omega^T m'_1} \\ \vdots \\ e^{j\omega^T m'_{D-1}} \end{bmatrix}\right) = \frac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} X\left(\begin{bmatrix} e^{j(\omega^T m'_0 - 2\pi k^T m'_0)} \\ e^{j(\omega^T m'_1 - 2\pi k^T m'_1)} \\ \vdots \\ e^{j(\omega^T m'_{D-1} - 2\pi k^T m'_{D-1})} \end{bmatrix}\right).$$

Further simplification yields

$$\hat{y}(\omega) = \frac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \hat{x}\left(\begin{bmatrix} m'^T_0(\omega - 2\pi k) \\ m'^T_1(\omega - 2\pi k) \\ \vdots \\ m'^T_{D-1}(\omega - 2\pi k) \end{bmatrix}\right) = \frac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \hat{x}\left(\begin{bmatrix} m'^T_0 \\ m'^T_1 \\ \vdots \\ m'^T_{D-1} \end{bmatrix}(\omega - 2\pi k)\right).$$

Finally, we have

$$\hat{y}(\omega) = \frac{1}{|\det M|} \sum_{k \in \mathcal{N}(M^T)} \hat{x}(M^{-T}(\omega - 2\pi k)).$$

□

6.2.2 Upsampling

Another basic operation in multirate signal processing is that of increasing the sampling density. This operation is called upsampling and is performed by a processing element known as an upsampler. In particular, upsampling is defined as follows.

Definition 6.2 (Upsampling)**.** Let M denote a $D \times D$ sampling matrix. The M-fold **upsampler**, depicted in Figure 6.2, takes an input sequence x and produces the output sequence y, where

$$y[n] = (\uparrow M)x[n] = \begin{cases} x[M^{-1}n] & n \in M\mathbb{Z}^D \\ 0 & \text{otherwise.} \end{cases}$$

In simple terms, upsampling copies the samples of the original sequence to one coset of a higher density lattice while setting the samples of the other cosets to zero.

Often, it is convenient to work in the $\mathcal{Z}$-domain. Consequently, the $\mathcal{Z}$-domain characterization of upsampling, given by the theorem below, is of great interest.

Theorem 6.3 (Upsampling in $\mathcal{Z}$ domain)**.** *Let x and y be two sequences defined on $\mathbb{Z}^D$, where $y = (\uparrow M)x$. Let X and Y denote the $\mathcal{Z}$ transforms of x and y, respectively. Then Y, which we abbreviate as $(\uparrow M)X$, is given by*

$$Y(z) = (\uparrow M)X(z) = X(z^M).$$

Proof. Using the definition of the $\mathcal{Z}$ transform and the fact that $y[n] = 0$ for $n \notin M\mathbb{Z}^D$, we can write

$$\begin{aligned} Y(z) &= \sum_{n \in \mathbb{Z}^D} y[n] z^{-n} \\ &= \sum_{n \in M\mathbb{Z}^D} y[n] z^{-n}. \end{aligned}$$

Since $y[n] = x[M^{-1}n]$ for $n \in M\mathbb{Z}^D$, we have

$$Y(z) = \sum_{n \in M\mathbb{Z}^D} x[M^{-1}n] z^{-n}.$$

Now, we employ a change of variable. Let $n' = M^{-1}n$. So, $n = Mn'$. Applying the change of variable and dropping the primes yields

$$Y(z) = \sum_{n \in \mathbb{Z}^D} x[n] z^{-Mn}.$$

Straightforward simplification using (5.1) yields

$$\begin{aligned} Y(z) &= \sum_{n \in \mathbb{Z}^D} x[n] z^{M(-n)} \\ &= \sum_{n \in \mathbb{Z}^D} x[n] (z^M)^{-n} \\ &= X(z^M). \end{aligned}$$

□

From the above result, the Fourier-domain characterization of upsampling can be deduced to be as given by the theorem below.

Theorem 6.4 (Upsampling in Fourier domain)**.** *Let M be a $D \times D$ sampling matrix, and let x and y be sequences defined on $\mathbb{Z}^D$, where $y = (\uparrow M)x$. Then, we have*

$$\hat{y}(\omega) = \hat{x}(M^T \omega).$$

Proof. Evaluating $Y(z)$ on the unit hypersphere (i.e., for $z = e_I(\omega)$), we have

$$\begin{aligned} \hat{y}(\omega) &= Y(e_I(\omega)) \\ &= X(e_I(\omega)^M) \\ &= X(e_M(\omega)) \\ &= X\left(e_I(M^T \omega)\right) \\ &= \hat{x}(M^T \omega). \end{aligned}$$

(In the above simplification, we used (5.7) and (5.6).) □

6.2.3 Cascaded Upsampling and Downsampling Identities

In multirate signal processing, we often encounter cascaded downsampling operations or cascaded upsampling operations. Thus, we would like to consider the effects of cascading such operations. In this regard, one can shown that the theorem below holds.

Theorem 6.5 (Cascaded upsampling/downsampling)**.** *Let L and M denote $D \times D$ sampling matrices. Then, we have*

$$(\downarrow M)(\downarrow L) = \downarrow (LM) \quad \text{and} \tag{6.1}$$

$$(\uparrow M)(\uparrow L) = \uparrow (ML). \tag{6.2}$$

These relationships are illustrated in Figure 6.3.

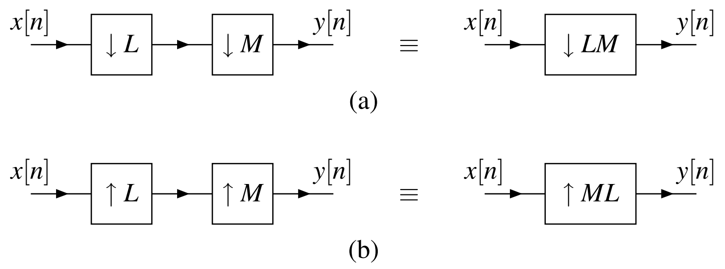


Figure 6.3: Identities for (a) cascaded downsampling and (b) cascaded upsampling operations.

Proof. First, we consider (6.1). Define $v = (\downarrow L)x$ and $y = (\downarrow M)v$ so that $y = (\downarrow M)(\downarrow L)x$. Then, we have

$$v[n] = x[Ln].$$

So, we can write

$$\begin{aligned} y[n] &= v[Mn] \\ &= x[LMn] \\ &= (\downarrow LM)x. \end{aligned}$$

Thus, (6.1) holds.

Now, we consider (6.2). Define $v = (\uparrow L)x$ and $y = (\uparrow M)v$ so that $y = (\uparrow M)(\uparrow L)x$. We have

$$\hat{v}(\omega) = \hat{x}(L^T \omega).$$

So, we have

$$\begin{aligned} \hat{y}(\omega) &= \hat{v}(M^T \omega) \\ &= \hat{x}(L^T M^T \omega) \\ &= \hat{x}((ML)^T \omega). \end{aligned}$$

Thus, $y = (\uparrow ML)x$ and (6.2) holds. □

Since matrix multiplication does not generally commute (i.e., usually $ML \neq LM$), one must be careful about the order in which cascaded downsampling or cascaded upsampling operations are applied. Also, note the subtle difference in the order of L and M in (6.1) and (6.2).

6.2.4 Commutativity of Upsampling and Downsampling

The circumstances under which downsampling and upsampling commute are significantly more complicated in the multidimensional case than the one-dimensional case. This additional complexity originates, in part, from the fact that matrix multiplication does not generally commute. For more details regarding the commutativity of upsampling and downsampling the reader is referred to [1, 4, 10, 14, 17, 18].

6.2.5 Noble Identities

Often, a downsampler or upsampler appears in cascade with a filter. Although it is not possible (except in trivial cases) to interchange the order of upsampling/downsampling and filtering without changing system behavior, it is sometimes possible to find an equivalent system with the order of these operations reversed, through the use of two very important relationships called the noble identities. These identities are presented in the theorem below.

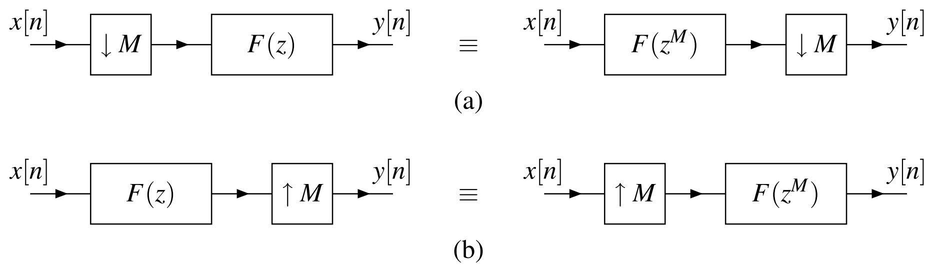


Figure 6.4: The noble identities. The (a) first and (b) second noble identities.

Theorem 6.6 (Noble identities)**.** *For any two sequences with $\mathcal{Z}$ transforms $X(z)$ and $F(z)$, the following identities hold:*

$$F(z)\left[(\downarrow M)X(z)\right] = (\downarrow M)\left[F(z^M)X(z)\right] \quad \textit{and}$$
$$(\uparrow M)\left[F(z)X(z)\right] = F(z^M)\left[(\uparrow M)X(z)\right].$$

These relationships are known as the first and second ***noble identities****, respectively. These identities are illustrated in Figure 6.4.*

6.2.6 Polyphase Representations of Signals and Filters

A fundamental concept in the study of multirate systems is that of polyphase signal representations. The notion of polyphase signal representations easily extends from the one-dimensional case to the multidimensional case. In particular, the polyphase representation of a signal is as defined below.

Definition 6.3 (Polyphase representation)**.** The **polyphase representation** of the sequence x, with respect to the sampling matrix M and its associated coset representatives $\{m_k\}_{k=0}^{\mathfrak{J}(M)-1}$, is defined as

$$x[n] = \sum_{k=0}^{\mathfrak{J}(M)-1} ((\uparrow M)x_k)[n+m_k], \tag{6.3}$$

where

$$x_k[n] = (\downarrow M)(x[n-m_k]) = x[Mn-m_k]$$

and the set $\{m_k\}_{k=0}^{\mathfrak{J}(M)-1}$ is chosen such that all members from the set are in distinct cosets of $\mathbb{Z}^D/(M\mathbb{Z}^D)$. As a matter of terminology, $x_0, x_1, \ldots, x_{\mathfrak{J}(M)-1}$ are called **polyphase components**.

Clearly, considerable freedom exists in the choice of polyphase representations. The number of possible choices for the sampling matrix M and its associated coset representatives $\{m_k\}_{k=0}^{\mathfrak{J}(M)-1}$ is infinite. Even for a fixed choice of M, the number of possible choices for $\{m_k\}_{k=0}^{\mathfrak{J}(M)-1}$ is infinite. In order to fully specify a polyphase representation, both the sampling matrix M and its associated coset vectors $\{m_k\}_{k=0}^{\mathfrak{J}(M)-1}$ must be specified.

The above polyphase representation can also be expressed in the $\mathcal{Z}$ domain. Let $X(z) = \mathcal{Z}\{x[n]\}$ and $X_k(z) = \mathcal{Z}\{x_k[n]\}$ for $k = 0, 1, \ldots, \mathfrak{J}(M)-1$. Then, by taking the $\mathcal{Z}$ transform of (6.3), we obtain

$$X(z) = \sum_{k=0}^{\mathfrak{J}(M)-1} z^{m_k} X_k(z^M) \tag{6.4a}$$

where

$$X_k(z) = (\downarrow M) z^{-m_k} X(z). \tag{6.4b}$$

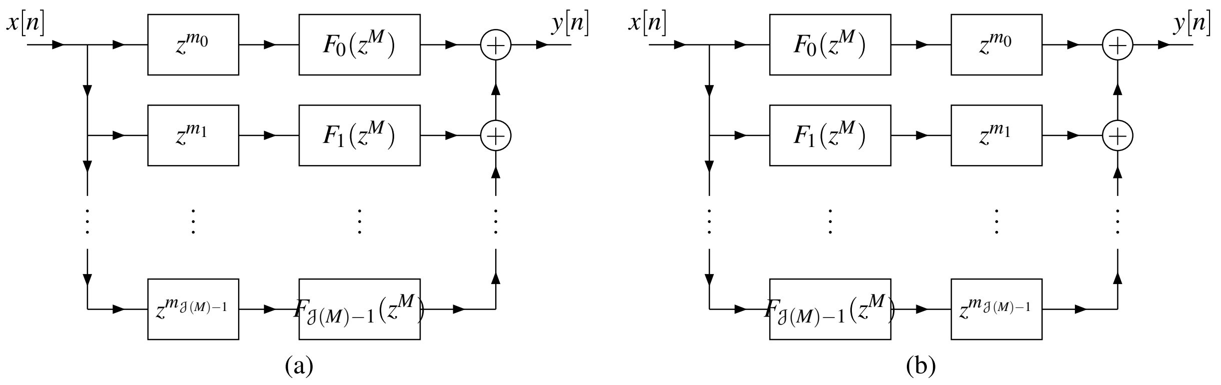


Figure 6.5: Polyphase representation of a filter.

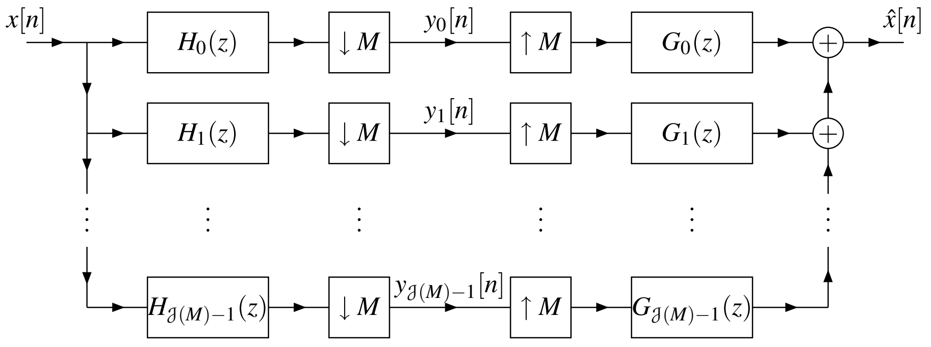


Figure 6.6: Canonical form of a UMD filter bank.

A polyphase representation can be used for the impulse response of a filter. This leads to the polyphase realization of a filter. Such a realization is associated with the structures shown in Figure 6.5.

6.3 UMD Filter Banks

The general structure of a UMD filter bank is shown in Figure 6.6. This structure is essentially the same as in the one-dimensional case, except that the filters and upsamplers/downsamplers have been replaced by their multidimensional counterparts. Since the upsampling and downsampling operations use the sampling matrix M, the (maximally-decimated) filter bank has $\mathfrak{J}(M)$ channels.

6.3.1 Polyphase Representation of UMD Filter Banks

Consider the UMD filter banks shown in Figure 6.6. As in the one-dimensional case, we can represent each of the analysis and synthesis filters in polyphase form. This leads to the structure shown in Figure 6.7. Then, we can use the noble identities to further transform this system into the one shown in Figure 6.8. This is the polyphase representation of the filter bank.

6.4 Design and Implementation of UMD Filter Banks

To date, many techniques have been proposed for the design of filter banks. Some of these techniques include: applying a transformation to the filters of a one-dimensional filter bank to produce the filters of a multidimensional

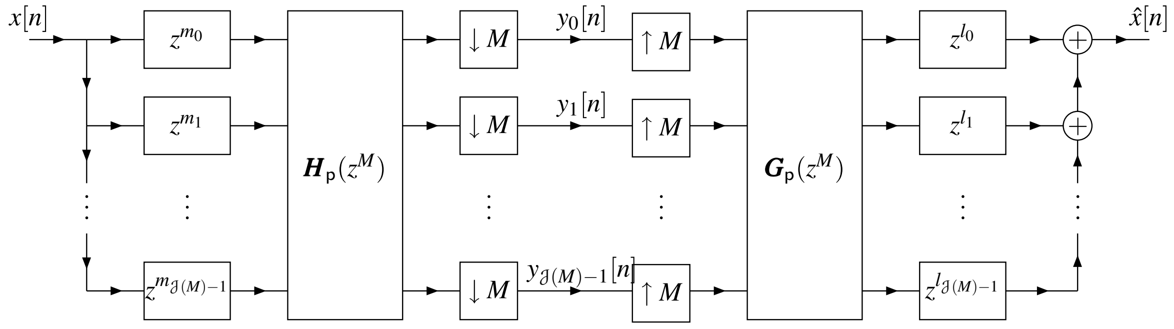

$x[n]$
z^{m_0}
z^{m_1}
$z^{m_{\mathcal{J}(M)-1}}$
$\boldsymbol{H}_\mathsf{p}(z^M)$
$\downarrow M$
$y_0[n]$
$y_1[n]$
$y_{\mathcal{J}(M)-1}[n]$
$\uparrow M$
$\boldsymbol{G}_\mathsf{p}(z^M)$
z^{l_0}
z^{l_1}
$z^{l_{\mathcal{J}(M)-1}}$
$\hat{x}[n]$

Figure 6.7: Polyphase representation of a UMD filter bank before simplification with the noble identities.

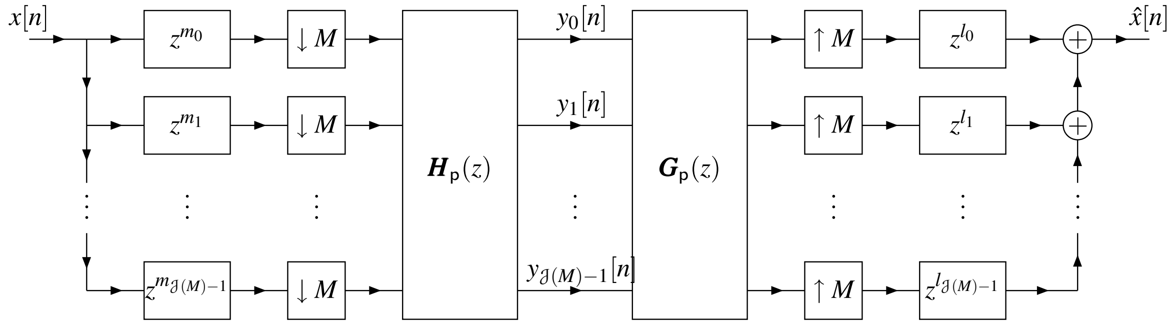

$x[n]$
z^{m_0}
z^{m_1}
$z^{m_{\mathcal{J}(M)-1}}$
$\downarrow M$
$\boldsymbol{H}_\mathsf{p}(z)$
$y_0[n]$
$y_1[n]$
$y_{\mathcal{J}(M)-1}[n]$
$\boldsymbol{G}_\mathsf{p}(z)$
$\uparrow M$
z^{l_0}
z^{l_1}
$z^{l_{\mathcal{J}(M)-1}}$
$\hat{x}[n]$

Figure 6.8: Polyphase representation of a UMD filter bank.

filter bank [2, 22, 24, 25, 26], lifting [6], the Cayley transform [34], Grobner bases [12], and others [8, 20, 23]. Some of these design methods also suggest an implementation strategy as well.

6.5 Additional Reading

Some additional references related to material in this chapter include: multidimensional multirate systems and filter banks [15, 16, 19, 29, 30], multidimensional multirate systems [3, 28, 31], and multidimensional sampling [9].

6.6 Problems

6.1 Let $x[n]$ be periodic with periodicity matrix P. Let $y[n]$ denote the output of a downsampler with sampling matrix M and input $x[n]$.
(a) Show that $y[n]$ is periodic.
(b) Derive a necessary and sufficient condition (involving M and P) for y to be $(M^{-1}P)$-periodic.

6.2 Let $x[n]$ be a symmetric/antisymmetric sequence with center of symmetry c. Let $y[n]$ denote the output of a downsampler with sampling matrix M and input $x[n]$. Determine the constraints on M and c that must be satisfied in order for y to be symmetric/antisymmetric.

6.7 Bibliography

[1] G. Cariolaro, P. Kraniauskas, and L. Vangelista. A novel general formulation of up/downsampling commutativity. *IEEE Trans. on Signal Processing*, 53(6):2124–2134, June 2005.

[2] T. Chen and P. P. Vaidyanathan. Multidimensional multirate filters and filter banks derived from one dimensional filters. *IEEE Trans. on Signal Processing*, 41(5):1749–1765, May 1993.

[3] T. Chen and P. P. Vaidyanathan. Recent developments in multidimensional multirate systems. *IEEE Trans. on Circuits and Systems for Video Technology*, 3(2):116–137, April 1993.

[4] T. Chen and P. P. Vaidyanathan. The role of integer matrices in multidimensional multirate systems. *IEEE Trans. on Signal Processing*, 41(3):1035–1047, March 1993.

[5] T. H. Chen and P. P. Vaidyanathan. Vector-space framework for unification of one-dimensional and multidimensional filter bank theory. *IEEE Trans. on Signal Processing*, 42(8):2006–2021, August 1994.

[6] Y. Chen, M. D. Adams, and W.-S. Lu. Design of optimal quincunx filter banks for image coding. *EURASIP Journal on Advances in Signal Processing*, 2007:18 pages, 2007. article ID 83858.

[7] T. Cooklev, A. Nishihara, T. Yoshida, and M. Sablatash. Regular multidimensional linear phase FIR digital filter banks and wavelet bases. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1464–1467, May 1995.

[8] T. Cooklev, A. Nishihara, T. Yoshida, and M. Sablatash. Multidimensional two-channel linear phase FIR filter banks and wavelet bases with vanishing moments. *Multidimensional Systems and Signal Processing*, 9(1):39–76, January 1998.

[9] E. Dubois. The sampling and reconstruction of time-varying imagery with application in video systems. *Proc. of IEEE*, 73(4):502–522, April 1985.

[10] B. L. Evans. Designing commutative cascades of multidimensional upsamplers and downsamplers. *IEEE Signal Processing Letters*, 4(11):313–316, November 1997.

[11] B. L. Evans, R. H. Bamberger, and J. H. McClellan. Rules for multidimensional multirate structures. *IEEE Trans. on Signal Processing*, 42(4):762–771, April 1994.

[12] J.-C. Faugere, F. Moreau de Saint-Martin, and F. Rouillier. Design of regular nonseparable bidimensional wavelets using grobner basis techniques. *IEEE Trans. on Signal Processing*, 46(4):845–856, April 1998.

[13] A. A. C. Kalker and I. A. Shah. Ladder structures for multidimensional linear phase perfect reconstruction filter banks and wavelets. In *Proc. of SPIE*, volume 1818, pages 12–20, Boston, MA, USA, November 1992.

[14] A. A. C. M. Kalker. Commutativity of up/down sampling. *Electronics Letters*, 28(6):567–569, March 1992.

[15] T. A. C. M. Kalker and I. A. Shah. A group theoretic approach to multidimensional filter banks: Theory and applications. *IEEE Trans. on Signal Processing*, 44(6):1392–1405, June 1996.

[16] G. Karlsson and M. Vetterli. Theory of two dimensional multirate filter banks. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 38:925–937, June 1990.

[17] M. R. K. Khansari and T. Chen. Finding commutative multidimensional downsamplers and upsamplers. *IEEE Trans. on Signal Processing*, 43(8):2002–2004, August 1995.

[18] J. Kovacevic and M. Vetterli. The commutativity of up/downsampling in two dimensions. *IEEE Trans. on Information Theory*, 37(3):695–698, May 1991.

[19] J. Kovacevic and M. Vetterli. Nonseparable multidimensional perfect reconstruction filter banks and wavelet bases for R^n. *IEEE Trans. on Information Theory*, 38(2):533–555, March 1992.

[20] Y.-P. Lin and P. P. Vaidyanathan. Theory and design of two-dimensional filter banks: A review. *Multidimensional Systems and Signal Processing*, 7(3-4):263–330, July 1996.

[21] H. Park, T. Kalker, and M. Vetterli. Grobner bases and multidimensional FIR multirate systems. *Multidimensional Systems and Signal Processing*, 8(1–2):11–30, 1997.

[22] I. Shah and A. Kalker. Theory and design of multidimensional QMF sub-band filters from 1-D filters and polynomials using transforms. *IEE Proceedings I: Communications, Speech and Vision*, 140(1):67–71, February 1993.

[23] J. Shyu and T. Ho. Lagrange multiplier approach to the design of multiplierless two-channel two-dimensional linear-phase FIR filter banks. *Signal Processing*, 45(3):389–395, September 1995.

[24] D. B. H. Tay. Design of filter banks/ wavelets using TROV: A survey. *Digital Signal Processing : A Review Journal*, 7(4):229–238, October 1997.

[25] D. B. H. Tay and N. G. Kingbury. Flexible design of multidimensional perfect reconstruction FIR two-band filters using transformation of variables. *IEEE Trans. on Image Processing*, 2(4):466–480, October 1993.

[26] D. B. H. Tay and N. G. Kingsbury. Design of 2-D perfect reconstruction filter banks using transformation of variables : Iir case. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 43(3):274–279, March 1996.

[27] L. Tolhuizen, H. Hollmann, and T. A. C. M. Kalker. On the realizability of biorthogonal *m*-dimensional two-band filter banks. *IEEE Trans. on Signal Processing*, 43(3):640–648, March 1995.

[28] P. P. Vaidyanathan. Fundamentals of multidimensional multirate digital signal processing. *Sadhana*, 15(3):157–176, November 1990.

[29] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[30] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[31] E. Viscito and J. P. Allebach. The analysis and design of multidimensional FIR perfect reconstruction filter banks for arbitrary sampling lattices. *IEEE Trans. on Circuits and Systems*, 38(1):29–41, January 1991.

[32] X. G. Xia, B. W. Suter, and M. E. Oxley. On necessary and sufficient conditions for perfect reconstruction multidimensional delay chain systems. *IEEE Trans. on Signal Processing*, 43(6):1515–1519, June 1995.

[33] H. Yan and M. Ikehara. Two-dimensional perfect reconstruction FIR filter banks with triangular supports. *IEEE Trans. on Signal Processing*, 47(4):1003–1009, April 1999.

[34] J. Zhou, M. N. Do, and J. Kovacevic. Multidimensional orthogonal filter bank characterization and design using the Cayley transform. *IEEE Trans. on Image Processing*, 14(6):760–769, June 2005.

Chapter 7

Multivariate Wavelet Systems

7.1 Introduction

In this chapter, we consider how wavelet systems generalize to the multivariate case.

7.2 Multiresolution Approximation

As in the univariate case, in the multivariate case, the multiresolution approximation (MRA) is a fundamental component of wavelet systems. Before we can state the definition of a MRA in the multivariate context, we first need to introduce the notion of a dilation matrix.

Definition 7.1 (Dilation matrix)**.** A **dilation matrix** is a nonsingular integer matrix such that each of its eigenvalues has a magnitude strictly greater than one.

The constraint on the magnitude of eigenvalues in the above definition ensures a dilation in each dimension. For example, $\left[\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ is not a valid dilation matrix, as it only dilates in one dimension, while $\left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ is a valid dilation matrix, as it dilates in all (i.e., two) dimensions. Since the linear transformation associated with a dilation matrix M dilates in every dimension, we have

$$\text{for all } k \in \mathbb{R}^d \setminus \{0\}, \ \lim_{q \to \infty} \left\| (M^T)^q k \right\| = \infty, \quad \text{and}$$
$$\text{for all } k \in \mathbb{R}^d, \ \lim_{q \to \infty} \left\| (M^{-T})^q k \right\| = 0.$$

These relationships are often helpful in subsequent derivations. With the above said, we can now introduce the definition of a MRA.

Definition 7.2 (Multiresolution approximation)**.** Let M denote a dilation matrix. A sequence $\{V_p\}_{p \in \mathbb{Z}}$ of closed subspaces of $L^2(\mathbb{R}^D)$ is said to be an M-dilation **multiresolution approximation** (MRA) if the following properties hold:

1. for all $p \in \mathbb{Z}$, $V_p \subset V_{p-1}$ (nesting);
2. $\lim_{p \to \infty} V_p = \bigcap_{p \in \mathbb{Z}} V_p = \{0\}$ (separation);
3. $\lim_{p \to -\infty} V_p = \operatorname{clos}\left(\bigcup_{p \in \mathbb{Z}} V_p \right) = L^2(\mathbb{R}^D)$ (density);
4. for all $p \in \mathbb{Z}$, $f(t) \in V_p \Leftrightarrow f(Mt) \in V_{p-1}$ (scale invariance);

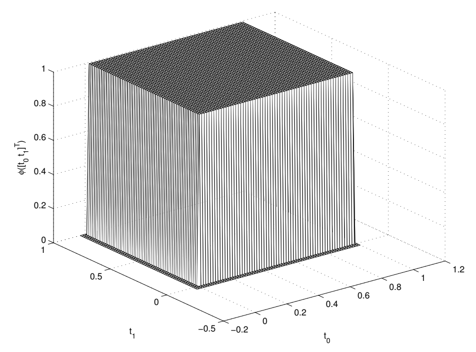


Figure 7.1: Two-dimensional Haar scaling function.

5. for all $k \in \mathbb{Z}^D$, $f(t) \in V_0 \Leftrightarrow f(t-k) \in V_0$ (shift invariance); and
6. there exists ϕ such that $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ is a Riesz basis of V_0 (shift-invariant Riesz basis).

In what follows, we consider an example of a MRA.

Example 7.1 (Piecewise constant approximations)**.** In the multivariate case, the simplest MRA is associated with piecewise constant approximations. In this case, the space V_0 is comprised of all functions $f \in L^2(\mathbb{R}^D)$ such that f is constant on regions of the form $n + [0,1)^D$, where $n \in \mathbb{Z}^D$ (i.e., D-dimensional unit hypercubes with vertices on the D-dimensional integer lattice). More generally, V_p is comprised of all functions $f \in L^2(\mathbb{R}^D)$ such that f is constant on regions of the form $2^{-p}(n + [0,1)^D)$, where $n \in \mathbb{Z}^D$.

One can show that an orthonormal basis of V_0 is given by $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}^D}$, where

$$\phi(t) = \chi_{[0,1)^D}(t) = \begin{cases} 1 & \text{for } t \in [0,1)^D \\ 0 & \text{otherwise.} \end{cases}$$

For the 2-dimensional case (i.e., $D = 2$), a plot of ϕ is given in Figure 7.1.

7.3 Existence of Riesz Basis

Recall the Riesz basis property of a MRA. Due to this property, one might wonder under what conditions the integer-vector translates of a function constitute a Riesz basis for their closed linear span. The answer to this question is given by the theorem below.

Theorem 7.1 (Condition for Riesz basis)**.** *A family $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}^D}$ is a Riesz basis of the space V_0 it generates (i.e., its closed linear span) if and only if there exist $A > 0$ and $B > 0$ such that*

$$\text{for } \omega \in [-\pi,\pi]^D,\ A \le \sum_{k\in\mathbb{Z}^D} \left|\hat{\theta}(\omega - 2\pi k)\right|^2 \le B \quad \text{a.e.}. \tag{7.1}$$

If such A and B do exist, they are the lower and upper Riesz bounds of $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}^D}$, respectively.

7.4 Wavelet Spaces

Consider an M-dilation MRA with the approximation space sequence $\{V_p\}_{p\in\mathbb{Z}}$. Let $m = |\det M|$. Since V_p is a proper subspace of V_{p-1}, there must be some space U_p which is the algebraic complement of V_p in V_{p-1}. In other words, we have

$$V_{p-1} = V_p \oplus U_p.$$

$$\begin{tikzcd}
L^2(\mathbb{R}^D) & \cdots & V_{-2} \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_{-1} \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_0 \arrow[r] \arrow[dr] \arrow[ddr] \arrow[ddddr] & V_1 & \cdots & \{0\} \\
& & & W_{1,-1} & W_{1,0} & W_{1,1} & & \\
& & & W_{2,-1} & W_{2,0} & W_{2,1} & & \\
& & & \vdots & \vdots & \vdots & & \\
& & & W_{m-1,-1} & W_{m-1,0} & W_{m-1,1} & &
\end{tikzcd}$$

Figure 7.2: MRA decomposition of $L^2(\mathbb{R}^D)$.

Unfortunately, if $m > 2$, it is not possible for the integer-vector shifts of a single function to form a basis for U_0. In fact, one can show that we need the integer-vector shifts of $m-1$ functions. Thus, we need to further decompose the space U_p into the $m-1$ subspaces $\{W_{q,p}\}_{q\in\{1,2,\ldots,m-1\}}$. That is, we have

$$U_p = \bigoplus_{q=1}^{m-1} W_{q,p}.$$

This leads to the relationship

$$V_{p-1} = V_p \oplus \left(\bigoplus_{q=1}^{m-1} W_{q,p} \right).$$

We refer to the spaces $\{W_{q,p}\}$ as **wavelet spaces**. Notice, however, that we now have $m-1$ wavelet spaces at each level in the MRA. In other words, we obtain the decomposition of $L^2(\mathbb{R}^D)$ illustrated in Figure 7.2.

One can show that the various spaces have the following properties:

$$\begin{gathered}
V_k \cap V_l = V_l \quad \text{for } k < l \\
W_{q,p} \cap W_{l,k} = \{0\} \quad \text{for } (q,p) \neq (l,k)\text{; and} \\
V_k \cap W_{q,p} = \{0\} \quad \text{for } k \geq p.
\end{gathered}$$

In addition, the wavelet spaces have a number of other important properties given by the theorem below.

Theorem 7.2 (Properties of wavelet spaces)**.** *Let $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$ denote the wavelet spaces associated with an M-dilation MRA. Then, these spaces are such that:*

1. $\operatorname{clos}\left(\bigoplus_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}} W_{q,p}\right) = L^2(\mathbb{R}^D)$ *(density);*
2. *for each $q \in \{1,2,\ldots,m-1\}$, $f \in W_{q,p} \Leftrightarrow f(M\cdot) \in W_{q,p-1}$ for all $p \in \mathbb{Z}$ (scale invariance);*
3. *for each $q \in \{1,2,\ldots,m-1\}$, $f \in W_{q,0} \Leftrightarrow f(\cdot - n) \in W_{q,0}$ for all $n \in \mathbb{Z}^D$ (shift invariance); and*
4. *for each $q \in \{1,2,\ldots,m-1\}$, there exists ψ_q such that $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}^D}$ is a Riesz basis of $W_{q,0}$ (shift-invariant Riesz basis).*

The basis of each wavelet space $W_{q,0}$ is generated by the integer-vector shifts of a single function ψ_q. We refer to ψ_q as a **wavelet function**. Notice that we have $m-1$ wavelet functions (since there are $m-1$ wavelet spaces at each level in the MRA).

7.5 Bases of Scaling and Wavelet Spaces

Consider an M-dilation MRA $\{V_p\}_{p\in\mathbb{Z}}$. Let $m = |\det M|$. Let $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ denote the Riesz basis of V_0. For each $q \in \{1,2,\ldots,m-1\}$, let $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}^D}$ denote the Riesz basis of $W_{q,0}$. Suppose that ϕ and $\{\psi_q\}_{q\in\{1,2,\ldots,m-1\}}$ are known. Then, just like in the univariate case, we can determine a basis for each of the other approximation and wavelet spaces. In multivariate context, we have the result below.

Theorem 7.3 (Bases of approximation spaces)**.** *Suppose that we have an M-dilation MRA $\{V_p\}_{p\in\mathbb{Z}}$ and $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ is a Riesz basis of V_0 with the dual basis $\{\tilde{\phi}(\cdot - n)\}_{n\in\mathbb{Z}^D}$. Let $m = |\det M|$. Then, for each $p \in \mathbb{Z}$, the set $\{\phi_{p,n}\}_{n\in\mathbb{Z}^D}$ given by*

$$\phi_{p,n}(t) = m^{-p/2}\phi(M^{-p}t - n)$$

is a Riesz basis of V_p with the same Riesz bounds as $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ and the dual basis $\{\tilde{\phi}_{p,n}\}_{n\in\mathbb{Z}^D}$ given by

$$\tilde{\phi}_{p,n}(t) = m^{-p/2}\tilde{\phi}(M^{-p}t - n).$$

Now, we consider the bases the wavelet spaces. The basis of each of these spaces can be determined using the theorem below.

Theorem 7.4 (Bases of wavelet spaces)**.** *Suppose that we have an M-dilation MRA $\{V_p\}_{p\in\mathbb{Z}}$ with the corresponding wavelet spaces $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$, where $m = |\det M|$, and $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of $W_{q,0}$ with the dual basis $\{\tilde{\psi}_q(\cdot - n)\}_{n\in\mathbb{Z}^D}$. Then, for each $p \in \mathbb{Z}$, $\{\psi_{q,p,n}\}_{n\in\mathbb{Z}^D}$ given by*

$$\psi_{q,p,n}(t) = m^{-p/2}\psi_q(M^{-p}t - n)$$

is a Riesz basis of $W_{q,p}$ with the same Riesz bounds as $\{\psi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ and dual basis $\{\tilde{\psi}_{q,p,n}\}_{n\in\mathbb{Z}^D}$ given by

$$\tilde{\psi}_{q,p,n}(t) = m^{-p/2}\tilde{\psi}_q(M^{-p}t - n).$$

7.6 Scaling and Wavelet Equations

In the multivariate context, a refinement equation is defined as specified below.

Definition 7.3 (Refinement equation)**.** Let M denote a $D \times D$ dilation matrix. An equation of the form

$$\phi(t) = \sum_{k\in\mathbb{Z}^D} c[k]\phi(Mt - k) \tag{7.2}$$

is called a **refinement equation**. The sequence c is referred to as a **refinement mask**. The solution of (7.2) is called a (refinable or) M**-refinable function**.

In what follows, we provide some examples of refinable functions.

Example 7.2 (Two-dimensional Haar scaling function)**.** Consider the function

$$\phi(t) = \chi_{[0,1)^2}(t) = \begin{cases} 1 & \text{for } t \in [0,1)^2 \\ 0 & \text{otherwise.} \end{cases}$$

One can confirm that ϕ satisfies the refinement equation

$$\phi(t) = \sum_{k\in\mathcal{N}(M)} \phi(Mt - k)$$

where $M = 2I$. A plot of ϕ is given in Figure 7.3.

Example 7.3. Consider the refinement equation

$$\phi(t) = \sum_{k\in\mathbb{Z}^2} c[k]\phi(Mt - k),$$

where $M = 2I$ and the nonzero elements of c are given by

$$\begin{bmatrix} c_{-1,1} & \cdots & c_{1,1} \\ \vdots & \iddots & \vdots \\ c_{-1,-1} & \cdots & c_{1,-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

One can show that the above refinement equation has the solution ϕ plotted in Figure 7.4.

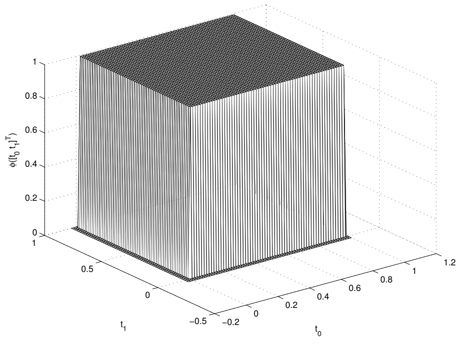


Figure 7.3: Two-dimensional Haar scaling function.

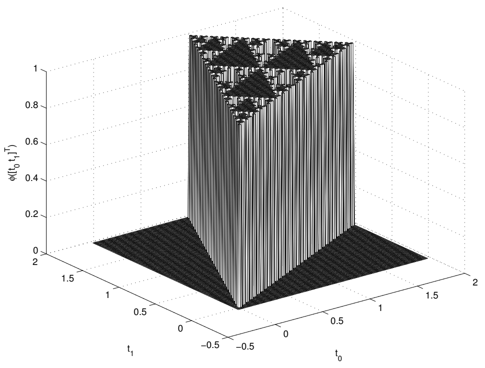


Figure 7.4: Refinable function.

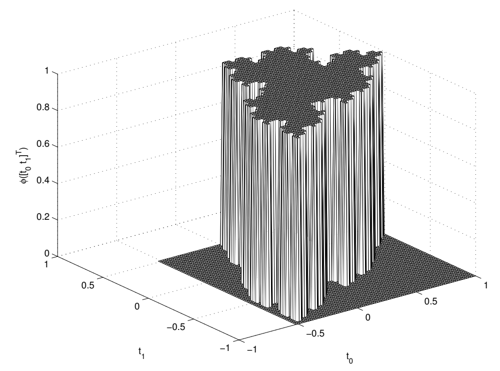


Figure 7.5: Refinable function.

Example 7.4. Consider the refinement equation

$$\phi(t) = \sum_{k\in\mathbb{Z}^2} c[k]\phi(Mt-k),$$

where $M = \left[\begin{smallmatrix} 2 & -1 \\ 1 & -2 \end{smallmatrix}\right]$ and c is given by

$$\begin{bmatrix} c_{0,0} & c_{1,0} \\ c_{0,-1} & c_{1,-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

(with the other remaining elements of c being zero). One can show that this refinement equation has the solution ϕ illustrated in Figure 7.5.

Just like in the univariate case, scaling functions are refinable and wavelet functions can be expressed in terms of scaling functions. In other words, we have the scaling and wavelet equations and their Fourier transforms as introduced below.

Theorem 7.5 (Scaling equation)**.** *Suppose that we have an M-dilation MRA $\{V_p\}_{p\in\mathbb{Z}}$ and V_0 has the Riesz basis $\{\phi(\cdot-n)\}_{n\in\mathbb{Z}^d}$. Then, ϕ satisfies a refinement equation of the form*

$$\phi(t) = |\det M|^{1/2} \sum_{k\in\mathbb{Z}^D} c[k]\phi(Mt-k)$$

where

$$c[n] = \left\langle \phi(\cdot), |\det M|^{1/2}\,\tilde{\phi}(M\cdot -n)\right\rangle.$$

Theorem 7.6 (Fourier transform of scaling equation)**.** *Let ϕ be an M-refinable scaling function with coefficient sequence c. Then, $\hat{\phi}$ is given by*

$$\hat{\phi}(\omega) = |\det M|^{-1/2}\,\hat{c}(M^{-T}\omega)\hat{\phi}(M^{-T}\omega)$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\phi}(\omega) = \hat{\phi}(0)\prod_{p=1}^{\infty} |\det M|^{-1/2}\,\hat{c}((M^{-T})^p\omega).$$

Proof. Taking the Fourier transform of the scaling equation, we have

$$\begin{aligned}\hat{\phi}(\omega) &= |\det M|^{-1/2}\sum_{k\in\mathbb{Z}^D} c[k]e^{-j\omega^T M^{-1}k}\hat{\phi}(M^{-T}\omega) \\ &= |\det M|^{-1/2}\left(\sum_{k\in\mathbb{Z}^D} c[k]e^{-j(M^{-T}\omega)^T k}\right)\hat{\phi}(M^{-T}\omega) \\ &= |\det M|^{-1/2}\,\hat{c}(M^{-T}\omega)\hat{\phi}(M^{-T}\omega).\end{aligned}$$

Applying the above relationship recursively, we obtain

$$\begin{aligned}\hat{\phi}(\omega) &= |\det M|^{-1/2}\hat{c}(M^{-T}\omega)\hat{\phi}(M^{-T}\omega)\\ &= |\det M|^{-1/2}\hat{c}(M^{-T}\omega)\left[|\det M|^{-1/2}\hat{c}((M^{-T})^2\omega)\hat{\phi}((M^{-T})^2\omega)\right]\\ &= \hat{\phi}((M^{-T})^N\omega)\prod_{p=1}^{N}|\det M|^{-1/2}\hat{c}((M^{-T})^p\omega)\\ &= \hat{\phi}(0)\prod_{p=1}^{\infty}|\det M|^{-1/2}\hat{c}((M^{-T})^p\omega).\end{aligned}$$

(Since $\left|\det M^{-T}\right| < 1$ and M dilates in all dimensions, we have that, for all $\omega \in \mathbb{R}^D$, $\lim_{q\to\infty}\left\|(M^{-T})^q\omega\right\| = 0$.) □

Theorem 7.7 (Wavelet equation)**.** *Suppose that we have an M-dilation MRA* $\{V_p\}_{p\in\mathbb{Z}}$ *with corresponding wavelet subspaces* $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$, *where* V_0 *has the Riesz basis* $\{\phi(\cdot - n)\}_{n\in\mathbb{Z}^D}$ *and* $W_{q,0}$ *has the Riesz basis* $\{\psi_q(\cdot - n)\}_{n\in\mathbb{Z}^D}$. *Then,* ψ_q *can be expressed in terms of an equation of the form*

$$\psi_q(t) = |\det M|^{1/2}\sum_{k\in\mathbb{Z}^D} d_q[k]\phi(Mt-k)$$

where

$$d_q[n] = \left\langle \psi_q, |\det M|^{1/2}\tilde{\phi}(M\cdot -n)\right\rangle.$$

Theorem 7.8 (Fourier transform of wavelet equation)**.** *Let* ϕ *and* $\{\psi_q\}_{q\in\{1,2,\ldots,m-1\}}$ *be the scaling and wavelet functions of a MRA. Suppose that* ϕ *has the scaling equation coefficient sequence* c *and* ψ_q *has the wavelet equation coefficient sequence* d_q. *Then,* $\hat{\psi}_q$ *is given by*

$$\hat{\psi}_q(\omega) = |\det M|^{-1/2}\hat{d}_q(M^{-T}\omega)\hat{\phi}(M^{-T}\omega)$$

which can be equivalently expressed in terms of an infinite product as

$$\hat{\psi}_q(\omega) = \hat{\phi}(0)\prod_{p=1}^{\infty}|\det M|^{-1/2}\hat{d}_q((M^{-T})^p\omega).$$

7.7 Dual MRAs

Just like in the univariate case, each MRA is associated with a dual MRA. In other words, we have the results given in the theorems below.

Theorem 7.9 (Dual MRAs)**.** *Let* $\{V_p\}_{p\in\mathbb{Z}}$ *be an M-dilation MRA with* $|\det M| = m$, *scaling function* ϕ, *wavelet spaces* $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$, *and wavelet functions* $\{\psi_q\}_{q\in\{1,2,\ldots,m-1\}}$. *Suppose that the dual Riesz bases of* $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}^D}$ *and* $\{\psi_q(\cdot - k)\}_{k\in\mathbb{Z}^D}$ *are given by* $\{\tilde{\phi}(\cdot - k)\}_{k\in\mathbb{Z}^D}$ *and* $\{\tilde{\psi}_q(\cdot - k)\}_{k\in\mathbb{Z}^D}$, *respectively. Then,* $\tilde{\phi}$ *is the scaling function of an M-dilation MRA* $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ *with wavelet spaces* $\{\tilde{W}_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$ *and wavelet functions* $\{\tilde{\psi}_q\}_{q\in\{1,2,\ldots,m-1\}}$.

Theorem 7.10. *Suppose that* $\{V_p\}_{p\in\mathbb{Z}}$ *and* $\{\tilde{V}_p\}_{p\in\mathbb{Z}}$ *are dual MRAs with wavelet spaces* $\{W_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$ *and* $\{\tilde{W}_{q,p}\}_{(q,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$, *respectively. Then, we have*

$$\text{for all } (q,p)\in\{1,2,\ldots,M-1\}\times\mathbb{Z}, \quad V_p \perp \tilde{W}_{q,p} \text{ and } W_{q,p}\perp\tilde{V}_p.$$

7.8 Wavelet Systems

In what follows, we introduce several examples of wavelet systems. Some of these systems are separable (i.e., formed from tensor-product constructions), while others are not.

Example 7.5 (Separable Haar wavelet system)**.** Let $\phi^{(1D)}(t)$ and $\psi^{(1D)}(t)$ denote the scaling and wavelet functions, respectively, of the 1-dimensional Haar wavelet system. We can construct a 2-dimensional wavelet system via tensor products. This yields a 2-dimensional wavelet system based on the dilation matrix $M = 2I$. (Note that $|\det M| = 4$.) The scaling function and wavelet functions are given by

$$\begin{aligned}\phi([\begin{smallmatrix} t_0 & t_1 \end{smallmatrix}]^T) &= \phi^{(1D)}(t_0)\phi^{(1D)}(t_1),\\ \psi_1([\begin{smallmatrix} t_0 & t_1 \end{smallmatrix}]^T) &= \phi^{(1D)}(t_0)\psi^{(1D)}(t_1),\\ \psi_2([\begin{smallmatrix} t_0 & t_1 \end{smallmatrix}]^T) &= \psi^{(1D)}(t_0)\phi^{(1D)}(t_1), \quad \text{and}\\ \psi_3([\begin{smallmatrix} t_0 & t_1 \end{smallmatrix}]^T) &= \psi^{(1D)}(t_0)\psi^{(1D)}(t_1).\end{aligned}$$

With this construction, ϕ satisfies the refinement equation

$$\phi(t) = \sum_{k\in\mathbb{Z}^2} c[k]\phi(Mt-k),$$

where the nonzero elements of c are given by

$$\begin{bmatrix} c[0,1] & c[1,1] \\ c[0,0] & c[1,0] \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Similarly, the wavelet functions $\{\psi_q\}_{q=1}^{3}$ can be expressed in terms of ϕ as

$$\psi_q(t) = \sum_{k\in\mathbb{Z}^2} d_q[k]\phi(Mt-k),$$

where the nonzero elements of $\{d_q\}_{q=1}^{3}$ are given by

$$\begin{aligned}\begin{bmatrix} d_1[0,1] & d_1[1,1] \\ d_1[0,0] & d_1[1,0] \end{bmatrix} &= \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix},\\ \begin{bmatrix} d_2[0,1] & d_2[1,1] \\ d_2[0,0] & d_2[1,0] \end{bmatrix} &= \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \quad \text{and}\\ \begin{bmatrix} d_3[0,1] & d_3[1,1] \\ d_3[0,0] & d_3[1,0] \end{bmatrix} &= \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}.\end{aligned}$$

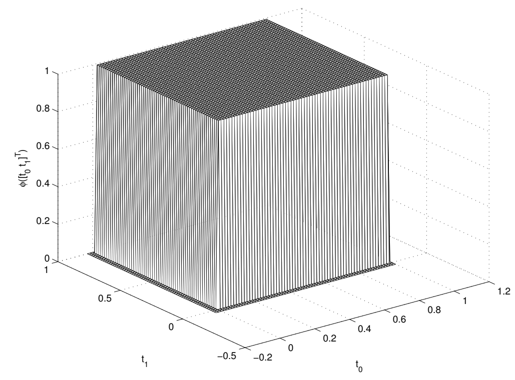
The scaling and wavelet functions are plotted in Figure 7.6.

Example 7.6 (Separable Le Gall 5/3 wavelet system)**.** In a similar fashion as in the previous example, we can construct a two-dimensional version of the Le Gall 5/3 wavelet system via tensor products. This results in a 4-adic wavelet system. The primal scaling and wavelet functions are plotted in Figures 7.7(a) to (d). The dual scaling and wavelet functions are plotted in Figures 7.7(e) to (h).

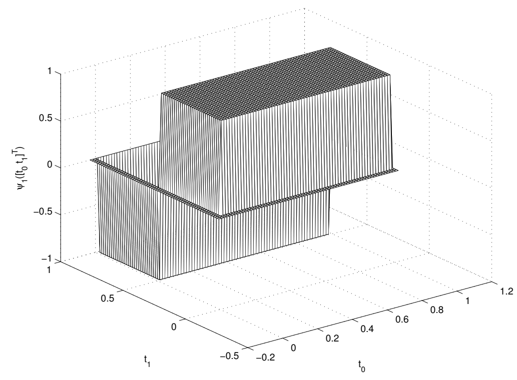
Example 7.7 (Separable CDF 9/7 wavelet system)**.** We can construct a two-dimensional version of the CDF 9/7 wavelet system via tensor products. This results in a 4-adic wavelet system. The primal scaling and wavelet functions are plotted in Figures 7.8(a) to (d). The dual scaling and wavelet functions are plotted in Figures 7.8(e) to (h).

Example 7.8 (Twin dragon wavelet system)**.** A classic example of a 2-dimensional wavelet system is the twin dragon wavelet system. This system is orthonormal. The dilation matrix M is given by

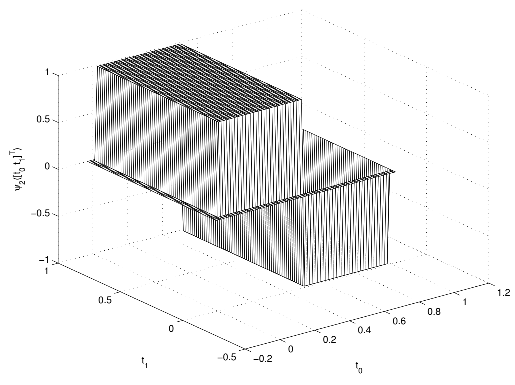
$$M = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$



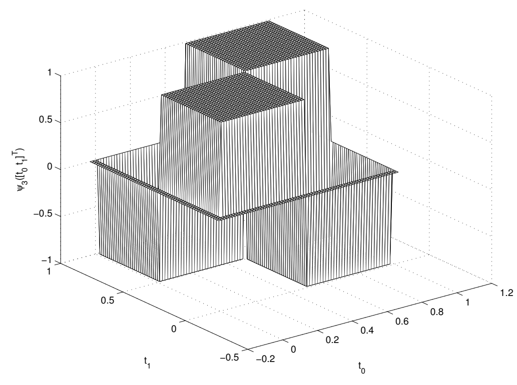
(a)



(b)

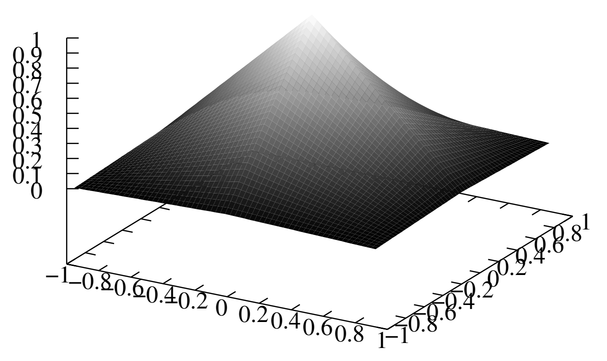


(c)

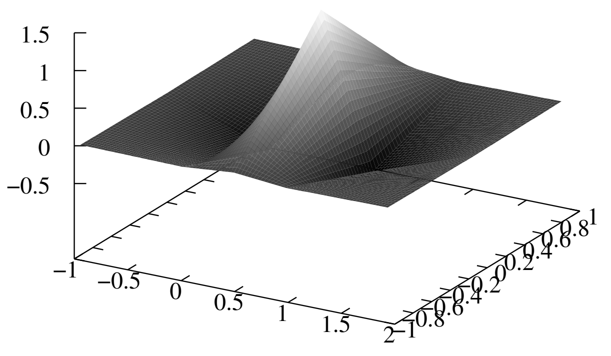


(d)

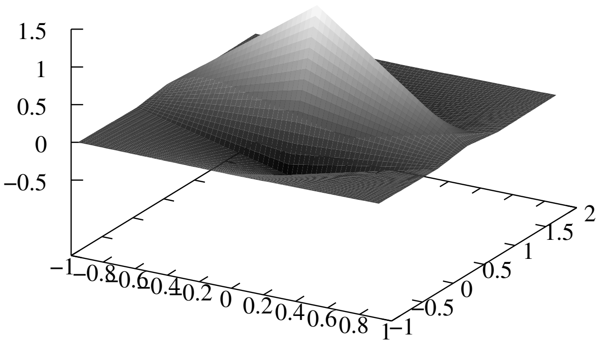
Figure 7.6: Separable Haar wavelet system. (a) Scaling and (b), (c), (d) wavelet functions.



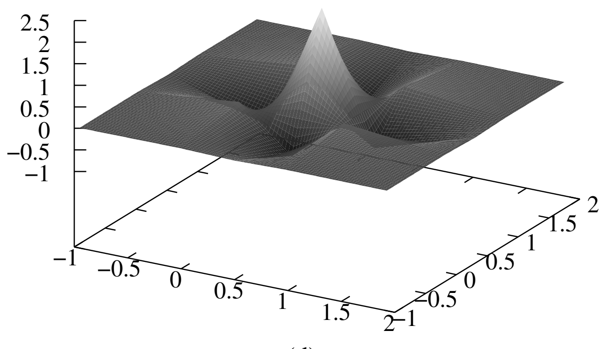
(a)



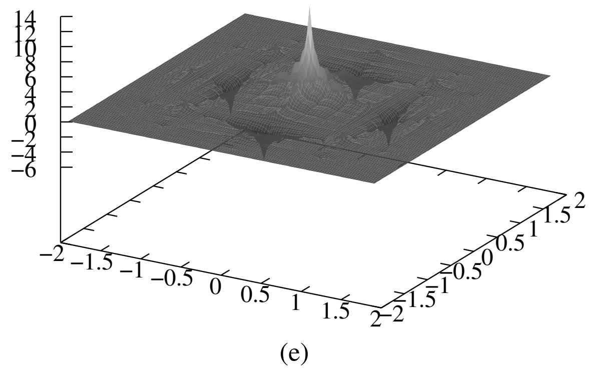
(b)



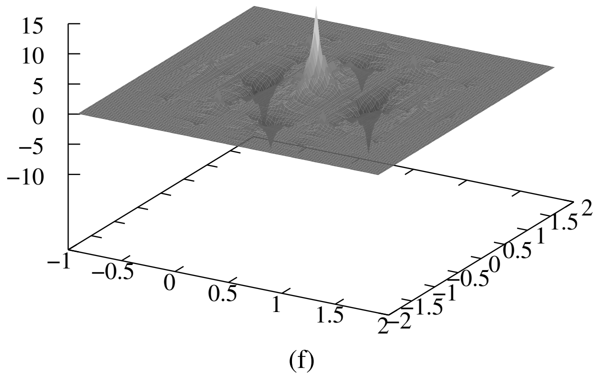
(c)



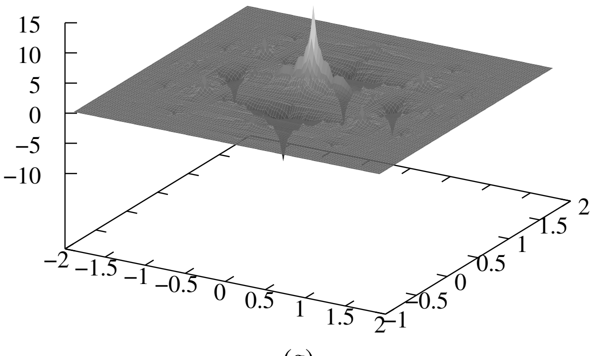
(d)



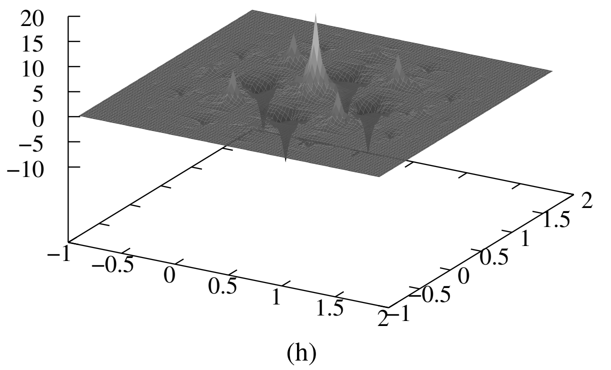
(e)



(f)

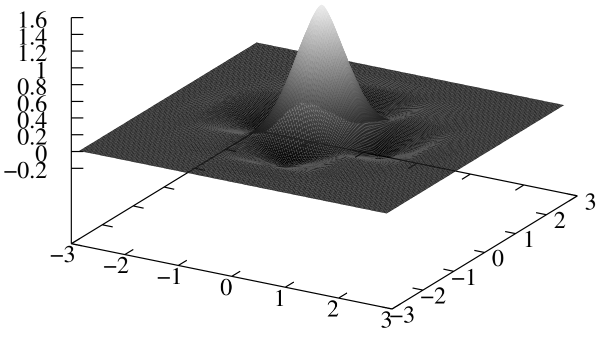


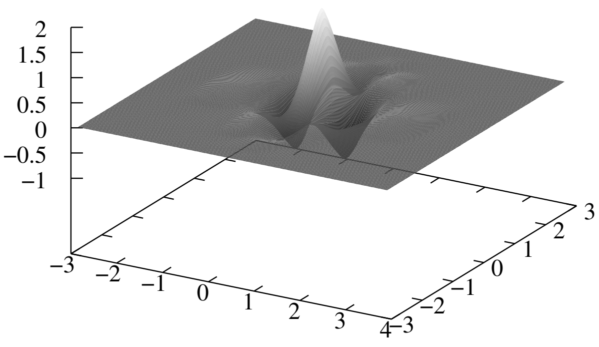
(g)

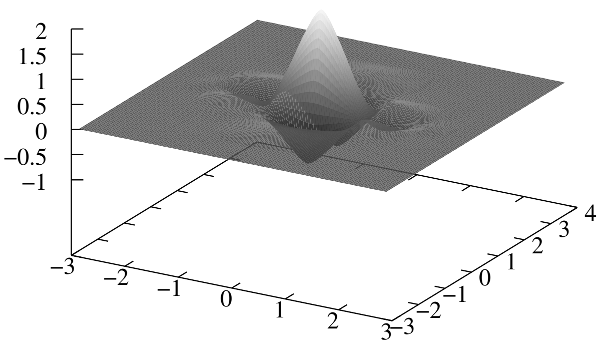


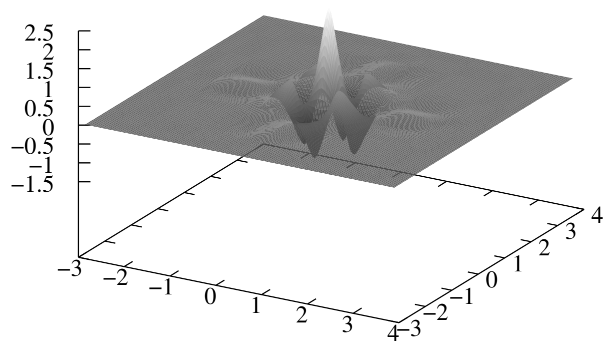
(h)

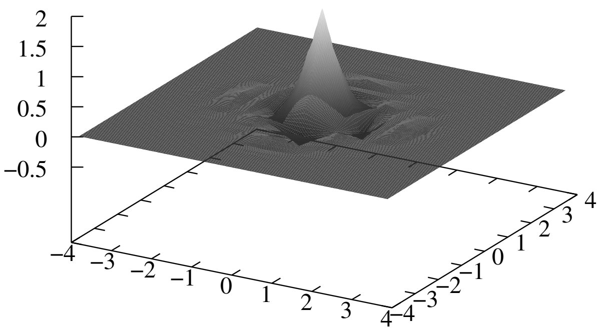
Figure 7.7: Separable Le Gall 5/3 wavelet system. Primal (a) scaling and (b), (c), (d) wavelet functions. Dual (e) scaling and (f), (g), (h) wavelet functions.

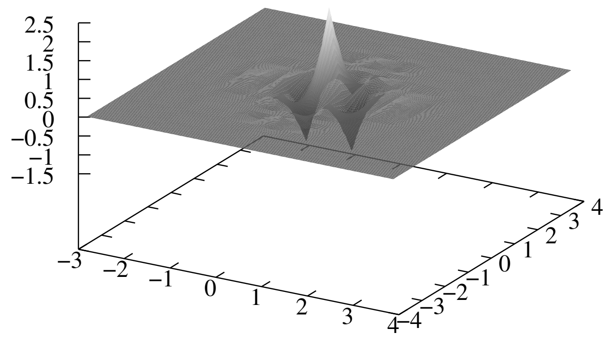

(a)

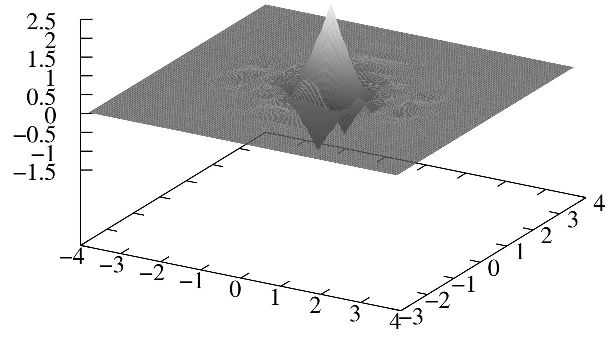

(b)


(c)


(d)


(e)


(f)


(g)

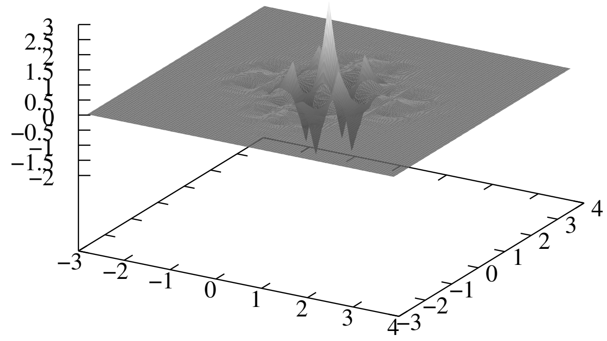

(h)

Figure 7.8: Separable CDF 9/7 wavelet system. Primal (a) scaling and (b), (c), (d) wavelet functions. Dual (e) scaling and (f), (g), (h) wavelet functions.

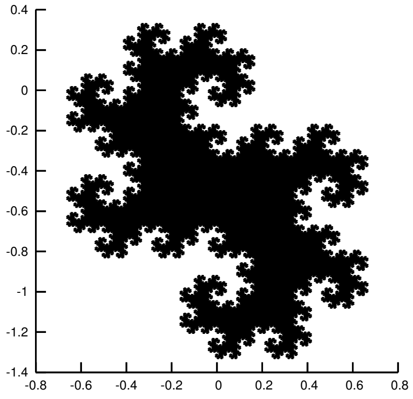
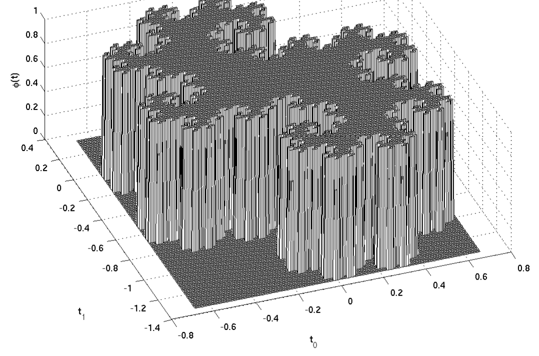
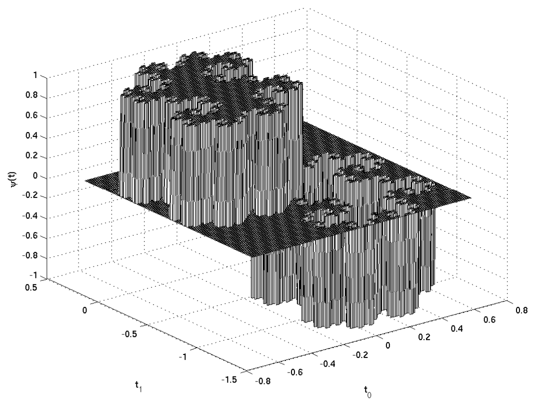


Figure 7.9: Twin dragon set.

(a) (b)

Figure 7.10: Twin dragon wavelet system. (a) Scaling and (b) wavelet functions.

(Note that $|\det M| = 2$.) The scaling function ϕ satisfies the refinement equation

$$\phi(t) = \phi(Mt) + \phi(Mt - \begin{bmatrix} 1 & 0 \end{bmatrix}^T).$$

The wavelet function ψ can be expressed in terms of the scaling function as

$$\psi(t) = \phi(Mt) - \phi(Mt - \begin{bmatrix} 1 & 0 \end{bmatrix}^T).$$

The scaling function ϕ can be shown to be

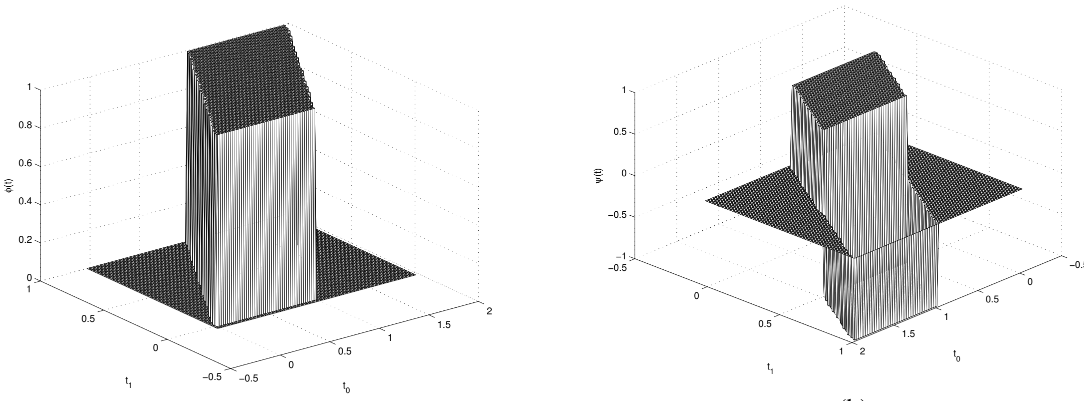
$$\phi(t) = \chi_S(t),$$

where

$$S = \left\{ \sum_{n \in \mathbb{N}} a_n \left(\tfrac{1-j}{2} \right)^n : \{a_n\}_{n \in \mathbb{N}} \in \{0,1\}^{\mathbb{N}} \right\}.$$

The set S is plotted in Figure 7.9. The boundary of the set S is a fractal curve.

The solution of a refinement equation is highly sensitive to the particular choice of dilation matrix. To illustrate this point, we consider another wavelet system with scaling and wavelet equations that are identical to those of the previous example, except for a different choice of dilation matrix.



(a) (b)

Figure 7.11: Haar-like wavelet system. (a) Scaling and (b) wavelet functions.

Example 7.9 (Haar-like wavelet system)**.** The dilation matrix M is given by

$$M = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

(Note that $|\det M| = 2$.) The scaling function ϕ satisfies the refinement equation

$$\phi(t) = \phi(Mt) + \phi(Mt - \begin{bmatrix} 1 & 0 \end{bmatrix}^T).$$

The wavelet function ψ can be expressed in terms of the scaling function as

$$\psi(t) = \phi(Mt) - \phi(Mt - \begin{bmatrix} 1 & 0 \end{bmatrix}^T).$$

The functions ϕ and ψ are plotted in Figure 7.11. Observe that the scaling and wavelet equations here are identical to those of the previous example, except for a different choice of dilation matrix. This different choice of dilation matrix, however, clearly has a very profound impact on the form of the scaling and wavelet functions.

7.9 Relationship Between Wavelets and Filter Banks

Consider the M-dilation MRA $\{V_p\}_{p\in\mathbb{Z}}$ with $|\det M| = m$ and the wavelet spaces $\{W_{k,p}\}_{(k,p)\in\{1,2,\ldots,m-1\}\times\mathbb{Z}}$. Suppose that we have a function $f \in V_p$. Since $f \in V_p$, f has an expansion in terms of the basis of V_p given by

$$f(t) = \sum_{n\in\mathbb{Z}^D} a_p[n]\phi_{p,n}(t). \tag{7.3}$$

Furthermore, as $V_p = V_{p+1} \oplus (\oplus_{k=1}^{m-1} W_{k,p+1})$, we can also expand f in terms of the bases of V_{p+1} and $\{W_{k,p+1}\}_{k=1}^{m-1}$ to obtain

$$f(t) = \sum_{n\in\mathbb{Z}^D} a_{p+1}[n]\phi_{p+1,n}(t) + \sum_{k=1}^{m-1}\sum_{n\in\mathbb{Z}^D} b_{k,p+1}[n]\psi_{k,p+1,n}(t) \tag{7.4}$$

(i.e., f is the sum of functions from V_{p+1} and each of $\{W_{k,p+1}\}_{k=1}^{m-1}$). Thus, we have two different representations of f. One might wonder if there exists a simple technique for computing (7.3) from (7.4) and vice versa. In other words, given a_p, we would like to be able to determine a_{p+1} and $\{b_{k,p+1}\}_{k=1}^{m-1}$; or given a_{p+1} and $\{b_{k,p+1}\}_{k=1}^{m-1}$, we would like to be able to determine a_p. Fortunately, there is a very elegant technique for accomplishing exactly this. Just like in the univariate case, we use a UMD filter bank, as shown in Figure 7.12. This is the so called **Mallat algorithm** in the multivariate context.

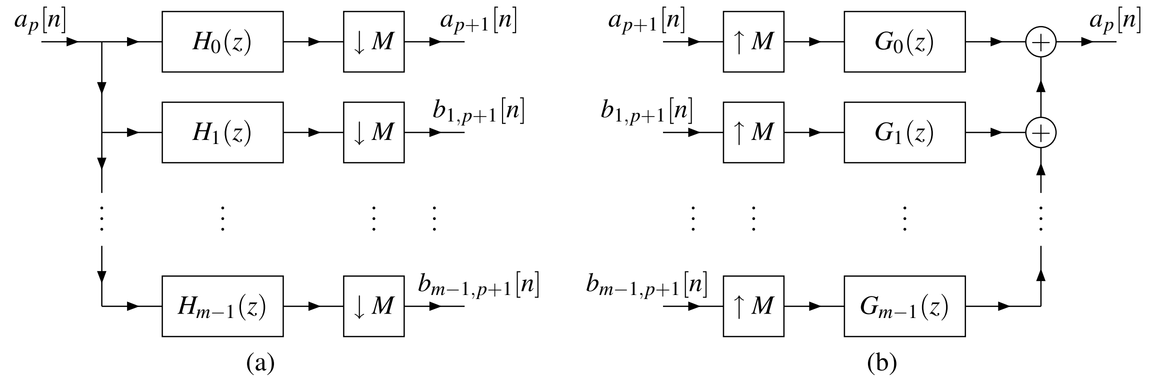


Figure 7.12: Computational structure associated with the Mallat algorithm.

7.10 Properties of Scaling Function

Due to the importance of scaling functions, we are interested in some of the properties that they possess. Some of the key properties are given by the theorem below.

Theorem 7.11 (Properties of scaling function)**.** *Let ϕ be a compactly supported M-refinable function in $L^2(\mathbb{R}^D)$ with nonvanishing zeroth-order moment μ_0. Then, we have*

$$\begin{aligned}\sum_{k\in\mathbb{Z}^D}\phi(t-k) &= \mu_0,\\ \hat{\phi}(2\pi k) &= 0 \quad \textit{for all } k\in\mathbb{Z}^D\setminus\{0\},\\ \hat{c}(0) &= |\det M|^{1/2}, \quad \textit{and}\\ \hat{c}(2\pi M^{-T}k) &= 0 \quad \textit{for all } k\in\mathcal{N}(M^T)\setminus\{0\}.\end{aligned}$$

Proof. Since $\hat{\phi}$ and $\hat{c}$ are continuous, the Fourier transform of the scaling equation must hold at every point. Thus, we can write

$$\hat{\phi}(0) = |\det M|^{-1/2}\hat{c}(0)\hat{\phi}(0).$$

Since, by assumption, $\hat{\phi}(0)\neq 0$, the preceding equation implies

$$\hat{c}(0) = |\det M|^{1/2}.$$

As $\hat{c}(0) = |\det M|^{1/2}$ and $\hat{c}$ is $2\pi I$-periodic, we have

$$\hat{c}(2\pi k) = |\det M|^{1/2} \quad \text{for all } k\in\mathbb{Z}^D.$$

Substituting $\omega = 2\pi(M^T)^N k$ into the Fourier transform of the scaling equation, we obtain

$$\begin{aligned}\phi(2\pi(M^T)^N k) &= |\det M|^{-1/2}\hat{c}(M^{-T}[2\pi(M^T)^N k])\hat{\phi}(M^{-T}[2\pi(M^T)^N k])\\ &= |\det M|^{-1/2}\hat{c}(2\pi(M^T)^{N-1}k)\hat{\phi}(2\pi(M^T)^{N-1}k)\\ &= |\det M|^{-1/2}\hat{c}(0)\hat{\phi}(2\pi(M^T)^{N-1}k)\\ &= \hat{\phi}(2\pi(M^T)^{N-1}k).\end{aligned}$$

(In the above simplification, we used the fact that $\hat{c}$ is $2\pi I$-periodic and $(M^T)^{N-1}k\in\mathbb{Z}^D$.) Repeating the above argument $N-1$ times, we obtain

$$\phi(2\pi(M^T)^N k) = \hat{\phi}(2\pi k).$$

Now, we take the limit of both sides of the preceding equation as $N \to \infty$, yielding

$$\lim_{N\to\infty} \hat{\phi}(2\pi(M^T)^N k) = \hat{\phi}(2\pi k).$$

Since $\left|\det M^T\right| > 1$ and M dilates in every dimension, as $N \to \infty$, $\left\|2\pi(M^T)^N k\right\| \to \infty$ for $k \neq 0$. Moreover, from the Riemann-Lebesgue lemma, $\lim_{\|\omega\|\to\infty} \hat{\phi}(\omega) = 0$. Thus, we have

$$\hat{\phi}(2\pi k) = 0 \quad \text{for all } k \in \mathbb{Z}^D \setminus \{0\}.$$

By definition, $\hat{\phi}(0) = \mu_0$. So, we have

$$\hat{\phi}(2\pi k) = \mu_0 \delta[k].$$

Using the Poisson summation formula, we have

$$\begin{aligned}
\sum_{k\in\mathbb{Z}^D} \phi(t-k) &= \sum_{k\in\mathbb{Z}^D} \hat{\phi}(2\pi k) e^{j2\pi k^T t} \\
&= \sum_{k\in\mathbb{Z}^D} \hat{\phi}(0)\delta[k] e^{j2\pi k^T t} \\
&= \hat{\phi}(0) \\
&= \mu_0.
\end{aligned}$$

Let $N(\phi) \triangleq \{\omega \in \mathbb{R}^d : \hat{\phi}(\omega + 2\pi k) = 0 \text{ for all } k \in \mathbb{Z}^D\}$. Since ϕ is compactly supported in $L^2(\mathbb{R}^D)$, $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}^D}$ is a Riesz basis if and only if $N(\phi)$ is the empty set. Let $\omega \in \mathcal{N}(M^T) \setminus \{0\}$. There must be some $\beta \in \mathbb{Z}^D$ such that $\hat{\phi}(\gamma) \neq 0$ for $\gamma = 2\pi\beta + 2\pi(M^T)^{-1}\omega$. Thus, in the neighbourhood of γ, $1/\hat{\phi}(\gamma)$ is well defined, and we rearrange the Fourier transform of the scaling equation as

$$\begin{aligned}
\hat{\phi}(M^T\gamma) = \left|\det M\right|^{-1/2} \hat{c}(\gamma)\hat{\phi}(\gamma) \Rightarrow \\
\hat{c}(\gamma) = \left|\det M\right|^{1/2} \hat{\phi}(M^T\gamma)/\hat{\phi}(\gamma).
\end{aligned}$$

Now, we make a few key observations. First, we have

$$M^T\gamma = M^T(2\pi\beta + 2\pi(M^T)^{-1}\omega) = 2\pi M^T\beta + 2\pi\omega \in 2\pi\mathbb{Z}^D \setminus \{0\}.$$

Thus, $\hat{\phi}(M^T\gamma) = 0$. Furthermore, since $\hat{c}$ is $2\pi I$-periodic, $\hat{c}(\gamma) = \hat{c}(2\pi(M^T)^{-1}\omega)$. From these observations, it then follows that

$$\hat{c}(2\pi(M^T)^{-1}\omega) = 0 \quad \text{for all } \omega \in \mathcal{N}(M^T) \setminus \{0\}.$$

□

7.11 Order of Approximation

As we saw in the univariate case, the approximation properties of scaling functions are of great practical interest. In the multivariate case, we have the following result concerning the approximation accuracy of scaling functions.

Theorem 7.12 (Approximation accuracy). *Let ϕ be an M-refinable scaling function with $\left|\det M\right| = m$ and the scaling equation coefficient sequence c. Let $\{\tilde{\psi}_q\}_{q=1}^{m-1}$ denote the corresponding dual wavelet functions with the wavelet equation coefficient sequences $\{\tilde{d}_q\}_{q=1}^{m-1}$. Suppose that $\hat{\phi}$ and $\{\hat{\tilde{\psi}}_q\}_{q=1}^{m-1}$ are $\eta - 1$ times differentiable. Then, the following statements are equivalent:*

1. *ϕ has approximation order η;*
2. *linear combinations of $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}^D}$ can locally reproduce polynomials of degree less than η;*
3. *$\hat{\phi}$ has a ηth order zero at $2\pi k$ for all $k \in \mathbb{Z}^D \setminus \{0\}$;*
4. *$\hat{c}$ has a ηth order zero at $2\pi M^{-T}k$ for all $k \in \mathcal{N}(M^T) \setminus \{0\}$; and*
5. *for $q \in \{1, 2, \ldots, m-1\}$, $\tilde{\psi}_q$ has all of its moments of order less than η vanish.*

7.12 Additional References

Some additional references related to topics in this chapter include: [1, 2, 3, 4, 5, 6, 7, 8].

7.13 Problems

7.1 Let $m \in \mathbb{Z}$, $m \geq 2$. Let $\phi^{(1D)}(t)$ be an m-refinable function defined on $\mathbb{R}$ with refinement mask c. Now, define $\phi([\,t_0\ t_1\,]^T) = \phi^{(1D)}(t_0)\phi^{(1D)}(t_1)$. (That is, ϕ is a function defined on $\mathbb{R}^2$.) Show that ϕ is M-refinable with $M = mI_2$ (where I_2 denotes the 2×2 identity matrix).

7.14 Bibliography

[1] A. Cohen and I. Daubechies. Non-separable bidimensional wavelet bases. *Revista Matematica Iberoamericana*, 9(1):51–137, 1993.

[2] F. C. A. Fernandes, R. L. C. van Spaendonck, and C. S. Burrus. Multidimensional, mapping-based complex wavelet transforms. *IEEE Trans. on Image Processing*, 14(1):110–124, January 2005.

[3] B. Han. Bivariate (two-dimensional) wavelets. In R. A. Meyers, editor, *Computational Complexity: Theory, Techniques, and Applications*, pages 275–285. Springer, 2012.

[4] J. Kovacevic and W. Sweldens. Wavelet families of increasing order in arbitrary dimensions. *IEEE Trans. on Image Processing*, 9(3):480–496, March 2000.

[5] J. Kovacevic and M. Vetterli. Nonseparable multidimensional perfect reconstruction filter banks and wavelet bases for R^n. *IEEE Trans. on Information Theory*, 38(2):533–555, March 1992.

[6] J. Kovacevic and M. Vetterli. Nonseparable two- and three-dimensional wavelets. *IEEE Trans. on Signal Processing*, 43(5):1269–1273, May 1995.

[7] W. Lawton, S. L. Lee, and Z. Shen. Convergence of multidimensional cascade algorithm. *Numerische Mathematik*, 78:427–438, 1998.

[8] P. Maass. Families of orthogonal two-dimensional wavelets. *SIAM Journal of Mathematical Analysis*, 27(5):1454–1481, September 1996.

Part IV

Subdivision Surfaces and Subdivision Wavelets

Chapter 8

Geometry Processing Preliminaries

8.1 Introduction

As digital computing devices have become more powerful, the complexity of the datasets processed with these devices has also increased. First came digital audio, then digital imagery, and then digital video. More recently, we have seen the rise of digital geometry, that is, digital representations of geometric objects such as polyhedra and surfaces. Digital geometry processing deals with the representation and manipulation of geometric objects in digital form. Digital geometry processing has applications in many diverse areas, including: multimedia, animation, gaming, biomedical computing, computer-aided design and manufacturing, geometric modelling, finite element analysis, computational fluid dynamics, and scientific visualization.

In the case of traditional signal processing, signals are essentially functions defined on a Euclidean domain $\mathbb{R}^n$. For example, an audio signal is a function defined on $\mathbb{R}$, where the domain of the function corresponds to time. An image signal is a function defined on $\mathbb{R}^2$, where the domain of the function corresponds to horizontal and vertical position. A video signal can be viewed as a function defined on $\mathbb{R}^3$, where the domain of the function corresponds to horizontal position, vertical position, and time. In the case of geometry processing, the signals are not functions. Rather, the signals are typically what are known as manifolds (with or without boundaries). In the sections that follow, we present some fundamentals relevant to geometry processing and introduce the formal definition of a manifold.

8.2 Linear Algebra

To begin, we first introduce some basic concepts from linear algebra.

Definition 8.1 (Cross product)**.** The **cross product** of two vectors $v = (v_1, v_2, v_3)$ and $w = (w_1, w_2, w_3)$ in $\mathbb{R}^3$, denoted $v \times w$, is defined as

$$v \times w = (v_2w_3 - v_3w_2, v_3w_1 - v_1w_3, v_1w_2 - v_2w_1). \tag{8.1}$$

The formula (8.1) for the cross product can be written in a more easily remembered form as

$$v \times w = \det \begin{bmatrix} i & j & k \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix},$$

where i, j, and k denote unit vectors in the x, y, and z directions, respectively.

Theorem 8.1 (Properties of cross product)**.** *The cross product in $\mathbb{R}^3$ has the following properties:*

1. $v \times w = -w \times v$*; and*

2. $\|v \times w\| = \|v\| \, \|w\| \sin \theta_{v,w}$*, where $\theta_{v,w}$ is the angle between the vectors v and w.*

For v, w in $\mathbb{R}^3$, $v \times w \perp v$ and $v \times w \perp w$.

Example 8.1. Find a vector $u \in \mathbb{R}^3$ that is orthogonal to both $v = (1,2,3)$ and $w = (1,-2,1)$.

Solution. The cross product $v \times w$ is orthogonal to both v and w. We have

$$\begin{aligned} v \times w = (1,2,3) \times (1,-2,1) &= \det \begin{bmatrix} i & j & k \\ 1 & 2 & 3 \\ 1 & -2 & 1 \end{bmatrix} \\ &= i(2+6) - j(1-3) + k(-2-2) = 8i + 2j - 4k \\ &= (8,2,-4). \end{aligned}$$

□

Definition 8.2 (Affine combination)**.** An **affine combination** of vectors $v_1, v_2, \ldots, v_n$ in a vector space V over the field F is an expression of the form

$$\sum_{k=1}^{n} a_k v_k \quad \text{where} \quad \sum_{k=1}^{n} a_k = 1,$$

and $a_1, a_2, \ldots, a_n \in F$ (i.e., an affine combination is a linear combination for which the sum of the coefficients is one).

Definition 8.3 (Affine hull)**.** The **affine hull** of $X \subset \mathbb{R}^n$, denoted aff$X$, is the intersection of all hyperplanes in $\mathbb{R}^n$ that contain X.

Equivalently, the affine hull of $X \subset \mathbb{R}^n$ is the set of all affine combinations of elements in X. The affine hull of a line segment is a line. The affine hull of a planar polygon is a plane. The affine hull of four points that are not coplanar is $\mathbb{R}^3$.

Definition 8.4 (Convex combination)**.** A **convex combination** of vectors $v_1, v_2, \ldots, v_n$ in a real vector space V is an expression of the form

$$\sum_{k=1}^{n} a_k v_k \quad \text{where} \quad \sum_{k=1}^{n} a_k = 1 \quad \text{and} \quad a_k \geq 0.$$

(Note that $\sum_{k=1}^{n} a_k = 1$ and $a_k \geq 0$ together imply that $a_k \in [0,1]$.)

A convex combination is simply an affine combination with nonnegative coefficients.

Definition 8.5 (Convex set)**.** A subset X of $\mathbb{R}^n$ is said to be **convex** if, for every pair p, q of points in X, the line segment $[p,q]$ is completely contained in X.

An example of a convex set is shown in Figure 8.1, while an example of a nonconvex set is depicted in Figure 8.2.

Definition 8.6 (Convex hull)**.** The **convex hull** of $X \subset \mathbb{R}^n$, denoted conv$X$, is defined as the intersection of all convex sets containing X (i.e., the smallest convex set that contains X).

Equivalently, the convex hull of $X \subset \mathbb{R}^n$ is the set of all convex combinations of elements in X. The convex hull is a subset of the affine hull. An example of the convex hull of a set of points in shown in Figure 8.3. The boundary of conv$X$ is a convex polygon with vertices in X. The boundary of conv$X$ can be visualized in terms of an elastic band stretched to encompass all of the points in X as shown in Figure 8.4. A point $p \in$ conv$X$ that does not lie on any open line segment joining two points in X called an extreme point (i.e., "corner").

Consider a triangle T whose vertices v_1, v_2, v_3 lie in the plane P in $\mathbb{R}^3$. The convex hull of T is the interior of the triangle plus its boundary, while the affine hull of T is the entire plane P.

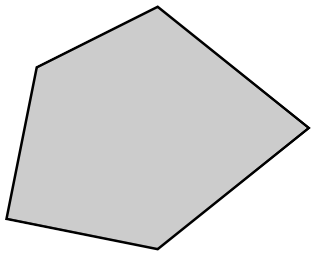


Figure 8.1: Example of a convex set in $\mathbb{R}^2$.

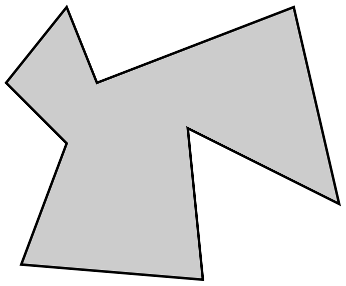


Figure 8.2: Example of a nonconvex set in $\mathbb{R}^2$.

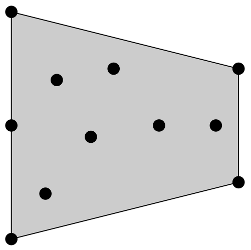


Figure 8.3: Example of the convex hull of a set of points.

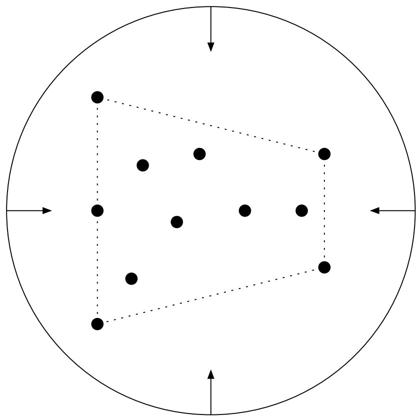


Figure 8.4: The elastic-band visualization of the convex-hull boundary.

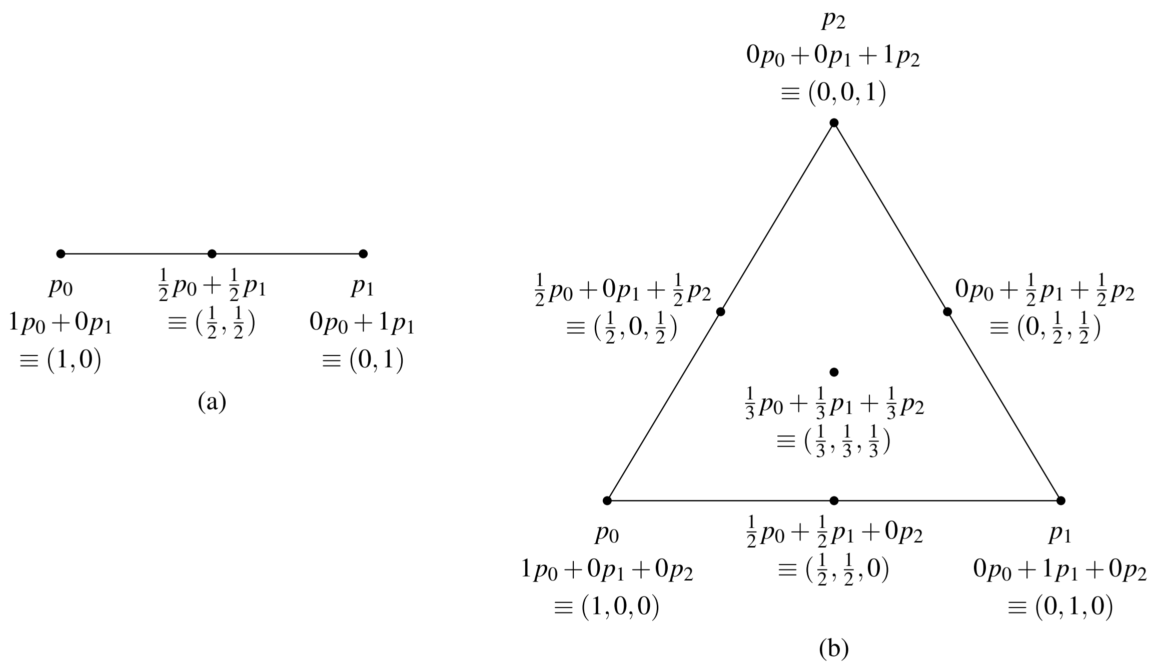

p_0
$1p_0+0p_1$
$\equiv(1,0)$
$\frac{1}{2}p_0+\frac{1}{2}p_1$
$\equiv(\frac{1}{2},\frac{1}{2})$
p_1
$0p_0+1p_1$
$\equiv(0,1)$
(a)
p_2
$0p_0+0p_1+1p_2$
$\equiv(0,0,1)$
$\frac{1}{2}p_0+0p_1+\frac{1}{2}p_2$
$\equiv(\frac{1}{2},0,\frac{1}{2})$
$0p_0+\frac{1}{2}p_1+\frac{1}{2}p_2$
$\equiv(0,\frac{1}{2},\frac{1}{2})$
$\frac{1}{3}p_0+\frac{1}{3}p_1+\frac{1}{3}p_2$
$\equiv(\frac{1}{3},\frac{1}{3},\frac{1}{3})$
p_0
$1p_0+0p_1+0p_2$
$\equiv(1,0,0)$
$\frac{1}{2}p_0+\frac{1}{2}p_1+0p_2$
$\equiv(\frac{1}{2},\frac{1}{2},0)$
p_1
$0p_0+1p_1+0p_2$
$\equiv(0,1,0)$
(b)

Figure 8.5: Barycentric coordinate examples. (a) Barycentric coordinates with respect to points p_0 and p_1. (b) Barycentric coordinates with respect to points p_0, p_1, and p_2.

Theorem 8.2. *Let $v_0, v_1, \ldots, v_m$ be $m+1$ linearly independent points in $\mathbb{R}^n$ (where, clearly, $m \leq n$). Every point in the convex hull of $v_0, v_1, \ldots, v_m$ can be expressed uniquely as a convex combination of $v_0, v_1, \ldots, v_m$. That is, every point w in the convex hull of $v_0, v_1, \ldots, v_m$ has a unique representation of the form*

$$w = \sum_{k=0}^{m} a_k v_k, \quad \text{where} \quad a_k \in [0,1] \quad \text{and} \quad \sum_{k=0}^{m} a_k = 1. \tag{8.2}$$

The $\{a_k\}_{k\in\{0,1,\ldots,m\}}$ in (8.2) are called the **barycentric coordinates** of w with respect to the points $v_0, v_1, \ldots, v_m$. Some examples of barycentric coordinates are shown in Figure 8.5.

8.3 Affine Transformations

Although linear transformations are extremely useful, a more general class of transformations, called affine transformations, is often of interest. An affine transformation is defined as follows.

Definition 8.7 (Affine transformation)**.** A one-to-one and onto mapping $T : \mathbb{R}^n \to \mathbb{R}^n$ that preserves the collinearity of points (i.e., maps lines onto lines) is called an **affine transformation**.

Affine transformations include scalings, rotations, shears, and translations. Although every linear transformation is an affine transformation, the converse is not true. For example, a translation is a transformation that is affine but not linear.

Since affine transformations are of great importance, it is helpful to know some of their properties. Some of these properties are given below.

Theorem 8.3. *Every affine transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ can be described by an equation of the form*

$$T\boldsymbol{x} = \boldsymbol{A}x + \boldsymbol{b},$$

where $\boldsymbol{A}$ is an $n \times n$ (real) matrix and $\boldsymbol{b}$ is an n-dimensional (real) column vector (i.e., an affine transformation can be expressed as the composition of a linear transformation and a translation).

Theorem 8.4. *Affine transformations preserve convex sets.*

A transformation T is said to preserve barycentric coordinates if, for all $v_k \in \mathbb{R}^n$ and real numbers a_i

$$T\left(\sum_{k=1}^{m} a_k v_k\right) = \sum_{k=1}^{m} a_k T(v_k), \quad \text{whenever} \quad \sum_{k=1}^{m} a_k = 1.$$

Theorem 8.5. *Affine maps in $\mathbb{R}^n$ preserve barycentric coordinates. Conversely, any one-to-one and onto transformation that preserves barycentric coordinates is an affine map.*

8.4 Homogeneous Coordinates

A point in $\mathbb{R}^3$ can be represented in many ways (e.g., Cartesian coordinates, spherical coordinates, cylindrical coordinates). Which representation is most convenient depends on the application at hand. Homogeneous coordinates provide yet another way to represent a point. The **homogeneous coordinates** of a point $p = (p_\mathsf{x}, p_\mathsf{y}, p_\mathsf{z})$ in $\mathbb{R}^3$ is a 4-tuple $(q_\mathsf{x}, q_\mathsf{y}, q_\mathsf{z}, q_\mathsf{w})$ satisfying $q_\mathsf{w} \neq 0$, and $p_\mathsf{x} = q_\mathsf{x}/q_\mathsf{w}$, $p_\mathsf{y} = q_\mathsf{y}/q_\mathsf{w}$, and $p_\mathsf{z} = q_\mathsf{z}/q_\mathsf{w}$. The homogeneous coordinates of a point are not unique. If $(p_\mathsf{x}, p_\mathsf{y}, p_\mathsf{z}, p_\mathsf{w})$ is the homogeneous coordinates of a point, then so too is $(kp_\mathsf{x}, kp_\mathsf{y}, kp_\mathsf{z}, kp_\mathsf{w})$ for any real $k \neq 0$. Two homogeneous coordinates represent same point if and only if one is a scalar multiple of other.

8.5 Homogeneous-Coordinate Transformations

When Cartesian coordinates are used along with 3×3 transformation matrices, translations and perspective projections are problematic as they have no corresponding matrix representation. As we shall see, the situation is quite different with homogeneous coordinates. As a matter of terminology, a transformation that operates on points expressed in homogeneous coordinates is referred to as **homogeneous-coordinate transformation**. Homogeneous-coordinate transformations are also commonly called "homogeneous transformations", but this choice of terminology is a poor one, since it conflicts with the standard mathematical definition of "homogeneous transformation". Since homogeneous coordinates are a 4-tuple, homogeneous-coordinate transformations are associated with 4×4 matrices. As it turns out, every affine transformation (including translations), perspective projection, or composition thereof, can be represented by a homogeneous-coordinate transformation matrix. The main benefit of the homogeneous representation is uniformity. All transformations of interest can be characterized by a matrix and the application/composition of transformations is achieved by matrix multiplication.

8.6 Translation, Scaling, and Rotation

The homogeneous-coordinate transformation matrix $T(d)$ that corresponds to a translation by $d = (d_\mathsf{x}, d_\mathsf{y}, d_\mathsf{z})$ is given by

$$T(d) = \begin{bmatrix} 1 & 0 & 0 & d_\mathsf{x} \\ 0 & 1 & 0 & d_\mathsf{y} \\ 0 & 0 & 1 & d_\mathsf{z} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point p, we obtain

$$T(d)\begin{bmatrix} p_\mathsf{x} \\ p_\mathsf{y} \\ p_\mathsf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_\mathsf{x} \\ 0 & 1 & 0 & d_\mathsf{y} \\ 0 & 0 & 1 & d_\mathsf{z} \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_\mathsf{x} \\ p_\mathsf{y} \\ p_\mathsf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} p_\mathsf{x} + d_\mathsf{x} \\ p_\mathsf{y} + d_\mathsf{y} \\ p_\mathsf{z} + d_\mathsf{z} \\ 1 \end{bmatrix}.$$

The homogeneous-coordinate transformation matrix $T(s)$, with $s = (s_x, s_y, s_z)$, that corresponds to a scaling in the x, y, and z directions by s_x, s_y, and s_z, respectively, is given by

$$S(s) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point $p = (p_x, p_y, p_z)$, we obtain

$$S(s)\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x p_x \\ s_y p_y \\ s_z p_z \\ 1 \end{bmatrix}.$$

The homogeneous-coordinate transformation matrix $R_z(\theta)$ that corresponds to a rotation of θ about the z axis is given by

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point (p_x, p_y, p_z), we obtain

$$R_z(\theta)\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x\cos\theta - p_y\sin\theta \\ p_x\sin\theta + p_y\cos\theta \\ p_z \\ 1 \end{bmatrix}.$$

The homogeneous-coordinate transformation matrix $R_x(\theta)$ that corresponds to a rotation of θ about the x axis is given by

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point (p_x, p_y, p_z), we obtain

$$R_x(\theta)\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y\cos\theta - p_z\sin\theta \\ p_y\sin\theta + p_z\cos\theta \\ 1 \end{bmatrix}.$$

The homogeneous-coordinate transformation matrix $R_y(\theta)$ that corresponds to a rotation of θ about the y axis is given by

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point (p_x, p_y, p_z), we obtain

$$R_y(\theta)\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x\cos\theta + p_z\sin\theta \\ p_y \\ -p_x\sin\theta + p_z\cos\theta \\ 1 \end{bmatrix}.$$

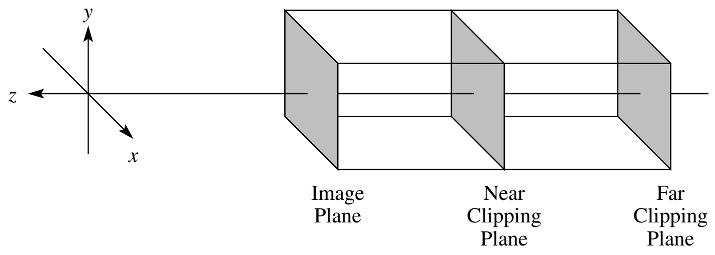


Figure 8.6: Orthographic projection.

The composition of any number of rotations can always be represented as a single rotation about an arbitrary axis. The homogeneous-coordinate transformation matrix $R(a,\theta)$ that corresponds to a rotation of θ about the axis in the direction of the unit vector $a = (a_\mathrm{x}, a_\mathrm{y}, a_\mathrm{z})$ is given by

$$R(a,\theta) = \begin{bmatrix} a_\mathrm{x}^2 + c_\theta(1-a_\mathrm{x}^2) & a_\mathrm{x}a_\mathrm{y}(1-c_\theta) - a_\mathrm{z}s_\theta & a_\mathrm{x}a_\mathrm{z}(1-c_\theta) + a_\mathrm{y}s_\theta & 0 \\ a_\mathrm{x}a_\mathrm{y}(1-c_\theta) + a_\mathrm{z}s_\theta & a_\mathrm{y}^2 + c_\theta(1-a_\mathrm{y}^2) & a_\mathrm{y}a_\mathrm{z}(1-c_\theta) - a_\mathrm{x}s_\theta & 0 \\ a_\mathrm{x}a_\mathrm{z}(1-c_\theta) - a_\mathrm{y}s_\theta & a_\mathrm{y}a_\mathrm{z}(1-c_\theta) + a_\mathrm{x}s_\theta & a_\mathrm{z}^2 + c_\theta(1-a_\mathrm{z}^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $c_\theta = \cos\theta$ and $s_\theta = \sin\theta$.

8.7 Orthographic Projection

Orthographic projection is illustrated in Figure 8.6. The homogeneous-coordinate transformation matrix P that corresponds to an orthographic projection onto the image plane $z = 0$ is given by

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Applying the above transformation to the point $(p_\mathrm{x}, p_\mathrm{y}, p_\mathrm{z})$, we obtain

$$P \begin{bmatrix} p_\mathrm{x} \\ p_\mathrm{y} \\ p_\mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_\mathrm{x} \\ p_\mathrm{y} \\ p_\mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} p_\mathrm{x} \\ p_\mathrm{y} \\ 0 \\ 1 \end{bmatrix}.$$

More complicated orthographic projections can be constructed by combining the above transformation with scaling, rotation, and translation to move the image plane and center of projection wherever they are desired.

8.8 Perspective Projection

Perspective projection is illustrated in Figure 8.7. The homogeneous-coordinate transformation matrix P that corresponds to a perspective projection, with the origin as the center of the projection and $z = 1$ as the image plane, is given by

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

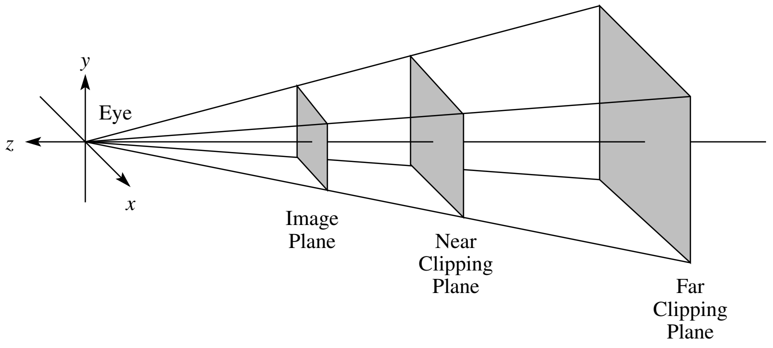


Figure 8.7: Perspective projection.

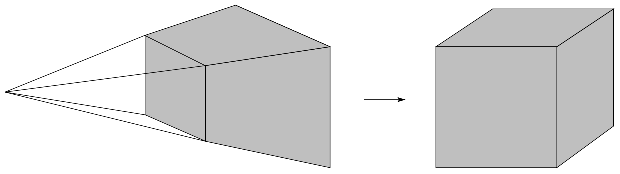


Figure 8.8: Mapping the viewing frustum into a cube.

Applying the above transformation to the point $(p_{\mathsf{x}}, p_{\mathsf{y}}, p_{\mathsf{z}})$, we obtain

$$P\begin{bmatrix}p_{\mathsf{x}}\\p_{\mathsf{y}}\\p_{\mathsf{z}}\\1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}p_{\mathsf{x}}\\p_{\mathsf{y}}\\p_{\mathsf{z}}\\1\end{bmatrix}=\begin{bmatrix}p_{\mathsf{x}}\\p_{\mathsf{y}}\\p_{\mathsf{z}}\\p_{\mathsf{z}}\end{bmatrix}\sim\begin{bmatrix}p_{\mathsf{x}}/p_{\mathsf{z}}\\p_{\mathsf{y}}/p_{\mathsf{z}}\\1\\1\end{bmatrix}.$$

More complicated perspective projections can be constructed by combining the above transformation with scaling, rotation, and translation to move the image plane and center of projection wherever they are desired.

Perspective projection can always be decomposed into two separate transformations, namely, a warping followed by orthographic projection (each of which has a homogeneous-coordinate transformation matrix representation). The warping maps the viewing frustum associated with perspective projection into a cube as shown in Figure 8.8.

8.9 Transformations as a Change in Coordinate System

For a transformation matrix T and a point p expressed as a column vector, consider the product Tp. The product Tp can be interpreted in two distinct but equivalent ways:

1. As the transformation of a point: The product Tp is the new point produced by applying the transformation T to the point p.
2. As the transformation of a coordinate system: The product Tp is the new point obtained by applying a transformation to the coordinate-system axes and then interpreting p relative to these new coordinate-system axes.

Although the first interpretation is, perhaps, the most straightforward, the second interpretation is often very useful in computer graphics applications. Note: In the case that T above is a composite transformation $T = T_n T_{n-1} \cdots T_1$, recall that $T^{-1} = T_1^{-1} T_2^{-1} \cdots T_n^{-1}$.

8.10 Quaternions

The quaternions, denoted $\mathbb{H}$, can be viewed as an extension of complex numbers, and were discovered by William Rowan Hamilton in 1843. Quaternions are extremely useful for representing rotations in 3-D, and have application in areas such as computer graphics and robotics. A **quaternion** q is a number of the form

$$q = w + xi + yj + zk, \quad \text{where} \quad w,x,y,z \in \mathbb{R} \quad \text{and} \quad i^2 = j^2 = k^2 = ijk = -1.$$

A quaternion $q = w + xi + yj + zk$ can be viewed as consisting of a **scalar part**, namely w, and a **vector part** with components in the i, j, and k directions, namely (x,y,z). The quaternion $q = w + xi + yj + zk$ with the scalar part $s = w$ and vector part $v = (x,y,z)$ is denoted as (s,v). For quaternions, addition is defined as

$$(s_1,v_1) + (s_2,v_2) = (s_1 + s_2, v_1 + v_2).$$

Multiplication is defined as

$$(s_1,v_1)(s_2,v_2) = (s_1 s_2 - v_1 \cdot v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2).$$

The conjugate of $q = (s,v)$, denoted q^*, is defined as $q^* = (s,-v)$. The norm of q is defined as $\|q\| = \sqrt{qq^*}$. The multiplicative inverse of q, denoted q^{-1}, is given by $q^*/\|q\|^2$. Addition is commutative. Multiplication is not commutative.

A rotation by the angle θ about the axis v (using a right-hand rule) can be represented by the unit-norm quaternion

$$\left(\cos\theta/2, \tfrac{1}{\|v\|} v \sin\theta/2\right).$$

Quaternion multiplication then corresponds to rotation matrix multiplication. Let q_1 and q_2 be quaternions associated with rotations R_1 and R_2, respectively. Then, the quaternion product $q_2 q_1$ corresponds to the rotation $R_2 R_1$ (i.e., R_1 followed by R_2). Let v_1 and v_2 denote points on the unit sphere in $\mathbb{R}^3$. The quaternion quotient $(0,v_2)/(0,v_1)$ corresponds to a rotation along the great circle arc from v_1 to v_2.

8.11 Topological Spaces

In geometry processing, we typically deal with manifolds. Before we can present the definition of a manifold, however, we must first introduce some concepts related to topological spaces.

Definition 8.8 (Topology)**.** A **topology** on a set X is a set T of subsets of X satisfying:

1. Both $\emptyset$ and X belong to T.
2. The union of any collection of sets from T also belongs to T.
3. The intersection of any finite collection of sets from T belongs to T.

The sets in T will be called the **open** sets of X.

Basically, a topology on a set X specifies which points in X are close to one another (i.e., which points are neighbours). The notion of a topology leads to the concept of a topological space as defined below.

Definition 8.9 (Topological space)**.** A **topological space** is a set X with a topology T (on X), and is denoted as (X,T), or simply X when T is clear from the context.

A metric space is a topological space, since a metric induces a topology on a set. Not all topologies can be described in terms of a metric, however. So, not all topological spaces are metric spaces.

In the case of a metric space, a metric serves as the starting point for defining a topology on a set. In the more general topological-space setting, the topology on a set X is specified directly by identifying all open subsets of X. Then, closed sets and neighbourhoods are defined in terms of open sets. Other concepts (such as closure, boundary, and interior) then follow.

More formally, we define a closed set and neighbourhood as follows.

Definition 8.10 (Closed set)**.** A subset S of a topological space X is said to be **closed** if $X \setminus S$ is open.

Definition 8.11 (Neighbourhood)**.** A subset S of a topological space X is said to be a **neighbourhood** of a point $p \in X$ if there exists an open set U such that $p \in U \subset S$ (i.e., S contains an open set that contains the point p).

A type of topological space that is of particular interest is the one given in the following definition.

Definition 8.12 (Hausdorff space)**.** A topological space X is said to be a **Hausdorff space** if any two distinct points of X have disjoint neighbourhoods.

Every metric space is a Hausdorff space. Not every topological space is a Hausdorff space, however. Since topological spaces that are not Hausdorff have very strange topologies, we are usually only interested in Hausdorff spaces.

Just to show that topological spaces that are not Hausdorff do exist, we provide a simple example.

Example 8.2 (Trivial topology)**.** Let X be any nonempty set, and let T be the set of subsets of X consisting of $\emptyset$ and X (i.e., $T = \{\emptyset, X\}$). (a) Show that T is a topology on X, and hence (X,T) is a topological space. (Incidentally, this particular choice of T is known as the **trivial topology**.) (b) Show that (X,T) is not a Hausdorff space.

Solution. (a) To show that T is a topology, we must show that T meets the three requirements of a topology (as identified in Definition 8.8). Trivially, we have that $\emptyset \in T$ and $X \in T$. Now, we consider all possible collections (with at least two elements) of sets from T. Clearly, there is only one such collection, namely, $\{\emptyset, X\}$. In this case, we have $\emptyset \cup X = X \in T$. So, the union of any collection of elements in T is also in T. Next, we consider all possible finite collections (with at least two elements) of sets from T. Again, clearly, there is only one such collection, namely, $\{\emptyset, X\}$. In this case, we have $\emptyset \cap X = \emptyset \in T$. So, the intersection of any finite collection of elements in T is also in T. Thus, T is a topology on X.

(b) For any $p \in X$, $X \in T$ is a neighbourhood of p. Furthermore, X is the only neighbourhood of p (since the only other element in T is $\emptyset$ and $\emptyset$ cannot be the neighbourhood of any point). Since p was chosen arbitrarily, this means that each point in X has exactly one neighbourhood and this neighbourhood is X. Thus, for any $p, q \in X$, every (i.e., the only one) neighbourhood of p has a nonempty intersection with every (i.e, the only one) neighbourhood of q, which violates the Hausdorff condition. Therefore, (X,T) is not a Hausdorff space. □

In the context of topological spaces, an important type of transformation is a homeomorphism, which is defined as follows.

Definition 8.13 (Homeomorphism)**.** A mapping $T : X \to Y$ that is one-to-one and onto, where T and T^{-1} are both continuous, is called a **homeomorphism**.

In simple terms, a homeomorphism can be thought of as an elastic deformation (i.e., stretching/compressing, bending/twisting, but no cutting/tearing/splitting/joining). Two spaces X and Y are said to be **homeomorphic** if there exists a homeomorphism $T : X \to Y$. Two topological spaces X and Y are equivalent, denoted $X = Y$, if they are homeomorphic.

Example 8.3. Let C and D denote the surfaces of the coffee cup and donut, respectively, as shown in Figure 8.9. Since C can be transformed into D (or vice versa) by an elastic deformation, C and D are homeomorphic. Thus, $C = D$. That is, a coffee cup and donut are topologically equivalent. Of course, this leads to the joke that a mathematical topologist cannot hold a job at the local donut shop, since he/she cannot tell the difference between a coffee cup and a donut.

8.12 Manifolds

Having introduced various concepts related to topological spaces, we can now define a manifold.

Definition 8.14 (Manifold)**.** An **n-dimensional manifold** (called an n-manifold) is a Hausdorff space M such that each point $p \in M$ has a neighbourhood homeomorphic to the open n-dimensional unit disc $U^n = \{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n : \sum_{k=1}^{n} x_k^2 < 1\}$.

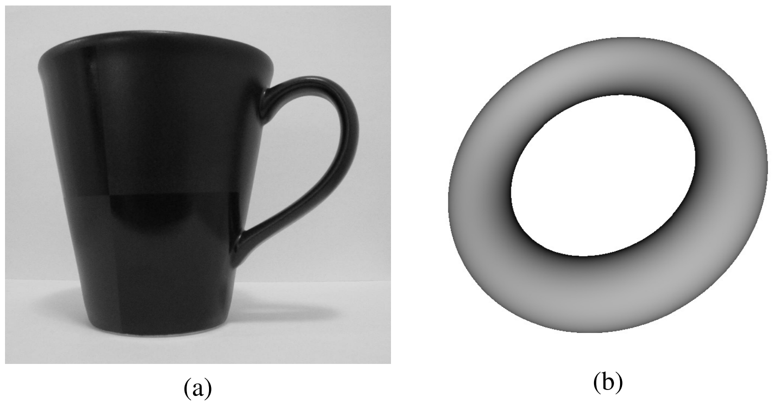


Figure 8.9: Example of homeomorphic topological spaces. (a) Coffee cup and (b) donut.

Definition 8.15 (Manifold with boundary)**.** An ***n*-dimensional manifold with boundary** (called an n-manifold with boundary) is a Hausdorff space M such that each point $p \in M$ has a neighbourhood homeomorphic to either the open n-dimensional unit disc U^n or the open n-dimensional unit half-disc $U_+^n = \{(x_1, x_2, \ldots, x_n) : \sum_{k=1}^n x_k^2 < 1 \text{ and } x_1 \geq 0\}$.

A 2-manifold is called a **surface**. An n-manifold (with or without boundary) locally has the same properties as $\mathbb{R}^n$. For example, an infinitesimally small bug crawling along a 2-manifold M could not distinguish M from the plane $\mathbb{R}^2$ if the range of the bug's vision were restricted to only its local neighbourhood. Examples of 1-manifolds include a line and circle. Some examples of 2-manifolds are shown in Figure 8.10. Some examples of 3-manifolds are shown in Figure 8.11. An example of an n-manifold is $\mathbb{R}^n$. In geometry processing, manifolds play a crucial role, since many geometric objects of practical interest are manifolds. Some examples of non-manifolds are shown in Figure 8.12.

A manifold is said to be **orientable** if one can consistently define a clockwise (or equivalently counterclockwise) direction for all loops (i.e., closed paths) in the manifold. We are interested in the orientability of surfaces (i.e., 2-manifolds). Most surfaces encountered in the physical world are orientable. Examples of orientable surfaces include: a sphere, plane, torus, and the surface of a polyhedron. An example of a non-orientable surface is a Mobius strip. An example of an orientable and non-orientable manifold is shown in Figure 8.13.

The signals dealt with in geometry processing are most commonly manifolds. In traditional signal processing, a signal is a function, which is a vector in an inner-product space. In geometry processing, however, a signal is a manifold, which is a topological space. In most practical applications, the n-manifold M (with or without boundary) of interest is imbedded into a Euclidean space $\mathbb{R}^d$, where $n \leq d$ (i.e., $M \subset \mathbb{R}^d$). Generally speaking, 2- and 3-manifolds (with or without boundaries) tend to be of interest most frequently. Since we are interested in subdivision surfaces, we are not surprisingly interested in surfaces (i.e., 2-manifolds) with or without boundaries.

8.13 Representations of Surfaces

To facilitate the processing of surfaces (i.e., 2-manifolds), we need a way to define them. Numerous ways in which to specify a surface exist, each with its own advantages and disadvantages.

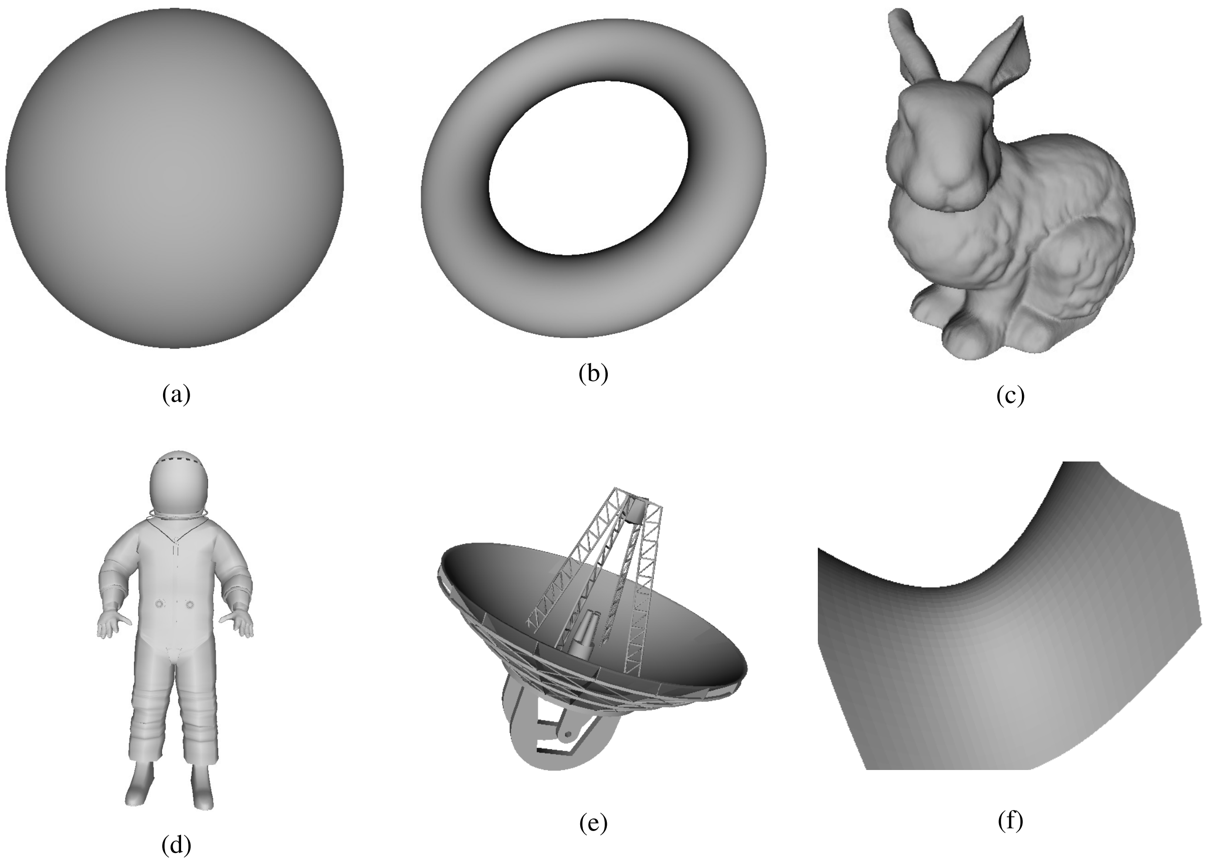
A surface can be defined implicitly. That is, a surface can be defined to consist of all of the points (x, y, z) satisfying a particular equation

$$f(x, y, z) = 0.$$

Example 8.4 (Implicit representation of a sphere)**.** The surface of a sphere with radius ρ and center at the origin consists of the points $\{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = \rho^2\}$.

A surface can be defined parametrically. That is, a surface can be defined to consist of the points in the range of a vector-valued function

$$p(u, v) = (f_x(u, v), f_y(u, v), f_z(u, v)).$$



(a) (b) (c) (d) (e) (f)

Figure 8.10: Examples of 2-manifolds with or without boundaries. (a) A sphere. (b) A torus. (c) The surface of a rabbit. (d) The surface of a spacesuit. (e) The surface of a telescope dish. (f) A simple surface.

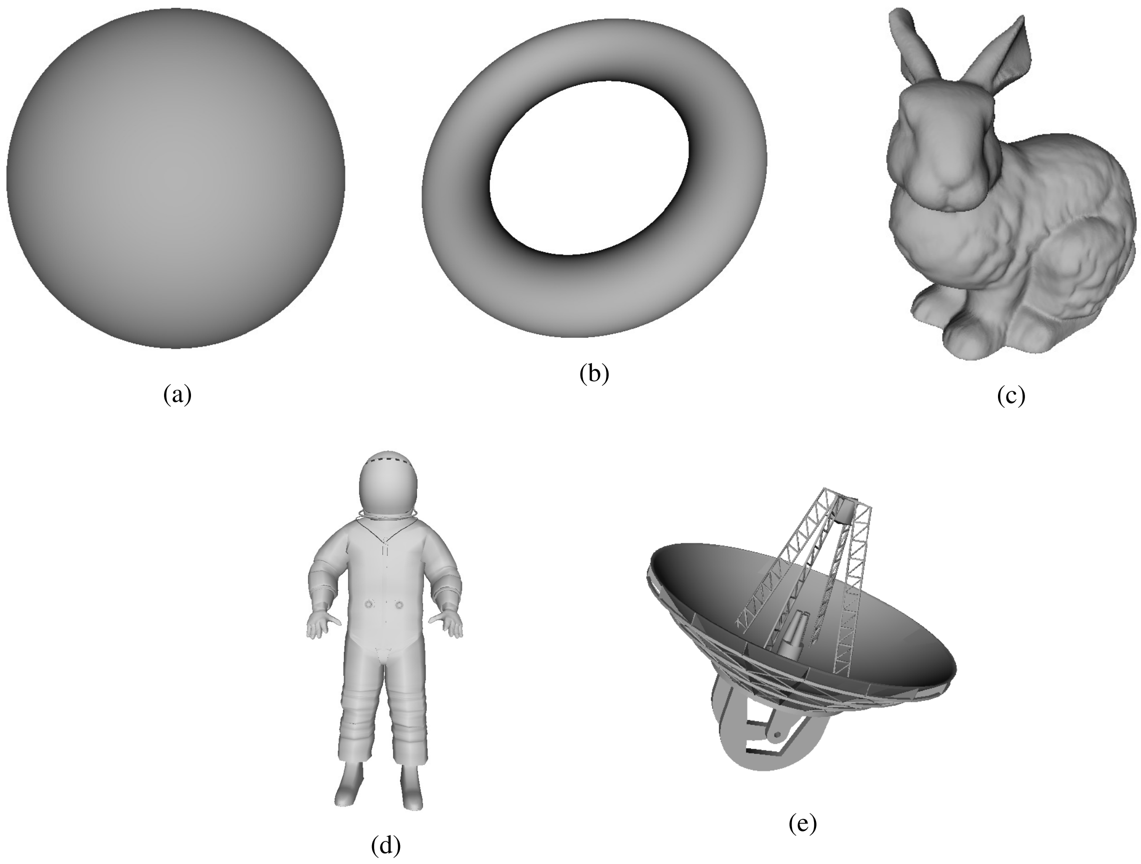


Figure 8.11: Examples of 3-manifolds with boundaries. (a) A ball (i.e., a sphere and its interior). (b) A toroid (i.e., a torus and its interior). (c) A rabbit and its interior. (d) A spacesuit and its interior. (e) A telescope dish and its interior.

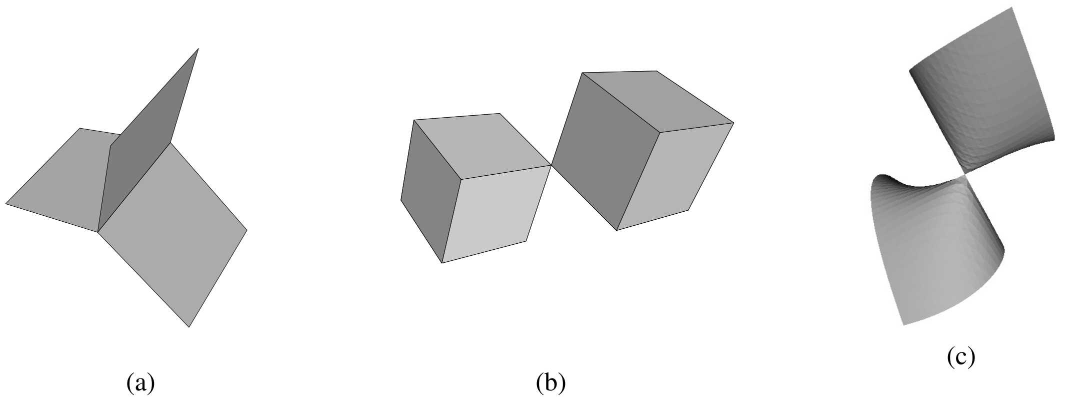


Figure 8.12: Examples of non-manifolds. (a) Three planar quadrilaterals sharing a common edge. (b) Two cubes intersecting at a single shared vertex. (c) Two surfaces touching at a single point.

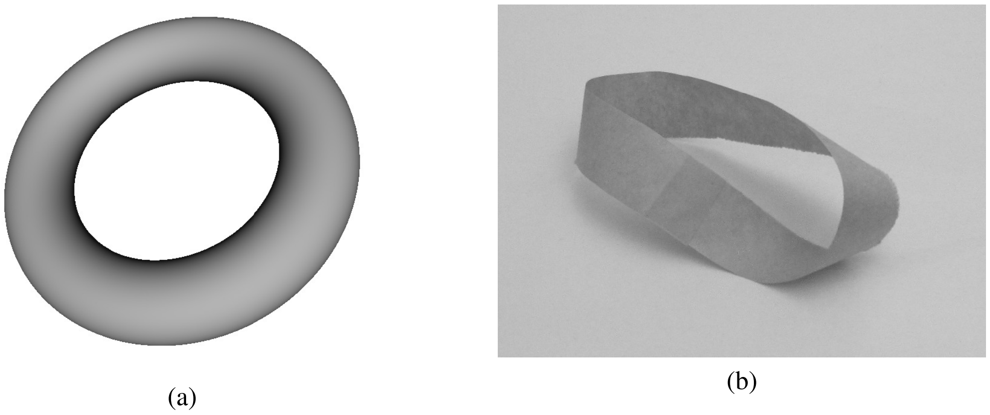


Figure 8.13: Orientability example. (a) A torus, which is orientable. (b) A Mobius strip, which is not orientable.

Example 8.5 (Parametric representation of a sphere)**.** The surface of a sphere with radius ρ and center at the origin consists of the points in the range of the (vector-valued) function $p(u,v) = (\rho\cos(u)\cos(v), \rho\sin(u)\cos(v), \rho\sin(v))$, where $u \in [0,2\pi), v \in [0,\pi]$.

Often, implicit and parametric representations are not the most convenient to utilize. Yet another approach is to define a surface (or an approximation thereof) using a polygon mesh, which we will consider in detail later.

8.14 Parametric Continuity

Often, we wish to quantify the smoothness of a surface. Parametric continuity is a measure of the smoothness of a surface. The parametric form of a surface S (imbedded in $\mathbb{R}^3$) is given by $p(u,v) = \begin{bmatrix} x(u,v) & y(u,v) & z(u,v) \end{bmatrix}^T$. The surface S is traced out by p as the parameters u and v are varied over their domain. The surface S is said to be C^n **continuous** if all partial derivatives of p of order n or less exist and are continuous. A tangent plane to S is determined by the first-order partial derivatives

$$\tfrac{\partial p(u,v)}{\partial u} = \begin{bmatrix} \tfrac{\partial x(u,v)}{\partial u} & \tfrac{\partial y(u,v)}{\partial u} & \tfrac{\partial z(u,v)}{\partial u} \end{bmatrix}^T \quad \text{and} \quad \tfrac{\partial p(u,v)}{\partial v} = \begin{bmatrix} \tfrac{\partial x(u,v)}{\partial v} & \tfrac{\partial y(u,v)}{\partial v} & \tfrac{\partial z(u,v)}{\partial v} \end{bmatrix}^T.$$

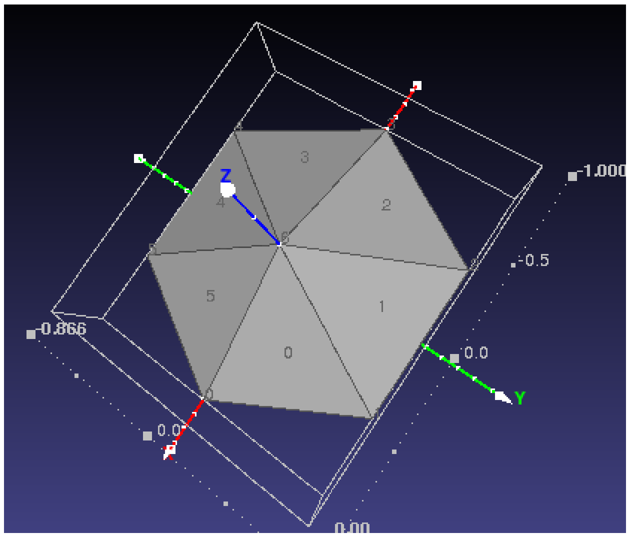
The parameterization of a surface is not unique. Different parameterizations of the same surface can have different parametric continuity. That is, parametric continuity depends not only on the surface being parameterized, but also on the parameterization itself.

8.15 Geometric Continuity

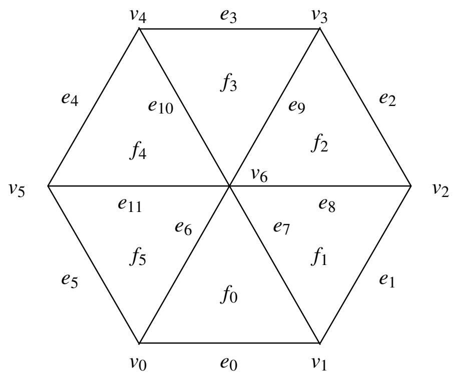
Geometric continuity is a measure of the smoothness of a surface that is independent of any surface parameterization. A surface is G^0 **continuous** if it does not have any jumps (i.e., is continuous). A surface is G^1 **continuous** if it has a continuously varying tangent plane. A surface is G^2 **continuous** if it has continuously varying curvature. A polyhedral surface is G^0 continuous but (typically) not G^1 continuous (since the tangent plane does not vary continuously between faces of a polyhedron). Parametric continuity is a stronger form of continuity than geometric continuity. That is, C^n continuity implies G^n continuity (but G^n continuity does not necessarily imply C^n continuity).

8.16 Polygon Meshes

In practice, one very useful representation scheme for surfaces is the polygon mesh, as defined below.



(a)



for $k \in \{0,1,2,3,4,5\}$:

$v_k = (\cos(\frac{\pi}{3}[k-2]), \sin(\frac{\pi}{3}[k-2]), 0)$

$v_6 = (0,0,\frac{1}{2})$

(b)

Figure 8.14: Example of a polygon mesh. (a) Three-dimensional view of the mesh. (b) Topology of the mesh.

Definition 8.16 (Polygon mesh)**.** A **polygon mesh** is a collection of vertices, edges, and (polygonal) faces, and an incidence relationship amongst them. An edge connects two vertices, and a face is a closed sequence of edges.

A polygon mesh consists of two types of information:

1. geometric information: the positions of the vertices (in the Euclidean space in which the mesh is imbedded); and

2. topologic (i.e., connectivity) information: how the vertices are connected together to form edges and faces.

In practice, the faces in a mesh are most commonly all triangles, all quadrilaterals, or a mixture of triangles and quadrilaterals. A polygon mesh with all triangle faces is called a **triangle mesh**. A polygon mesh with all quadrilateral faces is called a **quadrilateral mesh** (or **quad mesh**). A polygon mesh can be manifold or non-manifold. Also, it can be orientable or non-orientable.

Example 8.6. An example of a triangle mesh is shown in Figure 8.14. A three-dimensional view of the mesh is shown in Figure 8.14(a), and the topology of the mesh is shown in Figure 8.14(b). The mesh is manifold and orientable, and consists of 7 vertices, 12 edges, and 6 (triangle) faces. The face f_0, for example, consists of the vertices v_0, v_1, v_6 and the edges e_0, e_7, e_6.

Example 8.7. Some more exotic examples of polygon meshes are shown in Figure 8.15.

The **valence** of a vertex is the number of edges incident on that vertex. The **1-ring** of a vertex v is the set of all vertices in the mesh that are directly connected by an edge to v. The 1-ring is illustrated pictorially in Figure 8.16. In the figure, the vertices in the 1-ring of the vertex v are marked by dots. The **valence** of a face is the number of edges incident on that face (or equivalently, the number of vertices incident on that face). For example, a triangle face has a valence of three, while a quadrilateral face has a valence of four.

It is important to understand that a mesh is not determined by either its geometry or topology alone. The character of a mesh is determined jointly by both its geometry and topology. Two meshes with identical geometry but distinct topologies can have radically different appearances. Similarly, two meshes with identical topology but distinct geometries can also be radically different in appearance. An example of two quadrilateral meshes with the same geometry but different topologies is shown in Figure 8.17. An example of two quadrilateral meshes with the same topology but different geometries is shown in Figure 8.18.

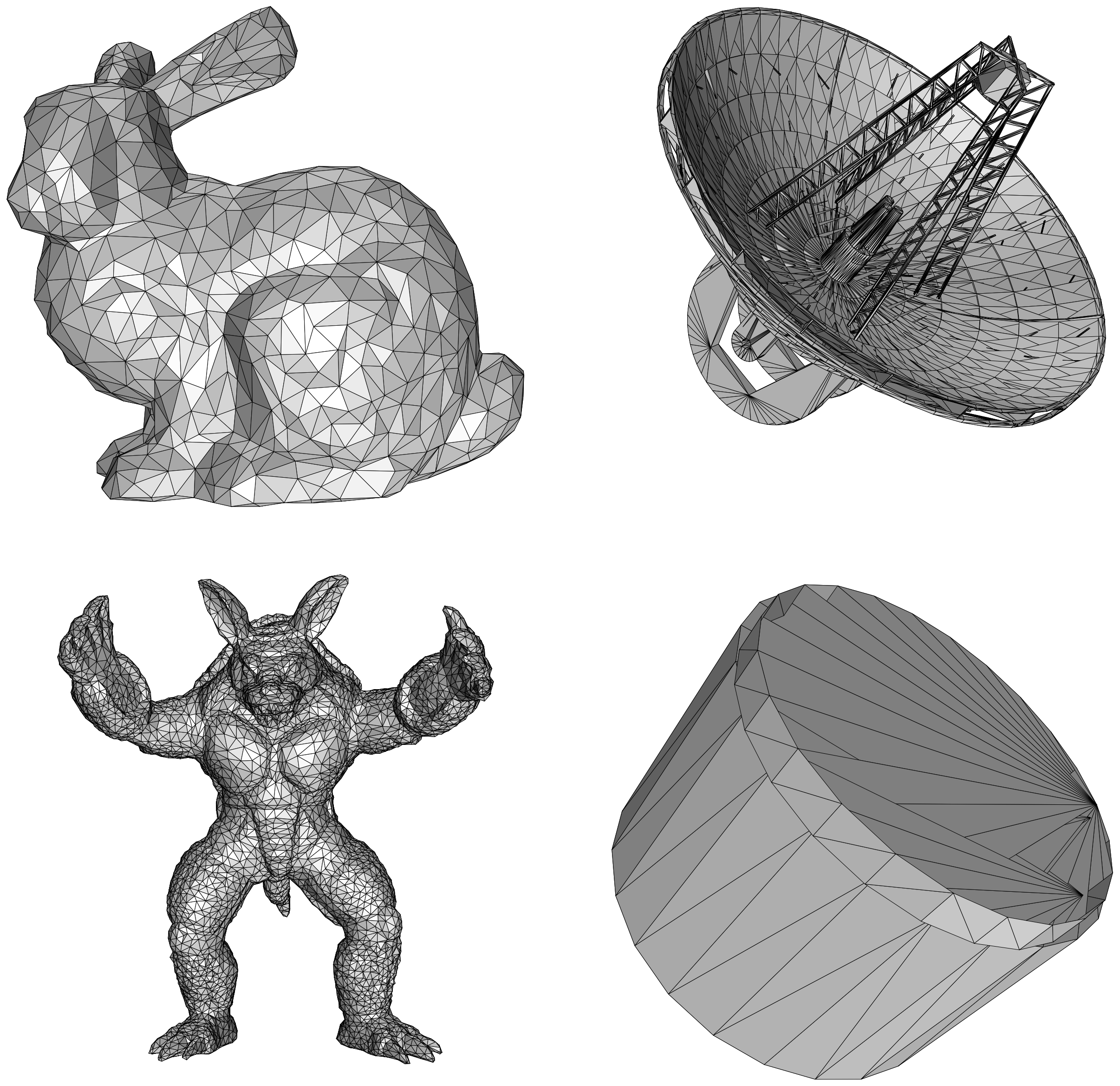


Figure 8.15: Examples of polygon meshes.

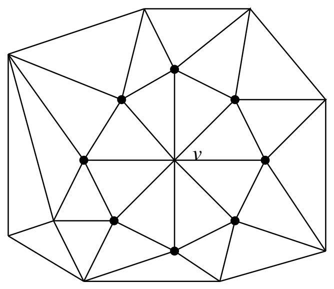


Figure 8.16: The 1-ring of a vertex.

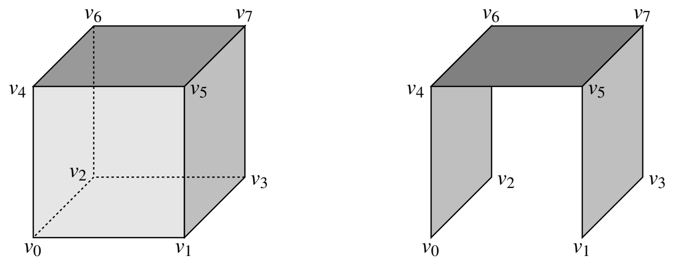


Figure 8.17: An example of two quadrilateral meshes with the same geometry but different topologies.

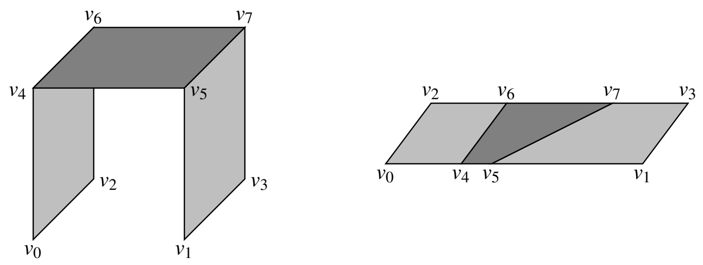
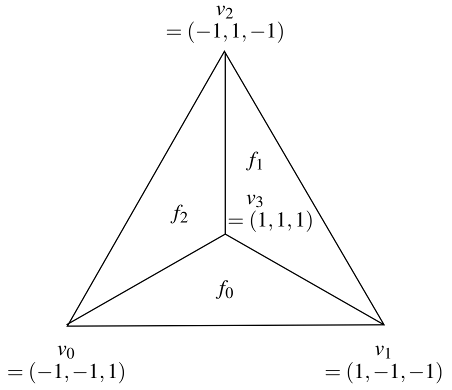


Figure 8.18: An example of two quadrilateral meshes with the same topology but different geometries.

```
Vertex vertices[numVertices]; // vertex array
Face faces[numFaces]; // face array

int faceValence; // number of vertices per face
struct Face {
    int vertexIndexes[faceValence]; // indexes of vertices for this face
};
```

Figure 8.19: Definition of naive data structure.



(a)

Vertex Array

Array Index	Array Element
0	(-1,-1,1)
1	(1,-1,-1)
2	(-1,1,-1)
3	(1,1,1)

Face Array

Array Index	Array Element
0	0, 1, 3
1	1, 2, 3
2	0, 3, 2

(b)

Figure 8.20: Pictorial view of naive data structure. (a) A mesh and (b) the corresponding data structure.

8.17 Data Structures for Polygon Meshes

To use a polygon mesh in software, clearly some data structure is needed to represent the mesh. Over the years, numerous data structures have been proposed for representing meshes, each having its own advantages and disadvantages. In the development of data structures for meshes, an important consideration is to what degree a data structure allows for efficient algorithms. In this regard, it is important to note that adjacency queries are one of the most frequent operations in most geometric algorithms. (An adjacency query simply finds the adjacent vertices/edges/faces to a given vertex/edge/face.) For this reason, it is usually critical that adjacency queries be fast (i.e., requiring constant time). In the sections that follow, we consider several data structures for polygon meshes.

8.17.1 Naive Data Structure

The simplest data structure for representing a polygon mesh is what we refer to as the naive data structure. With this data structure, a mesh is represented using two arrays, namely, a vertex array and a face array. Edges are not explicitly represented. Each entry in the vertex array holds the coordinates of a vertex in the mesh. Each entry in the face array holds the indices in the vertex array of the vertices associated with a single face in the mesh (in, say, CCW order). Pseudocode for the data structure is given in Figure 8.19. For a simple mesh, this data structure is illustrated in Figure 8.20.

The main advantage of the naive data structure is its simplicity. Also, this representation may require less memory than other approaches. Unfortunately, with the naive data structure, adjacency information is not readily accessible. For example, to find a neighboring face of a given face f, we must scan through the face array looking for a face with two vertices in common with f, which takes time that is linear in the number of faces (or vertices) in the mesh. Since adjacency information is not readily available, geometric algorithms employing this data structure will be grossly inefficient (in terms of execution time).

```
struct Edge {
    Vertex* pvt; // first vertex
    Vertex* nvt; // second vertex
    Face* pface; // first face (defined to be to left of directed line
                 // segment from first to second vertex)
    Face* nface; // second face (defined as other face incident on edge)
    Edge* pccw;  // neighboring edge on first face in CCW direction
    Edge* pcw;   // neighboring edge on first face in CW direction
    Edge* nccw;  // neighboring edge on second face in CCW direction
    Edge* ncw;   // neighboring edge on second face in CW direction
};
```

Figure 8.21: Definition of winged-edge data structure.

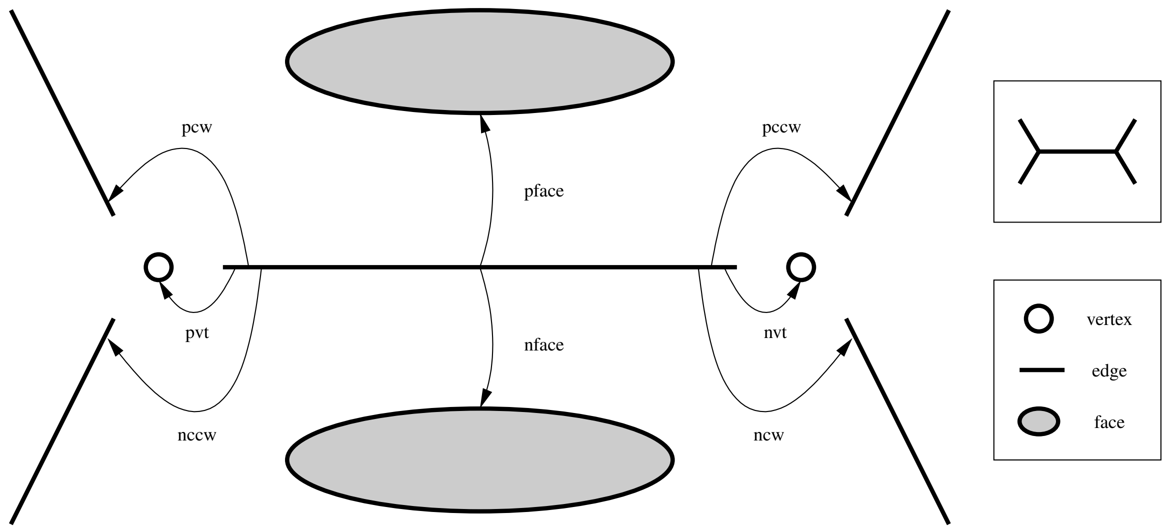


Figure 8.22: Pictorial view of winged-edge data structure.

In practice, the naive data structure is rarely ever used, due to its inability to efficiently handle adjacency queries. Instead, data structures that allow for more efficient access to adjacency information are employed.

8.17.2 Winged-Edge Data Structure

The next data structure for polygon meshes to be considered is the winged-edge data structure, proposed in [3]. This data structure explicitly represents vertices, edges, and faces, with edges serving as the glue that holds vertices and faces together. Because edges are explicitly represented and play such an important role in this data structure, it is called an edge-based representation. Pseudocode for the winged-edge data structure is given in Figure 8.21. Also, a diagram showing the structure associated with the small part of a mesh is shown in Figure 8.22. As can be seen from the diagram, each edge points to two incident faces, two incident vertices, and four incident edges that share the same faces and vertices (i.e., the four “wing” edges as they are called). In terms of memory cost, the structure requires 8 pointers per edge.

Relative to the naive data structure, the winged-edge data structure allows for much more efficient adjacency queries. For example, the faces incident on a given edge can be found in constant time by dereferencing the corresponding `pface` and `nface` pointers.

In an edge-based representation (such as the winged-edge data structure), the manner in which an edge is represented is extremely important. With the winged-edge data structure, an edge is represented in a very simple direct

```
struct Edge {
    HalfEdge e[2]; // pair of symmetric half-edges
};

struct HalfEdge {
    int index;      // index of half-edge in parent edge
    HalfEdge* next; // next CCW half-edge around left face
    Vertex* term;   // terminal vertex
    Face* left;     // left face
};
```

Figure 8.23: Definition of half-edge data structure.

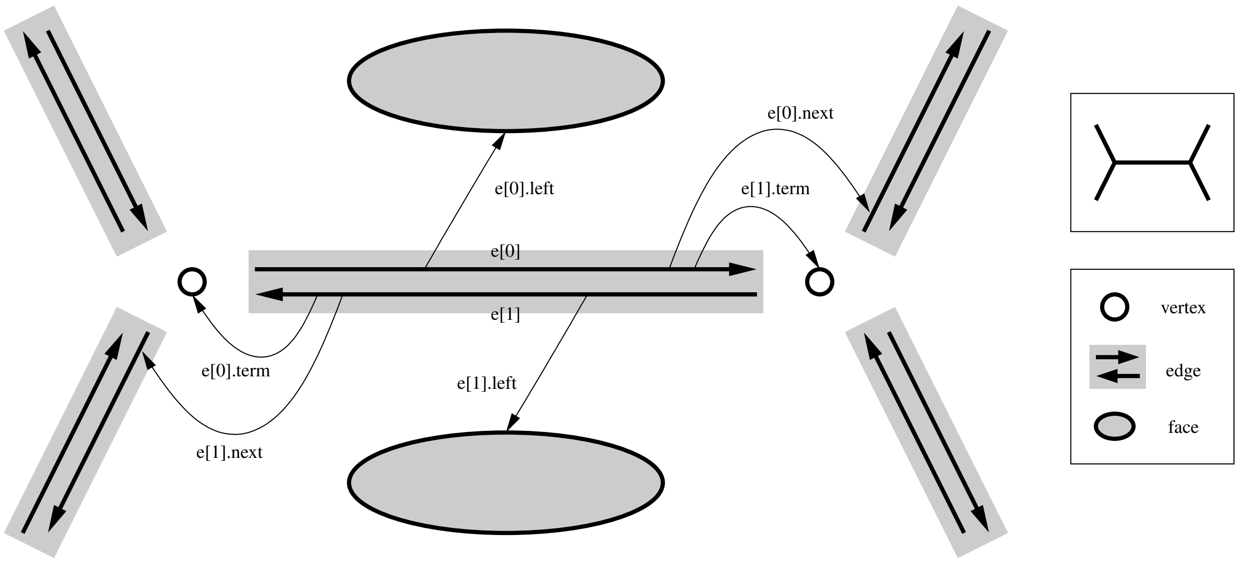


Figure 8.24: Pictorial view of half-edge data structure.

manner as a single atomic entity. Consequently, the way in which an edge is represented is perfectly symmetric. Although such symmetry might seem desirable at first glance, it is not. In fact, this symmetry often tends to complicate algorithms.

The winged-edge data structure improves upon the naive data structure. In practice, however, the winged-edge data structure is still not the most frequently used. Instead, other edge-based representations to be introduced shortly tend to be employed more often.

8.17.3 Half-Edge Data Structure

One popular data structure for polygon meshes is the half-edge data structure described in [11], known by the name face-edge (F-E) structure therein. The half-edge data structure is also referred to as a doubly-connected edge list (DCEL) in [5, p. 31], although the original reference for the DCEL [10] describes a different data structure. The half-edge data structure is another edge-based representation. Unlike in the case of the winged-edge data structure, however, an edge is not represented as an atomic entity. Instead, an (undirected) edge is represented as pair of directed edges that are oriented in opposite directions. Each of these directed edges is called a half edge. Pseudocode for this data structure is given in Figure 8.23. Also, a diagram showing the structure associated with the small part of a mesh is illustrated in Figure 8.24. In terms of memory cost, the data structure requires 6 pointers plus 2 bits (i.e., 2 one-bit integers) per edge. The data structure can accommodate, in an efficient manner, navigating around: 1) all of the edges incident on a face in the CCW direction; and 2) all of the faces incident on a vertex in CW direction. (Note that moving in the opposite directions cannot be done efficiently without modifications to the data structure.)

As it turns out, representing edges as pairs of directed edges has some significant practical benefits in terms of algorithms. For example, a half edge can be used to uniquely identify a particular vertex or face in the mesh. That is, one can employ the convention that a half edge is used to name the vertex at its terminus or the face on its left. Edges (as opposed to half edges), however, cannot be used to uniquely identify vertices or faces in this manner. This is due to the fact that one cannot speak meaningfully of the vertex at the terminus of an edge or the face to the left of an edge, as an edge is perfectly symmetric and has no left/right side or origin/terminus. Also, because a half edge has an orientation, this orientation can be used to simplify algorithms that must navigate around the mesh. For example, instead of needing to convey an edge and a direction in which to move, one can simply indicate a half edge, since a direction is implicit in the half edge itself.

Due to its numerous advantages, the half-edge data structure is used quite heavily in practice. Many software applications and libraries utilize this data structure, including the Computational Geometry Algorithms Library (CGAL) [4].

8.17.4 Quad-Edge Data Structure

Another popular data structure for polygon meshes is the quad-edge data structure, proposed in [8]. This data structure is another example of an edge-based representation. Instead of representing each edge as an atomic entity, an edge is represented as a set of two pairs of directed edges. One pair corresponds to edges in the mesh and one pair corresponds to edges in the dual mesh. This allows the quad-edge data structure to represent the mesh and its dual simultaneously. In effect, each edge belongs to four circular singly-linked lists, one for each of the two vertices and two faces incident on the edge. Pseudocode for the data structure is given in Figure 8.25. Also, a diagram showing the structure for a small part of a mesh is illustrated in Figure 8.26. In terms of memory cost, the data structure requires 8 pointers and 4 two-bit integers per edge. The data structure can accommodate, in an efficient manner, navigating around: 1) all of the edges incident on a face in both the CCW and CW directions; and 2) all of the edges incident on a vertex in both the CCW and CW directions.

Compared to the half-edge data structure, the quad-edge data structure has a higher memory cost, but allows for more flexible navigation around elements in the mesh and simultaneous representation of the dual mesh. Consequently, the half-edge data structure is probably preferable unless on one these additional functionalities is desired.

The quad-edge data structure is quite commonly used in practice. Some examples of software using this data structure include the Scape terrain-simplification software [6] and Dani Lischinski's constrained Delaunay-triangulation software [9].

8.18 File Formats for Polygon Meshes

Many different file formats are employed for the interchange of polygon mesh data. One commonly-used format is the object-file format (OFF), which is discussed in the section that follows.

8.18.1 Object File Format (OFF)

The object-file format (OFF) is commonly used to interchange polygon mesh data. The format employs a very simple scheme for encoding the geometry and topology of a polygon mesh, and also has provisions for including color and normal information. In what follows, we provide an example illustrating the basics of the format.

Example 8.8 (Triangle mesh)**.** The OFF format encoding of the triangle mesh shown in Figure 8.27(a) is given in Figure 8.27(b).

Example 8.9 (Quadrilateral mesh)**.** The OFF format encoding of the quadrilateral mesh shown in Figure 8.28(a) is given in Figure 8.28(b).

Example 8.10 (Quadrilateral mesh)**.** The OFF format encoding of the quadrilateral mesh shown in Figure 8.29(a) is given in Figure 8.29(b).

```
struct Edge {
    QuadEdge* e[4]; // four quad-edges of edge
};

struct QuadEdge {
    int index;      // index of quad-edge in parent edge
    QuadEdge* next; // next CCW quad-edge with same origin
    void* data;     // face or vertex
}
```

Figure 8.25: Definition of quad-edge data structure.

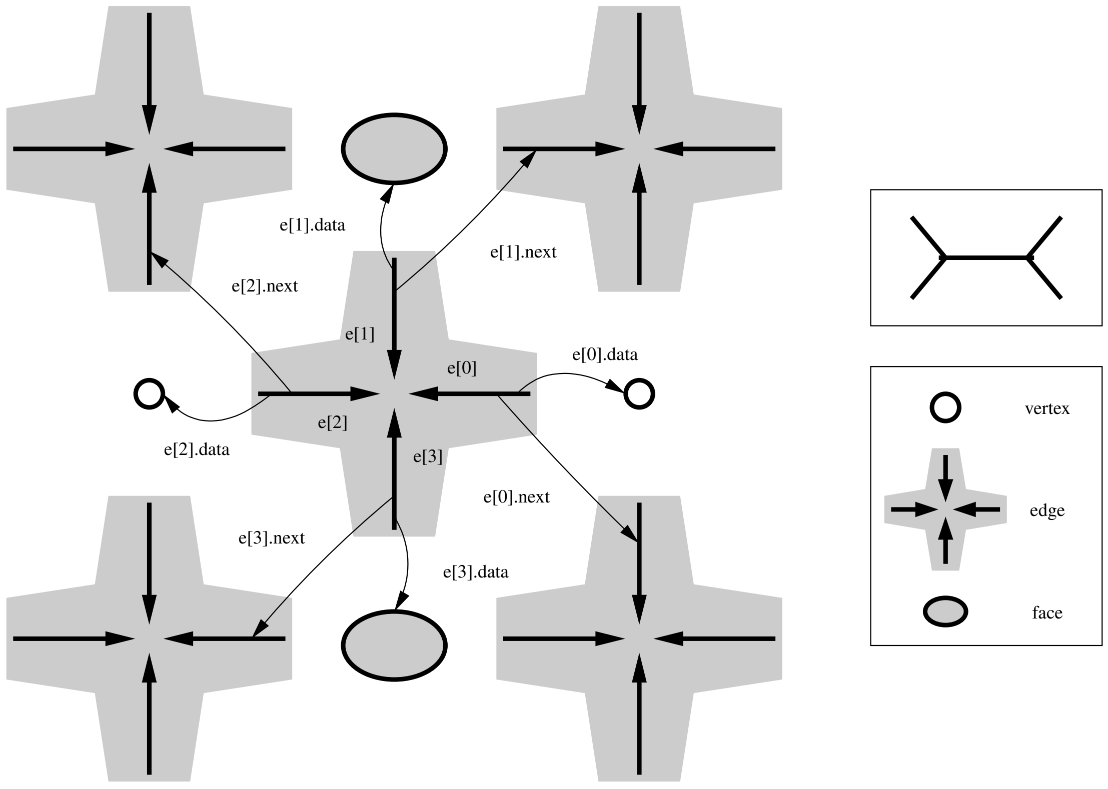
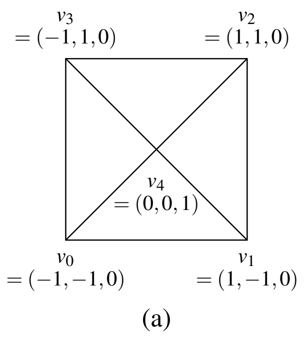


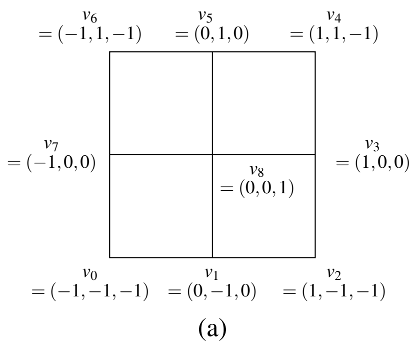
Figure 8.26: Pictorial view of quad-edge data structure.



```
OFF
5 4 0
-1 -1 0
 1 -1 0
 1  1 0
-1  1 0
 0  0 1
3 0 1 4
3 1 2 4
3 2 3 4
3 0 4 3
```

(b)

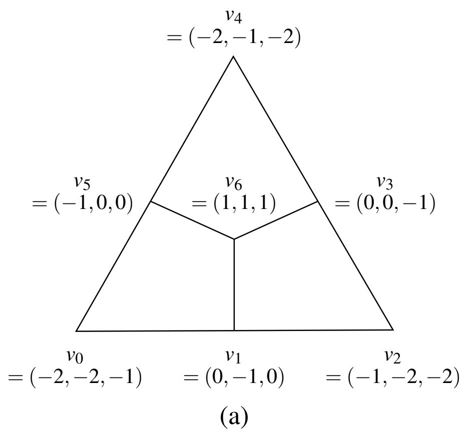
Figure 8.27: Triangle mesh example. (a) The mesh and (b) its corresponding OFF file.



```
OFF
9 4 0
-1 -1 -1
 0 -1  0
 1 -1 -1
 1  0  0
 1  1 -1
 0  1  0
-1  1 -1
-1  0  0
 0  0  1
4 0 1 8 7
4 1 2 3 8
4 8 3 4 5
4 7 8 5 6
```

(b)

Figure 8.28: Quadrilateral mesh example. (a) The mesh and (b) its corresponding OFF file.



```
OFF
7 3 0
-2 -2 -1
 0 -1  0
-1 -2 -2
 0  0 -1
-2 -1 -2
-1  0  0
 1  1  1
4 0 1 6 5
4 1 2 3 6
4 6 3 4 5
```

(b)

Figure 8.29: Quadrilateral mesh example. (a) The mesh and (b) its corresponding OFF file.

8.19 Additional Reading

For a much more detailed and comprehensive treatment of many topics related geometry processing, the reader is referred to the excellent books [2] and [1].

8.20 Problems

8.1 For each of the following cases, determine whether the set S is convex:
(a) the points S belonging to a triangle and its interior;
(b) the points S belonging to the quadrilateral Q and its interior, where the vertices of Q in counterclockwise (CCW) order are $(0,0)$, $(-1,1)$, $(0,-1)$, $(1,1)$;
(c) $S = \{(x,y,z) \in \mathbb{R}^3 : x^2+y^2+z^2 = 1\}$; and
(d) $S = \{(x,y) \in \mathbb{R}^2 : x^2+y^2 \leq 1\}$.

8.2 Find a unit vector n that is perpendicular to the vectors $v = (1,2,-1)$ and $w = (-1,2,1)$.

8.3 Consider the product T of homogeneous-coordinate-transformation matrices given by

$$T = \begin{bmatrix} \pi & 0 & 0 & 0 \\ 0 & \pi & 0 & 0 \\ 0 & 0 & \pi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \pi \\ 0 & 1 & 0 & -\pi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

(a) When T is viewed as a transformation applied to a point, determine the sequence of transformations (e.g., translations, rotations, scalings) to which T directly corresponds.
(b) When T is viewed as transformation applied to the coordinate system axes, determine the sequence of transformations to which T directly corresponds.
(c) When T^{-1} is viewed as a transformation applied to a point, determine the sequence of transformations (e.g., translations, rotations, scalings) to which T^{-1} directly corresponds.
(d) When T^{-1} is viewed as transformation applied to the coordinate system axes, determine the sequence of transformations to which T^{-1} directly corresponds.

8.21 Bibliography

[1] M. K. Agoston. *Computer Graphics and Geometric Modeling: Implementation and Algorithms*. Springer, London, UK, 2005.

[2] M. K. Agoston. *Computer Graphics and Geometric Modeling: Mathematics*. Springer, London, UK, 2005.

[3] B. G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, pages 589–596, 1975.

[4] *CGAL User and Reference Manual: All Parts (Release 4.0.2)*, July 2012. Available online from `http://www.cgal.org`.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, New York, NY, USA, 2nd edition, 2000.

[6] M. Garland and P. S. Heckbert. Fast triangular approximation of terrains and height fields, 1997. Draft (May 2, 1997, 19 pages).

[7] R. N. Goldman. Illicit expressions in vector algebra. *ACM Transactions on Graphics*, 4(3):223–243, July 1985.

[8] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[9] D. Lischinski. Incremental Delaunay triangulation. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 47–59, San Diego, CA, USA, 1994. Academic Press.

[10] D. F. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.

[11] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, January 1985.

Chapter 9

Subdivision Surfaces and Subdivision Wavelets

9.1 Introduction

As it turns out, the mathematical structure associated with wavelets, as discussed in earlier chapters, can be generalized to handle geometric objects. This chapter examines topics related to this generalization.

9.2 Subdivision Surfaces

In many applications, polygon meshes are used to model surfaces. Modelling smooth surfaces with polygon meshes is problematic, however. To obtain a reasonably good approximation of a smooth surface with a polygon mesh, a very fine mesh with an extremely large number of faces is typically required. Subdivision solves this problem by representing smooth surfaces in terms of a coarse mesh. Subdivision provides a set of well-defined rules for producing successively refined versions of a mesh. Typically, these rules are chosen so that repeating the refinement process ad infinitum produces, in the limit, a smooth (or mostly smooth) surface. A subdivision example is shown in Figure 9.1.

Subdivision is a general method for constructing a finer mesh from a coarser one, through the introduction of new vertices as well as edges and faces. Or put another way, subdivision is essentially an approximation/interpolation technique. The coarser mesh that serves as the starting point for subdivision is called a **control mesh**. The addition of new vertices to a mesh requires modifications to both the topology (i.e., connectivity) and geometry (i.e., vertex positions) of the mesh. For this reason, each subdivision scheme requires the specification of two rules:

1. a **topologic refinement rule** that describes how the connectivity of the mesh is to be modified in order to incorporate the new vertices being added to the mesh; and

2. a **geometric refinement rule** that describes how the geometry of the mesh is to be changed in order to accommodate the new vertices being added (where these modifications may affect the position of previously-existing vertices).

In practice, the refinement rules are applied repeatedly until a mesh of the desired fineness is obtained. Provided that a subdivision scheme is well behaved, if the refinement rules are applied repeatedly ad infinitum, the vertices in the mesh will converge, in the limit, to a surface. Such a surface is called a **limit surface**.

If the same topologic and geometric rules are used in each level of a subdivision scheme, the scheme is said to be **stationary**. If the same geometric rule is used to determine all of the vertices within a single level of a subdivision scheme, the scheme is said to be **uniform**.

A subdivision scheme is said to be **interpolating** if it always produces refined meshes that pass through all of the vertices of the original control mesh. Otherwise, the scheme is said to be **approximating**.

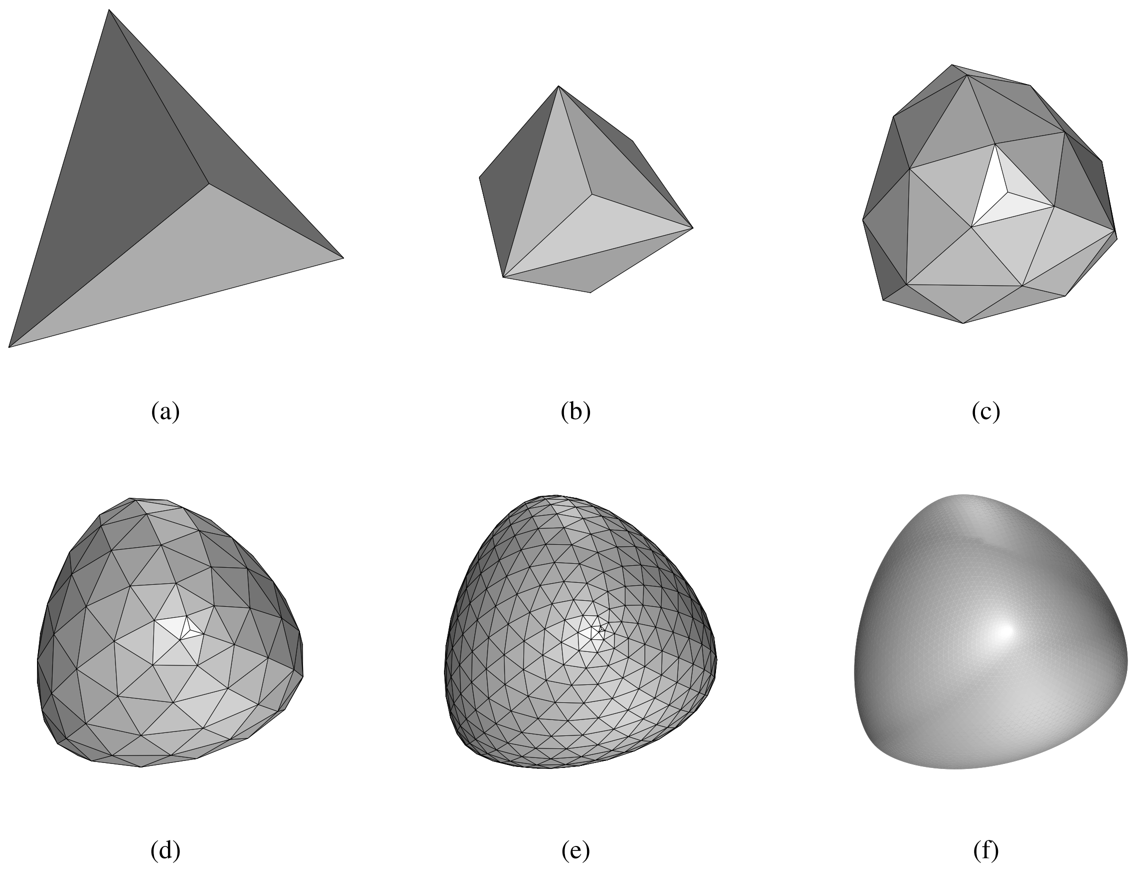


Figure 9.1: Subdivision example. (a) Control mesh (a tetrahedron). The refined mesh after (b) one, (c) two, (d) three, and (e) four levels of subdivision. (f) The limit surface.

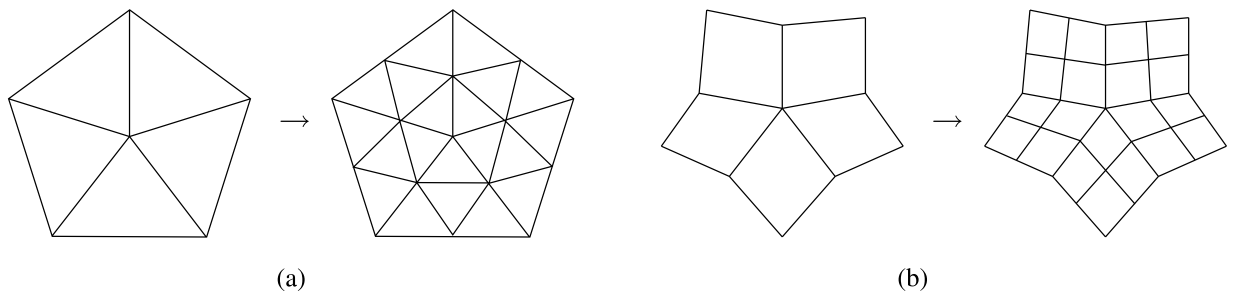


Figure 9.2: Examples of topologic refinement rules. (a) Primal triangle quadrisection and (b) primal quadrilateral quadrisection.

The (topologic and geometric) refinement rules employed in subdivision are chosen in order to obtain refined meshes with certain desirable properties. Perhaps, most importantly, the refinement rules are chosen to ensure that subdivision will always converge to a limit surface. Without a guarantee of convergence, the subdivision process would likely produce very poorly-behaved meshes. Furthermore, in many applications, smooth meshes are highly desirable. Consequently, the refinement rules are usually chosen to ensure that the limit surface be as smooth as possible. For example, in many computer-graphics applications, at least C^2 continuity is desirable.

9.2.1 Topologic Refinement Rules

The topologic refinement rule specifies how the connectivity of the mesh is to be modified in order to incorporate new vertices into the mesh. Such a rule, however, does not say anything about the coordinates of vertices, as this constitutes geometric information. Generally, there are two types of topologic rules: primal and dual. A primal scheme splits faces. A dual scheme splits vertices.

Since a topologic rule in subdivision introduces new vertices such that they are connected in a regular (i.e., highly structured) fashion, new vertices will always have particular valences. New vertices introduced in the interior of the mesh will all have the same valence. In the case of a triangle mesh, new vertices introduced in the interior have a valence of six. In the case of a quadrilateral mesh, new vertices introduced in the interior have a valence of four. A vertex with this special valence value is said to be **regular**. A vertex that is not regular is said to be **extraordinary**.

Some examples of topologic refinement rules are illustrated in Figure 9.2.

9.2.2 Geometric Refinement Rules

The geometric refinement rule specifies how the geometry of the mesh (i.e., the position of vertices) is to be changed in order to accommodate the new vertices being added. Clearly, the rule must specify how to determine the position of any new vertices introduced by the topologic refinement rule. In addition, the rule must also specify how to handle the old (i.e., previously existing) vertices, which may be either modified or left unchanged. If the rule leaves the old vertex positions unchanged, this leads to an interpolating scheme. The geometric refinement rule can be viewed as a filtering operation for meshes. The rule is specified in terms of masks, which are often expressed pictorially. Some examples of geometric refinement masks are illustrated in Figure 9.3.

In a geometric refinement rule, the mask coefficients normally sum to one in order to correspond to affine combinations of points. By taking affine combinations of points, we ensure that the subdivision process is affine invariant (i.e., independent of the coordinate system employed). Due to affine invariance, applying a translation/rotation/scaling to a control mesh will result in an identical translation/rotation/scaling of the refined meshes and limit surface produced by subdivision. In other words, subdivision and affine transformations commute.

Normally, in the case of meshes with a boundary, the masks used to determine the positions of vertices on the boundary only depend on boundary vertices. This ensures that the boundary vertices of a refined mesh depend only on the vertices on the boundary of the control mesh (and not vertices in the interior of the control mesh). As a consequence of this, two control meshes that align on their boundaries will also produce refined meshes that align along their corresponding boundaries.

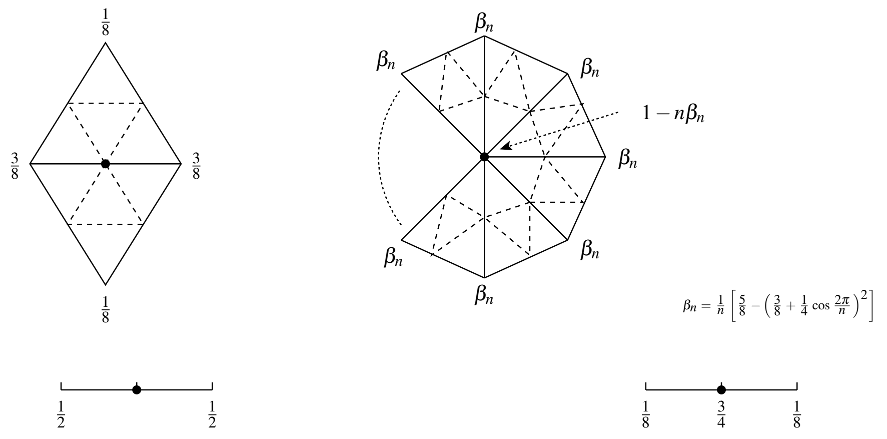


Figure 9.3: Examples of geometric refinement rules.

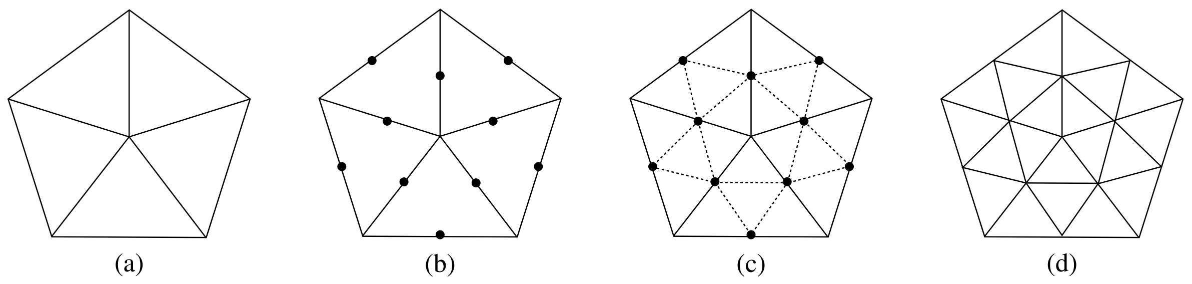


Figure 9.4: Primal triangle quadrisection. (a) The topology of the original mesh. (b) The topology of the mesh after edge splitting and new vertex insertion. (c) The topology of the mesh after the addition of new edges. (d) The topology of the refined mesh.

9.2.3 Primal Triangle Quadrisection

For triangle meshes, one of the most common topologic refinement rules is primal triangle quadrisection. To update the topology of a mesh, primal triangle quadrisection works as follows. Each edge in the original mesh is split in two, with a new vertex being inserted at the location of the split. Each of these newly added vertices is referred to as an **edge vertex**. Each new vertex is then connected by an edge to each of the other new vertices that originated from the same face (before edge splitting). The above process is illustrated in Figure 9.4.

With this topologic refinement rule, each triangle from the original mesh is replaced by four new triangles. Observe that all new vertices added in the interior of the mesh have valence six, while all new vertices added on the boundary have valence four. Note that no mention is made as to what the coordinates of the new vertices are. They are essentially undefined. A geometric refinement rule is needed to specify how to assign coordinates to the new vertices.

9.2.4 $\sqrt{3}$ Topologic Refinement

For triangle meshes, a more exotic topologic refinement rule is $\sqrt{3}$ refinement. This topologic refinement scheme employs two slightly different rules, where one is used in even iterations and one in odd iterations. These two cases only differ in how they handle boundaries. So, in the case of a mesh without a boundary, there is no difference in the processing of even and odd iterations.

First, we consider even iterations (where iterations are numbered starting from zero). In the interior of each face, a new vertex is added. Then, each new vertex is connected by edges to each of its three surrounding old vertices. Lastly, every original edge that connects two old vertices is flipped, except for boundary edges (which cannot be flipped). This process is illustrated pictorially in Figure 9.5.

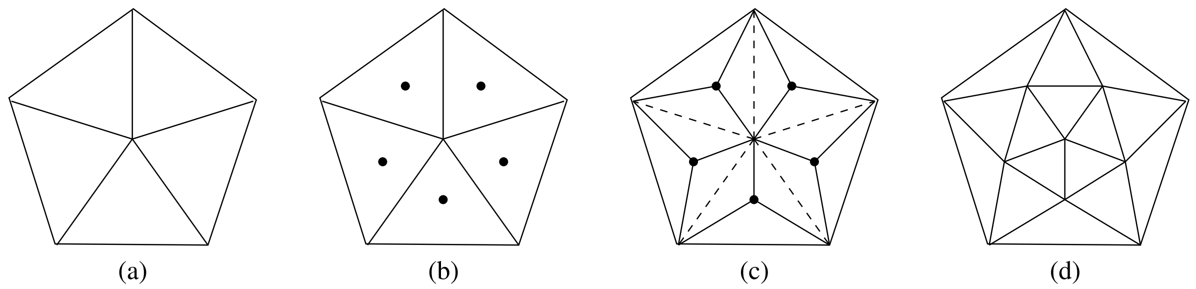


Figure 9.5: $\sqrt{3}$ topologic refinement rule (even iteration). (a) The initial mesh. (b) The mesh after adding a new vertex to each face. (c) The mesh prior to edge flipping. (d) The mesh after topologic refinement.

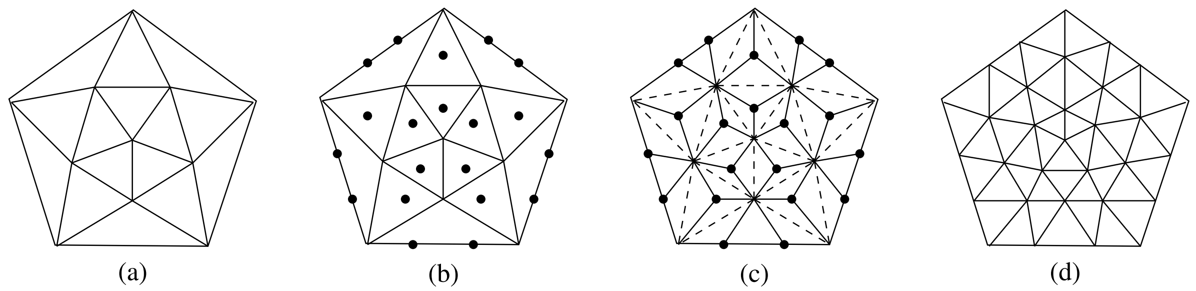


Figure 9.6: $\sqrt{3}$ topologic refinement rule (odd iteration). (a) The initial mesh. (b) The mesh after adding new vertices. (c) The mesh prior to edge flipping. (d) The mesh after topologic refinement.

In the case of odd iterations, the process is identical to above, except for how boundary edges are handled at the beginning of the refinement step. Due to the manner in which even iterations are handled, in an odd iteration, a triangle can have at most one edge on the boundary. In odd iterations, for each triangle with an edge on the boundary, the boundary edge is split twice inserting two new vertices along the edge. Then, the two new vertices are connected to the old vertex on the side of the triangle opposite the edge on which the two new vertices were inserted. The remainder of the refinement process is carried out identically to the even iteration case. This process is illustrated pictorially in Figure 9.6.

Every two iterations of the topologic refinement process results in each original triangle being replaced by nine new triangles. All new vertices in the interior that are not incident on a boundary face have valence six. Applying topologic refinement twice results in a uniform refinement with trisection of every original edge. A single iteration can be viewed as the "square root" of a three-fold split of each edge, hence the name $\sqrt{3}$.

9.2.5 Subdivision Schemes

Many subdivision schemes have been proposed to date. Such schemes can be classified using a variety of criteria:

1. the type of mesh that can be handled by the scheme (e.g., triangle, quadrilateral, triangle/quadrilateral, hexagonal);
2. whether the scheme is approximating or interpolating;
3. whether the scheme is primal (i.e., based on face splitting) or dual (i.e., based on vertex splitting);
4. the smoothness of the limit surface produced by the scheme (e.g., C^0, C^1, or C^2 continuity).

Several subdivision schemes and their attributes are listed in Table 9.1.

Table 9.1: Several subdivision schemes

Scheme	Attributes
Linear [14]	triangle; interpolating; C^0
Bilinear [14]	quadrilateral; interpolating; C^0
Midedge [5, 11]	quadrilateral; approximating; dual; C^1
Doo-Sabin [3]	quadrilateral; approximating; dual; C^1
Catmull-Clark [1]	polygon/quadrilateral; approximating; primal; C^2 everywhere except at extraordinary points where C^1
Loop [8]	triangle; approximating; primal; C^2 everywhere except at extraordinary points where C^1
Butterfly [4]	triangle; interpolating; primal; C^1 continuous everywhere except at extraordinary points
Modified butterfly [15]	triangle; interpolating; primal; C^1 continuous everywhere
Kobbelt $\sqrt{3}$ [7]	triangle; approximating; primal; C^2 everywhere except at extraordinary points where C^1
Stam-Loop [12]	quadrilateral/triangle; approximating; primal

9.2.6 Linear Subdivision

One of the simplest subdivision schemes is linear subdivision. Linear subdivision is defined for triangle meshes. It is primal and interpolating. With this scheme, the limit surface is always identical to the original control mesh. Therefore, this surface is only guaranteed to be C^0 continuous.

The topologic refinement rule employed is primal triangle quadrisection, described earlier in Section 9.2.3. The geometric rule places each new edge vertex at the midpoint of the edge (in the unrefined mesh) from which the new vertex was generated. Let v denote a new edge vertex associated with an edge in the unrefined mesh with vertices v_1 and v_2. Then, we choose $v = \frac{1}{2}(v_1 + v_2)$.

This scheme is not particularly interesting, as the limit surface is simply equal to the original control mesh. Consequently, this subdivision scheme is of somewhat limited practical value. Nevertheless, the scheme is useful for illustrating the basics of subdivision.

Example 9.1. Consider the control mesh shown in Figure 9.7(a). This mesh corresponds to the surface (with boundary) formed by removing the bottom face of a tetrahedron. Applying one level of linear subdivision to the control mesh, we obtain the refined mesh in Figure 9.7(b).

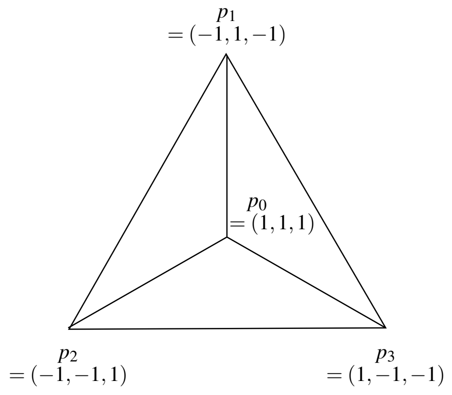
Example 9.2. Consider the control mesh, shown in Figure 9.8(a), which corresponds to the surface of a tetrahedron. Applying linear subdivision to this control mesh, we obtain the limit surface shown in Figure 9.8(b).

9.2.7 Loop Subdivision

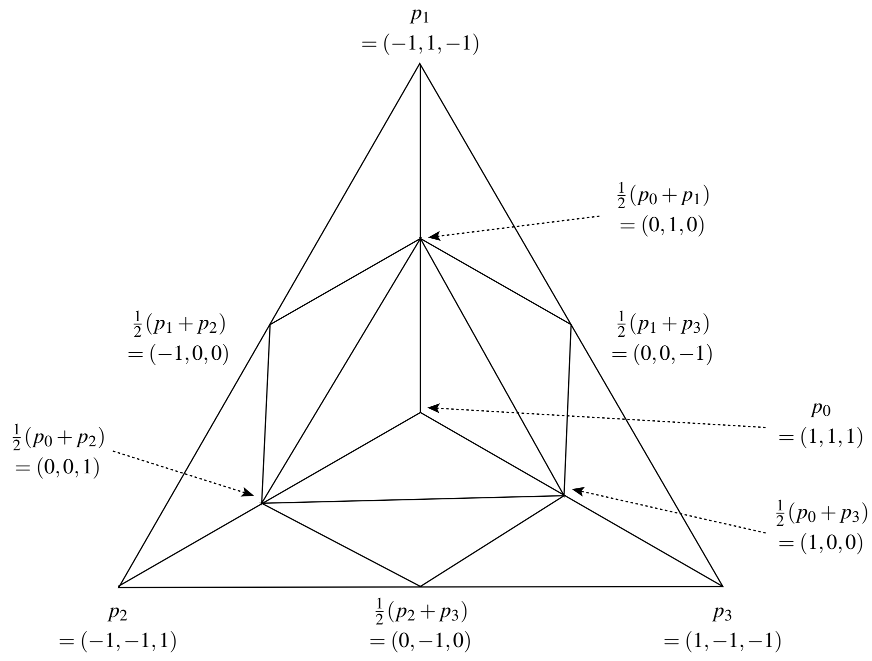
One quite popular subdivision scheme is Loop subdivision, originally proposed in [8]. This scheme is defined for triangle meshes. It is approximating and primal. This scheme produces limit surfaces that are C^2 continuous everywhere, except at extraordinary vertices where C^1 continuity is achieved.

In Loop subdivision, the topologic refinement rule employed is primal triangle quadrisection, described earlier in Section 9.2.3. After topologic refinement, there are two types of vertices in the mesh: non-edge (i.e., old) and edge (i.e., new) vertices. To fully specify the geometric refinement rule, we must consider how to handle both non-edge and edge vertices. Furthermore, since vertices are treated differently depending on whether they fall on the mesh boundary, we have the following cases to consider: 1) an interior edge vertex, 2) an interior non-edge vertex, 3) a boundary edge vertex, and 4) a boundary non-edge vertex. In what follows, we consider each of these cases in turn.

INTERIOR EDGE VERTEX. First, let us consider an edge vertex v that is not on the boundary of the mesh. Suppose that v was produced by splitting the edge e connecting vertices v_1 and v_2 in the unrefined mesh. The edge e (in the unrefined mesh) has two incident faces whose union forms a quadrilateral. Let v_3 and v_4 denote the two vertices of



(a)



(b)

Figure 9.7: Linear subdivision example. (a) Control mesh (a surface with boundary). (b) Refined mesh obtained after one round of subdivision.

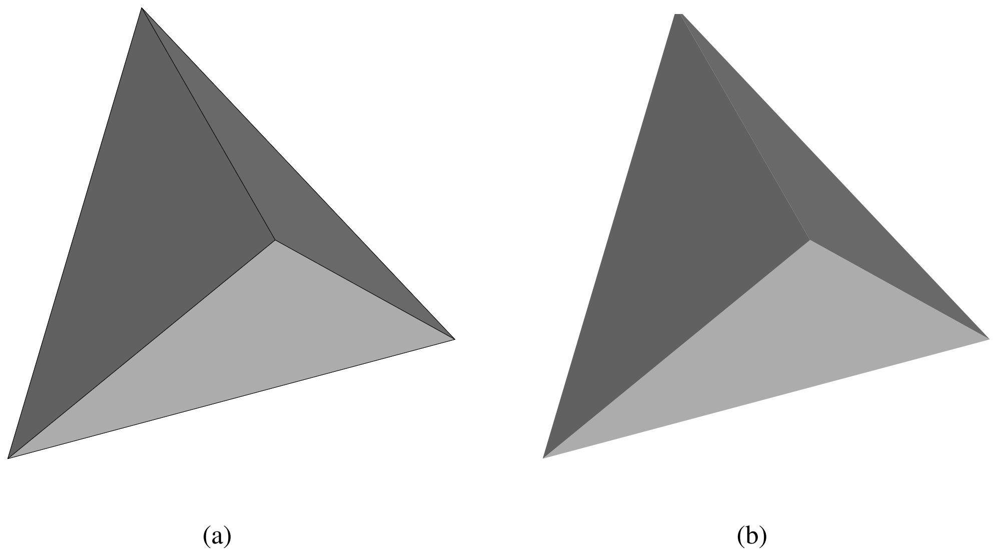


Figure 9.8: Linear subdivision example. (a) Control mesh (a tetrahedron). (b) Limit surface.

this quadrilateral other than v_1 and v_2. Then, v is chosen as

$$v = \tfrac{3}{8}(v_1 + v_2) + \tfrac{1}{8}(v_3 + v_4).$$

This corresponds to the mask shown in Figure 9.9(a).

INTERIOR NON-EDGE VERTEX. Next, let us consider a non-edge vertex v that is not on the boundary of the mesh. Let $\{v_k\}_{k=1}^{n}$ denote the n 1-ring neighbours of v in the unrefined mesh. Then, the updated value v' for the vertex v is given by

$$v' = (1 - n\beta_n)v + \beta_n \sum_{k=1}^{n} v_k,$$

where

$$\beta_n = \tfrac{1}{n}\left[\tfrac{5}{8} - \left(\tfrac{3}{8} + \tfrac{1}{4}\cos\tfrac{2\pi}{n}\right)^2\right]. \tag{9.1}$$

This corresponds to the mask shown in Figure 9.9(b).

BOUNDARY EDGE VERTEX. Next, let us consider an edge vertex v that is on the boundary of the mesh. Suppose that v was produced by splitting the edge connecting vertices v_1 and v_2 in the unrefined mesh. Then, v is chosen as

$$v = \tfrac{1}{2}(v_1 + v_2).$$

In other words, v is the midpoint of the edge joining v_1 and v_2. This calculation corresponds to the mask shown in Figure 9.9(c).

BOUNDARY NON-EDGE VERTEX. Next, let us consider a non-edge vertex that is on the boundary of the mesh. Let v_1 and v_2 denote the two neighbours of v along the boundary of the mesh prior to topologic refinement. The updated value v' for v is given by

$$v' = \tfrac{3}{4}v + \tfrac{1}{8}(v_1 + v_2).$$

This computation corresponds to the mask shown in Figure 9.9(d).

Example 9.3 (Loop subdivision)**.** Consider the control mesh given in Figure 9.10(a). Applying one level of Loop subdivision to this mesh, we obtain the refined mesh shown in Figure 9.10(b).

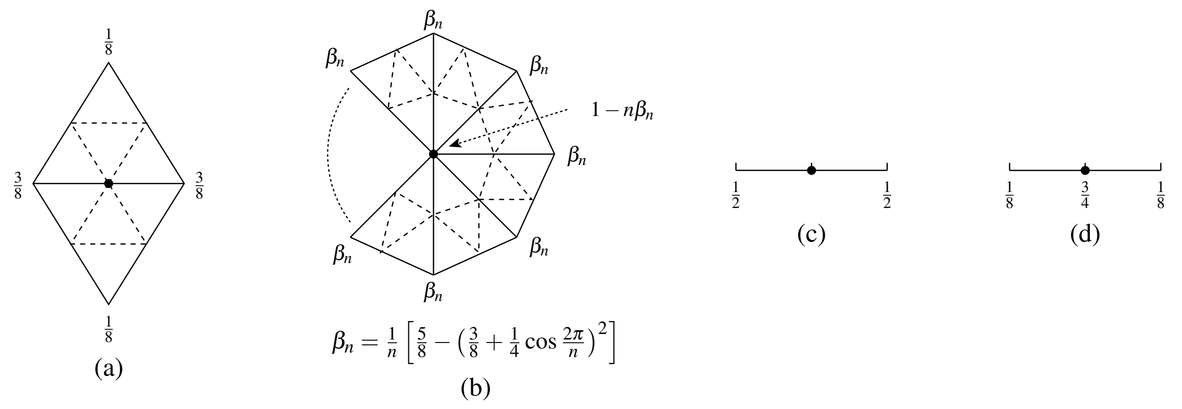


Figure 9.9: Geometric refinement masks for Loop subdivision. The masks for (a) interior edge vertices (regular), (b) interior non-edge vertices of valence n (regular and extraordinary), (c) boundary/crease edge vertices, and (d) boundary/crease non-edge vertices.

Table 9.2: Comparison of weights employed in the Loop and modified Loop schemes

n	β_n	β'_n
3	$\frac{3}{16} = 0.1875$	$\frac{3}{16} = 0.1875$
4	$\frac{31}{256} = 0.12109375$	$\frac{3}{32} = 0.09375$
5	$\approx 0.0840932189257829$	$\frac{3}{40} = 0.075$
6	$\frac{1}{16} = 0.0625$	$\frac{1}{16} = 0.0625$
7	$\approx 0.0490249201910889$	$\frac{3}{56} \approx 0.0535714285714286$
8	$\approx 0.0400678098159403$	$\frac{3}{64} = 0.046875$
9	$\approx 0.0337850179292333$	$\frac{1}{24} = 0.0416666666666667$
10	$\approx 0.0291777532480480$	$\frac{3}{80} = 0.0375$

Example 9.4. Consider a control mesh that corresponds to the boundary of a tetrahedron as shown in Figure 9.11(a). Applying Loop subdivision to this mesh, we obtain the refined meshes shown in Figures 9.11(b) to (e) and the limit surface shown in Figure 9.11(f).

It is worthwhile to note that numerous variations on Loop subdivision have been proposed. For example, sometimes we may want to introduce creases in the limit surface resulting from subdivision. In this case, we treat the location of a crease as a boundary and apply the appropriate boundary rule.

A modified version of Loop subdivision proposed by Warren and Weimer [14, p. 230] chooses β_n in (9.1) as $\beta_n = \beta'_n$, where

$$\beta'_n = \begin{cases} \frac{3}{8n} & n > 3 \\ \frac{3}{16} & n = 3. \end{cases}$$

The resulting coefficients have “nicer” values than those in the original Loop scheme (which has potential computational advantages). A comparison of the weights employed in the Loop and modified Loop schemes is provided in Table 9.2.

9.2.8 Butterfly Subdivision

Another popular subdivision scheme is butterfly subdivision, originally proposed by Dyn et al. [4]. This subdivision scheme is defined for triangle meshes. It is interpolating and primal. The limit surfaces produced by this scheme are C^1 continuous everywhere, except at extraordinary vertices of valence k where $k = 3$ or $k > 7$. At these points, the surface is only C^0 continuous.

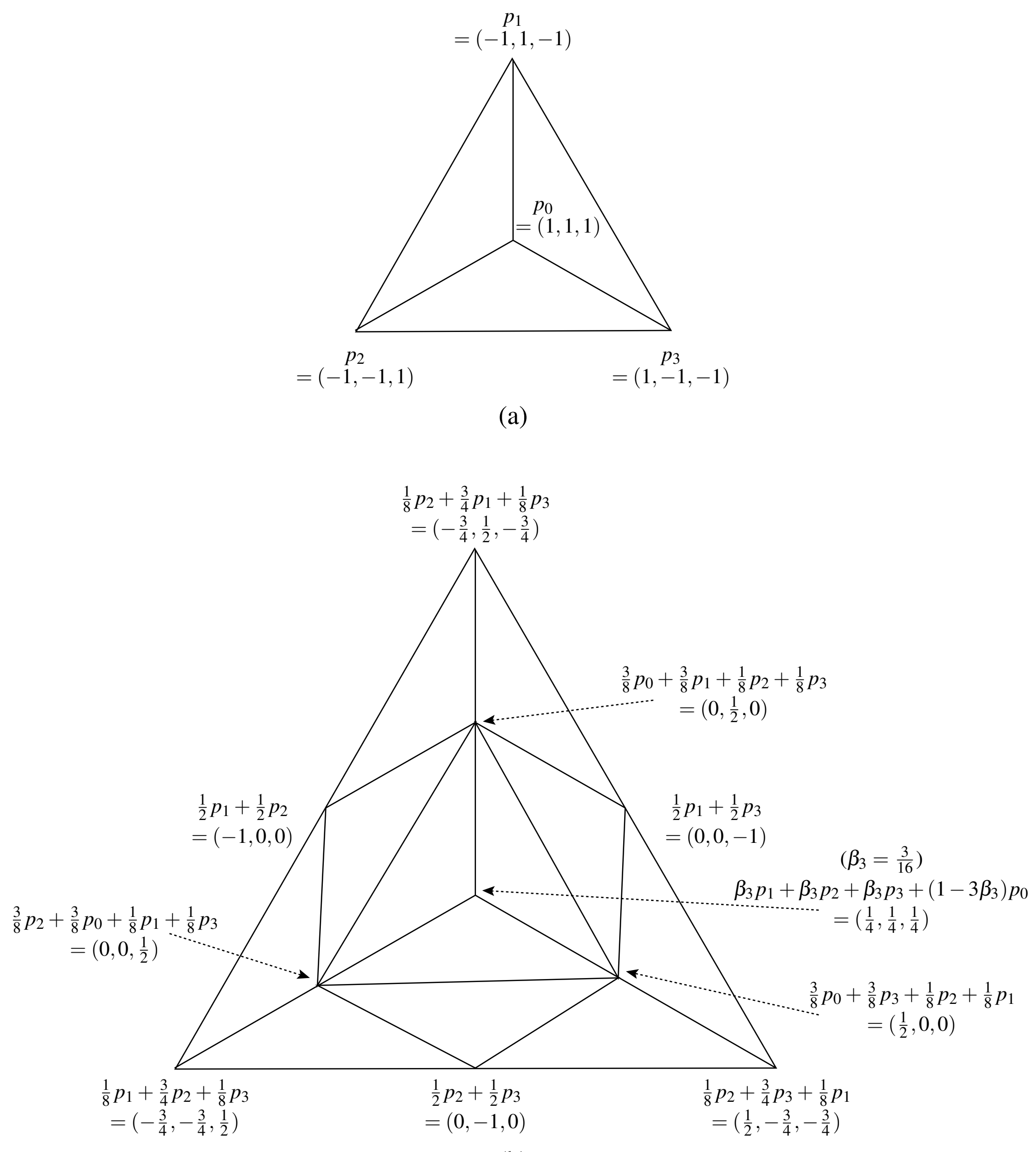

p_1
$= (-1, 1, -1)$
p_0
$= (1, 1, 1)$
p_2
$= (-1, -1, 1)$
p_3
$= (1, -1, -1)$
(a)
$\frac{1}{8}p_2 + \frac{3}{4}p_1 + \frac{1}{8}p_3$
$= (-\frac{3}{4}, \frac{1}{2}, -\frac{3}{4})$
$\frac{3}{8}p_0 + \frac{3}{8}p_1 + \frac{1}{8}p_2 + \frac{1}{8}p_3$
$= (0, \frac{1}{2}, 0)$
$\frac{1}{2}p_1 + \frac{1}{2}p_2$
$= (-1, 0, 0)$
$\frac{1}{2}p_1 + \frac{1}{2}p_3$
$= (0, 0, -1)$
$(\beta_3 = \frac{3}{16})$
$\beta_3 p_1 + \beta_3 p_2 + \beta_3 p_3 + (1 - 3\beta_3)p_0$
$= (\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$
$\frac{3}{8}p_2 + \frac{3}{8}p_0 + \frac{1}{8}p_1 + \frac{1}{8}p_3$
$= (0, 0, \frac{1}{2})$
$\frac{3}{8}p_0 + \frac{3}{8}p_3 + \frac{1}{8}p_2 + \frac{1}{8}p_1$
$= (\frac{1}{2}, 0, 0)$
$\frac{1}{8}p_1 + \frac{3}{4}p_2 + \frac{1}{8}p_3$
$= (-\frac{3}{4}, -\frac{3}{4}, \frac{1}{2})$
$\frac{1}{2}p_2 + \frac{1}{2}p_3$
$= (0, -1, 0)$
$\frac{1}{8}p_2 + \frac{3}{4}p_3 + \frac{1}{8}p_1$
$= (\frac{1}{2}, -\frac{3}{4}, -\frac{3}{4})$
(b)

Figure 9.10: Loop subdivision example. (a) Control mesh (a surface with boundary). (b) Refined mesh obtained after one round of subdivision.

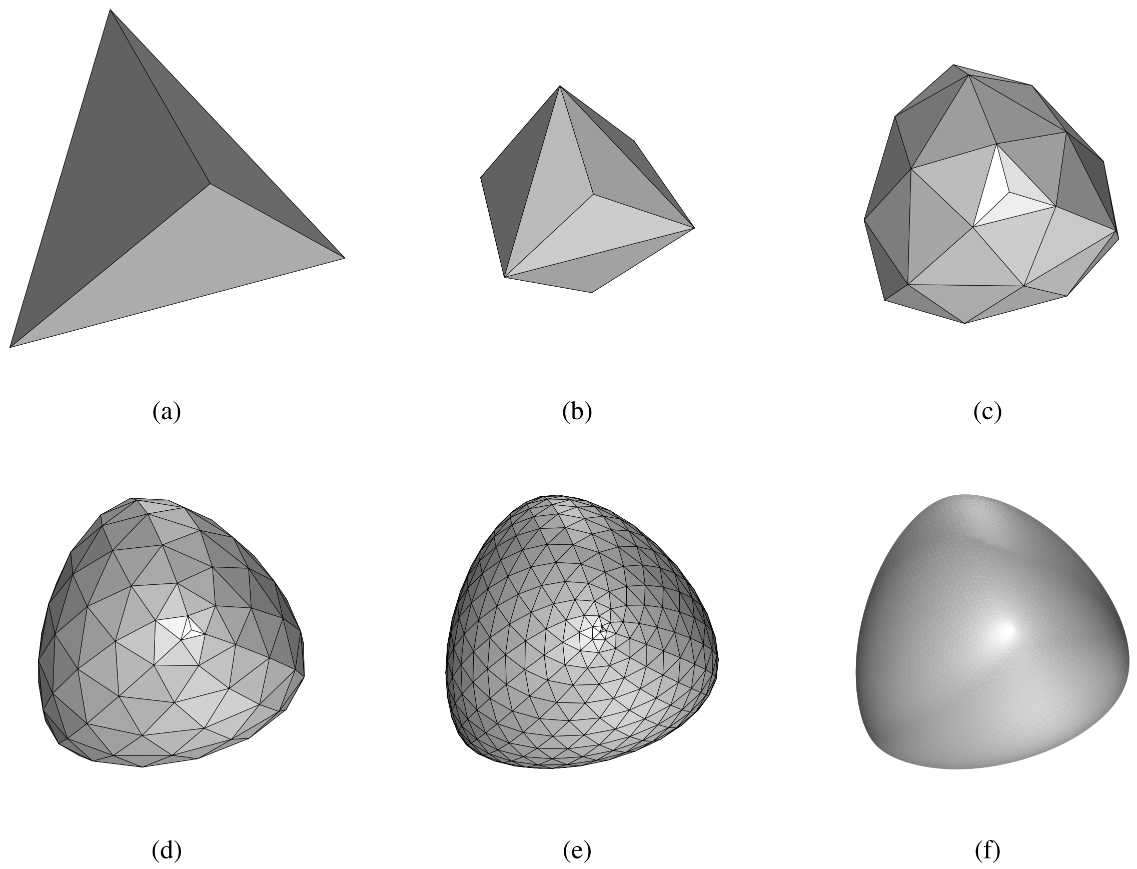


Figure 9.11: Loop subdivision example. (a) Control mesh (a tetrahedron). The refined mesh after (b) one, (c) two, (d) three, and (e) four levels of subdivision. (f) The limit surface.

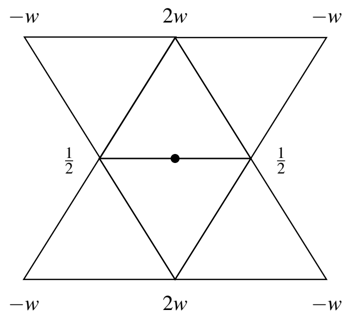


Figure 9.12: Geometric refinement mask for butterfly subdivision (for regular edge vertices).

The topologic refinement rule employed is primal triangle quadrisection, described earlier in Section 9.2.3. A tension parameter allows for local control over smoothness. For example, it can be used to form creases in the limit surface. The geometric refinement mask is shown in Figure 9.12 (for regular edge vertices). The boundary cases are somewhat more complicated and ignored here. The tension parameter w is nominally chosen as $w = \frac{1}{16}$.

Example 9.5. Consider a control mesh that corresponds to the surface of a tetrahedron as shown in Figure 9.13(a). Applying butterfly subdivision to this mesh, we obtain the refined meshes shown in Figures 9.13(b) to (e) and the limit surface shown in Figure 9.13(f).

A variant of the butterfly subdivision scheme called the modified butterfly scheme has been proposed in [15]. The limit surfaces produced are C^1 continuous everywhere (including all extraordinary points).

9.2.9 Kobbelt $\sqrt{3}$ Subdivision

Yet another subdivision scheme is the Kobbelt $\sqrt{3}$ scheme, originally proposed in [7]. The scheme is defined for triangle meshes. It is primal and approximating. The limit surfaces produced by this method are C^2 continuous everywhere, except at extraordinary points where C^1 continuity is achieved.

The topologic refinement rule employed is the $\sqrt{3}$ rule, described earlier in Section 9.2.4. After topologic refinement, there are two types of vertices in the mesh: new and old vertices. To fully specify the geometric refinement rule, we must consider how to handle both new and old vertices. Furthermore, since vertices are treated differently depending on whether they fall on the mesh boundary, we have the following cases to consider: 1) a new interior vertex, 2) an old interior vertex, 3) a new boundary vertex, and 4) an old boundary vertex. In what follows, we consider each of these cases in turn.

NEW INTERIOR VERTEX. First, let us consider the case of a new interior vertex. Let p be the new vertex inserted into the face of the unrefined mesh having vertices p_1, p_2, p_3. Then, we choose

$$p = \tfrac{1}{3}(p_1 + p_2 + p_3)$$

(i.e., p is inserted at the barycenter of its associated face). The geometric refinement rule for this case is illustrated in Figure 9.14(a).

OLD INTERIOR VERTEX. Next, let us consider the case of an old interior vertex. Let p be the old vertex to be updated, and let $p_1, p_2, \ldots, p_n$ be the 1-ring neighbours of p in the unrefined mesh. Then, we choose the updated value p' of the vertex p as given by

$$p' = (1 - n\beta_n)p + \beta_n \sum_{k=1}^{n} p_k,$$

where

$$\beta_n = \tfrac{1}{9n}\left[4 - 2\cos\left(\tfrac{2\pi}{n}\right)\right].$$

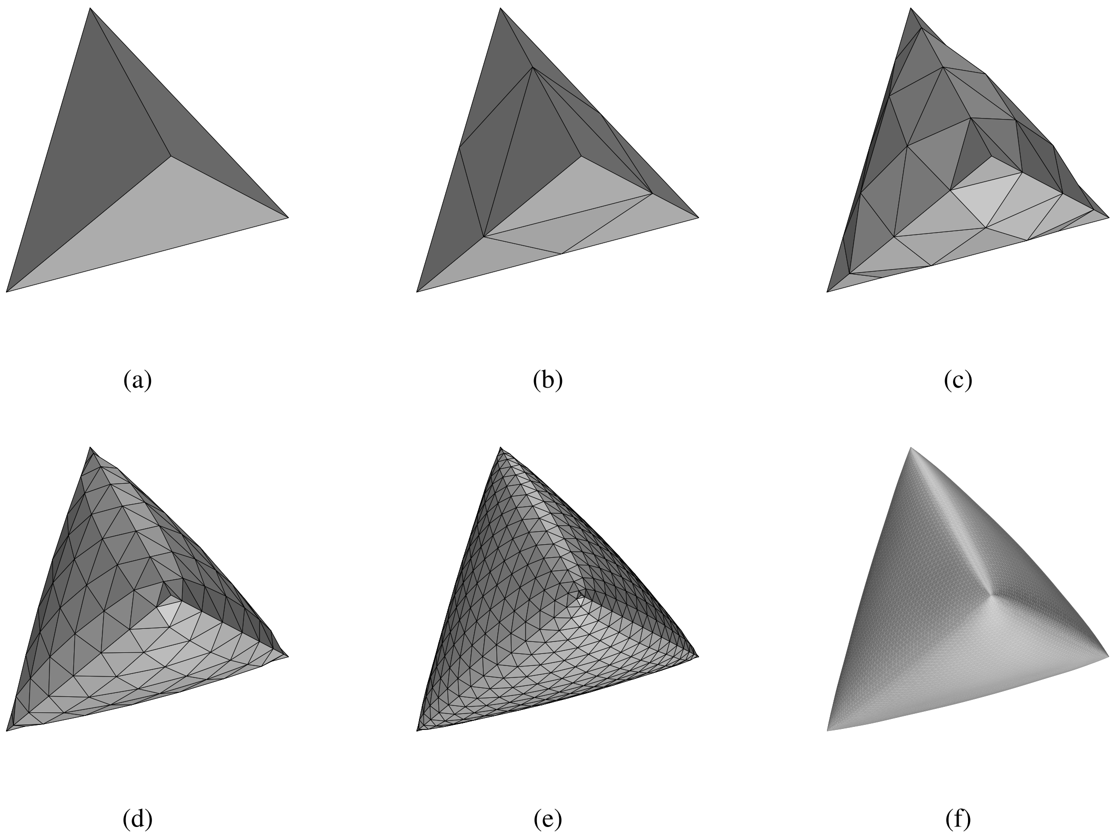


Figure 9.13: Butterfly subdivision example. (a) Control mesh (a tetrahedron). The refined mesh obtained after (b) one, (c) two, (d) three, and (e) four levels of subdivision. (f) The limit surface.

Table 9.3: The quantity β_n (in Kobbelt $\sqrt{3}$ subdivision) for several values of n

n	β_n
3	$\frac{5}{27} \approx 0.185185185185185$
4	$\frac{1}{9} \approx 0.111111111111111$
5	$\approx 0.0751548002500023$
6	$\frac{3}{54} \approx 0.0555555555555556$
7	$\approx 0.0436987364489291$
8	$\frac{4-\sqrt{2}}{72} \approx 0.0359137005225959$
9	$\approx 0.0304680384415067$
10	$\approx 0.0264662890138901$

The values for β_n for several values of n are listed in Table 9.3. The geometric refinement rule for this case is illustrated in Figure 9.14(b).

NEW BOUNDARY VERTEX. Next, let us consider the case of a new boundary vertex (in odd iterations). Let p be the new boundary vertex whose position is to be determined. Let e be the edge in the unrefined mesh into which p has been inserted. Let p_1 denote the old vertex that is a neighbour of p along the boundary in the refined mesh. Let p_2 and p_3 denote the neighbours of p_1 in the unrefined mesh, with p_2 being the neighbour that is an endpoint of e. Then, we choose p as given by

$$p = \tfrac{16}{27}p_1 + \tfrac{10}{27}p_2 + \tfrac{1}{27}p_3.$$

This geometric refinement rule corresponds to the masks shown in Figures 9.14(d) and (e). The two masks shown in these figures are essentially the same, as one is simply the reflection (i.e., mirror image) of the other.

OLD BOUNDARY VERTEX. Lastly, let us consider the case of an old boundary vertex (in odd iterations). Let p denote the old boundary vertex whose position is to be updated. Let p_1 and p_2 denote the two neighbours of p along the boundary in the unrefined mesh. Then, we choose the updated value p' of the vertex p as given by

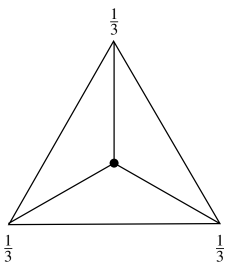
$$p' = \tfrac{19}{27}p + \tfrac{4}{27}p_1 + \tfrac{4}{27}p_2.$$

This geometric refinement rule is shown in Figure 9.14(c).

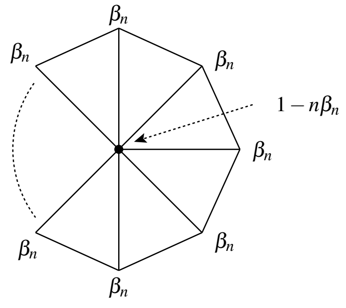
Example 9.6. Consider the control mesh shown in Figure 9.15(a). This mesh corresponds to the surface (with boundary) formed by removing the bottom face of a tetrahedron. In this example, we apply two iterations of Kobbelt $\sqrt{3}$ subdivision to this control mesh.

The topologic refinement process for the first iteration of subdivision (i.e., iteration 0) is illustrated in Figures 9.15(b) and 9.15(c). In going from Figure 9.15(a) (which shows the initial control mesh) to Figure 9.15(b), a vertex is added to each (interior and boundary) face and each new vertex is connected to the three vertices of the face in which the new vertex was inserted. (In this particular circumstance, there are no interior faces since all three faces are boundary faces. So, only boundary faces need be considered here.) In Figure 9.15(b), the edges that still remain to be flipped, in order to complete the topologic refinement process, are shown as dashed lines. In going from Figure 9.15(b) to 9.15(c) each of the dashed-line edges has been flipped. Thus, Figure 9.15(c) shows the connectivity of the mesh after the topologic refinement process completes in the first subdivision iteration (i.e., iteration 0). Figure 9.15(d) then shows the results of the geometric refinement process (i.e., the initialization of the new vertex positions and updating of the old vertex positions). Thus, the result of the first full iteration of subdivision is the mesh shown in Figure 9.15(d).

At the start of the second iteration of subdivision (i.e., iteration 1), we have the mesh shown in Figure 9.16(a), which is the same mesh as shown in Figure 9.15(d) but with the vertices relabelled for convenience so as to avoid an overly messy diagram (e.g., $p'_2 = p_2$, $p'_5 = \frac{1}{3}(p_0 + p_2 + p_3)$, and so on). The topologic refinement process in the second iteration of subdivision (i.e., iteration 1) is depicted in Figures 9.16(b) and 9.16(c). In going from Figure 9.16(a) to 9.16(b), two things happen. First, we insert a new vertex into each interior face, and then connect each new vertex to the three vertices of the face into which the new vertex was inserted. Second, for each boundary face, we split the

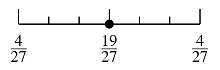


(a)

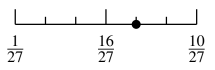


$\beta_n = \frac{1}{9n}\left[4 - 2\cos\left(\frac{2\pi}{n}\right)\right]$

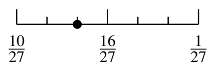
(b)



(c)



(d)



(e)

Figure 9.14: The geometric refinement mask for Kobbelt $\sqrt{3}$ subdivision. The case of a (a) new interior vertex and (b) old interior vertex. (c) The mask for updating an old boundary vertex in odd iterations. (d) and (e) The masks for a new boundary vertex in the odd iterations.

boundary edge of the face twice, inserting a new vertex at each splitting point (resulting in two new vertices being added to the edge) and then the two new vertices are connected to the opposing vertex within the face. This results in the trisection of each boundary triangle. In Figure 9.16(b), the old edges that will need to be flipped in order to complete the topologic refinement process are shown as dashed lines. In going from Figure 9.16(b) to 9.16(c), each dashed-line edge is flipped. Thus, Figure 9.16(c) shows the connectivity of the mesh after the topologic refinement process completes in the second iteration of subdivision (i.e., iteration 1). Notice that each face from the original control mesh from Figure 9.15(a) has been replaced by nine new triangles in the new mesh in Figure 9.16(c). Finally, Figure 9.16(d) shows the results of the geometric refinement process (i.e., the initialization of the new vertex positions and updating of the old positions). Thus, the result of the second full subdivision iteration is the mesh shown in Figure 9.16(d).

Example 9.7. Consider the control mesh corresponding to the surface of a tetrahedron, shown in Figure 9.17(a). Applying Kobbelt $\sqrt{3}$ subdivision to this mesh, we obtain the refined meshes shown in Figures 9.17(b) to (e) and the limit surface shown in Figure 9.17(f).

Kobbelt $\sqrt{3}$ subdivision has some potential advantages in relation to other schemes, with these benefits coming largely from its use of $\sqrt{3}$ topologic refinement. Recall that $\sqrt{3}$ refinement increases the number of faces in the mesh by a smaller factor than primal triangle quadrisection. In particular, every two iterations of $\sqrt{3}$ refinement increase the number of faces by factor of nine, whereas in the case of primal triangle quadrisection this factor is sixteen. Thus, the number of faces increases more slowly with Kobbelt $\sqrt{3}$ subdivision than with schemes employing primal triangle quadrisection (such as Loop and butterfly subdivision). Having finer control over the growth in face count can be beneficial in many situations. Lastly, Kobbelt $\sqrt{3}$ subdivision is better suited to adaptive refinement strategies, since it more easily allows for local refinement without cracks.

9.2.10 Catmull-Clark Subdivision

Perhaps, one of the best known subdivision schemes is Catmull-Clark subdivision, originally proposed in [1]. Although Catmull-Clark subdivision was historically one of the first subdivision methods proposed, it is still used frequently in practice today, especially in computer animation. This method is a generalization of bicubic B-splines to arbitrary meshes. That is, in the case of a regular quadrilateral mesh, Catmull-Clark subdivision yields a bicubic B-spline surface. This scheme can be applied to polygon meshes with any type of faces (e.g., triangle, quadrilateral,

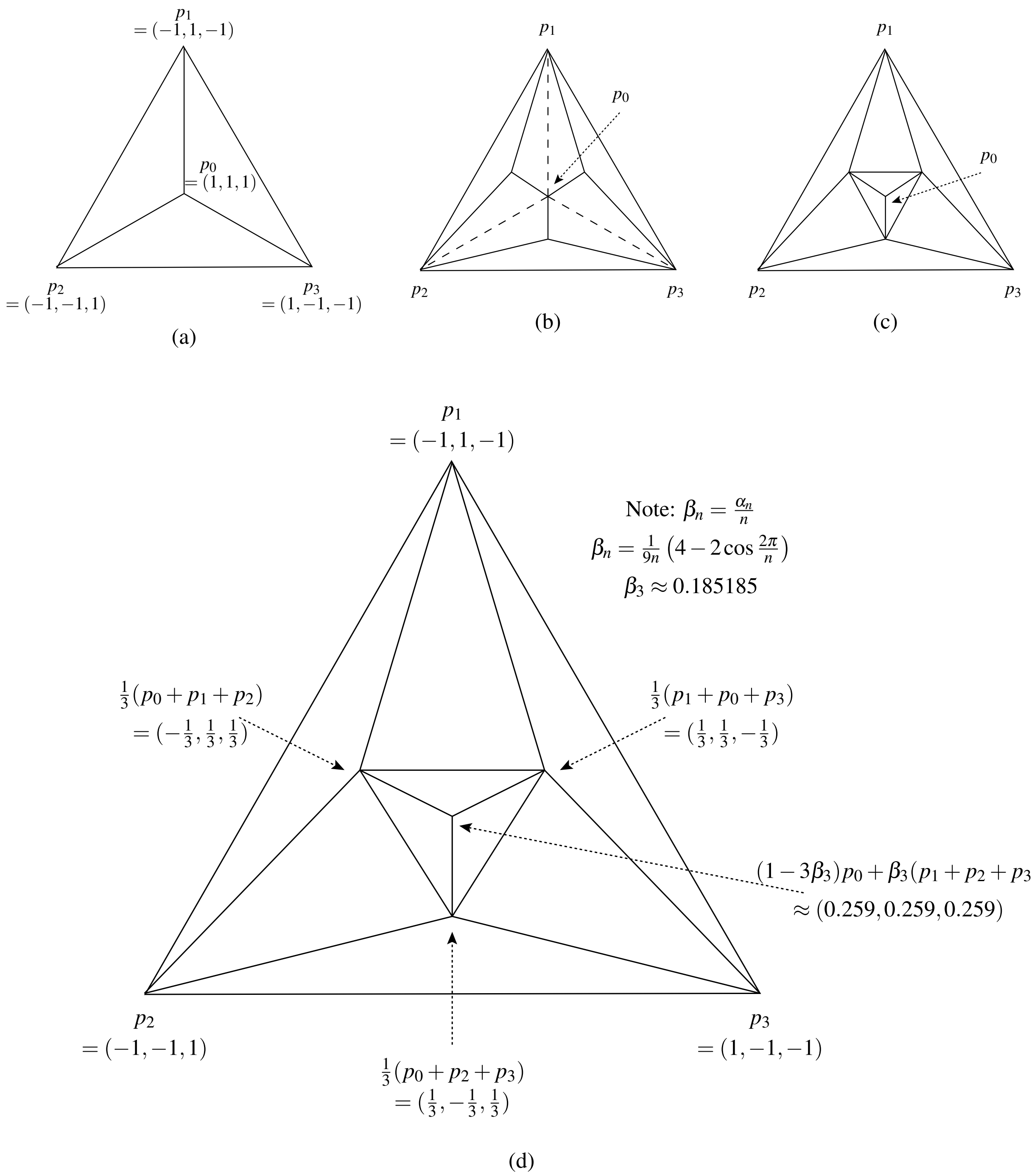

$p_1 = (-1,1,-1)$
$p_0 = (1,1,1)$
$p_2 = (-1,-1,1)$
$p_3 = (1,-1,-1)$
(a)
p_1
p_0
p_2
p_3
(b)
p_1
p_0
p_2
p_3
(c)
$p_1 = (-1,1,-1)$
Note: $\beta_n = \frac{\alpha_n}{n}$
$\beta_n = \frac{1}{9n}\left(4-2\cos\frac{2\pi}{n}\right)$
$\beta_3 \approx 0.185185$
$\frac{1}{3}(p_0+p_1+p_2) = (-\frac{1}{3},\frac{1}{3},\frac{1}{3})$
$\frac{1}{3}(p_1+p_0+p_3) = (\frac{1}{3},\frac{1}{3},-\frac{1}{3})$
$(1-3\beta_3)p_0+\beta_3(p_1+p_2+p_3) \approx (0.259,0.259,0.259)$
$p_2 = (-1,-1,1)$
$p_3 = (1,-1,-1)$
$\frac{1}{3}(p_0+p_2+p_3) = (\frac{1}{3},-\frac{1}{3},\frac{1}{3})$
(d)

Figure 9.15: $\sqrt{3}$ subdivision example. (a) Control mesh (a surface with boundary). (b) Intermediate mesh after inserting new vertices and establishing their initial connectivity (i.e., before edge flipping). (c) Mesh after completion of topological refinement (i.e., after edge flipping). (d) Refined mesh after the first round of subdivision.

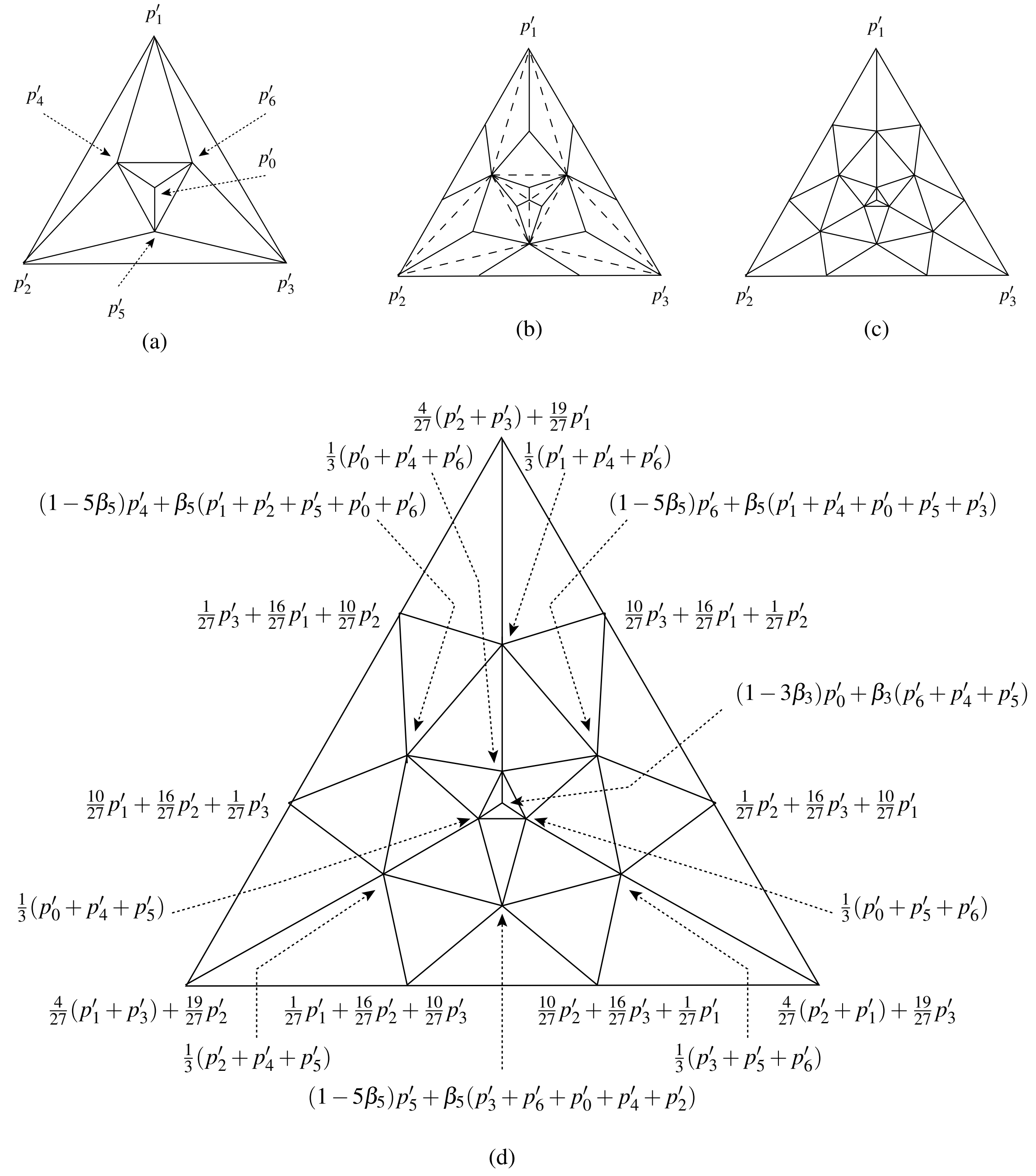

p'_1
p'_4
p'_6
p'_0
p'_2
p'_3
p'_5
(a)
p'_1
p'_2
p'_3
(b)
p'_1
p'_2
p'_3
(c)
$\frac{4}{27}(p'_2+p'_3)+\frac{19}{27}p'_1$
$\frac{1}{3}(p'_0+p'_4+p'_6)$
$\frac{1}{3}(p'_1+p'_4+p'_6)$
$(1-5\beta_5)p'_4+\beta_5(p'_1+p'_2+p'_5+p'_0+p'_6)$
$(1-5\beta_5)p'_6+\beta_5(p'_1+p'_4+p'_0+p'_5+p'_3)$
$\frac{1}{27}p'_3+\frac{16}{27}p'_1+\frac{10}{27}p'_2$
$\frac{10}{27}p'_3+\frac{16}{27}p'_1+\frac{1}{27}p'_2$
$(1-3\beta_3)p'_0+\beta_3(p'_6+p'_4+p'_5)$
$\frac{10}{27}p'_1+\frac{16}{27}p'_2+\frac{1}{27}p'_3$
$\frac{1}{27}p'_2+\frac{16}{27}p'_3+\frac{10}{27}p'_1$
$\frac{1}{3}(p'_0+p'_4+p'_5)$
$\frac{1}{3}(p'_0+p'_5+p'_6)$
$\frac{4}{27}(p'_1+p'_3)+\frac{19}{27}p'_2$
$\frac{1}{27}p'_1+\frac{16}{27}p'_2+\frac{10}{27}p'_3$
$\frac{10}{27}p'_2+\frac{16}{27}p'_3+\frac{1}{27}p'_1$
$\frac{4}{27}(p'_2+p'_1)+\frac{19}{27}p'_3$
$\frac{1}{3}(p'_2+p'_4+p'_5)$
$\frac{1}{3}(p'_3+p'_5+p'_6)$
$(1-5\beta_5)p'_5+\beta_5(p'_3+p'_6+p'_0+p'_4+p'_2)$
(d)

Figure 9.16: $\sqrt{3}$ subdivision example (continued). (a) Initial mesh for the second round of subdivision. (b) Intermediate mesh after inserting new vertices and establishing their initial connectivity (i.e., before edge flipping). (c) Mesh after completion of topological refinement (i.e., after edge flipping). (d) Final refined mesh after the second round of subdivision.

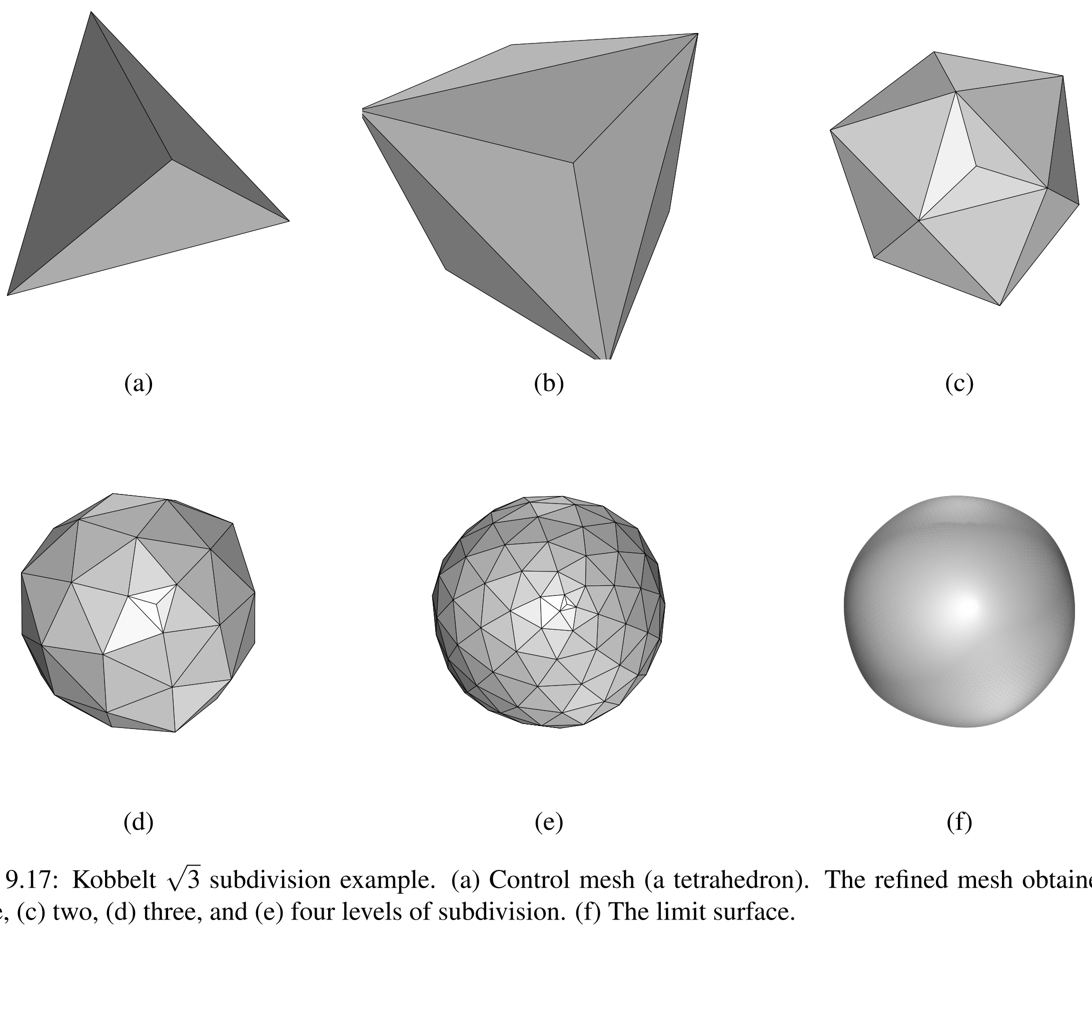


Figure 9.17: Kobbelt $\sqrt{3}$ subdivision example. (a) Control mesh (a tetrahedron). The refined mesh obtained after (b) one, (c) two, (d) three, and (e) four levels of subdivision. (f) The limit surface.

triangle/quadrilateral, hexagonal). The scheme is primal (i.e., face splitting) and approximating. The limit surfaces produced have C^2 continuity everywhere, except at extraordinary points where only C^1 continuity is achieved. The subdivision process always produces quadrilateral faces, regardless of the types of polygons in the control mesh.

The topologic refinement rule is defined as follows. A new vertex is inserted in each face. Such a vertex is called a **face point**. Each edge is split in two by inserting a new vertex on the edge. Such a vertex is called an **edge point**. Then, each face point is connected by an edge to each of the edge points associated with the same face. This topologic refinement process is illustrated for three different meshes in Figures 9.18, 9.19, and 9.20. This topologic refinement rule effectively splits an n-gon into n quadrilaterals. For this reason, subdivision always produces quadrilateral meshes. For a quadrilateral input mesh, this topologic refinement rule is simply equivalent to primal quadrilateral quadrisection. The first iteration of subdivision may introduce new extraordinary vertices. All subsequent iterations, however, introduce only regular (i.e., valence four) vertices in the interior of the mesh.

As we have just seen above, topologic refinement introduces two types of new vertices, namely, face points and edge points. To fully specify the geometric refinement rule, we must consider how to handle both types of new vertices (i.e., face points and edge points) as well as old vertices. Since vertices are treated differently depending on whether they fall on the mesh boundary, we have the following cases to consider: 1) a face point (which, by definition, cannot be on the boundary), 2) an interior edge point, 3) a boundary edge point, 4) an old interior vertex, and 5) an old boundary vertex. In what follows, we consider each of these cases in turn.

FACE POINT. First, let us consider a face point. A face point v is chosen as the average of the vertices defining its associated face. That is, a face point v associated with the old face having vertices $v_0, v_1, \ldots, v_{n-1}$ is given by

$$v = \frac{1}{n}\sum_{k=0}^{n-1} v_k.$$

INTERIOR EDGE POINT. Next, let us consider an interior edge point. An interior edge point v is chosen as the average of the midpoint of the old edge and the average of the two new face points of the faces sharing the edge. Suppose that an edge point v originated from splitting an old edge that has vertices v_0, v_1 and is incident on the old faces associated with face points f_0, f_1. Then,

$$v = \frac{1}{2}\left[\frac{1}{2}(v_0+v_1)+\frac{1}{2}(f_0+f_1)\right] = \frac{1}{4}(v_0+v_1+f_0+f_1).$$

BOUNDARY EDGE POINT. Next, we consider a boundary edge point. A boundary edge point is chosen as the midpoint of its corresponding old edge. That is, for an edge point v originating from splitting an old edge with vertices v_0, v_1, we choose

$$v = \frac{1}{2}(v_0+v_1).$$

OLD INTERIOR VERTEX. Next, let us consider an old interior vertex. Consider an old interior vertex v with valence n in the old mesh. Let $v_0, v_1, \ldots, v_{n-1}$ be the old vertices that are 1-ring neighbours of v (before refinement). Let $f_0, f_1, \ldots, f_{n-1}$ be the face points of all faces incident on v. Then, the new value v' for the old vertex is given by

$$v' = \frac{n-3}{n}v + \frac{1}{n}q + \frac{2}{n}r,$$

where $q = \frac{1}{n}\sum_{k=0}^{n-1} f_k$, and $r = \frac{1}{n}\sum_{k=0}^{n-1}\frac{1}{2}(v+v_k)$ (i.e., q is the average of the new face points of all faces incident on the old vertex point, and r is the average of the midpoints of all old edges incident on the old vertex point). Alternatively, one can show that

$$v' = \frac{n-2}{n}v + \frac{1}{n^2}\left(\sum_{k=0}^{n-1} v_k + \sum_{k=0}^{n-1} f_k\right).$$

OLD BOUNDARY VERTEX. Lastly, we consider an old boundary vertex. The old boundary vertex v with the neighbouring boundary vertices v_0, v_1 has its new value v' chosen as

$$v' = \frac{3}{4}v + \frac{1}{8}(v_0+v_1).$$

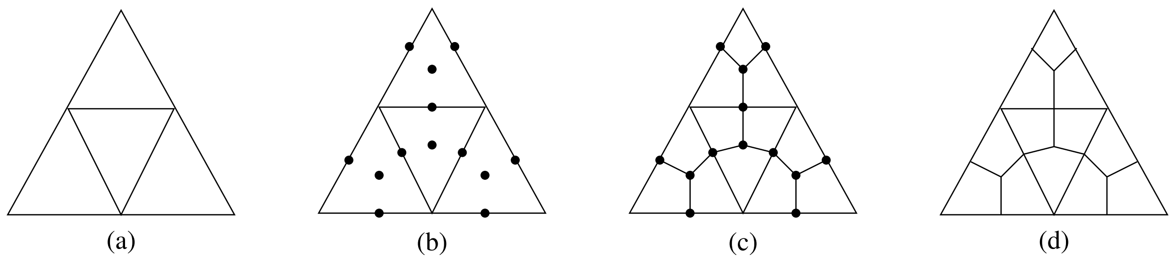


Figure 9.18: Example of topologic refinement for Catmull-Clark subdivision. (a) The topology of the initial mesh. (b) The topology after introducing new face and edge points. (c) The topology after connecting the new points in the mesh. (d) The topology of the refined mesh.

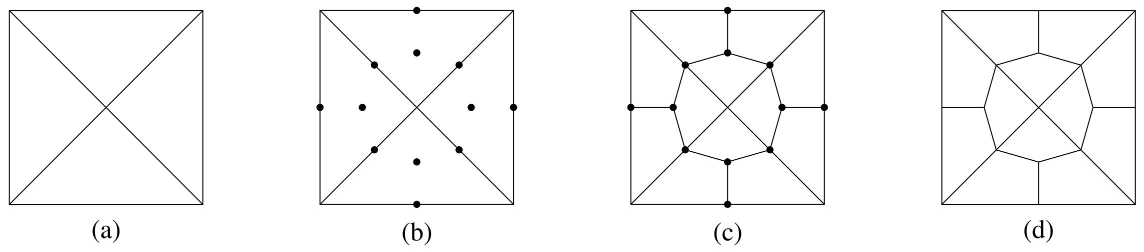


Figure 9.19: Example of topologic refinement for Catmull-Clark subdivision. (a) The topology of the initial mesh. (b) The topology after introducing new face and edge points. (c) The topology after connecting the new points in the mesh. (d) The topology of the refined mesh.

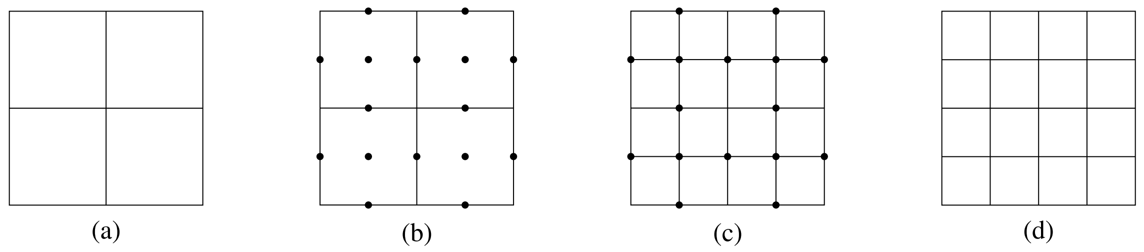
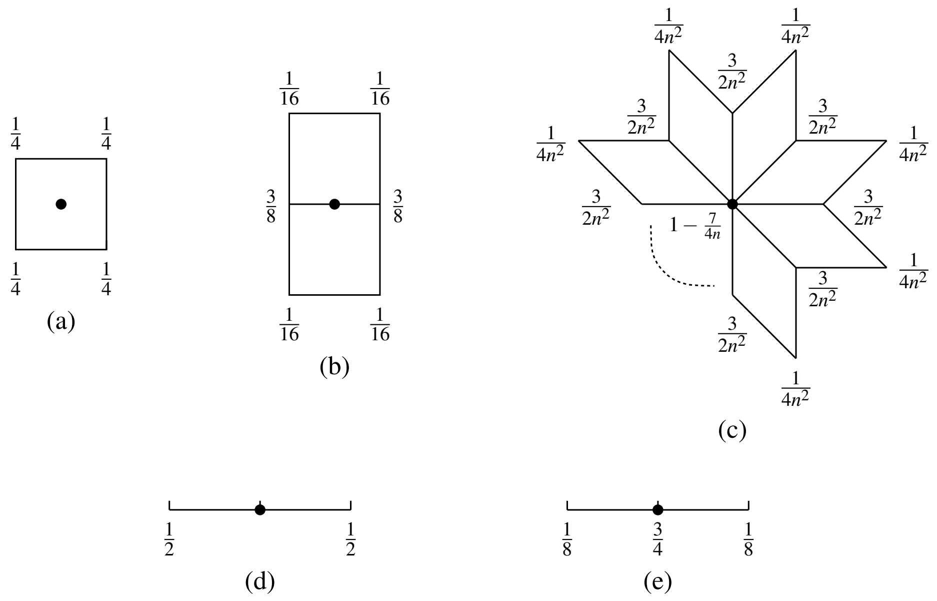


Figure 9.20: Example of topologic refinement for Catmull-Clark subdivision. (a) The topology of the initial mesh. (b) The topology after introducing new face and edge points. (c) The topology after connecting the new points in the mesh. (d) The topology of the refined mesh.



$\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$
(a)
$\frac{1}{16}$ $\frac{1}{16}$ $\frac{3}{8}$ $\frac{3}{8}$ $\frac{1}{16}$ $\frac{1}{16}$
(b)
$\frac{1}{4n^2}$ $\frac{3}{2n^2}$ $1-\frac{7}{4n}$
(c)
$\frac{1}{2}$ $\frac{1}{2}$
(d)
$\frac{1}{8}$ $\frac{3}{4}$ $\frac{1}{8}$
(e)

Figure 9.21: Geometric refinement masks for Catmull-Clark subdivision in the case of quadrilateral faces. The geometric mask for (a) a face point, (b) an interior edge point, (c) an interior old vertex, (d) a boundary edge point, and (e) a boundary old vertex.

Since the control mesh can be a polygon mesh with any types of faces (i.e., triangles, quadrilaterals, and so on), it is not practical to enumerate all possible cases of geometric masks in pictorial form. In the case that the mesh to be refined is a quadrilateral mesh, however, the masks have the form shown in Figure 9.21. In particular, Figures 9.21(a), (b), (c), (d), and (e), show the masks used for a face point, an edge point, an old interior vertex, a boundary edge point, and an old boundary vertex. Since, after the first iteration of subdivision, we always have a quadrilateral mesh, the preceding special case almost always applies in practice.

Although all new faces introduced by subdivision are quadrilaterals, these faces are not necessarily planar. In many situations, nonplanar faces are undesirable. For example, many rendering engines will not properly handle nonplanar faces, which can lead to undesirable artifacts appearing in the rendered image. In practice, we can work around this problem by splitting quadrilateral faces into pairs of triangular faces, which yields a triangle mesh with all planar faces.

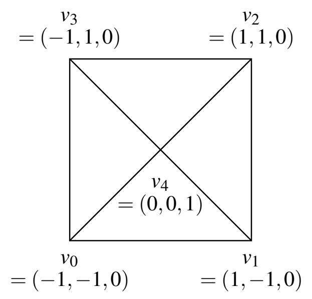
Example 9.8. Consider the control mesh shown in Figure 9.22(a). This mesh corresponds to the surface (with boundary) formed by removing the bottom face of a pyramid. Applying one level of Catmull-Clark subdivision to the control mesh, we obtain the refined mesh in Figure 9.22(b).

Example 9.9. Consider the control mesh shown in Figure 9.23(a). Applying one level of Catmull-Clark subdivision to the control mesh, we obtain the refined mesh in Figure 9.23(b).

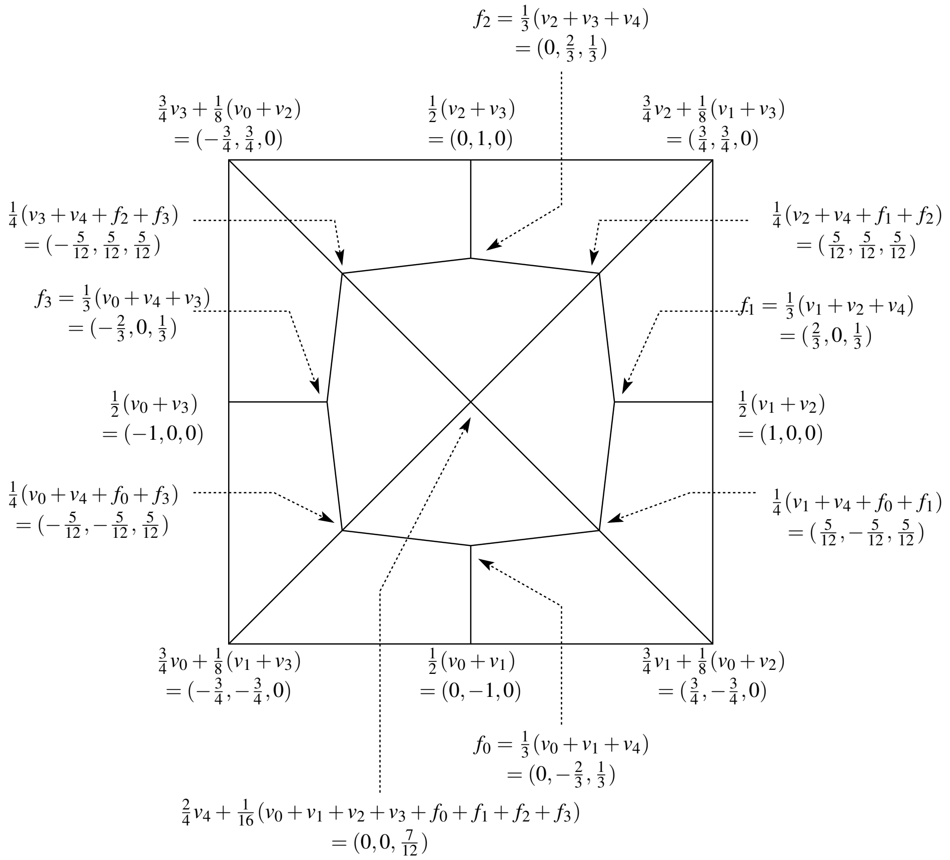
Example 9.10. Consider a mesh that corresponds to the surface of a tetrahedron as shown in Figure 9.24(a). Applying Catmull-Clark subdivision to this mesh, we obtain the refined meshes shown in Figures 9.24(b) to (e) and the limit surface shown in Figure 9.24(f).

9.2.11 Comparison of Subdivision Schemes

Let us consider a control mesh corresponding to the boundary of a tetrahedron as shown in Figure 9.25(a). The limit surfaces produced from this control mesh using several subdivision schemes are shown in Figures 9.25(b) to (e). Note that interpolating methods, such as linear and butterfly subdivision, produce limit surfaces that more closely resemble the original control mesh (as compared to approximating methods like Loop, Kobbelt $\sqrt{3}$, and Catmull-Clark



(a)


$f_2 = \frac{1}{3}(v_2+v_3+v_4) = (0,\frac{2}{3},\frac{1}{3})$
$\frac{3}{4}v_3 + \frac{1}{8}(v_0+v_2) = (-\frac{3}{4},\frac{3}{4},0)$
$\frac{1}{2}(v_2+v_3) = (0,1,0)$
$\frac{3}{4}v_2 + \frac{1}{8}(v_1+v_3) = (\frac{3}{4},\frac{3}{4},0)$
$\frac{1}{4}(v_3+v_4+f_2+f_3) = (-\frac{5}{12},\frac{5}{12},\frac{5}{12})$
$\frac{1}{4}(v_2+v_4+f_1+f_2) = (\frac{5}{12},\frac{5}{12},\frac{5}{12})$
$f_3 = \frac{1}{3}(v_0+v_4+v_3) = (-\frac{2}{3},0,\frac{1}{3})$
$f_1 = \frac{1}{3}(v_1+v_2+v_4) = (\frac{2}{3},0,\frac{1}{3})$
$\frac{1}{2}(v_0+v_3) = (-1,0,0)$
$\frac{1}{2}(v_1+v_2) = (1,0,0)$
$\frac{1}{4}(v_0+v_4+f_0+f_3) = (-\frac{5}{12},-\frac{5}{12},\frac{5}{12})$
$\frac{1}{4}(v_1+v_4+f_0+f_1) = (\frac{5}{12},-\frac{5}{12},\frac{5}{12})$
$\frac{3}{4}v_0 + \frac{1}{8}(v_1+v_3) = (-\frac{3}{4},-\frac{3}{4},0)$
$\frac{1}{2}(v_0+v_1) = (0,-1,0)$
$\frac{3}{4}v_1 + \frac{1}{8}(v_0+v_2) = (\frac{3}{4},-\frac{3}{4},0)$
$f_0 = \frac{1}{3}(v_0+v_1+v_4) = (0,-\frac{2}{3},\frac{1}{3})$
$\frac{2}{4}v_4 + \frac{1}{16}(v_0+v_1+v_2+v_3+f_0+f_1+f_2+f_3) = (0,0,\frac{7}{12})$

(b)

Figure 9.22: Catmull-Clark subdivision example (for non-quadrilateral mesh). (a) Control mesh (a surface with boundary). (b) Refined mesh obtained after one round of subdivision.

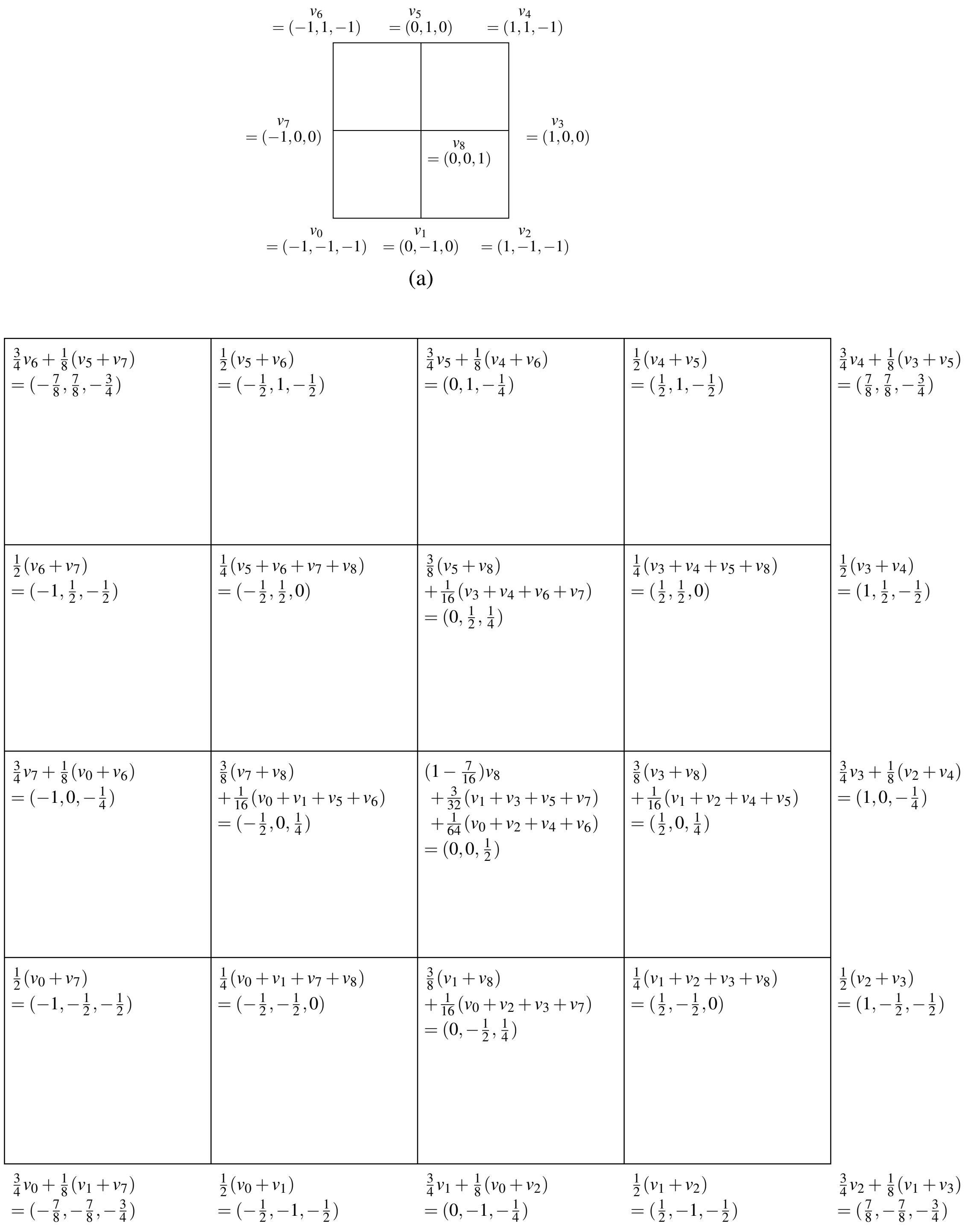

$v_6 = (-1,1,-1)$
$v_5 = (0,1,0)$
$v_4 = (1,1,-1)$
$v_7 = (-1,0,0)$
$v_8 = (0,0,1)$
$v_3 = (1,0,0)$
$v_0 = (-1,-1,-1)$
$v_1 = (0,-1,0)$
$v_2 = (1,-1,-1)$
(a)
$\frac{3}{4}v_6 + \frac{1}{8}(v_5+v_7) = (-\frac{7}{8}, \frac{7}{8}, -\frac{3}{4})$
$\frac{1}{2}(v_5+v_6) = (-\frac{1}{2}, 1, -\frac{1}{2})$
$\frac{3}{4}v_5 + \frac{1}{8}(v_4+v_6) = (0,1,-\frac{1}{4})$
$\frac{1}{2}(v_4+v_5) = (\frac{1}{2}, 1, -\frac{1}{2})$
$\frac{3}{4}v_4 + \frac{1}{8}(v_3+v_5) = (\frac{7}{8}, \frac{7}{8}, -\frac{3}{4})$
$\frac{1}{2}(v_6+v_7) = (-1, \frac{1}{2}, -\frac{1}{2})$
$\frac{1}{4}(v_5+v_6+v_7+v_8) = (-\frac{1}{2}, \frac{1}{2}, 0)$
$\frac{3}{8}(v_5+v_8) + \frac{1}{16}(v_3+v_4+v_6+v_7) = (0, \frac{1}{2}, \frac{1}{4})$
$\frac{1}{4}(v_3+v_4+v_5+v_8) = (\frac{1}{2}, \frac{1}{2}, 0)$
$\frac{1}{2}(v_3+v_4) = (1, \frac{1}{2}, -\frac{1}{2})$
$\frac{3}{4}v_7 + \frac{1}{8}(v_0+v_6) = (-1, 0, -\frac{1}{4})$
$\frac{3}{8}(v_7+v_8) + \frac{1}{16}(v_0+v_1+v_5+v_6) = (-\frac{1}{2}, 0, \frac{1}{4})$
$(1-\frac{7}{16})v_8 + \frac{3}{32}(v_1+v_3+v_5+v_7) + \frac{1}{64}(v_0+v_2+v_4+v_6) = (0,0,\frac{1}{2})$
$\frac{3}{8}(v_3+v_8) + \frac{1}{16}(v_1+v_2+v_4+v_5) = (\frac{1}{2}, 0, \frac{1}{4})$
$\frac{3}{4}v_3 + \frac{1}{8}(v_2+v_4) = (1, 0, -\frac{1}{4})$
$\frac{1}{2}(v_0+v_7) = (-1, -\frac{1}{2}, -\frac{1}{2})$
$\frac{1}{4}(v_0+v_1+v_7+v_8) = (-\frac{1}{2}, -\frac{1}{2}, 0)$
$\frac{3}{8}(v_1+v_8) + \frac{1}{16}(v_0+v_2+v_3+v_7) = (0, -\frac{1}{2}, \frac{1}{4})$
$\frac{1}{4}(v_1+v_2+v_3+v_8) = (\frac{1}{2}, -\frac{1}{2}, 0)$
$\frac{1}{2}(v_2+v_3) = (1, -\frac{1}{2}, -\frac{1}{2})$
$\frac{3}{4}v_0 + \frac{1}{8}(v_1+v_7) = (-\frac{7}{8}, -\frac{7}{8}, -\frac{3}{4})$
$\frac{1}{2}(v_0+v_1) = (-\frac{1}{2}, -1, -\frac{1}{2})$
$\frac{3}{4}v_1 + \frac{1}{8}(v_0+v_2) = (0, -1, -\frac{1}{4})$
$\frac{1}{2}(v_1+v_2) = (\frac{1}{2}, -1, -\frac{1}{2})$
$\frac{3}{4}v_2 + \frac{1}{8}(v_1+v_3) = (\frac{7}{8}, -\frac{7}{8}, -\frac{3}{4})$
(b)

Figure 9.23: Catmull-Clark subdivision example (for quadrilateral mesh). (a) Control mesh (a surface with boundary). (b) Refined mesh obtained after one round of subdivision.

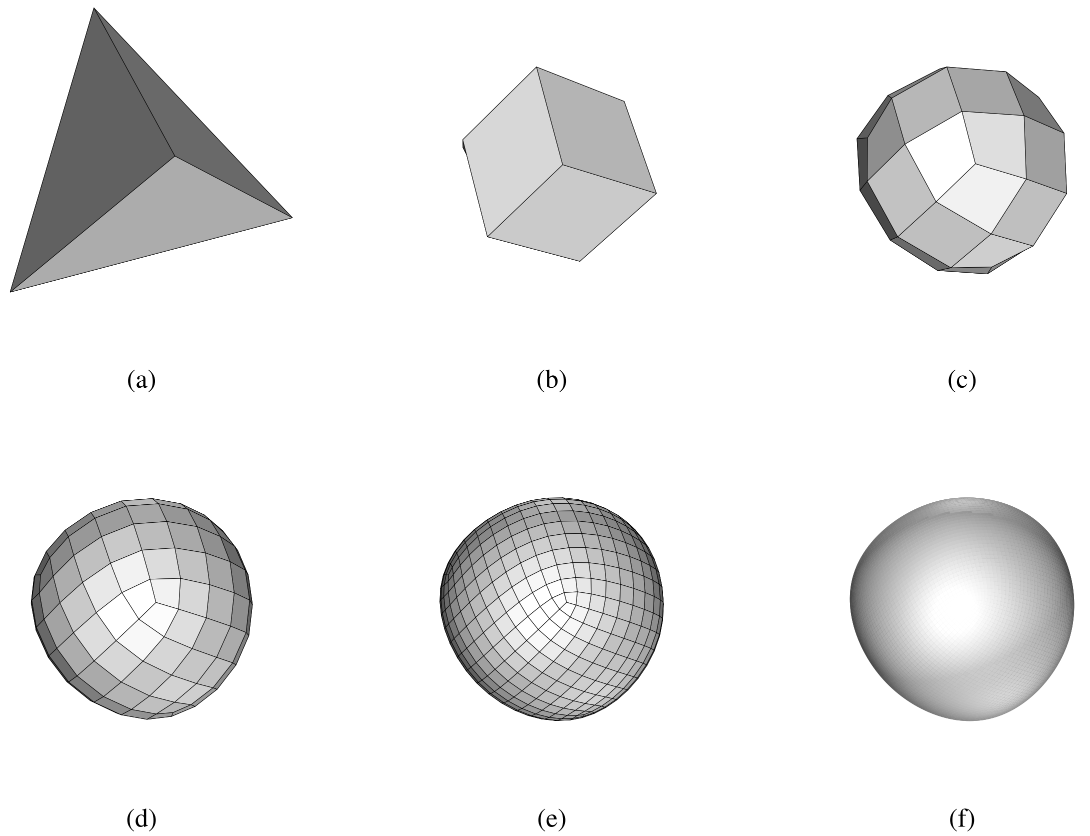


Figure 9.24: Catmull-Clark subdivision example. (a) Control mesh (a tetrahedron). The refined mesh obtained after (b) one, (c) two, (d) three, and (e) four levels of subdivision. (f) The limit surface.

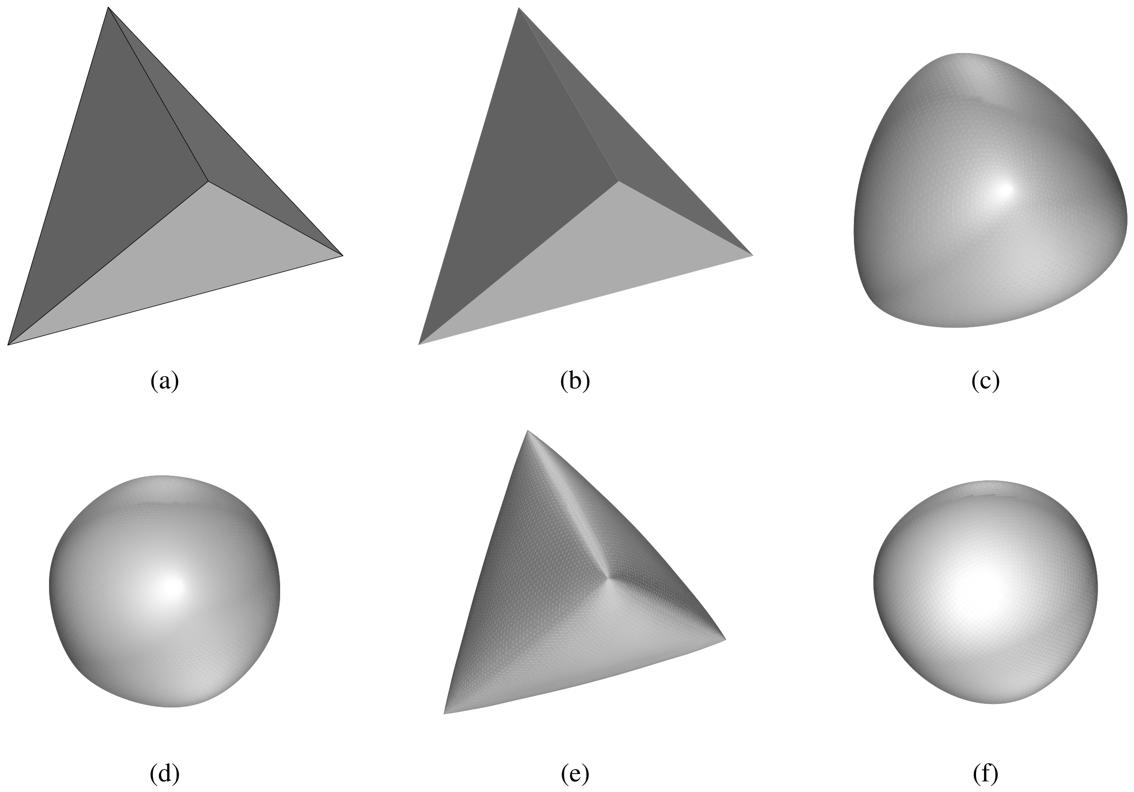


Figure 9.25: Comparison of the limit surfaces produced by various subdivision schemes. (a) Control mesh (a tetrahedron). The limit surface produced by the (b) linear, (c) Loop, (d) Kobbelt $\sqrt{3}$, (e) butterfly, and (f) Catmull-Clark subdivision schemes.

subdivision). On the other hand, the approximating methods produce much smoother limit surfaces (as compared to the interpolating methods).

9.2.12 Position and Tangent Masks

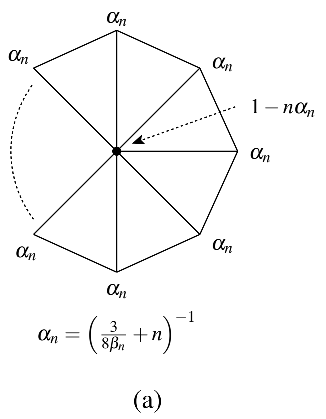
In the case of an approximating scheme, each iteration of subdivision results in a repositioning of old vertices. For a particular vertex in the mesh, we might want to determine where the vertex will be positioned in the limit (i.e., after an infinite number of iterations of subdivision are applied). As it turns out, the limit position of a vertex can be easily found by taking an appropriate linear combination of nearby mesh vertices. This linear combination can be represented pictorially as a **position mask**.

Similarly, we might want to determine a tangent vector to the limit surface at the limit position of a vertex. A tangent vector can also be computed using an appropriate linear combination of nearby mesh vertices. This linear combination can be represented pictorially as a **tangent mask**. Using two tangent masks, we can determine two tangent vectors, which can then be used to calculate a surface normal (via a vector cross product).

9.2.12.1 Loop Subdivision

Suppose that we want to calculate the limit position of a vertex for Loop subdivision. There are two cases to consider, corresponding to the vertex in question being: 1) an interior vertex or 2) a boundary vertex. We consider each of these cases in turn.

INTERIOR VERTEX. First, we consider the case of an interior vertex [10, Fig. 5], [6, p. 68, Fig. 4.8]. Let p denote the interior vertex whose limit position is to be determined. Let $\{p_i\}_{i=1}^{n}$ denote the 1-ring neighbours of p. Then, the



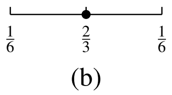


Figure 9.26: Position masks for Loop subdivision. The position masks for (a) an interior vertex and (b) a boundary vertex.

Table 9.4: The quantity α_n used in the position mask of Loop subdivision (for several small values of n)

n	α_n
3	$\frac{1}{5} = 0.2$
4	$\frac{31}{220} \approx 0.140909090909091$
5	$\approx 0.105715654625119$
6	$\frac{1}{12} \approx 0.0833333333333333$
7	$\approx 0.0682632482712396$
8	$\approx 0.0576065604228295$
9	$\approx 0.0497522474300052$
10	$\approx 0.0437593527476905$

limit position p' of p is given by

$$p' = (1-n\alpha_n)p + \sum_{i=1}^{n} \alpha_n p_n,$$

where

$$\alpha_n = \left(\frac{3}{8\beta_n}+n\right)^{-1},$$

and β_n is as given by (9.1). The corresponding position mask is shown in Figure 9.26(a). The quantity α_n is given for several values of n in Table 9.4.

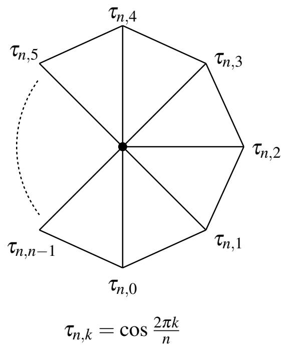
BOUNDARY VERTEX. Next, let us consider the case of a boundary vertex. Let p denote the boundary vertex whose limit position is to be determined. Let p_1 and p_2 denote the two neighbours of p along the mesh boundary. Then, the limit position p' of p is given by

$$p' = \tfrac{2}{3}p + \tfrac{1}{6}(p_1+p_2).$$

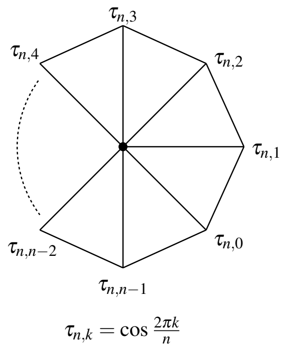
The corresponding position mask is shown in Figure 9.26(b).

For Loop subdivision, a vector t tangent to the limit surface at the limit position of the vertex p can be computed as follows. Let $\{p_i\}_{i=1}^{n}$ denote the 1-ring neighbours of p. Then, t is given by

$$t = \sum_{i=1}^{n} \tau_{n,i-1} p_i,$$



(a)



(b)

Figure 9.27: Two tangent masks for Loop subdivision. (a) First and (b) second mask.

Table 9.5: The quantity $\tau_{n,k}$ used in the tangent masks of Loop subdivision (for some small values of n)

n	$\{\tau_{n,0}, \tau_{n,1}, \ldots, \tau_{n,n-1}\}$
3	$1, -\frac{1}{2}, -\frac{1}{2}$
4	1, 0, -1, 0
5	1, $\cos(\frac{2\pi}{5}) \approx 0.309016994374947$, $\cos(\frac{4\pi}{5}) \approx -0.809016994374947$, $\cos(\frac{6\pi}{5}) \approx -0.809016994374948$, $\cos(\frac{8\pi}{5}) \approx 0.309016994374947$
6	1, $\frac{1}{2}$, $-\frac{1}{2}$, -1, $-\frac{1}{2}$, $\frac{1}{2}$

where

$$\tau_{n,k} = \cos\left(\tfrac{2\pi k}{n}\right).$$

A corresponding tangent mask is shown in Figure 9.27(a) [6, p. 68, Fig. 4.8]. Additional tangent masks can be generated by circularly shifting the coefficients in the first mask. For example, a second mask is given in Figure 9.27(b). The coefficients $\{\tau_{n,k}\}$ are provided for several values of n in Table 9.5.

Example 9.11. Consider the application of Loop subdivision to the control mesh shown in Figure 9.28. Find the limit position of each vertex in the control mesh. Also, for each interior vertex in the control mesh, find a normal to the limit surface at the corresponding vertex limit position.

Solution. For $k \in \{0,1,2,3\}$, let v'_k denote the limit position of the vertex v_k in the control mesh. To find the limit position of a vertex, we apply the appropriate position mask. Using the appropriate position masks, we have

$$\begin{aligned}
v'_3 &= (1-3\alpha_3)v_3 + \alpha_3(v_0+v_1+v_2)\\
&= \tfrac{2}{5}v_3 + \tfrac{1}{5}(v_0+v_1+v_2) = (\tfrac{2}{5},\tfrac{2}{5},\tfrac{2}{5}) + (-\tfrac{1}{5},-\tfrac{1}{5},\tfrac{1}{5}) + (\tfrac{1}{5},-\tfrac{1}{5},-\tfrac{1}{5}) + (-\tfrac{1}{5},\tfrac{1}{5},-\tfrac{1}{5})\\
&= (\tfrac{1}{5},\tfrac{1}{5},\tfrac{1}{5}),
\end{aligned}$$

$$\begin{aligned}
v'_0 &= \tfrac{2}{3}v_0 + \tfrac{1}{6}(v_1+v_2)\\
&= \tfrac{2}{3}(-1,-1,1) + \tfrac{1}{6}(0,0,-2) = (-\tfrac{2}{3},-\tfrac{2}{3},\tfrac{2}{3}) + (0,0,-\tfrac{1}{3})\\
&= (-\tfrac{2}{3},-\tfrac{2}{3},\tfrac{1}{3}),
\end{aligned}$$

$$\begin{aligned}
v'_1 &= \tfrac{2}{3}v_1 + \tfrac{1}{6}(v_0+v_2)\\
&= \tfrac{2}{3}(1,-1,-1) + \tfrac{1}{6}(-2,0,0) = (\tfrac{2}{3},-\tfrac{2}{3},-\tfrac{2}{3}) + (-\tfrac{1}{3},0,0)\\
&= (\tfrac{1}{3},-\tfrac{2}{3},-\tfrac{2}{3}), \quad \text{and}
\end{aligned}$$

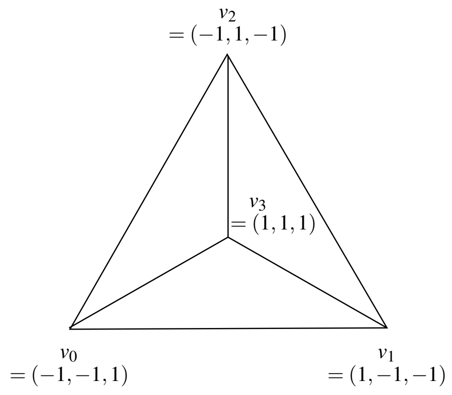


Figure 9.28: Control mesh.

$$\begin{aligned} v_2' &= \tfrac{2}{3}v_2 + \tfrac{1}{6}(v_0+v_1) \\ &= \tfrac{2}{3}(-1,1,-1) + \tfrac{1}{6}(0,-2,0) = (-\tfrac{2}{3},\tfrac{2}{3},-\tfrac{2}{3}) + (0,-\tfrac{1}{3},0) \\ &= (-\tfrac{2}{3},\tfrac{1}{3},-\tfrac{2}{3}). \end{aligned}$$

Only one of the vertices in the control mesh is in the interior. So, we only need to find a surface normal at one point on the limit surface, namely at v_3'. So, we proceed to find two tangent vectors t_0,t_1 to the limit surface at the point v_3'. To do this, we use the tangent masks. We have

$$\begin{aligned} t_0 &= v_0 - \tfrac{1}{2}v_1 - \tfrac{1}{2}v_2 \\ &= (-1,-1,1) - (\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2}) - (-\tfrac{1}{2},\tfrac{1}{2},-\tfrac{1}{2}) = (-1,-1,1) + (-\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}) + (\tfrac{1}{2},-\tfrac{1}{2},\tfrac{1}{2}) \\ &= (-1,-1,2), \quad \text{and} \end{aligned}$$

$$\begin{aligned} t_1 &= v_1 - \tfrac{1}{2}v_2 - \tfrac{1}{2}v_0 \\ &= (1,-1,-1) + (\tfrac{1}{2},-\tfrac{1}{2},\tfrac{1}{2}) + (\tfrac{1}{2},\tfrac{1}{2},-\tfrac{1}{2}) \\ &= (2,-1,-1). \end{aligned}$$

By computing the vector cross product of the two tangent vectors, we obtain the surface normal n. We have

$$n = t_0 \times t_1 = \det \begin{bmatrix} i & j & k \\ -1 & -1 & 2 \\ 2 & -1 & -1 \end{bmatrix} = i(3) - j(-3) + k(3) = (3,3,3).$$

Normalizing the length of n to one, we obtain the unit vector $\hat{n}$ given by

$$\hat{n} = \tfrac{1}{\|n\|}n = \tfrac{1}{\sqrt{3^2+3^2+3^2}}(3,3,3) = \tfrac{1}{\sqrt{27}}(3,3,3) = \tfrac{1}{3\sqrt{3}}(3,3,3) = (\tfrac{1}{\sqrt{3}},\tfrac{1}{\sqrt{3}},\tfrac{1}{\sqrt{3}}).$$

□

9.2.13 Surfaces with Creases

In many practical applications, we need to be able to represent surfaces that are mostly smooth but have some creases. A simple approach to representing creases is to treat edges associated with creases as boundary edges, during subdivision. Since geometric refinement masks for boundary vertices do not usually depend on non-boundary vertices, positional continuity is maintained by such an approach. The surface, however, need not be smooth across the crease edges. In this way, creases in a surface can be produced.

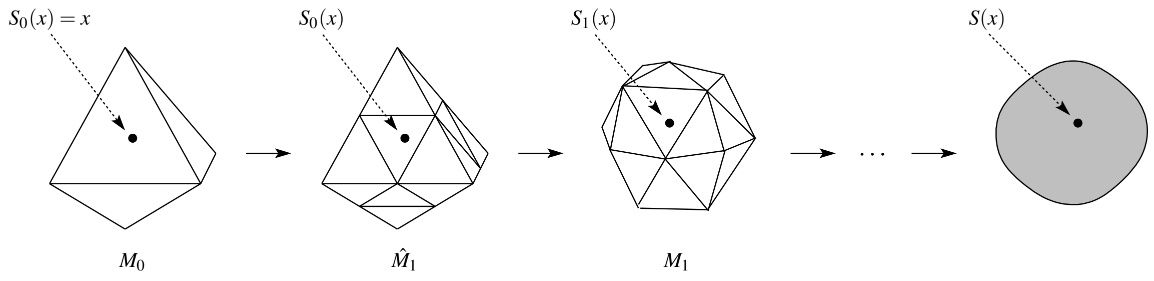


Figure 9.29: Tracking the movement of a point during subdivision.

9.2.14 Applications of Subdivision

Many applications require modelling of surfaces. Since subdivision provides a powerful tool for modelling surfaces, subdivision has potential application in many areas. Some of these application areas include: multimedia, animation, gaming, biomedical computing, computer-aided design and manufacturing, geometric modelling, finite element analysis, computational fluid dynamics, and scientific visualization. Some applications of subdivision are considered in Chapter/Appendix 11.

9.3 Subdivision Wavelets

Although not immediately obvious, subdivision surfaces have an implicit mathematical structure very similar to wavelets discussed in earlier chapters. We explore this relationship in more detail in the sections that follow.

9.3.1 Primal Subdivision as Splitting and Averaging

Subdivision iteratively refines a control polyhedron M_0 to produce increasingly faceted polyhedra $M_1, M_2, \ldots$ that converge to the limit surface M_∞. In each subdivision step, the vertices of $M_{\ell+1}$ are computed as affine combinations of the vertices of M_ℓ. The refinement process that transforms M_ℓ to $M_{\ell+1}$ can be viewed as consisting of two steps: 1) splitting and 2) averaging. In the splitting step, each face of M_ℓ is split into k new faces, yielding the intermediate mesh $\hat{M}_{\ell+1}$. In the case of primal triangle quadrisection, a face is split into $k = 3$ new faces by introducing a new vertex at the midpoint of each edge. The splitting step changes the topology of the mesh (by inserting new vertices) and establishes positions for the new vertices (which may be subsequently changed by averaging). The positions of the old vertices are not affected by splitting. In the averaging step, the vertex positions of $M_{\ell+1}$ are computed as affine combinations of the vertices of $\hat{M}_{\ell+1}$. The averaging step does not affect the topology of the mesh. Only the geometry of the mesh is affected. The old and new vertices may both be affected. Using this splitting and averaging view of subdivision, we can show that M_∞ can be expressed parametrically using M_0 as the domain of the parameterization.

9.3.2 Subdivision-Surface Parameterization

Since it will prove useful later, we seek a parametric representation of the limit surface produced by subdivision. To do this, we first establish a correspondence between the points on M_ℓ and $M_{\ell+1}$. Then, we use this correspondence to track the movement of an arbitrary point originating on M_0 from one subdivision level to the next, as subdivision is applied repeatedly. As the number of iterations approaches infinity, the point converges to a point on the limit surface. This process is illustrated in Figure 9.29. The above process establishes a correspondence between points on M_0 and points on the limit surface M_∞. The parametric representation of M_∞ immediately follows from this correspondence (which is nothing more than mapping from M_0 to M_∞).

Now, we describe the parameterization of the subdivision surface in more detail. Let I_ℓ denote the index set of the vertices in M_ℓ, and let $\{v_{\ell,n}\}_{n \in I_\ell}$ denote the corresponding vertices. The correspondence between points in M_ℓ and $M_{\ell+1}$ is illustrated in Figure 9.30. The vertex $v_{\ell+1,\beta_k}$ in $M_{\ell+1}$ is generated from the vertex $\hat{v}_{\ell+1,\beta_k}$ in $\hat{M}_{\ell+1}$ by averaging.

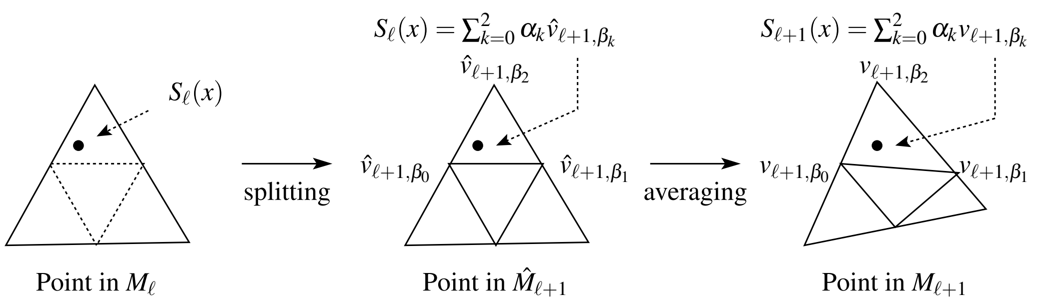

$S_\ell(x)$
Point in M_ℓ
splitting
$S_\ell(x) = \sum_{k=0}^{2} \alpha_k \hat{v}_{\ell+1,\beta_k}$
$\hat{v}_{\ell+1,\beta_2}$
$\hat{v}_{\ell+1,\beta_0}$
$\hat{v}_{\ell+1,\beta_1}$
Point in $\hat{M}_{\ell+1}$
averaging
$S_{\ell+1}(x) = \sum_{k=0}^{2} \alpha_k v_{\ell+1,\beta_k}$
$v_{\ell+1,\beta_2}$
$v_{\ell+1,\beta_0}$
$v_{\ell+1,\beta_1}$
Point in $M_{\ell+1}$

Figure 9.30: Correspondence between points on M_ℓ, $\hat{M}_{\ell+1}$, and $M_{\ell+1}$.

To begin, we consider an arbitrary point $S_\ell(x)$ on an arbitrary face of M_ℓ (as shown on the left side of Figure 9.30). Since splitting a face in M_ℓ produces a new set of faces whose union is the original (i.e., unsplit) face, $S_\ell(x)$ must lie on one of these new faces (which belong to $\hat{M}_{\ell+1}$). Suppose that $S_\ell(x)$ lies in the new triangle $(\hat{v}_{\ell+1,\beta_0}, \hat{v}_{\ell+1,\beta_1}, \hat{v}_{\ell+1,\beta_2})$ of $\hat{M}_{\ell+1}$ with the barycentric coordinates $(\alpha_0, \alpha_1, \alpha_2)$. That is,

$$S_\ell(x) = \sum_{k=0}^{2} \alpha_k \hat{v}_{\ell+1,\beta_k}.$$

Since $\hat{M}_{\ell+1}$ and $M_{\ell+1}$ have identical topologies, we have a trivial correspondence between faces on $\hat{M}_{\ell+1}$ and $M_{\ell+1}$. Suppose that the triangle $(\hat{v}_{\ell+1,\beta_0}, \hat{v}_{\ell+1,\beta_1}, \hat{v}_{\ell+1,\beta_2})$ in $\hat{M}_{\ell+1}$ has the corresponding triangle $(v_{\ell+1,\beta_0}, v_{\ell+1,\beta_1}, v_{\ell+1,\beta_2})$ in $M_{\ell+1}$. Then, a point $S_{\ell+1}(x)$ in $M_{\ell+1}$ corresponding to $S_\ell(x)$ in $\hat{M}_{\ell+1}$ can be chosen as

$$S_{\ell+1}(x) = \sum_{k=0}^{2} \alpha_k v_{\ell+1,\beta_k}.$$

That is, we choose the point having the same barycentric coordinates with respect to the geometrically repositioned face resulting from averaging. Note that $\{\hat{v}_{\ell+1,\beta_k}\}_{k=0}^{2}$ and $\{v_{\ell+1,\beta_k}\}_{k=0}^{2}$ are not generally the same except for linear subdivision. We have tried to emphasize this fact in the figure, by having the vertex positions appear noticeably different before and after averaging. The above establishes a correspondence between points on M_ℓ and $M_{\ell+1}$.

Now, we select

$$S_0(x) = x \text{ for } x \in M_0.$$

Then, using the relationship between $S_\ell(x)$ and $S_{\ell+1}(x)$ from above, we can map $S_0(x)$ to $S_1(x)$, and then $S_1(x)$ to $S_2(x)$, and so on, until we obtain $S_\ell(x)$. At this point, we have generated a mapping from a point x on M_0 to a point $S_\ell(x)$ on M_ℓ. This is nothing more than a parametric representation of M_ℓ. Continuing this mapping process ad infinitum (i.e., by letting $\ell \to \infty$), we obtain the parametric representation for M_∞ given by

$$S(x) = \lim_{\ell\to\infty} S_\ell(x).$$

Observe that the domain of the above parameterization is M_0. As we shall see, the above subdivision-surface parameterization leads to a wavelet representation for meshes.

9.3.3 Approximation Spaces and Scaling Functions

Now, we explore some of the properties of the parameterization just developed in more detail. First, we consider the parametric representation $S_\ell(x)$ for M_ℓ (i.e., the polyhedral surface obtained after ℓ levels of subdivision).

Lemma 9.1. *Let I_n denote the index set of the vertices of M_n and let $\{v_{n,k}\}_{k\in I_n}$ denote the corresponding vertices. For all $n \geq 0$ and all $\ell \geq n$, there exist functions $\phi_{\ell\leftarrow n,k} : M_0 \to \mathbb{R}$ such that*

$$S_\ell(x) = \sum_{k\in I_n} v_{n,k}\phi_{\ell\leftarrow n,k}(x). \tag{9.2}$$

Furthermore, each element of $\{\phi_{\ell\leftarrow n,k}\}_{k\in I_n}$ *can be expressed in terms of* $\{\phi_{\ell\leftarrow n+1,m}\}_{m\in I_{n+1}}$ *as*

$$\phi_{\ell\leftarrow n,k} = \sum_{m\in I_{n+1}} a_m \phi_{\ell\leftarrow n+1,m}, \tag{9.3}$$

for some choice of $\{a_m\}_{m\in I_{n+1}}$.

Proof. Let $\Phi_{\ell\leftarrow n}(x)$ denote the row vector whose kth element is $\phi_{\ell\leftarrow n,k}(x)$. Let V_n denote the matrix whose kth row is the x, y, and z coordinates of the kth vertex of M_n. To begin, we rewrite (9.2) and (9.3) in vector form as

$$S_\ell(x) = \Phi_{\ell\leftarrow n}(x)V_n \quad \text{and} \tag{9.4}$$
$$\Phi_{\ell\leftarrow n}(x) = \Phi_{\ell\leftarrow n+1}(x)A. \tag{9.5}$$

Thus, we need to show that relationships (9.4) and (9.5) hold. From the parametric representation of M_ℓ, we have that $S_\ell(x) = \sum_{k=0}^{2} \alpha_k v_{\ell,\beta_k}$, which can be rewritten in vector form as

$$S_\ell(x) = b_\ell(x)V_\ell, \tag{9.6}$$

where $b_\ell(x)$ is the row vector given by

$$(b_\ell(x))_n = \begin{cases} \alpha_n & n \in \{\beta_0, \beta_1, \beta_2\} \\ 0 & \text{otherwise} \end{cases}$$

(i.e., the vector $b_\ell(x)$ has all of its elements equal to zero except for the elements with indices β_0, β_1, and β_2, which have the respective values α_0, α_1, and α_2). Because the vertices of M_ℓ are affine combinations of the vertices of $M_{\ell-1}$, we have that, for some (nonsquare) matrix $P_{\ell-1}$,

$$V_\ell = P_{\ell-1}V_{\ell-1}.$$

Applying the preceding relationship recursively $\ell - n$ times, we obtain

$$V_\ell = P_{\ell-1}P_{\ell-2}\cdots P_n V_n.$$

Substituting the preceding equation into (9.6) yields

$$S_\ell(x) = b_\ell(x)P_{\ell-1}P_{\ell-2}\cdots P_n V_n.$$

Defining

$$\Phi_{\ell\leftarrow n}(x) = b_\ell(x)P_{\ell-1}P_{\ell-2}\cdots P_n, \tag{9.7}$$

we obtain (9.4). Thus, (9.2) holds. From (9.7), we observe that

$$\begin{aligned} \Phi_{\ell\leftarrow n}(x) &= b_\ell(x)P_{\ell-1}P_{\ell-2}\cdots P_{n+1}P_n. \\ &= [b_\ell(x)P_{\ell-1}P_{\ell-2}\cdots P_{n+1}]\,P_n. \\ &= \phi_{\ell\leftarrow n+1}(x)P_n. \end{aligned}$$

Thus, a relationship of the form of (9.5) holds. Therefore, (9.3) holds. □

Now, we consider some of the properties of the parameterization $S(x)$ of the limit surface M_∞.

Theorem 9.1. *For any subdivision scheme, and for any* $\ell \geq 0$*, given the vertices* $\{v_{\ell,k}\}_{k\in I_\ell}$ *of the mesh* M_ℓ*, there exist scalar-valued functions* $\{\phi_{\ell,k}\}$ *defined on* M_0 *such that*

$$S(x) = \sum_{k\in I_\ell} v_{\ell,k}\phi_{\ell,k}(x). \tag{9.8}$$

Proof. From Lemma 9.1, we have that

$$S_\ell(x) = \sum_{k \in I_\ell} v_{n,k} \phi_{\ell \leftarrow n,k}(x).$$

Taking the limit of both sides of the preceding equation as $\ell \to \infty$ (and using the fact that $S(x) = \lim_{\ell \to \infty} S_\ell(x)$), we obtain

$$S(x) = \lim_{\ell \to \infty} \sum_{k \in I_n} v_{n,k} \phi_{\ell \leftarrow n,k}(x).$$

Assuming that we can interchange the summation and limit (which turns out to be valid provided that the subdivision scheme is sufficiently well behaved), we obtain

$$S(x) = \sum_{k \in I_n} v_{n,k} \lim_{\ell \to \infty} \phi_{\ell \leftarrow n,k}(x).$$

Defining

$$\phi_{n,k}(x) = \lim_{\ell \to \infty} \phi_{\ell \leftarrow n,k}(x), \tag{9.9}$$

we can rewrite the preceding equation as

$$S(x) = \sum_{k \in I_n} v_{n,k} \phi_{n,k}(x).$$

Thus, (9.8) holds. □

As a matter of terminology, the $\{\phi_{\ell,k}\}$ (from the above theorem) are called **scaling functions**. As it turns out, these scaling functions are refinable in the sense considered by the theorem below.

Theorem 9.2. *The scaling functions $\{\phi_{\ell,k}\}_{k \in I_\ell}$ are refinable, in the sense that each element of $\{\phi_{\ell,k}\}_{k \in I_\ell}$ can be expressed in terms of $\{\phi_{\ell+1,k}\}_{k \in I_{\ell+1}}$ as*

$$\phi_{\ell,k} = \sum_{m \in I_{\ell+1}} a_m \phi_{\ell+1,m}, \tag{9.10}$$

where $\{a_m\}_{m \in I_{\ell+1}}$ is a scalar sequence.

Proof. In Lemma 9.1, we showed that

$$\phi_{\ell \leftarrow n,k} = \sum_{m \in I_{n+1}} a_m \phi_{\ell \leftarrow n+1,m}.$$

Taking the limit of both sides of the preceding equation as $\ell \to \infty$, we obtain

$$\lim_{\ell \to \infty} \phi_{\ell \leftarrow n,k} = \lim_{\ell \to \infty} \sum_{m \in I_{n+1}} a_m \phi_{\ell \leftarrow n+1,m}. \quad \Rightarrow$$
$$\lim_{\ell \to \infty} \phi_{\ell \leftarrow n,k} = \sum_{m \in I_{n+1}} a_m \lim_{\ell \to \infty} \phi_{\ell \leftarrow n+1,m}.$$

Using (9.9), the preceding equation reduces to

$$\phi_{n,k} = \sum_{m \in I_{n+1}} a_m \phi_{n+1,m}.$$

Thus, the scaling functions can be expressed in the form stated above. □

$$\begin{tikzcd}
V_\infty \arrow[r] & \cdots V_4 \arrow[r] \arrow[dr] & V_3 \arrow[r] \arrow[dr] & V_2 \arrow[r] \arrow[dr] & V_1 \arrow[r] \arrow[dr] & V_0 \\
& & W_3 & W_2 & W_1 & W_0
\end{tikzcd}$$

Figure 9.31: MRA decomposition of V_∞.

The relationship (9.10) is analogous to the scaling equation (i.e., a refinement equation) from classical wavelet systems, discussed earlier in Chapters 4 and 7.

Assuming that the subdivision scheme is sufficiently well behaved, it can be shown that, for each $\ell \in \mathbb{Z}^*$, $\{\phi_{\ell,k}\}_{k\in I_\ell}$ is independent (i.e., these functions form a basis for their span). From the scaling functions, we can define a sequence $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ of spaces, called **approximation spaces**, as

$$V_\ell = \operatorname{span}\{\phi_{\ell,k}\}_{k\in I_\ell}.$$

Provided that the subdivision scheme is sufficiently well behaved so that $\{\phi_{\ell,k}\}_{k\in I_\ell}$ is independent, a basis for V_ℓ is given by $\{\phi_{\ell,k}\}_{k\in I_\ell}$. Using the refinability of the scaling functions, we can show that the $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ are nested as

$$V_0 \subset V_1 \subset V_2 \subset \ldots.$$

The approximation spaces $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ are analogous to the spaces of a MRA in classical wavelet systems.

9.3.4 Wavelets Spaces and Wavelet Functions

Since $V_\ell \subset V_{\ell+1}$, there must exist some subspace W_ℓ of $V_{\ell+1}$ such that $V_{\ell+1} = V_\ell \oplus W_\ell$ (i.e., W_ℓ is the algebraic complement of V_ℓ in $V_{\ell+1}$). Thus, we can associate with $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ another sequence $\{W_\ell\}_{\ell\in\mathbb{Z}^*}$ of spaces. The $\{W_\ell\}_{\ell\in\mathbb{Z}^*}$ are called **wavelet spaces**. The wavelet spaces $\{W_\ell\}_{\ell\in\mathbb{Z}^*}$ can be shown to be mutually disjoint. For each wavelet space W_ℓ, we can find a basis $\{\psi_{\ell,k}\}_{k\in I_\ell}$. The $\{\psi_{\ell,k}\}$ are called **wavelet functions**. Since $\psi_{\ell,k} \in V_{\ell+1}$ (as $\psi_{\ell,k} \in W_\ell \subset V_{\ell+1}$) for all $k \in I_\ell$, $\psi_{\ell,k}$ can be expressed in terms of $\{\phi_{\ell+1,k}\}_{k\in I_{\ell+1}}$ (which is a basis of $V_{\ell+1}$) as

$$\psi_{\ell,k} = \sum_{k\in I_{\ell+1}} a_k \phi_{\ell+1,k}.$$

This is analogous to a wavelet equation from classical wavelet systems.

9.3.5 Wavelet Systems

Let $V_\infty = \lim_{\ell\to\infty} V_\ell$. The wavelet and approximation spaces decompose the space V_∞ as given by

$$V_\infty = V_0 \oplus \left(\bigoplus_{k\in\mathbb{Z}^*} W_k \right).$$

This decomposition is shown in pictorial form in Figure 9.31. Notice that the index set for the approximation and wavelet spaces is not $\mathbb{Z}$ but rather $\mathbb{Z}^*$. Also, observe that the elements of the approximation and wavelet spaces are functions defined on a manifold M_0, not functions defined on a Euclidean domain such as $\mathbb{R}$. Since the domain of the scaling/wavelet functions is M_0, which is a manifold, the type of variation on wavelets that we are considering here is sometimes referred to as wavelets on manifolds. (Classical wavelets are sometimes referred to as wavelets on Euclidean domains.)

Before proceeding further, it is instructive to examine a couple of expressions more carefully, to ensure a clear understanding of what they represent. Suppose that we have a sequence $\{\varphi_k\}_{k\in I}$ of functions, where $\varphi_k : M_0 \to \mathbb{R}$. The scaling and wavelet functions are examples of functions of this form.

First, consider a function f of the form

$$f(x) = \sum_{k\in I} a_k \varphi_k(x), \tag{9.11}$$

where $\{a_k\}_{k\in I}$ is a real sequence and $x \in M_0$. Clearly, the function f maps M_0 to $\mathbb{R}$. So, f is not a surface. (A surface would need to map M_0 to $\mathbb{R}^3$.) The elements of the approximation and wavelet spaces have the form of (9.11). In the case of the approximation space V_ℓ, $\varphi_k = \phi_{\ell,k}$, whereas in the case of the wavelet space W_ℓ, $\varphi_k = \psi_{\ell,k}$. So, the elements of the approximation spaces $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ and wavelet spaces $\{W_\ell\}_{\ell\in\mathbb{Z}^*}$ are not surfaces.

Now, consider a function s of the form

$$s(x) = \sum_{k\in I} v_k \varphi_k(x), \tag{9.12}$$

where $\{v_k\}_{k\in I}$ is a sequence of vectors in $\mathbb{R}^3$ and $x \in M_0$. Let the x, y, and z components of the vector-valued function s be denoted as s_x, s_y, and s_z, respectively. Let the x, y, and z coordinates of the vector v_k be denoted as $v_{k,\mathsf{x}}$, $v_{k,\mathsf{y}}$, and $v_{k,\mathsf{z}}$, respectively. Then, we can equivalently write (9.12) as

$$\begin{bmatrix} s_\mathsf{x}(x) \\ s_\mathsf{y}(x) \\ s_\mathsf{z}(x) \end{bmatrix} = \sum_{k\in I} \left(\begin{bmatrix} v_{k,\mathsf{x}} \\ v_{k,\mathsf{y}} \\ v_{k,\mathsf{z}} \end{bmatrix} \varphi_k(x) \right) = \sum_{k\in I} \begin{bmatrix} v_{k,\mathsf{x}}\varphi_k(x) \\ v_{k,\mathsf{y}}\varphi_k(x) \\ v_{k,\mathsf{z}}\varphi_k(x) \end{bmatrix} = \begin{bmatrix} \sum_{k\in I} v_{k,\mathsf{x}}\varphi_k(x) \\ \sum_{k\in I} v_{k,\mathsf{y}}\varphi_k(x) \\ \sum_{k\in I} v_{k,\mathsf{z}}\varphi_k(x) \end{bmatrix}.$$

From this, it is clear that the function s maps M_0 to $\mathbb{R}^3$. In other words, s is of the form of a surface. Furthermore, suppose that we define $U = \operatorname{span}\{\varphi_k\}_{k\in I}$. Then, $s \in U^3$ (since $s_\mathsf{x}, s_\mathsf{y}, s_\mathsf{z} \in U$). Thus, the elements of V_ℓ^3, which have the form of (9.12) with $\varphi_k = \phi_{\ell,k}$, are surfaces. Similarly, the elements of W_ℓ^3, which have the form of (9.12) with $\varphi_k = \psi_{\ell,k}$, are surfaces.

Recalling (9.8), we have that

$$S(x) = \sum_{k\in I_\ell} v_{\ell,k}\phi_{\ell,k}(x).$$

Let the x, y, and z components of the vector-valued function S be denoted as S_x, S_y, S_z, respectively. Since, from the above equation, $S_\mathsf{x}, S_\mathsf{y}, S_\mathsf{z} \in \operatorname{span}\{\phi_{\ell,k}\}_{k\in I_\ell}$, we can conclude that, for all $\ell \in \mathbb{Z}^*$, $S_\mathsf{x}, S_\mathsf{y}, S_\mathsf{z} \in V_\ell$, or equivalently, $S \in V_\ell^3$. So, strictly speaking, the multiresolution representation of surfaces is associated with the spaces $\{V_\ell^3\}_{\ell\in\mathbb{Z}^*}$ and $\{W_\ell^3\}_{\ell\in\mathbb{Z}^*}$. Of course, if one views surfaces in terms of their x, y, and z components separately, then the spaces of interest simply become $\{V_\ell\}_{\ell\in\mathbb{Z}^*}$ and $\{W_\ell\}_{\ell\in\mathbb{Z}^*}$.

9.3.6 Wavelet Analysis and Synthesis Operators

A surface $s \in V_\ell^3$ can be represented in two different ways. Since $s \in V_\ell^3$, s has an expansion in terms of the basis of V_ℓ given by

$$s = \sum_{k\in I_\ell} v_{\ell,k}\phi_{\ell,k}. \tag{9.13}$$

Furthermore, as $V_\ell = V_{\ell-1} \oplus W_{\ell-1}$, we can also expand s in terms of the bases for $V_{\ell-1}$ and $W_{\ell-1}$ to obtain

$$s = \sum_{k\in I_{\ell-1}} v_{\ell-1,k}\phi_{\ell-1,k} + \sum_{k\in I_{\ell-1}} w_{\ell-1,k}\psi_{\ell-1,k}. \tag{9.14}$$

In (9.14), the first summation corresponds to a coarse approximation of s that lies in $V_{\ell-1}^3$, while the second summation is associated with fine detail (missing from the coarse approximation) that lies in $W_{\ell-1}^3$. The $\{w_{\ell,k}\}$ are called **wavelet coefficients**. Note that the $\{v_{\ell,k}\}$ and $\{w_{\ell,k}\}$ are vectors. The transformation from (9.13) to (9.14) is a wavelet analysis operator. The transformation from (9.14) to (9.13) is a wavelet synthesis operator. By construction, the limit surface M_∞ is in V_ℓ^3 for all $\ell \in \mathbb{Z}^*$. Therefore, the wavelet coefficients in an expansion of M_∞ are always all zero. Thus, subdivision is essentially equivalent to a wavelet synthesis operator with all wavelet coefficients set to the zero vector.

9.3.7 Mallat Algorithm

The preceding mathematics implies that wavelet analysis and synthesis operators are associated with the geometric and topologic refinement processes of subdivision. Given $\{v_{\ell-1,k}\}_{k\in I_{\ell-1}}$ and $\{w_{\ell-1,k}\}_{k\in I_{\ell-1}}$, we can compute the corresponding $\{v_{\ell,k}\}_{k\in I_\ell}$ by a filtering operation (i.e., the application of geometric masks) and topologic refinement. This transformation is a wavelet synthesis operator. Given $\{v_{\ell,k}\}_{k\in I_\ell}$, we can compute the corresponding $\{v_{\ell-1,k}\}_{k\in I_{\ell-1}}$ and $\{w_{\ell-1,k}\}_{k\in I_{\ell-1}}$ by filtering operations (i.e., the application of geometric masks) and topologic refinement (i.e., the inverse of the topologic refinement rule used in the associated subdivision process). This transformation is a wavelet analysis operator.

9.3.8 Applications of Subdivision Wavelets

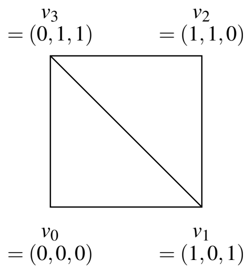
Some applications of subdivision wavelets include: polygon mesh compression, continuous level-of-detail control, compression of functions defined on surfaces, multiresolution editing of surfaces, surface optimization, and numerical solution of integral and differential equations (involving functions defined on a surface of arbitrary topological type).

9.4 Additional Information

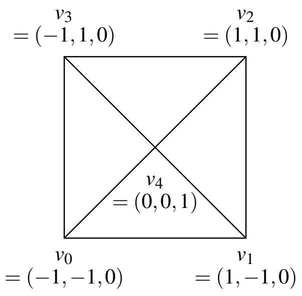
A few additional good references on subdivision include the following: subdivision and wavelets for computer graphics [13], subdivision for geometric modelling [14], and subdivision wavelets [2, 9].

9.5 Problems

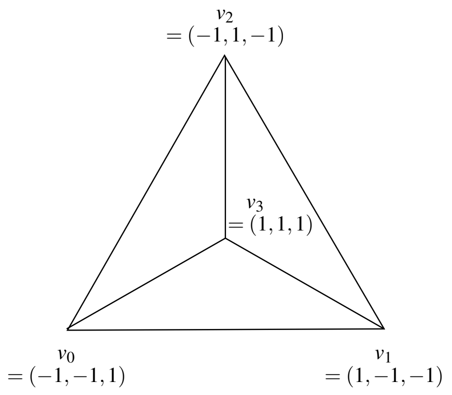
9.1 Compute exactly (i.e., do not use decimal approximations) the refined mesh obtained after applying one level of subdivision to the control mesh shown in the figure below, for each of the following subdivision schemes:
(a) linear;
(b) Loop;
(c) Kobbelt $\sqrt{3}$.



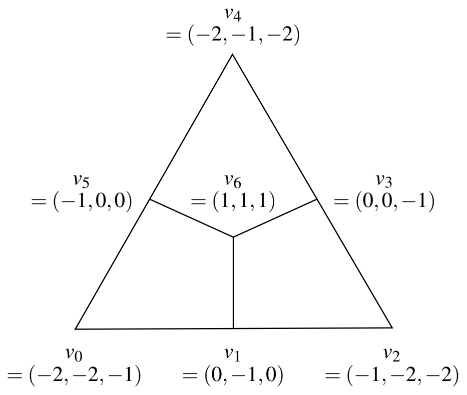
9.2 Compute exactly (i.e., do not use decimal approximations) the refined mesh obtained after applying one level of subdivision to the control mesh shown in the figure below, for each of the following subdivision schemes:
(a) linear;
(b) Loop;
(c) Kobbelt $\sqrt{3}$.



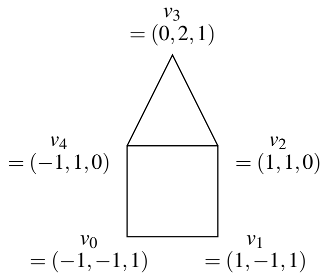
9.3 Compute exactly (i.e., do not use decimal approximations) the refined mesh obtained after applying one level of Catmull-Clark subdivision to the control mesh in each of the figures below.



(a)



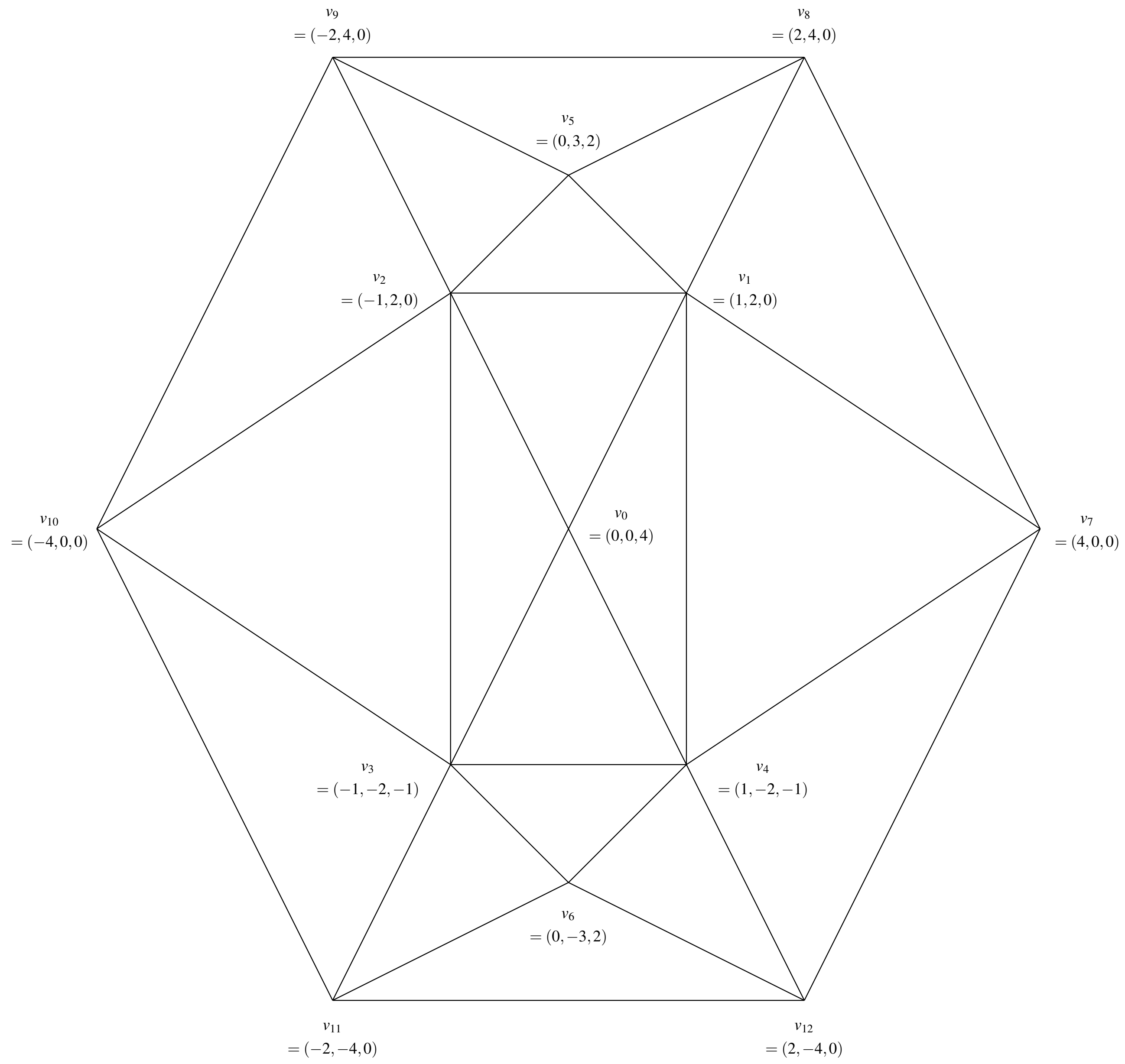
(b)



(c)

9.4 Show that the new vertices introduced by primal triangle quadrisection always have valence six in the interior of the mesh and valence four on the boundary of a mesh. [Hint: Draw diagrams for all of the possible interior/boundary cases that can occur.]

9.5 Consider the application of Loop subdivision to the control mesh shown in the figure below. Let v'_k denote the limit position of the vertex v_k.
(a) Compute exactly (i.e., do not use decimal approximations) v'_7, v'_{11}, v'_0, and v'_1.
(b) Compute exactly (i.e., do not use decimal approximations) a unit vector that is normal to the limit surface at v'_0 and v'_1.



9.6 Bibliography

[1] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[2] I. Daubechies, I. Guskov, P. Schroder, and W. Sweldens. Wavelets on irregular point sets. *Philosophical Transactions of the Royal Society London A*, 357:2397–2413, September 1999.

[3] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[4] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[5] A. Habib and J. Warren. Edge and vertex insertion for a class of C^1 subdivision surfaces. *Computer Aided Geometric Design*, 16(4):223–247, 1999.

[6] H. Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, University of Washington, Seattle, WA, USA, 1994.

[7] L. Kobbelt. $\sqrt{3}$ subdivision. In *Proc. of SIGGRAPH 2000*, pages 103–112, 2000.

[8] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, 1987.

[9] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997.

[10] W. Ma, X. Ma, S.-K. Tso, and Z. Pan. A direct approach for subdivision surface fitting from a dense triangle mesh. *Computer Aided Geometric Design*, 36:525–536, 2004.

[11] J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics*, 16(4):420–431, 1997.

[12] J. Stam and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum*, 22(1):1–7, 2003.

[13] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, USA, 1996.

[14] J. Warren and H. Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann, San Francisco, CA, USA, 2002.

[15] D. Zorin, P. Schroder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 96*, pages 189–192, 1996.

Part V

Applications

Chapter 10

Applications in Signal Processing

10.1 Introduction

Filter banks, wavelets, and other multirate systems (such as transmultiplexers) have many applications in signal processing. In this chapter, we consider a few of these applications, including signal coding, denoising, multiplexing, and adaptive filtering.

10.2 Signal Coding

The discipline of **signal coding** studies alternative representations of signals, usually with some specific purpose in mind. The goal may be to find a representation with less redundancy so that fewer bits are required to encode the signal, or to add redundancy in order to facilitate error detection/correction for information transmitted over noisy channels. The former type of coding is referred to as **signal compression**. Herein, we focus our attention exclusively on signal compression.

There are two general approaches to signal compression. In the first approach, the sample values of the original signal are encoded directly to generate the coded bitstream. Such an approach is often referred to as a **time-domain/spatial-domain approach**. In the second approach, a transformation is first applied to the samples of the original signal and then the transformed data are encoded to produce the compressed bitstream. A system employing this philosophy is said to be **transform based**. For lossy compression, transform-based coders are frequently employed as they tend to have much better performance for fixed complexity. On the other hand, in the case of lossless compression, time-domain/spatial-domain coders have traditionally been employed in the past. In recent years, however, coders employing reversible integer-to-integer transforms have been growing in popularity for lossless coding.

10.2.1 Transform/Subband Coding

The general structure of a transform-based signal compression system is shown in Figure 10.1. In this diagram, x represents the original signal, y denotes the compressed signal, and $\tilde{x}$ represents the reconstructed signal obtained from decompression. The goal, of course, is to design a system so that the coded signal y can be represented with fewer bits than the original signal x. The compressed bitstream may be stored or transmitted. In the case of storage, compression has the benefit of reducing disk or memory requirements, and in transmission scenarios, compression reduces the bandwidth (or time) required to send the data.

Rather than attempt to code the sample values of the original signal directly, a transform-based coder first applies a transform to the signal and then codes the resulting transform coefficients instead. The transform is used in an attempt to obtain coefficients that are easier to code.

Many signals of practical interest are characterized by a spectrum that decreases rapidly with increasing frequency. Image and audio data are good examples of classes of signals with this property. By employing transforms that decompose a signal into its various frequency components, one obtains many small or zero-valued coefficients which

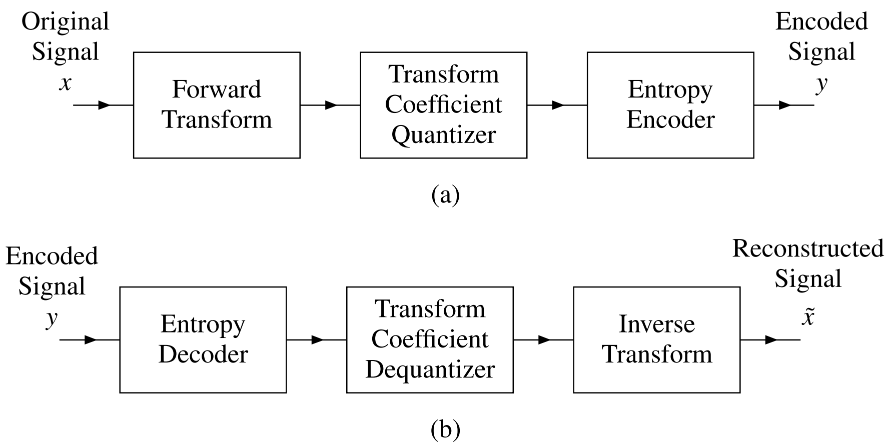


Figure 10.1: General structure of a transform-based signal compression system. The (a) encoder and (b) decoder.

correspond to the high-frequency components of the original signal. Due to the large number of small coefficients, the transformed signal is often easier to code than the original signal itself. As a practical matter, in order for a transform to be useful, it must be effective for a reasonably large class of signals and efficient to compute.

Consider now the compression process embodied by the encoder shown in Figure 10.1(a). First, the transform of the original signal to be coded is calculated. The particular transform employed is chosen so as to pack the energy from the original signal into a small number of coefficients that can be more efficiently coded. Next, the transform coefficients are quantized. Quantization is used to discard transform-coefficient information that is deemed to be insignificant. In the case of lossless compression, no quantization is performed since all transform coefficient information is essential, in which case, the quantized coefficient data is simply identical to the original coefficients. Finally, the quantized coefficients are entropy coded to produce the compressed bitstream. Normally, the entropy coding process is lossless (i.e., does not introduce any information loss). Moreover, the coding process typically exploits a statistical model in order to code symbols with a higher probability of occurrence using fewer bits. In so doing, the size of the compressed bitstream is reduced. Provided that the transform employed is truly invertible, the only potential for information loss is due to coefficient quantization, as the quantized coefficients are coded in a lossless manner.

Now, consider the decompression process embodied by the decoder shown in Figure 10.1(b). The decompression process simply mirrors the process used for compression. First, the compressed bitstream is decoded to obtain the quantized transform-coefficient information. Then, from the quantized data, the dequantization process produces transform coefficients that approximate those obtained during encoding. In the case of lossless coding, the approximation is exact (i.e., has no error). Finally, the inverse of the transform used in the encoder is employed to obtain the reconstructed signal $\tilde{x}$.

In the subband-coding context, the forward and inverse transforms correspond to the analysis and synthesis sides of a PR maximally-decimated filter bank. The forward transform splits the original signal into subbands. The inverse transform combines subband signals to produce a reconstructed signal. Subband coding is used for many types of signals including speech, audio, image, video, and ECG data. The basic idea behind subband coding is that the energy (or information) in the original signal is typically not distributed uniformly with frequency. So, some subband signals have more energy than others. By exploiting the nonuniform distribution of energy across subbands, it is possible to achieve higher coding efficiency. For example, for a transform derived from a one-dimensional M-channel UMD filter bank (with ideal frequency-selective filters), if the original signal were bandlimited to baseband frequencies in the range $[-\frac{\pi}{M}, \frac{\pi}{M}]$, only the first subband signal would be nonzero. Thus, only $\frac{1}{M}$ of the samples would need to be coded. In practice, however, all subbands usually have some energy, but this energy is not evenly distributed, with the lower frequency bands typically having more than other bands. The number of bits allocated for for each subband is then based on this information content. For speech, audio, image, and video data, the number of subbands, filter bandwidths, and bit allocation is chosen to exploit the perceptual properties of the human auditory and visual systems.

10.2.2 Coding Performance Measures

Obviously, in order to evaluate the performance of signal compression systems, we need a way to measure compression. For this purpose, the **compression ratio** metric is often employed and is defined as the ratio of the size of the original signal (in bits) to the size of the compressed signal (in bits). Often, compression is quantified by stating the **bit rate** achieved by compression in bits per sample (i.e., the size of the compressed signal in bits). In some cases, the **normalized bit rate** is more convenient to use, and is defined as the reciprocal of compression ratio. The author prefers to use the normalized bit rate and compression ratio metrics, since in order for the bit rate metric to be meaningful one must also state the initial bit rate. The normalized bit rate and compression ratio metrics, however, are meaningful if stated in isolation.

In the case of lossy compression, the reconstructed signal is only an approximation to the original. The difference between the original and reconstructed signal is referred to as approximation error or **distortion**. Although many metrics exist for quantifying distortion, the **mean-squared error (MSE)** is frequently used. For a signal x and its reconstruction $\tilde{x}$ sampled at the points in Λ, the MSE is simply

$$\mathsf{MSE} = |\Lambda|^{-1} \sum_{k \in \Lambda} (\tilde{x}(k) - x(k))^2 .$$

Frequently, the MSE is expressed in terms of the **peak-signal-to-noise ratio (PSNR)**, which is defined as

$$\mathsf{PSNR} = 20 \log_{10} \left(\frac{2^P - 1}{\sqrt{\mathsf{MSE}}} \right),$$

where P is the number of bits/sample. Note that, as MSE increases, PSNR decreases.

Generally speaking, distortion varies with the amount of compression. In other words, distortion is implicitly a function of rate (i.e., compression ratio). For this reason, plots (or tables) of distortion versus rate are often used to analyze lossy compression performance. Obviously, for any given rate, the lowest possible distortion is desired.

In the case of lossless compression, the reconstructed signal is an exact replica of the original signal. In other words, the distortion is always zero. Since the distortion is zero, we only need to consider the rate achieved when analyzing lossless compression performance. Obviously, the lower the rate, the better is the compression performance.

10.2.3 Coding Gain

In the context of signal coding applications, we are often interested in the energy compacting ability of a filter bank. This ability is often quantified using a measure known as the coding gain. More specifically, the **coding gain** is defined as the ratio between the reconstruction error variance obtained by quantizing a signal directly to that obtained by quantizing the corresponding subband coefficients using an optimal bit allocation strategy. In what follows, we will derive a formula for the coding gain of a filter bank. More information concerning coding gain can also be found in [19].

Suppose that we have a coder based on an m-channel shift-free PR (possibly non-uniform) maximally-decimated filter bank. Recall that the tree-structured filter bank associated with a wavelet system can always be converted from a tree structure to a flat structure via the noble identities. Thus, we consider a subband coder of the form shown in Figure 10.2. Here, Q_k denotes a quantizer with the corresponding bit rate r_k. Let ξ denote the reconstruction error of the filter bank and q_k denote the error introduced by the kth quantizer. In other words, we have

$$\begin{aligned} \xi[n] &= y[n] - x[n] \quad \text{and} \\ q_k[n] &= q_k'[n] - v_k[n]. \end{aligned}$$

We now proceed to derive an expression for the coding gain of the filter bank. To begin, we first make some assumptions about the input signal x and the quantization process embodied by the quantizers $\{Q_k\}$. First, we assume that the input x is a wide-sense stationary (WSS) process with zero mean. Second, we assume that the $\{Q_k\}$ are uniform scalar quantizers and are employed at a high bit rate. This implies that each quantization error q_k is white with zero mean (and WSS) and the $\{q_k\}$ are uncorrelated.

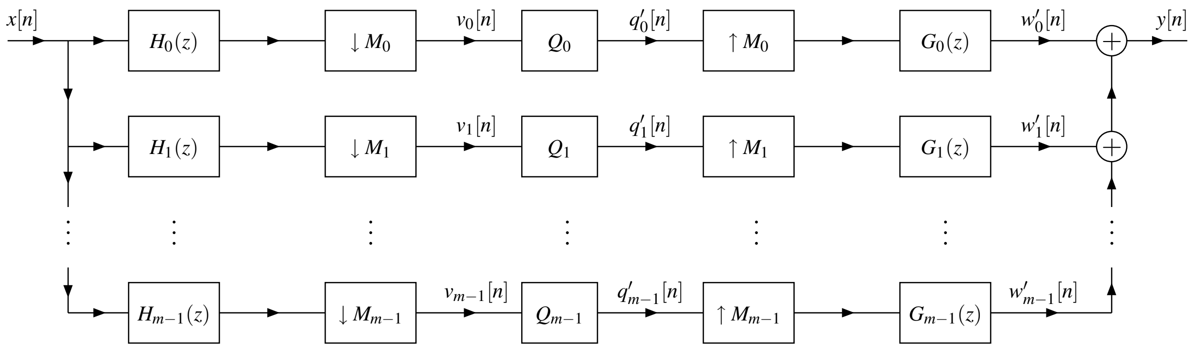


Figure 10.2: Model of subband coder.

VARIANCE OF SUBBAND SIGNALS. We begin by calculating the variance $\sigma_{v_k}^2$ of each subband signal v_k. From the system block diagram, we have

$$\begin{aligned} v_k[n] &= (\downarrow M_k)(x * h_k)[n] \\ &= \sum_{l \in \mathbb{Z}^d} x[M_k n - l] h_k[l]. \end{aligned}$$

Since the WSS property is preserved by downsampling and/or convolution, v_k is WSS. In what follows, let μ_{v_k} denote the mean of v_k (which is zero by assumption). Also, let R_{xx} denote the autocorrelation of x and let ρ_{xx} denote the normalized autocorrelation (i.e., $\rho_{xx} = \sigma_x^{-2} R_{xx}$). Computing the variance $\sigma_{v_k}^2$ of v_k, we obtain

$$\begin{aligned} \sigma_{v_k}^2 &= E\{(v_k[n] - \mu_{v_k})^2\} \\ &= E\{v_k^2[n]\} \\ &= E\left\{ \left(\sum_{l \in \mathbb{Z}^d} x[M_k n - l] h_k[l] \right) \left(\sum_{p \in \mathbb{Z}^d} x[M_k n - p] h_k[p] \right) \right\} \\ &= E\left\{ \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] x[M_k n - l] x[M_k n - p] \right\} \\ &= \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] E\left\{ x[M_k n - l] x[M_k n - p] \right\}. \end{aligned}$$

Since x is WSS, the preceding equation can be written as

$$\begin{aligned} \sigma_{v_k}^2 &= \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] E\left\{ x[0] x[p - l] \right\} \\ &= \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] R_{xx}[p - l] \\ &= \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] \rho_{xx}[p - l] \sigma_x^2. \end{aligned}$$

Thus, $\sigma_{v_k}^2$ and σ_x^2 are related as

$$\sigma_{v_k}^2 = A_k \sigma_x^2 \tag{10.1a}$$

where

$$A_k = \sum_{l \in \mathbb{Z}^d} \sum_{p \in \mathbb{Z}^d} h_k[l] h_k[p] \rho_{xx}[p - l]. \tag{10.1b}$$

In passing, we note that the above expression can be rewritten in the frequency domain as

$$\begin{aligned}A_k &= \sum_{p\in\mathbb{Z}^d} h_k[p] \sum_{l\in\mathbb{Z}^d} h_k[l]\rho_{xx}[p-l] \\ &= \sum_{p\in\mathbb{Z}^d} h_k[p](h_k * \rho_{xx})[p] \\ &= \sum_{p\in\mathbb{Z}^d} \tilde{h}_k[-p]\,(h_k * \rho_{xx}[p]) \\ &= (h_k * \tilde{h}_k * \rho_{xx})[0] \\ &= (2\pi)^{-d} \int_{[-\pi,\pi)^d} \hat{h}_k(\omega)\hat{\tilde{h}}_k(\omega)\hat{\rho}_{xx}(\omega)d\omega \\ &= (2\pi)^{-d} \int_{[-\pi,\pi)^d} \hat{h}_k(\omega)\hat{h}_k^*(\omega)\hat{\rho}_{xx}(\omega)d\omega \\ &= (2\pi)^{-d} \int_{[-\pi,\pi)^d} \hat{\rho}_{xx}(\omega)\left|\hat{h}_k(\omega)\right|^2 d\omega,\end{aligned}$$

where $\tilde{h}_k[n] = h_k^*[-n]$. (Note that h_k is real due to earlier assumptions.)

VARIANCE OF ERROR IN SYNTHESIS FILTER OUTPUTS. In each channel of the filter bank, the subband signal v_k is quantized to produce q'_k. Thus, q'_k effectively consists of two additive components, namely the original subband signal v_k and an error component q_k. That is, $q'_k = v_k + q_k$. Since the synthesis side of the filter bank is linear, the reconstruction error ξ depends only on the error components $\{q_k\}$. Therefore, the reconstruction error variance σ_ξ^2 (which we would eventually like to compute) is determined solely by the $\{q_k\}$. Consequently, in what follows, we need only consider what happens to the $\{q_k\}$ as they pass through the synthesis side of the filter bank.

Let w_k denote the response of the kth upsampler and synthesis filter to the error component input q_k. Now, we compute the variance $\sigma_{w_k}^2$ of the error (due to quantization) in each of the synthesis filter outputs. Let $\alpha_k = |\det M_k|^{-1}$. The system being maximally decimated is equivalent to

$$\sum_{k=0}^{m-1} \alpha_k = 1. \tag{10.2}$$

From the system block diagram, we have

$$\begin{aligned}w_k[n] &= (((\uparrow M_k)q_k) * g_k)[n] \\ &= \sum_{l\in\mathbb{Z}^d} q_k[l]g_k[n - M_k l].\end{aligned}$$

Since q_k is WSS and downsampling by M_k is LPTV with period M_k, w_k is cyclo-WSS with period M_k. So, we calculate the variance of w_k over a single coset (of $\mathbb{Z}^d/(M_k\mathbb{Z}^d)$). Let $\beta \in \mathbb{Z}^d$ and $\gamma \in \mathcal{N}(M_k)$. For fixed γ, we have

$$\begin{aligned}E\left\{w_k^2[M_k\beta+\gamma]\right\} &= E\left\{\left(\sum_{l\in\mathbb{Z}^d} q_k[l]g_k[M_k\beta+\gamma-M_k l]\right)\left(\sum_{p\in\mathbb{Z}^d} q_k[p]g_k[M_k\beta+\gamma-M_k p]\right)\right\} \\ &= E\left\{\sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} q_k[l]q_k[p]g_k[M_k(\beta-l)+\gamma]g_k[M_k(\beta-p)+\gamma]\right\} \\ &= \sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} g_k[M_k(\beta-l)+\gamma]g_k[M_k(\beta-p)+\gamma]E\left\{q_k[l]q_k[p]\right\}.\end{aligned}$$

Now, we employ a change of variable. Let $l' = \beta - l$ and $p' = \beta - p$ so that $l = \beta - l'$ and $p = \beta - p'$. Applying the change of variable and dropping the primes, we obtain

$$E\left\{w_k^2[M_k\beta+\gamma]\right\} = \sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} g_k[M_k l+\gamma]g_k[M_k p+\gamma]E\left\{q_k[\beta-l]q_k[\beta-p]\right\}.$$

Since q_k is WSS, we can write

$$E\left\{w_k^2[M_k\beta+\gamma]\right\} = \sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} g_k[M_kl+\gamma]g_k[M_kp+\gamma]R_{q_kq_k}[p-l].$$

Now, we compute the average variance $\sigma_{w_k}^2$ over a single period of the cyclo-WSS process associated with w_k. We have

$$\sigma_{w_k}^2 = \alpha_k \sum_{\gamma\in\mathcal{N}(M_k)}\sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} g_k[M_kl+\gamma]g_k[M_kp+\gamma]R_{q_kq_k}[l-p].$$

Since q_k is a white-noise process with variance $\sigma_{q_k}^2$, we have that $R_{q_kq_k}[n]=\sigma_{q_k}^2\delta[n]$, and we can write

$$\begin{aligned}
\sigma_{w_k}^2 &= \alpha_k \sum_{\gamma\in\mathcal{N}(M_k)}\sum_{l\in\mathbb{Z}^d}\sum_{p\in\mathbb{Z}^d} g_k[M_kl+\gamma]g_k[M_kp+\gamma]\sigma_{q_k}^2\delta[l-p] \\
&= \alpha_k \sum_{\gamma\in\mathcal{N}(M_k)}\sum_{l\in\mathbb{Z}^d} g_k[M_kl+\gamma]g_k[M_kl+\gamma]\sigma_{q_k}^2 \\
&= \alpha_k \sum_{\gamma\in\mathcal{N}(M_k)}\sum_{l\in\mathbb{Z}^d} g_k^2[M_kl+\gamma]\sigma_{q_k}^2 \\
&= \alpha_k \sum_{l\in\mathbb{Z}^d} g_k^2[l]\sigma_{q_k}^2.
\end{aligned}$$

Thus, $\sigma_{w_k}^2$ and $\sigma_{q_k}^2$ are related as

$$\sigma_{w_k}^2 = B_k\sigma_{q_k}^2 \tag{10.3a}$$

where

$$B_k = \alpha_k \sum_{l\in\mathbb{Z}^d} g_k^2[l]. \tag{10.3b}$$

(Note that $B_k=\alpha_k\left\|g_k\right\|_{l^2}^2$.) In passing, we observe that the $\{w_k\}$ are uncorrelated, since the $\{q_k\}$ are uncorrelated.

VARIANCE OF RECONSTRUCTION ERROR. Now, we proceed to calculate the variance σ_ξ^2 of the reconstruction error ξ. Since, by assumption, we are using uniform scalar quantizers and a high bit rate, we have [17]

$$\sigma_{q_k}^2 = \epsilon_k^2 2^{-2r_k}\sigma_{v_k}^2. \tag{10.4}$$

The reconstruction error ξ is given by

$$\xi[n] = \sum_{k=0}^{m-1} w_k[n].$$

Since the $\{w_k\}$ are uncorrelated, the variance of their sum equals the sum of their variances. So, from (10.3b), (10.4), and (10.1b), we can write

$$\begin{aligned}
\sigma_\xi^2 &= \sum_{k=0}^{m-1}\sigma_{w_k}^2 \\
&= \sum_{k=0}^{m-1} B_k\sigma_{q_k}^2 \\
&= \sum_{k=0}^{m-1} B_k\epsilon_k^2 2^{-2r_k}\sigma_{v_k}^2 \\
&= \sum_{k=0}^{m-1} A_kB_k\epsilon_k^2\sigma_x^2 2^{-2r_k}.
\end{aligned} \tag{10.5}$$

OPTIMAL RECONSTRUCTION ERROR VARIANCE. Now, we must minimize the reconstruction error variance σ_ξ^2 subject to the rate constraint

$$\sum_{k=0}^{m-1} \alpha_k r_k = r, \tag{10.6}$$

where r is the desired rate. To do this, we use the method of Lagrange multipliers. We define the Lagrangian function

$$\begin{aligned} C &= \sigma_\xi^2 + \lambda \left(\sum_{k=0}^{m-1} \alpha_k r_k - r \right) \\ &= \sum_{k=0}^{m-1} A_k B_k \varepsilon_k^2 \sigma_x^2 2^{-2r_k} + \lambda \left(\sum_{k=0}^{m-1} \alpha_k r_k - r \right), \end{aligned}$$

where λ is a Lagrange multiplier. Differentiating the Lagrangian with respect to r_k, we obtain

$$\begin{aligned} \tfrac{\partial C}{\partial r_k} &= A_k B_k \varepsilon_k^2 \sigma_x^2 (-2)(\ln 2) 2^{-2r_k} + \alpha_k \lambda \\ &= -2(\ln 2) A_k B_k \varepsilon_k^2 \sigma_x^2 2^{-2r_k} + \alpha_k \lambda. \end{aligned}$$

(Note that $\frac{\partial}{\partial x} 2^{ax} = a(\ln 2) 2^{ax}$.) Letting $\frac{\partial C}{\partial r_k} = 0$, we have

$$\begin{aligned} & \alpha_k \lambda = 2(\ln 2) A_k B_k \varepsilon_k^2 \sigma_x^2 2^{-2r_k} \\ \Rightarrow \quad & \frac{\alpha_k \lambda}{2(\ln 2) A_k B_k \varepsilon_k^2 \sigma_x^2} = 2^{-2r_k} \\ \Rightarrow \quad & -2r_k = \log_2 \frac{\alpha_k \lambda}{2(\ln 2) A_k B_k \varepsilon_k^2 \sigma_x^2}. \end{aligned}$$

Solving for r_k in the preceding equation, we obtain

$$\begin{aligned} r_k &= -\tfrac{1}{2} \log_2 \frac{\alpha_k \lambda}{2(\ln 2) A_k B_k \varepsilon_k^2 \sigma_x^2} \\ &= -\tfrac{1}{2} \log_2 \frac{\lambda}{2\ln 2} - \tfrac{1}{2} \log_2 \frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2}. \end{aligned} \tag{10.7}$$

Substituting (10.7) into (10.6), we obtain

$$\begin{aligned} r &= \sum_{k=0}^{m-1} \alpha_k \left(-\tfrac{1}{2} \log_2 \frac{\lambda}{2\ln 2} - \tfrac{1}{2} \log_2 \frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2} \right) \\ &= -\tfrac{1}{2} \left(\log_2 \frac{\lambda}{2\ln 2} \right) \sum_{k=0}^{m-1} \alpha_k - \tfrac{1}{2} \sum_{k=0}^{m-1} \alpha_k \log_2 \frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2}. \end{aligned}$$

Since the system is maximally decimated (i.e., (10.2) holds), the preceding equation becomes

$$r = -\tfrac{1}{2} \log_2 \frac{\lambda}{2\ln 2} - \tfrac{1}{2} \sum_{k=0}^{m-1} \log_2 \left(\left(\frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2} \right)^{\alpha_k} \right).$$

Rewriting the sum of logarithms as a logarithm of a product, we obtain

$$\begin{aligned} r &= -\tfrac{1}{2} \log_2 \frac{\lambda}{2\ln 2} - \tfrac{1}{2} \log_2 \left(\prod_{k=0}^{m-1} \left(\frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2} \right)^{\alpha_k} \right) \\ &= -\tfrac{1}{2} \log_2 \frac{\lambda}{2\ln 2} - \tfrac{1}{2} \log_2 \left(\frac{1}{\sigma_x^2} \prod_{k=0}^{m-1} \left(\frac{\alpha_k}{A_k B_k \varepsilon_k^2} \right)^{\alpha_k} \right). \end{aligned}$$

Rearranging, we have

$$\tfrac{1}{2}\log_2 \frac{\lambda}{2\ln 2} = -r - \tfrac{1}{2}\log_2\left(\frac{1}{\sigma_x^2}\prod_{k=0}^{m-1}\left(\frac{\alpha_k}{A_k B_k \varepsilon_k^2}\right)^{\alpha_k}\right).$$

Substituting the preceding equation into (10.7) and writing the difference of logarithms as the logarithm of a quotient, we obtain

$$\begin{aligned} r_k &= r + \tfrac{1}{2}\log_2\left(\frac{1}{\sigma_x^2}\prod_{l=0}^{m-1}\left(\frac{\alpha_l}{A_l B_l \varepsilon_l^2}\right)^{\alpha_l}\right) - \tfrac{1}{2}\log_2 \frac{\alpha_k}{A_k B_k \varepsilon_k^2 \sigma_x^2} \\ &= r + \tfrac{1}{2}\log_2\left(\frac{A_k B_k \varepsilon_k^2}{\alpha_k}\prod_{l=0}^{m-1}\left(\frac{\alpha_l}{A_l B_l \varepsilon_l^2}\right)^{\alpha_l}\right). \end{aligned}$$

Substituting the preceding equation into (10.5), we have

$$\begin{aligned} \sigma_\xi^2 &= \sum_{k=0}^{m-1} A_k B_k \varepsilon_k^2 \sigma_x^2 (2^{r_k})^{-2} \\ &= \sum_{k=0}^{m-1} A_k B_k \varepsilon_k^2 \sigma_x^2 (2^{r})^{-2}\left(\frac{A_k B_k \varepsilon_k^2}{\alpha_k}\prod_{l=0}^{m-1}\left(\frac{\alpha_l}{A_l B_l \varepsilon_l^2}\right)^{\alpha_l}\right)^{-1} \\ &= 2^{-2r}\sum_{k=0}^{m-1}\alpha_k \sigma_x^2 \prod_{l=0}^{m-1}\left(\frac{A_l B_l \varepsilon_l^2}{\alpha_l}\right)^{\alpha_l} \\ &= \sigma_x^2 2^{-2r}\prod_{l=0}^{m-1}\left(\frac{A_l B_l \varepsilon_l^2}{\alpha_l}\right)^{\alpha_l}. \end{aligned}$$

For the PCM case, one can show [17] that the error variance is given by $\varepsilon 2^{-2r}\sigma_x^2$. Suppose that, in the subband case, $\varepsilon_k = \varepsilon$. Then, the subband coding gain is calculated as

$$\begin{aligned} G_{\text{SBC}} &= \frac{\varepsilon^2 2^{-2r}\sigma_x^2}{\sigma_x^2 2^{-2r}\prod_{l=0}^{m-1}\left(\frac{A_l B_l \varepsilon_l^2}{\alpha_l}\right)^{\alpha_l}} \\ &= \frac{1}{\prod_{l=0}^{m-1}\left(\frac{A_l B_l}{\alpha_l}\right)^{\alpha_l}}. \end{aligned}$$

Therefore, the subband coding gain G_{SBC} is given by

$$G_{\text{SBC}} = \prod_{k=0}^{m-1}\left(\frac{\alpha_k}{A_k B_k}\right)^{\alpha_k}, \tag{10.8}$$

where A_k and B_k are given by (10.1b) and (10.3b), respectively.

10.2.4 Additional Reading

For a more detailed treatment of signal coding and data compression beyond that presented herein, the reader is referred to the following: data compression [31], image coding [3, 6, 25, 26], signal coding [17], and JPEG 2000 [8, 27, 29, 36, 37, 41].

10.3 Image Coding

Wavelet transforms have proven extremely effective for transform-based image compression. Since many of the wavelet transform coefficients for a typical image tend to be very small or zero, these coefficients can be easily coded. Thus, wavelet transforms are a useful tool for image compression.

The main advantage of wavelet transforms over other more traditional decomposition methods (like the DFT and DCT) is that the basis functions associated with a wavelet decomposition typically have both long and short support. The basis functions with long support are effective for representing slow variations in an image while the basis functions with short support can efficiently represent sharp transitions (i.e., edges). This makes wavelets ideal for representing signals having mostly low-frequency content mixed with a relatively small number of sharp transitions. With more traditional transforms techniques like the DFT and DCT, the basis functions have support over the entire signal domain, making it difficult to represent both slow variations and edges efficiently.

The wavelet transform decomposes a signal into frequency bands that are equally spaced on a logarithmic scale. The low-frequency bands have small bandwidths, while the high-frequency bands have large bandwidths. This logarithmic behavior of wavelet transforms can also be advantageous. Since human visual perception behaves logarithmically in many respects, the use of wavelet decompositions can sometimes make it easier to exploit characteristics of the human visual system in order to obtain improved subjective lossy compression results.

The wavelet transform is used in many image coding systems. For example, the wavelet transform is employed in the JPEG-2000 image compression standard [16] and the FBI fingerprint compression standard [11]. It is also used in the following coders: embedded block coding with optimal truncation (EBCOT) [35], set partitioning in hierarchical trees (SPIHT) [28], compression with reversible embedded wavelets (CREW) [40], and embedded zerotree wavelet (EZW) [32]. For a detailed overview of the JPEG-2000 image compression standard, the reader is referred to Appendix A.

10.3.1 Coding Performance Measures

Although MSE (or equivalently, PSNR) is frequently employed to measure distortion, it is important to note that MSE does not always correlate well with image quality as perceived by the human visual system. This is particularly true at high compression ratios (i.e., low bit rates). For this reason, one should ideally supplement any objective lossy compression performance measurements with subjective tests to ensure that the objective results are not misleading.

10.3.2 Coding Gain

Earlier, we presented the subband coding gain of a filter bank. The particular formula for the coding gain was given by (10.8). Since a wavelet transform is associated with a filter bank, a wavelet transform also has an associated coding gain. The coding gain formula requires the assumption of a statistical model for the signal being processed by the filter bank. In practice, for the case of images, we typically assume a first-order autoregressive (AR) model. Two variants of this model are commonly used, separable and isotropic. In these cases, ρ_{xx} in (10.8) is given by

$$\rho_{xx}[n] = \begin{cases} \rho^{\|n\|_{l^1}} & \text{for separable model} \\ \rho^{\|n\|_{l^2}} & \text{for isotropic model,} \end{cases}$$

where the correlation coefficient ρ satisfies $|\rho| \leq 1$. Typically, ρ is chosen to satisfy $\rho \in [0.90, 0.95]$.

10.3.3 Choice of Wavelet System

Although wavelet transforms with many different characteristics are possible, orthogonal transforms with symmetric finitely-supported basis functions are ideally most desirable for image compression. Separable transforms are preferred for reasons of computational efficiency. Orthogonality is beneficial as it ensures that transform coefficients do not become unreasonably large and also because it easily facilitates the selection of the most important transform coefficients in the sense of minimizing MSE. Symmetric basis functions are desirable in order to avoid phase distortion as a result of compression. If phase is not preserved, edges and lines can become severely distorted, resulting in poor subjective image quality. Moreover, the symmetric extension method for handling finite-length signals can only be applied to transforms with symmetric basis functions. This is yet another incentive for using transforms with symmetric basis functions. Unfortunately, in the case of two-band wavelet transforms, orthogonality, symmetry, and finite support can only be achieved in the trivial case of the Haar and other Haar-like transforms. For this reason, we usually choose to sacrifice orthogonality, and use biorthogonal transforms instead. In practice, this concession does not pose any serious problems. The choice of the wavelet transform also effects coding gain. Therefore, wavelet transforms with high coding gain are desirable.

Another important consideration in the choice of wavelet transform is the shape of the primal scaling and wavelet functions. The nature of the artifacts introduced by compression are closely related to the shapes of these functions. In a transform-based coding system, we are representing the signal x to be coded as

$$x = \sum_{k \in I} a_k f_k,$$

where the $\{f_k\}_{k \in I}$ are the primal basis functions and the $\{a_k\}_{k \in I}$ are the transform coefficients. Suppose now that we quantize the transform coefficients. We replace the $\{a_k\}_{k \in I}$ with their quantized versions $\{\tilde{a}_k\}_{k \in I}$, where $\tilde{a}_k = a_k + q_k$ and q_k corresponds to quantization error. In so doing, we obtain the quantized signal $\tilde{x}$, where

$$\tilde{x} = \sum_{k \in I} \tilde{a}_k f_k = \sum_{k \in I} (a_k + q_k) f_k.$$

Thus, the error q introduced by quantization is given by

$$q = \sum_{k \in I} q_k f_k.$$

In other words, the quantization error is a weighted sum of the primal basis functions $\{f_k\}_{k \in I}$. For a wavelet transform, the $\{f_k\}_{k \in I}$ are essentially sampled versions of approximations of the primal scaling and wavelet functions. Thus, the basis functions have a shape resembling the primal scaling and wavelet functions. For this reason, the shape of the primal scaling and wavelet functions determine the nature of the artifacts introduced by compression.

To better illustrate the importance of the shape of the primal scaling and wavelet functions, we consider the image coding example shown in Figure 10.3. The image shown in Figure 10.3(a) was compressed in a lossy manner at the same bit rate with three different wavelet transforms (namely, the Haar, twin dragon, and CDF 9/7 transforms) and then decoded to form a reconstructed image. The reconstructed images are shown in Figures 10.3(b) to (d). To allow differences between these images to be more clearly seen, a small region of interest is shown under magnification in Figure 10.4. Clearly, the nature of the artifacts obtained for each transform are quite different. The primal scaling and wavelet functions for the Haar, twin dragon, and CDF 9/7 transforms can be found in Figures 7.6 (on page 329), 7.10 (on page 332), and 7.8 (on page 331), respectively. In the cases of the Haar and twin dragon transforms, the shape of the basis functions is visually apparent in the reconstructed images. Thus, the shape of the primal scaling and wavelet functions are quite important when selecting a wavelet transform to be used in an image coding system.

10.4 Signal Denoising

Another common application of wavelets is in signal denoising [9, 10]. In this application, we have a signal that has been corrupted by noise, and we need to make a best estimate of the original signal (without noise).

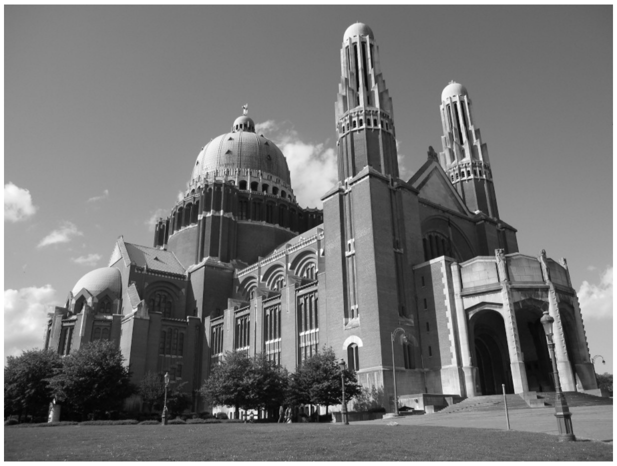
For simplicity, let us consider a simple noise model. A signal x is corrupted by additive noise v to produce a new signal y. This corresponds to the noise model shown in Figure 10.5. Now, by observing only y, we wish to make the best possible estimate of x. In what follows, let us further assume that v is white Gaussian noise with zero mean and variance σ^2.

One way to accomplish the above goal is through the use of a wavelet transform. Suppose that we have the signal x represented in terms of an orthonormal wavelet series expansion. We can apply a wavelet transform, threshold the resulting coefficients, and then perform the inverse wavelet transform. In other words, we have a system with the general structure shown in Figure 10.6. As it turns out, such a system can be quite effective for noise reduction.

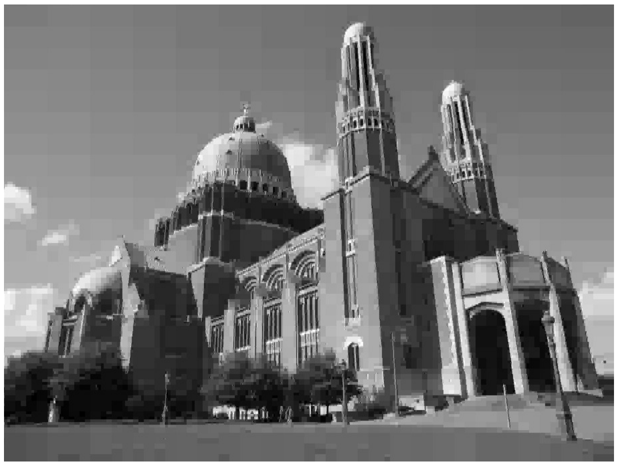
10.4.1 Thresholding Operators

In order to perform thresholding, we need to choose a thresholding operator. Three commonly used thresholding operators are the hard, soft, and hyperbolic thresholding operators, respectively, given by

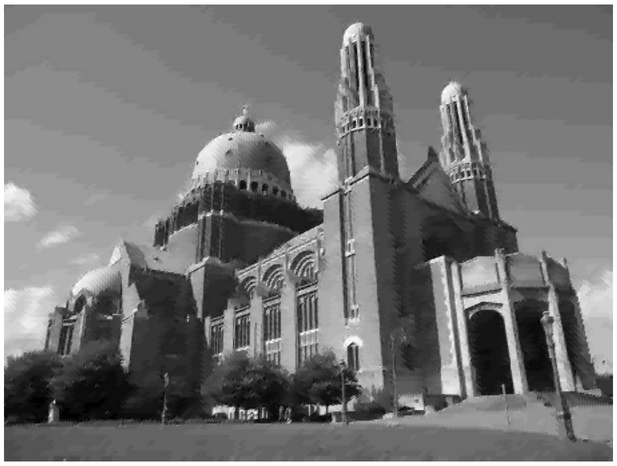
$$T_\varepsilon(x) = \begin{cases} x & \text{if } |x| \geq \varepsilon \\ 0 & \text{otherwise,} \end{cases}$$



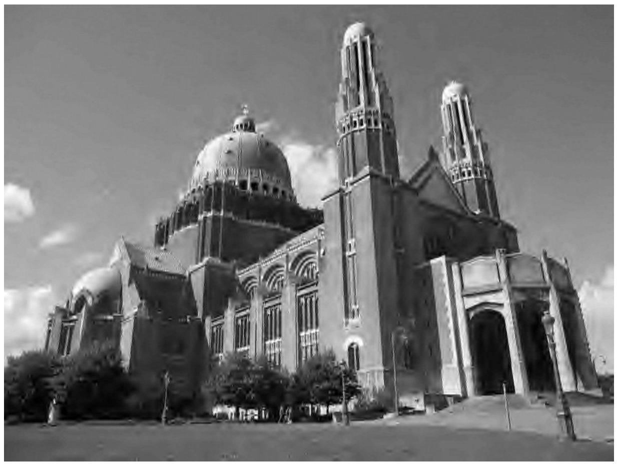
(a)



(b)

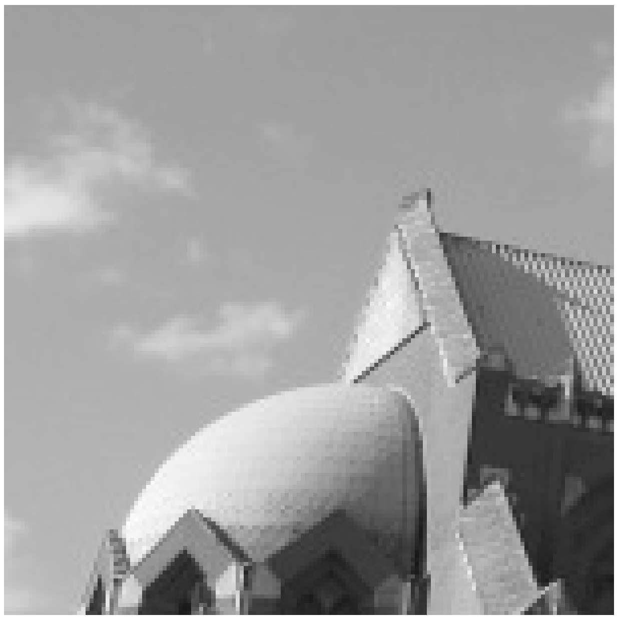


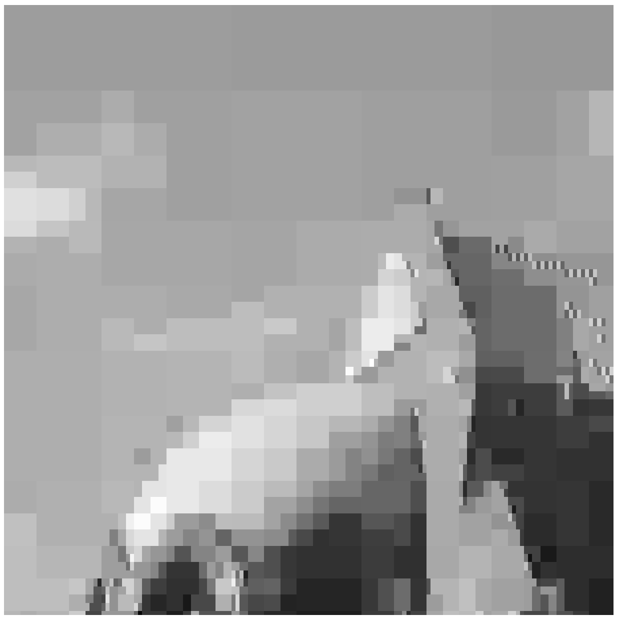
(c)



(d)

Figure 10.3: Lossy compression example. (a) Original image. Lossy reconstructions obtained at a compression ratio of 64 with the (b) Haar (26.55 dB), (c) twin dragon (24.83 dB), and (d) 9/7 wavelet (27.68 dB) transforms.


(a)


(b)

(c)

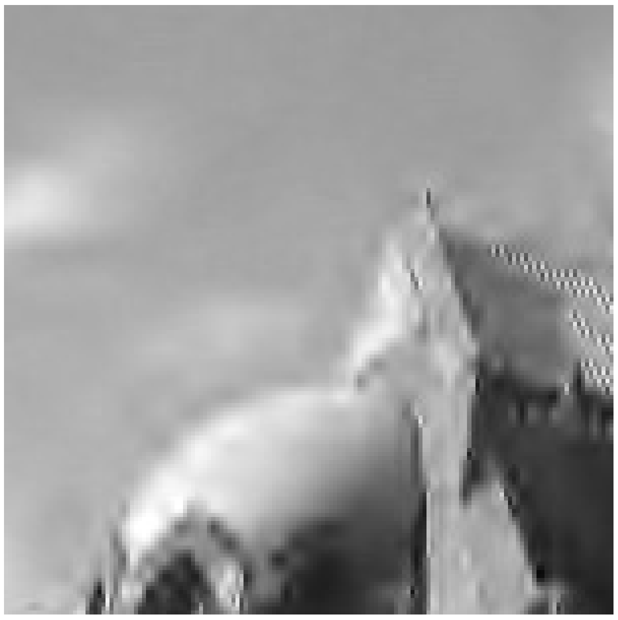

(d)

Figure 10.4: Lossy compression example (continued). Magnified region from (a) original image. Magnified region from lossy reconstructions obtained at a compression ratio of 64 with the (b) Haar, (c) twin dragon, and (d) 9/7 wavelet transforms.

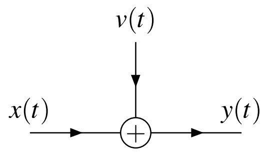


Figure 10.5: Noise model.

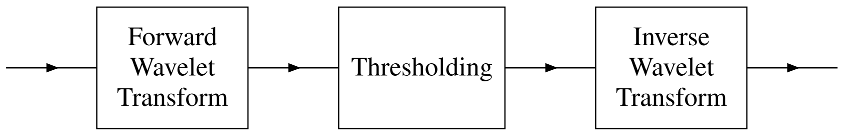


Figure 10.6: Noise reduction system.

$$T_\varepsilon(x) = \begin{cases} (\operatorname{sgn} x)(|x| - \varepsilon) & \text{if } |x| \geq \varepsilon \\ 0 & \text{otherwise,} \end{cases} \quad \text{and}$$

$$T_\varepsilon(x) = \begin{cases} (\operatorname{sgn} x)\sqrt{x^2 - \varepsilon^2} & \text{if } |x| \geq \varepsilon \\ 0 & \text{otherwise,} \end{cases}$$

where ε denotes the threshold value.

10.4.2 Choice of Threshold

A number of different schemes have been proposed for selecting an appropriate threshold value. Here, we consider what is sometimes referred to as the universal threshold scheme. In this case, the threshold is computed as

$$\varepsilon = \sigma\sqrt{2\ln N},$$

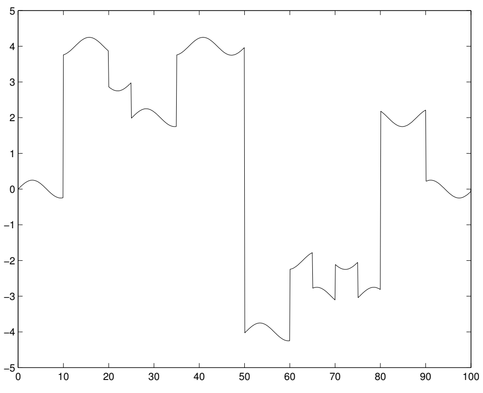
where σ^2 is an estimate of the noise variance and N is the number of samples. An estimate of the noise variance can be calculated using the variance of the highpass wavelet coefficients.

10.4.3 Examples

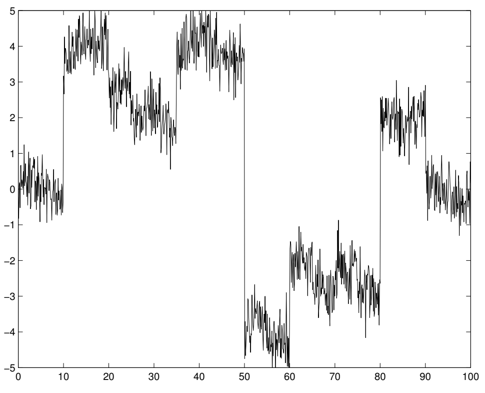
To demonstrate the effectiveness of wavelet-based denoising, we now provide an example. We consider the signal shown in Figure 10.7(a). This signal is corrupted with additive white Gaussian noise having variance 0.25 yielding the signal in Figure 10.7(b). Then, for each of the three thresholding methods, the wavelet-based denoising algorithm is applied using a five-level Daubechies-4 decomposition (i.e., “db4” from MATLAB). This process yields the three signals in Figures 10.7(c), (d), and (e). From these results, we can see that hyperbolic and soft thresholding perform best. In general, hard thresholding tends not to perform as well as soft or hyperbolic thresholding.

10.5 Multiplexing in Communication Systems

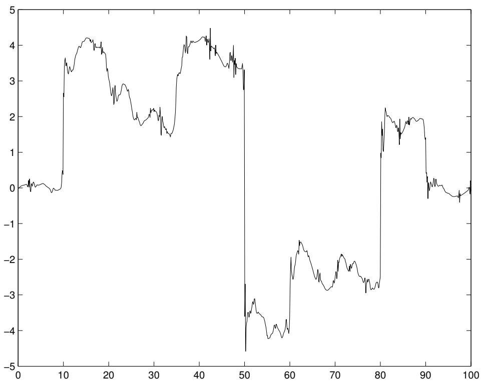
Earlier, in Section 3.11, we introduced a multirate structure called a transmultiplexer. Transmultiplexers have many applications in communications. Often, we have two or more signals that must be transmitted over a single communication channel. Thus, we need to be able to combine multiple signals into a single multiplexed signal for transmission, and then later split them apart. Transmultiplexers allow us to perform such a multiplexing operation. The need for multiplexing can arise in both single-user and multi-user communication systems. In a multi-user system, the signals from several users may need to be multiplexed on a single physical communication channel. For example, in a



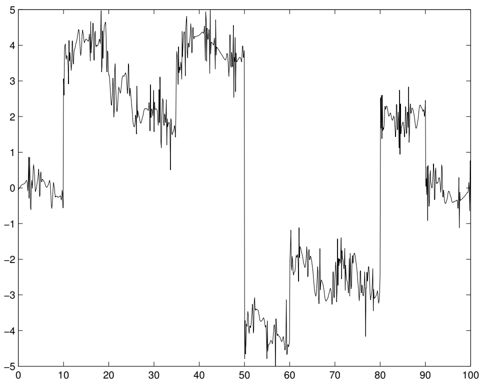
(a)



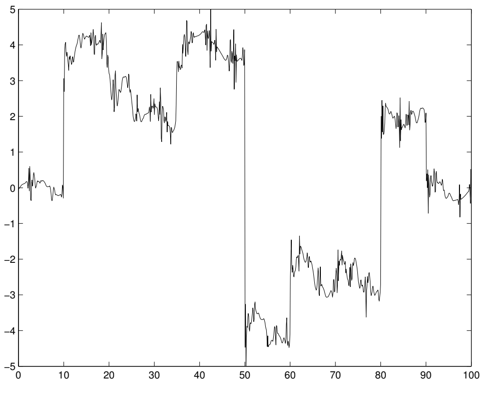
(b)



(c)



(d)



(e)

Figure 10.7: Denoising example. (a) Original signal. (b) Signal corrupted by additive Gaussian white noise with variance 0.25 (PSNR 20.50 dB). Denoised signal obtained using (c) soft thresholding (PSNR 29.42 dB), (d) hard thresholding (PSNR 27.97 dB), and (e) hyperbolic thresholding (PSNR 30.40 dB).

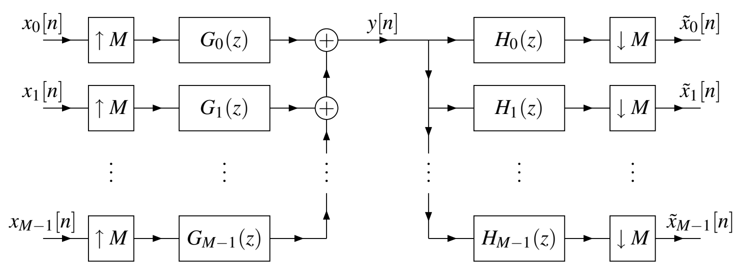


Figure 10.8: An M-channel transmultiplexer.

digital telephone network, multiple voice signals must often be sent over the same physical channel. In a single-user system, multiple data streams from the same user may need to be multiplexed over a single channel. Consequently, multiplexing techniques are of fundamental importance in communications. In the sections that follow, we explore multiplexing in more detail in the context of multirate systems.

10.5.1 Multichannel Communication Systems

One way in which to perform multiplexing is with a transmultiplexer. Consider the M-channel transmultiplexer shown in Figure 10.8, with synthesis filters $\{g_k\}_{k=0}^{M-1}$ and analysis filters $\{h_k\}_{k=0}^{M-1}$. The synthesis side of the transmultiplexer (on the left side of the diagram) multiplexes the M signals $\{x_k\}_{k=0}^{M-1}$ onto the single signal y. The analysis side of the transmultiplexer (on the right side of the diagram) demultiplexes y to produce the M signals $\{\tilde{x}_k\}_{k=0}^{M-1}$. If the transmultiplexer has the PR property, then, for each $k \in \{0,1,\ldots,M-1\}$, $\tilde{x}_k$ is simply a shifted version of x_k. In other words, if the PR property holds, we have a system that can be used to multiplex/demultiplex M signals. This is ideal for many communication applications.

Transmultiplexers are commonly used for frequency division multiple access (FDMA), orthogonal frequency division multiplexing (OFDM), code division multiple access (CDMA), and time division multiple access (TDMA). For FDMA, the synthesis/analysis filters are chosen to be frequency selective. For TDMA, the filters are chosen to have simple one-tap impulses responses. For CDMA, the filters are spread in both time and frequency.

Example 10.1 (FDMA with two-channel transmultiplexer)**.** Suppose that we have two signals x_0 and x_1 that we need multiplex on a single physical communication channel. This can be accomplished by using the two-channel transmultiplexer shown in Figure 10.9, where the frequency responses of the synthesis and analysis filters, for $\omega \in (-\pi,\pi]$, are given by

$$\hat{g}_0(\omega) = 2\chi_{[-\pi/2,\pi/2]}(\omega), \quad \hat{g}_1(\omega) = 2-\hat{g}_0(\omega), \quad \hat{h}_0(\omega) = \chi_{[-\pi/2,\pi/2]}(\omega), \quad \text{and} \quad \hat{h}_1(\omega) = 1-\hat{h}_0(\omega).$$

These frequency responses are shown in Figure 10.10. This system can be shown to have the shift-free PR property.

Suppose that the signals x_0 and x_1 to be transmitted over the communication channel have the frequency spectra shown in Figures 10.11(a) and (b), respectively. Consider the synthesis side of the transmultiplexer, which performs multiplexing. The upsampler outputs u_0 and u_1 have the spectra shown in Figures 10.11(c) and (d), respectively; and the synthesis filter outputs v_0 and v_1 have the spectra shown in Figures 10.11(e) and (f), respectively. Thus, the mutiplexed signal y has the spectrum shown in Figure 10.11(g). Now, consider the analysis side of the transmultiplexer, which performs demultiplexing. The analysis filter outputs w_0 and w_1 have the spectra shown in Figures 10.11(h) and (i), respectively; and the outputs $\tilde{x}_0$ and $\tilde{x}_1$ have the spectra shown in Figures 10.11(j) and (k), respectively. Clearly, $\tilde{x}_0 = x_0$ and $\tilde{x}_1 = x_1$. So, shift-free PR is achieved.

10.5.2 Multicarrier Modulation

Transmultiplexers can also be used in an approach called **multicarrier modulation (MCM)**. With MCM, the data to be transmitted is split into several streams and then used to modulate several carriers. Thus, instead of transmitting a single wideband signal over one channel, we transmit a set of narrowband signals. In effect, MCM provides a means for partitioning a single physical channel into multiple logical subchannels.

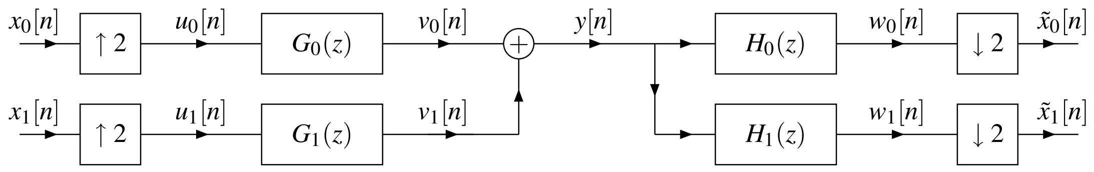


Figure 10.9: A two-channel transmultiplexer.

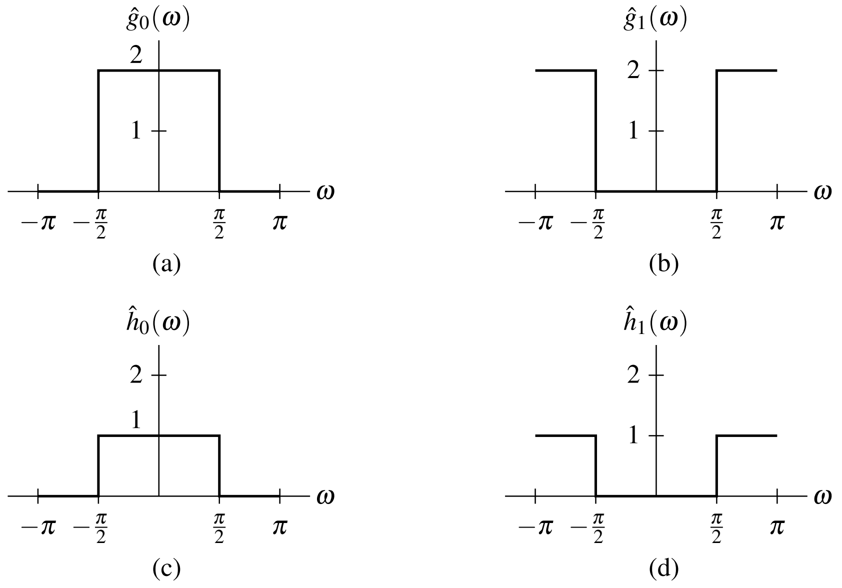


Figure 10.10: Synthesis and analysis filter frequency responses. Frequency response of the (a) synthesis lowpass filter, (b) synthesis highpass filter, (c) analysis lowpass filter, and (d) analysis highpass filter.

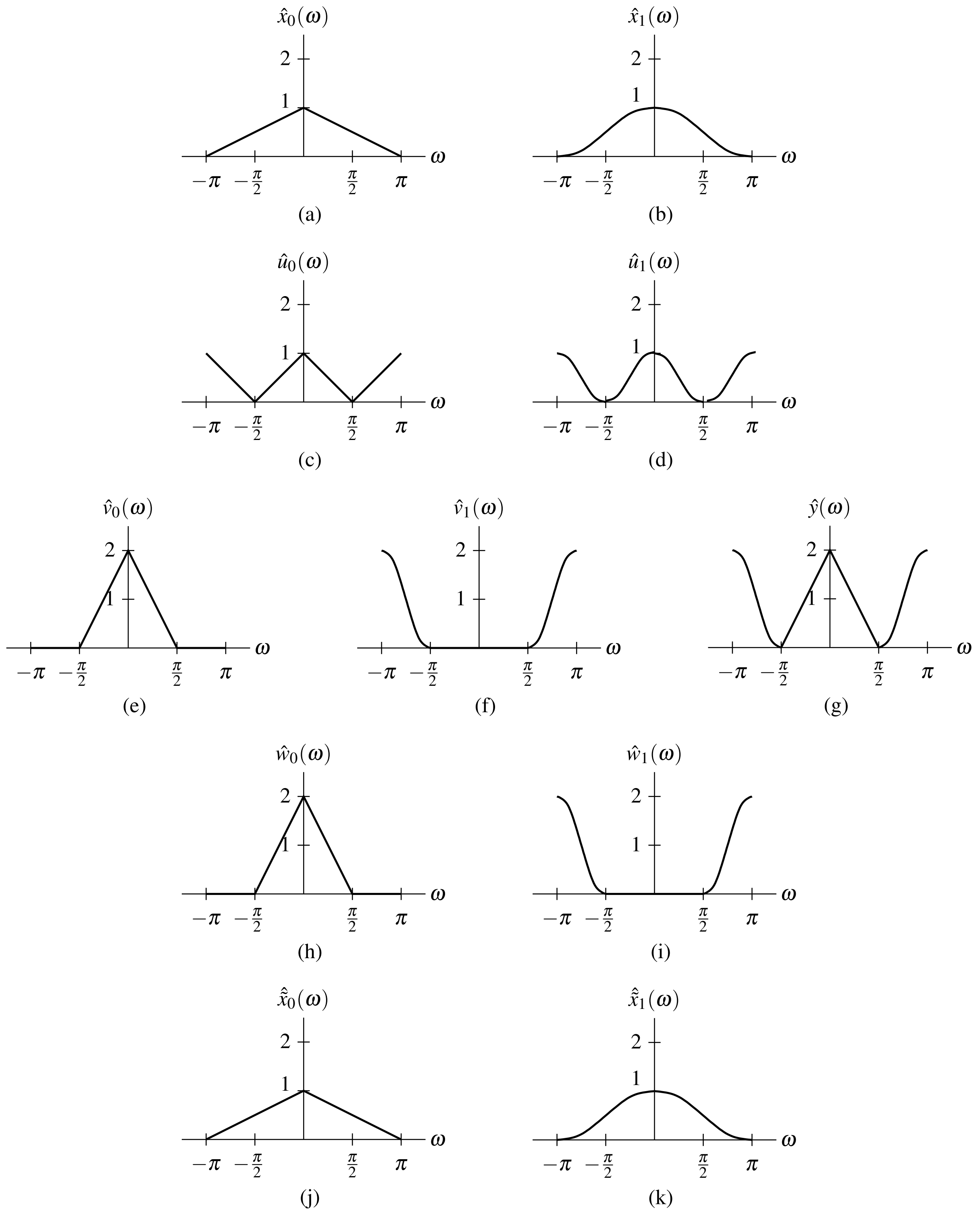


Figure 10.11: Two-channel transmultiplexer FDMA example. Frequency spectra of the (a) first and (b) second inputs, (c) first and (d) second upsampler outputs, (e) first and (f) second synthesis filter outputs, (g) multiplexed signal, (h) first and (i) second analysis filter outputs, and (j) first and (k) second downsampler outputs.

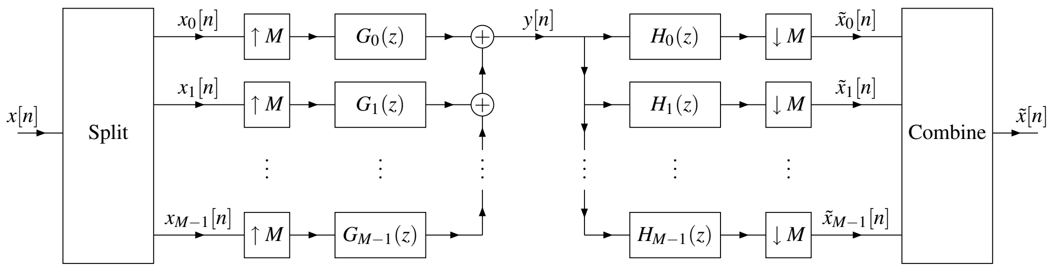


Figure 10.12: Multicarrier modulation system.

The general structure of a MCM system is shown in Figure 10.12. The system works as follows. First, the input signal x to be transmitted is split into M distinct data streams $\{x_k\}_{k=0}^{M-1}$. Then, all of the data streams are multiplexed together into a single signal y that is sent over the communication channel. Next, the received signal y undergoes demultiplexing to produce the data streams $\{\tilde{x}_k\}_{k=0}^{M-1}$, and the data streams are combined together to form the output signal $\tilde{x}$. Provided that the system is designed properly (e.g., with a PR transmultiplexer), the signal $\tilde{x}$ will equal x up to a time shift.

MCM is used in many applications. It was first employed for military HF radios in the late 1950s and early 1960s. Since the 1990s, the use of MCM has become widespread. It is employed for digital TV broadcasting in Europe [30], numerous wireless LAN/MAN standards such as 802.11a [13], 802.11g [14], and 802.16a [15]. Also, it is used for high-speed data transmission over the twisted-pair channel of digital subscriber lines (e.g., ADSL) [7].

MCM has several advantages over the classical single-carrier approach. First, MCM can adapt the data rates of the subchannels to the channel-noise characteristics (which is beneficial since channel noise is usually colored). For example, more important data can be transmitted in subchannels with less noise, and transmitting in extremely noisy subchannels can be avoided altogether. Second, MCM facilitates more effective coding schemes, improving robustness to transmission errors. Third, with MCM, channel effects can be more easily modelled. For example, since each subchannel has a much smaller bandwidth than the single wideband channel, it may be possible to model the transfer function of single (narrowband) subchannel simply as a constant.

Some additional references on MCM include: [18], and [5].

10.5.3 Additional Reading

A large body of literature exists on the use of transmultiplexers for communications. A few additional references include: [1], [38], [2, Section 7.3], and [34, pp. 391–392].

10.6 Adaptive Systems

Filter banks and multirate systems are often used in adaptive filtering applications, such as inverse filtering, room acoustics modelling, echo cancellation, and equalization. In such schemes, adaptive filtering is performed in the subbands, at a lower sampling rate. This use of a lower sampling rate has obvious advantages. Also, some subbands may be ignored if they contain little information. The advantages of using multirate structures for adaptive filtering often include: smaller filter lengths, lower implementation complexity, and faster convergence. A good tutorial on multirate adaptive filtering is provided in [33]. Some other additional references include: acoustic echo cancellation [12], adaptive filtering [20], channel estimation and equalization [24], and NLMS algorithm [22].

10.7 Bibliography

[1] A. N. Akansu, P. Duhamel, X. Lin, and M. de Courville. Orthogonal transmultiplexers in communication: A review. *IEEE Trans. on Signal Processing*, 46(4):979–995, April 1998.

[2] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*. Academic Press, San Diego, CA, USA, 2nd edition, 2001.

[3] R. Ansari, E. Ceran, and N. Memon. Near-lossless image compression techniques. In *Proc. of SPIE: Visual Communications and Image Processing*, volume 3309, pages 731–742, January 1998.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–220, April 1992.

[5] J. Bingham. Multicarrier modulation for data transmission: An idea whose time has come. *IEEE Communications Magazine*, pages 5–14, May 1990.

[6] B. Carpentieri, M. J. Weinberger, and G. Seroussi. Lossless compression of continuous-tone images. *Proc. of IEEE*, 88(11):1797–1809, November 2000.

[7] G. Cherubini, E. Eleftheriou, S. Olcer, and J. M. Cioffi. Filter bank modulation techniques for very high-speed digital subscriber lines. *IEEE Communications Magazine*, 38(5):98–104, May 2000.

[8] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG 2000 still image coding system: An overview. *IEEE Trans. on Consumer Electronics*, 46(4):1103–1127, November 2000.

[9] D. L. Donoho. De-noising by soft-thesholding. *IEEE Trans. on Information Theory*, 41(3):613–627, May 1995.

[10] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shinkage. *Biometrika*, 81(3):425–455, August 1994.

[11] Federal Bureau of Investigation, Washington, DC, USA. *WSQ Gray-Scale Fingerprint Image Compression Specification, IAFIS-IC-0110(V3)*, December 1997.

[12] A. Gilloire and M. Vetterli. Adaptive filtering in subbands with critical sampling: Analysis, experiments, and application to acoustic echo cancellation. *IEEE Trans. on Signal Processing*, 40(8):1862–1875, August 1992.

[13] IEEE. *IEEE Std 802.11a-1999 — Supplement to IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band*, 1999.

[14] IEEE. *IEEE Std 802.11g-2003 — IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2003.

[15] IEEE. *IEEE Std 802.16a-2003 — IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems — Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 2-11 GHz*, 2003.

[16] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.

[17] N. S. Jayant and P. Noll. *Digital Coding of Waveforms: Principals and Applications to Speech and Video*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1984.

[18] S. G. Kang and E. K. Joo. OFDM system with linear-phase transmultiplexer. *Electronics Letters*, 34(13):1292–1293, June 1998.

[19] J. Katto and Y. Yasuda. Performance evaluation of subband coding and optimization of its filter coefficients. In *SPIE VCIP*, volume 1605, pages 95–106, 1991.

[20] S. Kim, C. D. Yoo, and T. Q. Nguyen. Alias-free subband adaptive filtering with critical sampling. *IEEE Trans. on Signal Processing*, 56(5):1894–1904, May 2008.

[21] C.-W. King and C.-A. Lin. A unified approach to scrambling filter design. *IEEE Trans. on Signal Processing*, 43(8):1753–1765, August 1995.

[22] K. A. Lee and W. S. Gan. Improving convergence of the NLMS algorithm using constrained subband updates. *IEEE Signal Processing Letters*, 11(9):736–739, September 2004.

[23] A. S. Lewis and G. Knowles. Image compression using the 2-D wavelet transform. *IEEE Trans. on Image Processing*, 1(2):244–250, April 1992.

[24] D. Marelli and M. Fu. A subband approach to channel estimation and equalization for DMT and OFDM systems. *IEEE Trans. on Communications*, 53(11):1850–1858, November 2005.

[25] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Kluwer Academic, New York, NY, USA, 1992.

[26] M. Rabbani and P. W. Jones. *Digital Image Compression Techniques*. SPIE Optical Engineering Press, Bellingham, WA, USA, 1991.

[27] M. Rabbani and R. Joshi. An overview of the JPEG2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3–48, January 2002.

[28] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[29] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi. JPEG 2000 performance evaluation and assessment. *Signal Processing: Image Communication*, 17(1):113–130, January 2002.

[30] H. Sari, G. Karam, and I. Jeanclaude. Transmission techniques for digital terrestrial TV broadcasting. *IEEE Communications Magazine*, pages 100–109, February 1995.

[31] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, San Francicso, CA, USA, 3rd edition, 2006.

[32] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[33] J. J. Shynk. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, 9:14–37, January 1992.

[34] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, USA, 1996.

[35] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Trans. on Image Processing*, 9(7):1158–1170, July 2000.

[36] D. S. Taubman and M. W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic, Boston, MA, USA, 2002.

[37] D. S. Taubman and M. W. Marcellin. JPEG2000: Standard for interactive imaging. *Proc. of IEEE*, 90(8):1336–1357, August 2002.

[38] P. P. Vaidyanathan. Filter banks in digital communications. *IEEE Circuits and Systems Magazine*, 1(2):4–25, 2001.

[39] J. D. Villasenor, B. Belzer, and J. Liao. Wavelet filter evaluation for image compression. *IEEE Trans. on Image Processing*, 4(8):1053–1060, August 1995.

[40] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek. CREW: Compression with reversible embedded wavelets. In *Proc. of IEEE Data Compression Conference*, pages 212–221, Snowbird, UT, USA, March 1995.

[41] W. Zeng, S. Daly, and S. Lei. An overview of the visual optimization tools in JPEG 2000. *Signal Processing: Image Communication*, 17(1):85–104, January 2002.

Chapter 11

Applications in Geometry Processing

11.1 Introduction

Subdivision surfaces and wavelets have many applications in geometry processing. In this chapter, we briefly explore the how subdivision surfaces are employed in the rendering of geometric objects.

11.2 Practical Use of Subdivision: Rendering

In addition to the vertices of a mesh and their connectivity, to render a mesh, we need surface normals. In practice, given a control mesh, we would typically render the corresponding surface using a process like the following:

1. Apply several iterations of subdivision to the control mesh in order to produce a refined mesh.
2. Push the vertices in the refined mesh to their limit positions using position masks.
3. Using tangent masks, compute two tangent vectors at each point on the limit surface corresponding to the limit position of a vertex.
4. From the two tangent vectors, compute a corresponding surface normal (via a vector cross product).
5. Equipped with the vertex positions and normals from above, we then proceed to render the surface. We might, for example, use a rendering engine such as OpenGL in conjunction with a Phong lighting model.

As for the specific details as to how the above process can be accomplished in a real software application, the reader is referred to the material on the CGAL [1, 3] and OpenGL [2, 4] libraries in Appendices C and D, respectively.

11.3 Bibliography

[1] Computational geometry algorithms library (CGAL) home page, 2012. `http://www.cgal.org`.

[2] The OpenGL web site, 2013. `http://www.opengl.org`.

[3] *CGAL User and Reference Manual: All Parts (Release 4.0.2)*, July 2012. Available online from `http://www.cgal.org`.

[4] M. Segal and K. Akeley, editors. *The OpenGL Graphics System: A Specification (Version 4.4 (Core Profile))*. July 2013.

Part VI

Appendices

Appendix A

The JPEG-2000 Image Compression Standard

Overview

JPEG 2000, a new international standard for still image compression, is discussed at length. A high-level introduction to the JPEG-2000 standard is given, followed by a detailed technical description of the JPEG-2000 Part-1 codec.

A.1 Introduction

Digital imagery is pervasive in our world today. Consequently, standards for the efficient representation and interchange of digital images are essential. To date, some of the most successful still image compression standards have resulted from the ongoing work of the Joint Photographic Experts Group (JPEG). This group operates under the auspices of Joint Technical Committee 1, Subcommittee 29, Working Group 1 (JTC 1/SC 29/WG 1), a collaborative effort between the International Organization for Standardization (ISO) and International Telecommunication Union Standardization Sector (ITU-T). Both the JPEG [22, 35, 46] and JPEG-LS [13, 24, 47] standards were born from the work of the JPEG committee. More recently, the JPEG committee has developed a new standard known as JPEG 2000 (i.e., ISO/IEC 15444).

In this appendix, we provide a detailed technical description of the JPEG-2000 Part-1 codec, in addition to a brief overview of the JPEG-2000 standard. This exposition is intended to serve as a reader-friendly starting point for those interested in learning about JPEG 2000. Although many details are included in our presentation, some details are necessarily omitted. The reader should, therefore, refer to the standard [25] before attempting an implementation. The JPEG-2000 codec realization in the JasPer software [2, 4, 5] (developed by the author of this book) may also serve as a practical guide for implementors. (See Section A.5 for more information about JasPer.) The reader may also find [16, 36, 45] to be useful sources of information on the JPEG-2000 standard.

The remainder of this appendix is structured as follows. Section A.2 begins with a overview of the JPEG-2000 standard. This is followed, in Section A.3, by a detailed description of the JPEG-2000 Part-1 codec. Finally, we conclude with some closing remarks in Section A.4. Throughout our presentation, a basic understanding of image coding is assumed.

A.2 JPEG 2000

The JPEG-2000 standard supports lossy and lossless compression of single-component (e.g., grayscale) and multi-component (e.g., color) imagery. In addition to this basic compression functionality, however, numerous other features are provided, including: 1) progressive recovery of an image by fidelity or resolution; 2) region of interest coding, whereby different parts of an image can be coded with differing fidelity; 3) random access to particular regions of an image without needing to decode the entire code stream; 4) a flexible file format with provisions for specifying opacity information and image sequences; and 5) good error resilience. Due to its excellent coding performance and many attractive features, JPEG 2000 has a very large potential application base. Some possible application areas include: image archiving, Internet, web browsing, document imaging, digital photography, medical imaging, remote sensing, and desktop publishing.

A.2.1 Why JPEG 2000?

Work on the JPEG-2000 standard commenced with an initial call for contributions [21] in March 1997. The purpose of having a new standard was twofold. First, it would address a number of weaknesses in the original JPEG standard. Second, it would provide a number of new features not available in the original JPEG standard. The preceding points led to several key objectives for the new standard, namely that it should: 1) allow efficient lossy and lossless compression within a single unified coding framework, 2) provide superior image quality, both objectively and subjectively, at low bit rates, 3) support additional features such as rate and resolution scalability, region of interest coding, and a more flexible file format, and 4) avoid excessive computational and memory complexity. Undoubtedly, much of the success of the original JPEG standard can be attributed to its royalty-free nature. Consequently, considerable effort has been made to ensure that a minimally-compliant JPEG-2000 codec can be implemented free of royalties.

A.2.2 Structure of the Standard

The JPEG-2000 standard is comprised of numerous parts, with the parts listed in Table A.1 being defined at the time of this writing. For convenience, we will refer to the codec defined in Part 1 (i.e., [25]) of the standard as the baseline codec. The baseline codec is simply the core (or minimal functionality) JPEG-2000 coding system. Part 2 (i.e., [27]) describes extensions to the baseline codec that are useful for certain “niche” applications, while Part 3 (i.e., [28]) defines extensions for intraframe-style video compression. Part 5 (i.e., [30]) provides two reference software implementations of the Part-1 codec, and Part 4 (i.e., [29]) provides a methodology for testing implementations for compliance with the standard. In this appendix, we will, for the most part, limit our discussion to the baseline codec. Some of the extensions included in Part 2 will also be discussed briefly. Unless otherwise indicated, our exposition considers only the baseline system.

For the most part, the JPEG-2000 standard is written from the point of view of the decoder. That is, the decoder is defined quite precisely with many details being normative in nature (i.e., required for compliance), while many parts of the encoder are less rigidly specified. Obviously, implementors must make a very clear distinction between normative and informative clauses in the standard. For the purposes of our discussion, however, we will only make such distinctions when absolutely necessary.

A.3 JPEG-2000 Codec

Having briefly introduced the JPEG-2000 standard, we are now in a position to begin examining the JPEG-2000 codec in detail. The codec is based on wavelet/subband coding techniques [8, 33]. It handles both lossy and lossless compression using the same transform-based framework, and borrows heavily on ideas from the embedded block coding with optimized truncation (EBCOT) scheme [42, 43, 44]. In order to facilitate both lossy and lossless coding in an efficient manner, reversible integer-to-integer [3, 12, 19] and nonreversible real-to-real transforms are employed. To code transform data, the codec makes use of bit-plane coding techniques. For entropy coding, a context-based adaptive binary arithmetic coder [48] is used—more specifically, the MQ coder from the JBIG2 standard [23]. Two levels of syntax are employed to represent the coded image: a code stream and file format syntax. The code stream syntax is similar in spirit to that used in the original JPEG standard.

The remainder of Section A.3 is structured as follows. First, Sections A.3.1 to A.3.3, discuss the source image model and how an image is internally represented by the codec. Next, Section A.3.4 examines the basic structure of the codec. This is followed, in Sections A.3.5 to A.3.13 by a detailed explanation of the coding engine itself. Next, Sections A.3.14 and A.3.15 explain the syntax used to represent a coded image. Finally, Section A.3.16 briefly describes some of the extensions included in Part 2 of the standard.

Table A.1: Parts of the standard

Part	Title	Purpose	Document
1	Core coding system	Specifies the core (or minimal functionality) JPEG-2000 codec.	[25]
2	Extensions	Specifies additional functionalities that are useful in some applications but need not be supported by all codecs.	[27]
3	Motion JPEG 2000	Specifies extensions to JPEG-2000 for intraframe-style video compression.	[28]
4	Conformance testing	Specifies the procedure to be employed for compliance testing.	[29]
5	Reference software	Provides sample software implementations of the standard to serve as a guide for implementors.	[30]
6	Compound image file format	Defines a file format for compound documents.	[31]
8	Secure JPEG 2000	Defines mechanisms for conditional access, integrity/authentication, and intellectual property rights protection.	
9	Interactivity tools, APIs and protocols	Specifies a client-server protocol for efficiently communicating JPEG-2000 image data over networks.	
10	3D and floating-point data	Provides extensions for handling 3D (e.g., volumetric) and floating-point data.	
11	Wireless	Provides channel coding and error protection tools for wireless applications.	
12	ISO base media file format	Defines a common media file format used by Motion JPEG 2000 and MPEG 4.	[26]
13	Entry-level JPEG 2000 encoder	Specifies an entry-level JPEG-2000 encoder.	

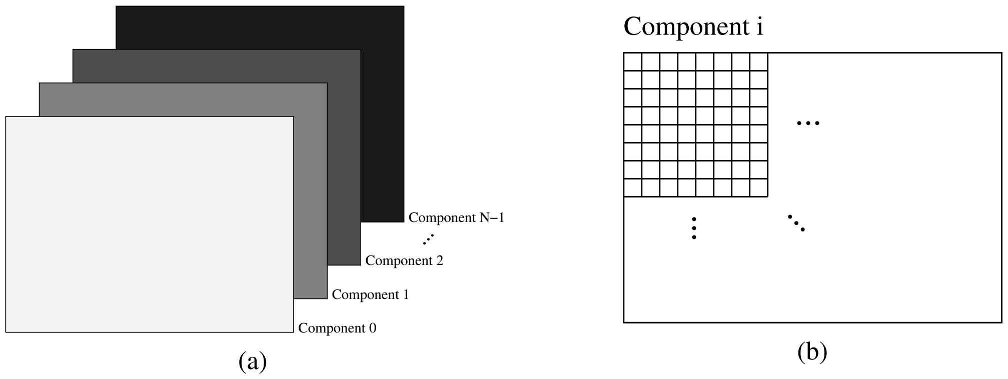


Figure A.1: Source image model. (a) An image with N components. (b) Individual component.

A.3.1 Source Image Model

Before examining the internals of the codec, it is important to understand the image model that it employs. From the codec's point of view, an image is comprised of one or more components (up to a limit of 2^{14}), as shown in Figure A.1(a). As illustrated in Figure A.1(b), each component consists of a rectangular array of samples. The sample values for each component are integer valued, and can be either signed or unsigned with a precision from 1 to 38 bits/sample. The signedness and precision of the sample data are specified on a per-component basis.

All of the components are associated with the same spatial extent in the source image, but represent different spectral or auxiliary information. For example, a RGB color image has three components with one component representing each of the red, green, and blue color planes. In the simple case of a grayscale image, there is only one component, corresponding to the luminance plane. The various components of an image need not be sampled at the same resolution. Consequently, the components themselves can have different sizes. For example, when color images are represented in a luminance-chrominance color space, the luminance information is often more finely sampled than the chrominance data.

A.3.2 Reference Grid

Given an image, the codec describes the geometry of the various components in terms of a rectangular grid called the reference grid. The reference grid has the general form shown in Figure A.2. The grid is of size $\mathrm{Xsiz} \times \mathrm{Ysiz}$ with the origin located at its top-left corner. The region with its top-left corner at $(\mathrm{XOsiz}, \mathrm{YOsiz})$ and bottom-right corner at $(\mathrm{Xsiz}-1, \mathrm{Ysiz}-1)$ is called the image area, and corresponds to the picture data to be represented. The width and height of the reference grid cannot exceed $2^{32}-1$ units, imposing an upper bound on the size of an image that can be handled by the codec.

All of the components are mapped onto the image area of the reference grid. Since components need not be sampled at the full resolution of the reference grid, additional information is required in order to establish this mapping. For each component, we indicate the horizontal and vertical sampling period in units of the reference grid, denoted as XRsiz and YRsiz, respectively. These two parameters uniquely specify a (rectangular) sampling grid consisting of all points whose horizontal and vertical positions are integer multiples of XRsiz and YRsiz, respectively. All such points that fall within the image area, constitute samples of the component in question. Thus, in terms of its own coordinate system, a component will have the size $\left(\left\lceil \frac{\mathrm{Xsiz}}{\mathrm{XRsiz}} \right\rceil - \left\lceil \frac{\mathrm{XOsiz}}{\mathrm{XRsiz}} \right\rceil\right) \times \left(\left\lceil \frac{\mathrm{Ysiz}}{\mathrm{YRsiz}} \right\rceil - \left\lceil \frac{\mathrm{YOsiz}}{\mathrm{YRsiz}} \right\rceil\right)$ and its top-left sample will correspond to the point $\left(\left\lceil \frac{\mathrm{XOsiz}}{\mathrm{XRsiz}} \right\rceil, \left\lceil \frac{\mathrm{YOsiz}}{\mathrm{YRsiz}} \right\rceil\right)$. Note that the reference grid also imposes a particular alignment of samples from the various components relative to one another.

From the diagram, the size of the image area is $(\mathrm{Xsiz}-\mathrm{XOsiz}) \times (\mathrm{Ysiz}-\mathrm{YOsiz})$. For a given image, many combinations of the Xsiz, Ysiz, XOsiz, and YOsiz parameters can be chosen to obtain an image area with the same size. Thus, one might wonder why the XOsiz and YOsiz parameters are not fixed at zero while the Xsiz and Ysiz parameters are set to the size of the image. As it turns out, there are subtle implications to changing the XOsiz and YOsiz parameters (while keeping the size of the image area constant). Such changes affect codec behavior in several

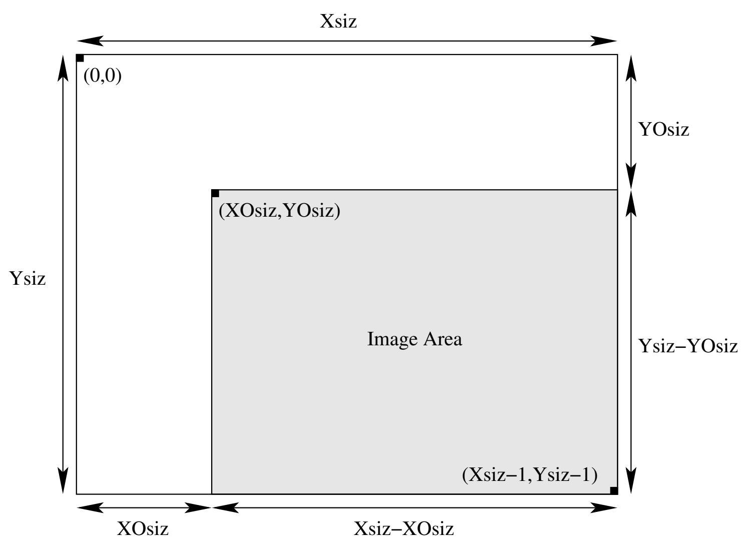


Figure A.2: Reference grid.

important ways, as will be described later. This behavior allows a number of basic operations to be performed more efficiently on coded images, such as cropping, horizontal/vertical flipping, and rotation by an integer multiple of 90 degrees.

A.3.3 Tiling

In some situations, an image may be quite large in comparison to the amount of memory available to the codec. Consequently, it is not always feasible to code the entire image as a single atomic unit. To solve this problem, the codec allows an image to be broken into smaller pieces, each of which is independently coded. More specifically, an image is partitioned into one or more disjoint rectangular regions called tiles. As shown in Figure A.3, this partitioning is performed with respect to the reference grid by overlaying the reference grid with a rectangular tiling grid having horizontal and vertical spacings of XTsiz and YTsiz, respectively. The origin of the tiling grid is aligned with the point $(\mathrm{XTOsiz}, \mathrm{YTOsiz})$. Tiles have a nominal size of $\mathrm{XTsiz} \times \mathrm{YTsiz}$, but those bordering on the edges of the image area may have a size which differs from the nominal size. The tiles are numbered in raster scan order (starting at zero).

By mapping the position of each tile from the reference grid to the coordinate systems of the individual components, a partitioning of the components themselves is obtained. For example, suppose that a tile has an upper left corner and lower right corner with coordinates $(\mathrm{tx}_0, \mathrm{ty}_0)$ and $(\mathrm{tx}_1 - 1, \mathrm{ty}_1 - 1)$, respectively. Then, in the coordinate space of a particular component, the tile would have an upper left corner and lower right corner with coordinates $(\mathrm{tcx}_0, \mathrm{tcy}_0)$ and $(\mathrm{tcx}_1 - 1, \mathrm{tcy}_1 - 1)$, respectively, where

$$(\mathrm{tcx}_0, \mathrm{tcy}_0) = (\lceil \mathrm{tx}_0/\mathrm{XRsiz} \rceil, \lceil \mathrm{ty}_0/\mathrm{YRsiz} \rceil) \quad \text{and} \tag{A.1a}$$

$$(\mathrm{tcx}_1, \mathrm{tcy}_1) = (\lceil \mathrm{tx}_1/\mathrm{XRsiz} \rceil, \lceil \mathrm{ty}_1/\mathrm{YRsiz} \rceil). \tag{A.1b}$$

These equations correspond to the illustration in Figure A.4. The portion of a component that corresponds to a single tile is referred to as a tile-component. Although the tiling grid is regular with respect to the reference grid, it is important to note that the grid may not necessarily be regular with respect to the coordinate systems of the components.

A.3.4 Codec Structure

The general structure of the codec is shown in Figure A.5 with the form of the encoder given by Figure A.5(a) and the decoder given by Figure A.5(b). From these diagrams, the key processes associated with the codec can be identified:

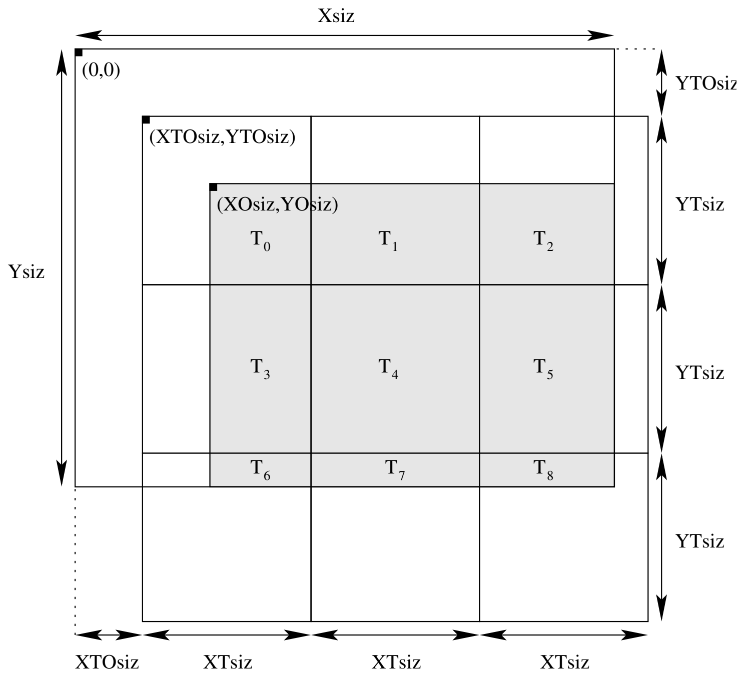


Figure A.3: Tiling on the reference grid.

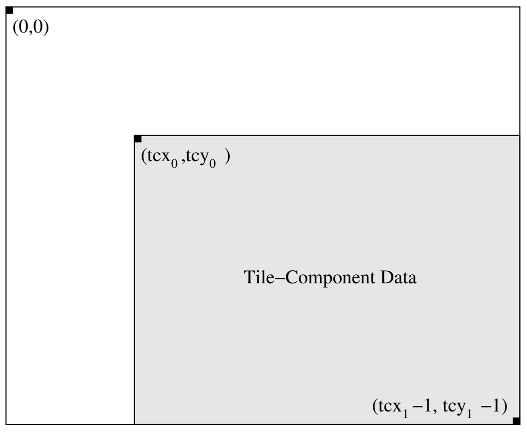
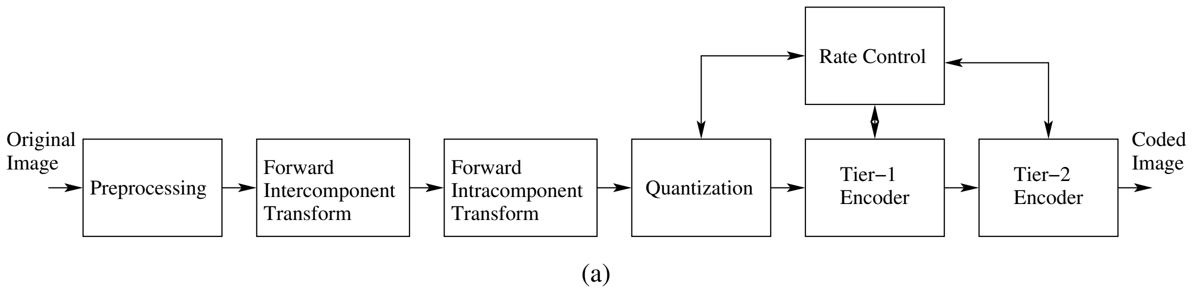


Figure A.4: Tile-component coordinate system.



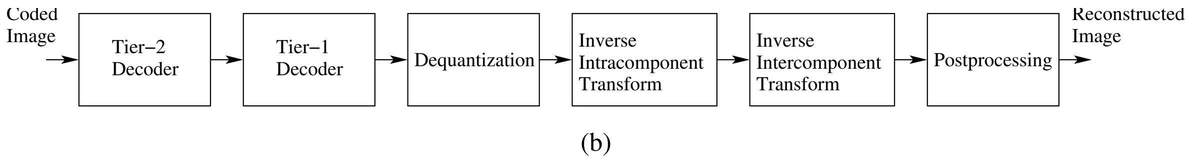


Figure A.5: Codec structure. The structure of the (a) encoder and (b) decoder.

1) preprocessing/postprocessing, 2) intercomponent transform, 3) intracomponent transform, 4) quantization/dequantization, 5) tier-1 coding, 6) tier-2 coding, and 7) rate control. The decoder structure essentially mirrors that of the encoder. That is, with the exception of rate control, there is a one-to-one correspondence between functional blocks in the encoder and decoder. Each functional block in the decoder either exactly or approximately inverts the effects of its corresponding block in the encoder. Since tiles are coded independently of one another, the input image is (conceptually, at least) processed one tile at a time. In the sections that follow, each of the above processes is examined in more detail.

A.3.5 Preprocessing/Postprocessing

The codec expects its input sample data to have a nominal dynamic range that is approximately centered about zero. The preprocessing stage of the encoder simply ensures that this expectation is met. Suppose that a particular component has P bits/sample. The samples may be either signed or unsigned, leading to a nominal dynamic range of $[-2^{P-1}, 2^{P-1}-1]$ or $[0, 2^P-1]$, respectively. If the sample values are unsigned, the nominal dynamic range is clearly not centered about zero. Thus, the nominal dynamic range of the samples is adjusted by subtracting a bias of 2^{P-1} from each of the sample values. If the sample values for a component are signed, the nominal dynamic range is already centered about zero, and no processing is required. By ensuring that the nominal dynamic range is centered about zero, a number of simplifying assumptions could be made in the design of the codec (e.g., with respect to context modeling, numerical overflow, etc.).

The postprocessing stage of the decoder essentially undoes the effects of preprocessing in the encoder. If the sample values for a component are unsigned, the original nominal dynamic range is restored. Lastly, in the case of lossy coding, clipping is performed to ensure that the sample values do not exceed the allowable range.

A.3.6 Intercomponent Transform

In the encoder, the preprocessing stage is followed by the forward intercomponent transform stage. Here, an intercomponent transform can be applied to the tile-component data. Such a transform operates on all of the components together, and serves to reduce the correlation between components, leading to improved coding efficiency.

Only two intercomponent transforms are defined in the baseline JPEG-2000 codec: the irreversible color transform (ICT) and reversible color transform (RCT). The ICT is nonreversible and real-to-real in nature, while the RCT is reversible and integer-to-integer. Both of these transforms essentially map image data from the RGB to YCrCb color space. The transforms are defined to operate on the first three components of an image, with the assumption that components 0, 1, and 2 correspond to the red, green, and blue color planes. Due to the nature of these transforms, the components on which they operate must be sampled at the same resolution (i.e., have the same size). As a consequence of the above facts, the ICT and RCT can only be employed when the image being coded has at least three components, and the first three components are sampled at the same resolution. The ICT may only be used in the case of lossy coding, while the RCT can be used in either the lossy or lossless case. Even if a transform can be legally employed, it is not necessary to do so. That is, the decision to use a multicomponent transform is left at the discretion of the encoder. After the intercomponent transform stage in the encoder, data from each component is treated independently.

The ICT is nothing more than the classic RGB to YCrCb color space transform. The forward transform is defined as

$$\begin{bmatrix} V_0(x,y) \\ V_1(x,y) \\ V_2(x,y) \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} U_0(x,y) \\ U_1(x,y) \\ U_2(x,y) \end{bmatrix}, \tag{A.2}$$

where $U_0(x,y)$, $U_1(x,y)$, and $U_2(x,y)$ are the input components corresponding to the red, green, and blue color planes, respectively, and $V_0(x,y)$, $V_1(x,y)$, and $V_2(x,y)$ are the output components corresponding to the Y, Cr, and Cb planes, respectively. The inverse transform can be shown to be

$$\begin{bmatrix} U_0(x,y) \\ U_1(x,y) \\ U_2(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34413 & -0.71414 \\ 1 & -1.772 & 0 \end{bmatrix} \begin{bmatrix} V_0(x,y) \\ V_1(x,y) \\ V_2(x,y) \end{bmatrix}. \tag{A.3}$$

The RCT is simply a reversible integer-to-integer approximation to the ICT (similar to that proposed in [19]). The forward transform is given by

$$V_0(x,y) = \left\lfloor \tfrac{1}{4}\left(U_0(x,y) + 2U_1(x,y) + U_2(x,y)\right) \right\rfloor, \tag{A.4a}$$
$$V_1(x,y) = U_2(x,y) - U_1(x,y), \quad \text{and} \tag{A.4b}$$
$$V_2(x,y) = U_0(x,y) - U_1(x,y), \tag{A.4c}$$

where $U_0(x,y)$, $U_1(x,y)$, $U_2(x,y)$, $V_0(x,y)$, $V_1(x,y)$, and $V_2(x,y)$ are defined as above. The inverse transform can be shown to be

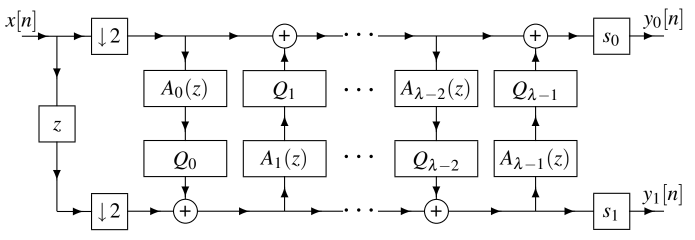
$$U_1(x,y) = V_0(x,y) - \left\lfloor \tfrac{1}{4}\left(V_1(x,y) + V_2(x,y)\right) \right\rfloor, \tag{A.5a}$$
$$U_0(x,y) = V_2(x,y) + U_1(x,y), \quad \text{and} \tag{A.5b}$$
$$U_2(x,y) = V_1(x,y) + U_1(x,y). \tag{A.5c}$$

The inverse intercomponent transform stage in the decoder essentially undoes the effects of the forward intercomponent transform stage in the encoder. If a multicomponent transform was applied during encoding, its inverse is applied here. Unless the transform is reversible, however, the inversion may only be approximate due to the effects of finite-precision arithmetic.

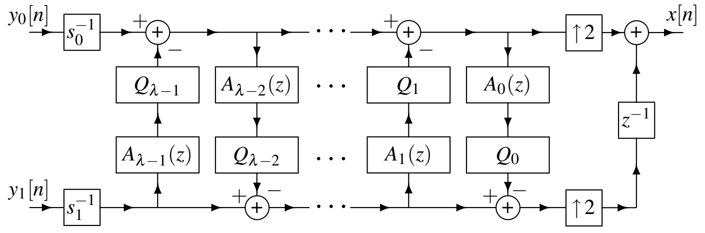
A.3.7 Intracomponent Transform

Following the intercomponent transform stage in the encoder is the intracomponent transform stage. In this stage, transforms that operate on individual components can be applied. The particular type of operator employed for this purpose is the wavelet transform. Through the application of the wavelet transform, a component is split into numerous frequency bands (i.e., subbands). Due to the statistical properties of these subband signals, the transformed data can usually be coded more efficiently than the original untransformed data.

Both reversible integer-to-integer [1, 3, 6, 12, 14] and nonreversible real-to-real wavelet transforms are employed by the baseline codec. The basic building block for such transforms is the 1-D 2-channel perfect-reconstruction (PR) uniformly-maximally-decimated (UMD) filter bank (FB) which has the general form shown in Figure A.6. Here, we focus on the lifting realization of the UMDFB [40, 41], as it can be used to implement the reversible integer-to-integer and nonreversible real-to-real wavelet transforms employed by the baseline codec. In fact, for this reason, it is likely that this realization strategy will be employed by many codec implementations. The analysis side of the UMDFB, depicted in Figure A.6(a), is associated with the forward transform, while the synthesis side, depicted in Figure A.6(b), is associated with the inverse transform. In the diagram, the $\{A_i(z)\}_{i=0}^{\lambda-1}$, $\{Q_i(x)\}_{i=0}^{\lambda-1}$, and $\{s_i\}_{i=0}^{1}$



(a)



(b)

Figure A.6: Lifting realization of a 1-D 2-channel PR UMDFB. (a) Analysis side. (b) Synthesis side.

denote filter transfer functions, quantization operators, and (scalar) gains, respectively. To obtain integer-to-integer mappings, the $\{Q_i(x)\}_{i=0}^{\lambda-1}$ are selected such that they always yield integer values, and the $\{s_i\}_{i=0}^{1}$ are chosen as integers. For real-to-real mappings, the $\{Q_i(x)\}_{i=0}^{\lambda-1}$ are simply chosen as the identity, and the $\{s_i\}_{i=0}^{1}$ are selected from the real numbers. To facilitate filtering at signal boundaries, symmetric extension [7, 10, 11] is employed. Since an image is a 2-D signal, clearly we need a 2-D UMDFB. By applying the 1-D UMDFB in both the horizontal and vertical directions, a 2-D UMDFB is effectively obtained. The wavelet transform is then calculated by recursively applying the 2-D UMDFB to the lowpass subband signal obtained at each level in the decomposition.

Suppose that a $(R-1)$-level wavelet transform is to be employed. To compute the forward transform, we apply the analysis side of the 2-D UMDFB to the tile-component data in an iterative manner, resulting in a number of subband signals being produced. Each application of the analysis side of the 2-D UMDFB yields four subbands: 1) horizontally and vertically lowpass (LL), 2) horizontally lowpass and vertically highpass (LH), 3) horizontally highpass and vertically lowpass (HL), and 4) horizontally and vertically highpass (HH). A $(R-1)$-level wavelet decomposition is associated with R resolution levels, numbered from 0 to $R-1$, with 0 and $R-1$ corresponding to the coarsest and finest resolutions, respectively. Each subband of the decomposition is identified by its orientation (e.g., LL, LH, HL, HH) and its corresponding resolution level (e.g., $0,1,\ldots,R-1$). The input tile-component signal is considered to be the LL_{R-1} band. At each resolution level (except the lowest) the LL band is further decomposed. For example, the LL_{R-1} band is decomposed to yield the LL_{R-2}, LH_{R-2}, HL_{R-2}, and HH_{R-2} bands. Then, at the next level, the LL_{R-2} band is decomposed, and so on. This process repeats until the LL_0 band is obtained, and results in the subband structure illustrated in Figure A.7. In the degenerate case where no transform is applied, $R=1$, and we effectively have only one subband (i.e., the LL_0 band).

As described above, the wavelet decomposition can be associated with data at R different resolutions. Suppose that the top-left and bottom-right samples of a tile-component have coordinates $(\text{tcx}_0, \text{tcy}_0)$ and $(\text{tcx}_1-1, \text{tcy}_1-1)$, respectively. This being the case, the top-left and bottom-right samples of the tile-component at resolution r have coordinates $(\text{trx}_0, \text{try}_0)$ and $(\text{trx}_1-1, \text{try}_1-1)$, respectively, given by

$$(\text{trx}_0, \text{try}_0) = \left(\left\lceil \text{tcx}_0/2^{R-r-1}\right\rceil, \left\lceil \text{tcy}_0/2^{R-r-1}\right\rceil\right) \quad \text{and} \tag{A.6a}$$

$$(\text{trx}_1, \text{try}_1) = \left(\left\lceil \text{tcx}_1/2^{R-r-1}\right\rceil, \left\lceil \text{tcy}_1/2^{R-r-1}\right\rceil\right), \tag{A.6b}$$

where r is the particular resolution of interest. Thus, the tile-component signal at a particular resolution has the size $(\text{trx}_1-\text{trx}_0)\times(\text{try}_1-\text{try}_0)$.

Not only are the coordinate systems of the resolution levels important, but so too are the coordinate systems for the various subbands. Suppose that we denote the coordinates of the upper left and lower right samples in a subband

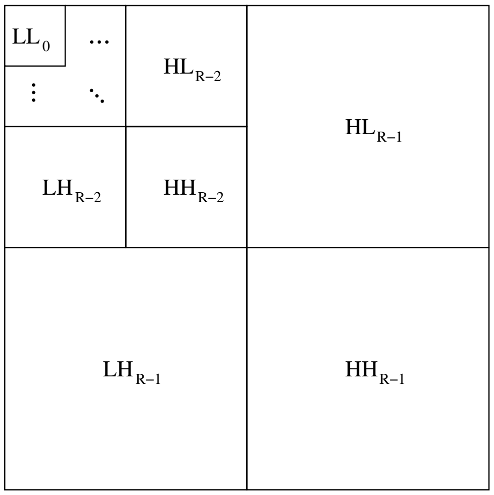


Figure A.7: Subband structure.

as $(\mathrm{tbx}_0, \mathrm{tby}_0)$ and $(\mathrm{tbx}_1 - 1, \mathrm{tby}_1 - 1)$, respectively. These quantities are computed as

$$(\mathrm{tbx}_0, \mathrm{tby}_0) = \begin{cases} \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} \right\rceil\right) & \text{for LL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} \right\rceil\right) & \text{for HL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for LH band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for HH band} \quad \text{and} \end{cases} \tag{A.7a}$$

$$(\mathrm{tbx}_1, \mathrm{tby}_1) = \begin{cases} \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} \right\rceil\right) & \text{for LL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} \right\rceil\right) & \text{for HL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for LH band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for HH band,} \end{cases} \tag{A.7b}$$

where r is the resolution level to which the band belongs, R is the number of resolution levels, and tcx_0, tcy_0, tcx_1, and tcy_1 are as defined in (A.1). Thus, a particular band has the size $(\mathrm{tbx}_1 - \mathrm{tbx}_0) \times (\mathrm{tby}_1 - \mathrm{tby}_0)$. From the above equations, we can also see that $(\mathrm{tbx}_0, \mathrm{tby}_0) = (\mathrm{trx}_0, \mathrm{try}_0)$ and $(\mathrm{tbx}_1, \mathrm{tby}_1) = (\mathrm{trx}_1, \mathrm{try}_1)$ for the LL_r band, as one would expect. (This should be the case since the LL_r band is equivalent to a reduced resolution version of the original data.) As will be seen, the coordinate systems for the various resolutions and subbands of a tile-component play an important role in codec behavior.

By examining (A.1), (A.6), and (A.7), we observe that the coordinates of the top-left sample for a particular subband, denoted $(\mathrm{tbx}_0, \mathrm{tby}_0)$, are partially determined by the XOsiz and YOsiz parameters of the reference grid. At each level of the decomposition, the parity (i.e., oddness/evenness) of tbx_0 and tby_0 affects the outcome of the downsampling process (since downsampling is shift variant). In this way, the XOsiz and YOsiz parameters have a subtle, yet important, effect on the transform calculation.

Having described the general transform framework, we now describe the two specific wavelet transforms supported by the baseline codec: the 5/3 and 9/7 transforms. The 5/3 transform is reversible, integer-to-integer, and nonlinear. This transform was proposed in [12], and is simply an approximation to a linear wavelet transform proposed in [18]. The 5/3 transform has an underlying 1-D UMDFB with the parameters:

$$\begin{gathered} \lambda = 2, \quad A_0(z) = -\tfrac{1}{2}(z+1), \quad A_1(z) = \tfrac{1}{4}(1+z^{-1}), \\ Q_0(x) = -\lfloor -x \rfloor, \quad Q_1(x) = \lfloor x + \tfrac{1}{2} \rfloor, \quad \text{and} \quad s_0 = s_1 = 1. \end{gathered} \tag{A.8}$$

The 9/7 transform is nonreversible and real-to-real. This transform, proposed in [8], is also employed in the FBI fingerprint compression standard [17] (although the normalizations differ). The 9/7 transform has an underlying 1-D UMDFB with the parameters:

$$\begin{gathered}\lambda = 4, \quad A_0(z) = \alpha_0(z+1), \quad A_1(z) = \alpha_1(1+z^{-1}), \\ A_2(z) = \alpha_2(z+1), \quad A_3(z) = \alpha_3(1+z^{-1}), \\ Q_i(x) = x \text{ for } i = 0,1,2,3, \\ \alpha_0 \approx -1.586134, \quad \alpha_1 \approx -0.052980, \quad \alpha_2 \approx 0.882911, \\ \alpha_3 \approx 0.443506, \quad s_0 \approx 1/1.230174, \quad \text{and} \quad s_1 = -1/s_0.\end{gathered} \tag{A.9}$$

Since the 5/3 transform is reversible, it can be employed for either lossy or lossless coding. The 9/7 transform, lacking the reversible property, can only be used for lossy coding. The number of resolution levels is a parameter of each transform. A typical value for this parameter is six (for a sufficiently large image). The encoder may transform all, none, or a subset of the components. This choice is at the encoder's discretion.

The inverse intracomponent transform stage in the decoder essentially undoes the effects of the forward intracomponent transform stage in the encoder. If a transform was applied to a particular component during encoding, the corresponding inverse transform is applied here. Due to the effects of finite-precision arithmetic, the inversion process is not guaranteed to be exact unless reversible transforms are employed.

A.3.8 Quantization/Dequantization

In the encoder, after the tile-component data has been transformed (by intercomponent and/or intracomponent transforms), the resulting coefficients are quantized. Quantization allows greater compression to be achieved, by representing transform coefficients with only the minimal precision required to obtain the desired level of image quality. Quantization of transform coefficients is one of the two primary sources of information loss in the coding path (the other source being the discarding of coding pass data as will be described later).

Transform coefficients are quantized using scalar quantization with a deadzone. A different quantizer is employed for the coefficients of each subband, and each quantizer has only one parameter, its step size. Mathematically, the quantization process is defined as

$$V(x,y) = \lfloor |U(x,y)| / \Delta \rfloor \operatorname{sgn} U(x,y), \tag{A.10}$$

where Δ is the quantizer step size, $U(x,y)$ is the input subband signal, and $V(x,y)$ denotes the output quantizer indices for the subband. Since this equation is specified in an informative clause of the standard, encoders need not use this precise formula. This said, however, it is likely that many encoders will, in fact, use the above equation.

The baseline codec has two distinct modes of operation, referred to herein as integer mode and real mode. In integer mode, all transforms employed are integer-to-integer in nature (e.g., RCT, 5/3 WT). In real mode, real-to-real transforms are employed (e.g., ICT, 9/7 WT). In integer mode, the quantizer step sizes are always fixed at one, effectively bypassing quantization and forcing the quantizer indices and transform coefficients to be one and the same. In this case, lossy coding is still possible, but rate control is achieved by another mechanism (to be discussed later). In the case of real mode (which implies lossy coding), the quantizer step sizes are chosen in conjunction with rate control. Numerous strategies are possible for the selection of the quantizer step sizes, as will be discussed later in Section A.3.12.

As one might expect, the quantizer step sizes used by the encoder are conveyed to the decoder via the code stream. In passing, we note that the step sizes specified in the code stream are relative and not absolute quantities. That is, the quantizer step size for each band is specified relative to the nominal dynamic range of the subband signal.

In the decoder, the dequantization stage tries to undo the effects of quantization. This process, however, is not usually invertible, and therefore results in some information loss. The quantized transform coefficient values are obtained from the quantizer indices. Mathematically, the dequantization process is defined as

$$U(x,y) = (V(x,y) + r \operatorname{sgn} V(x,y))\,\Delta, \tag{A.11}$$

where Δ is the quantizer step size, r is a bias parameter, $V(x,y)$ are the input quantizer indices for the subband, and $U(x,y)$ is the reconstructed subband signal. Although the value of r is not normatively specified in the standard, it is likely that many decoders will use the value of one half.

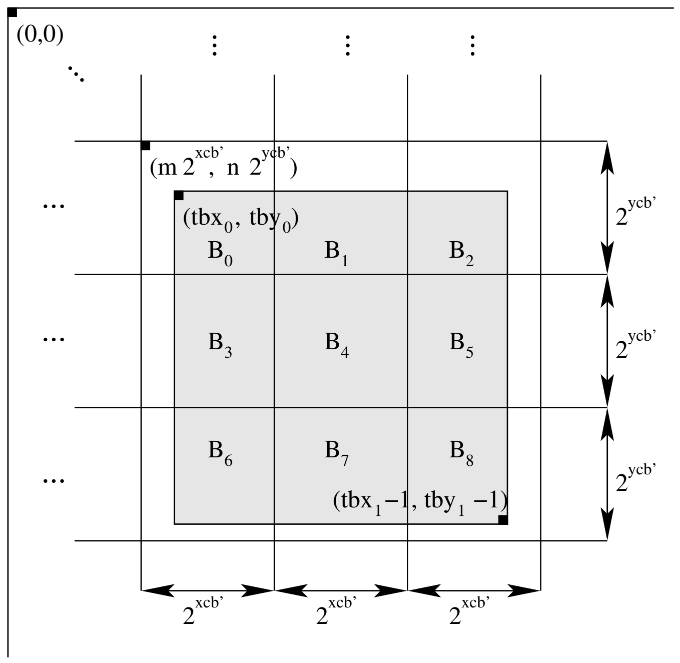


Figure A.8: Partitioning of a subband into code blocks.

A.3.9 Tier-1 Coding

After quantization is performed in the encoder, tier-1 coding takes place. This is the first of two coding stages. The quantizer indices for each subband are partitioned into code blocks. Code blocks are rectangular in shape, and their nominal size is a free parameter of the coding process, subject to certain constraints, most notably: 1) the nominal width and height of a code block must be an integer power of two, and 2) the product of the nominal width and height cannot exceed 4096.

Suppose that the nominal code block size is tentatively chosen to be $2^{\mathrm{xcb}} \times 2^{\mathrm{ycb}}$. In tier-2 coding, yet to be discussed, code blocks are grouped into what are called precincts. Since code blocks are not permitted to cross precinct boundaries, a reduction in the nominal code block size may be required if the precinct size is sufficiently small. Suppose that the nominal code block size after any such adjustment is $2^{\mathrm{xcb'}} \times 2^{\mathrm{ycb'}}$ where xcb' $\leq$ xcb and ycb' $\leq$ ycb. The subband is partitioned into code blocks by overlaying the subband with a rectangular grid having horizontal and vertical spacings of $2^{\mathrm{xcb'}}$ and $2^{\mathrm{ycb'}}$, respectively, as shown in Figure A.8. The origin of this grid is anchored at $(0,0)$ in the coordinate system of the subband. A typical choice for the nominal code block size is 64×64 (i.e., xcb $= 6$ and ycb $= 6$).

Let us, again, denote the coordinates of the top-left sample in a subband as $(\mathrm{tbx}_0, \mathrm{tby}_0)$. As explained in Section A.3.7, the quantity $(\mathrm{tbx}_0, \mathrm{tby}_0)$ is partially determined by the reference grid parameters XOsiz and YOsiz. In turn, the quantity $(\mathrm{tbx}_0, \mathrm{tby}_0)$ affects the position of code block boundaries within a subband. In this way, the XOsiz and YOsiz parameters have an important effect on the behavior of the tier-1 coding process (i.e., they affect the location of code block boundaries).

After a subband has been partitioned into code blocks, each of the code blocks is independently coded. The coding is performed using the bit-plane coder described later in Section A.3.10. For each code block, an embedded code is produced, comprised of numerous coding passes. The output of the tier-1 encoding process is, therefore, a collection of coding passes for the various code blocks.

On the decoder side, the bit-plane coding passes for the various code blocks are input to the tier-1 decoder, these passes are decoded, and the resulting data is assembled into subbands. In this way, we obtain the reconstructed quantizer indices for each subband. In the case of lossy coding, the reconstructed quantizer indices may only be approximations to the quantizer indices originally available at the encoder. This is attributable to the fact that the code stream may only include a subset of the coding passes generated by the tier-1 encoding process. In the lossless case, the reconstructed quantizer indices must be same as the original indices on the encoder side, since all coding passes must be included for lossless coding.

A.3.10 Bit-Plane Coding

The tier-1 coding process is essentially one of bit-plane coding. After all of the subbands have been partitioned into code blocks, each of the resulting code blocks is independently coded using a bit-plane coder. Although the bit-plane coding technique employed is similar to those used in the embedded zerotree wavelet (EZW) [39] and set partitioning in hierarchical trees (SPIHT) [37] codecs, there are two notable differences: 1) no interband dependencies are exploited, and 2) there are three coding passes per bit plane instead of two. The first difference follows from the fact that each code block is completely contained within a single subband, and code blocks are coded independently of one another. By not exploiting interband dependencies, improved error resilience can be achieved. The second difference is arguably less fundamental. Using three passes per bit plane instead of two reduces the amount of data associated with each coding pass, facilitating finer control over rate. Also, using an additional pass per bit plane allows better prioritization of important data, leading to improved coding efficiency.

As noted above, there are three coding passes per bit plane. In order, these passes are as follows: 1) significance, 2) refinement, and 3) cleanup. All three types of coding passes scan the samples of a code block in the same fixed order shown in Figure A.10. The code block is partitioned into horizontal stripes, each having a nominal height of four samples. If the code block height is not a multiple of four, the height of the bottom stripe will be less than this nominal value. As shown in the diagram, the stripes are scanned from top to bottom. Within a stripe, columns are scanned from left to right. Within a column, samples are scanned from top to bottom.

The bit-plane encoding process generates a sequence of symbols for each coding pass. Some or all of these symbols may be entropy coded. For the purposes of entropy coding, a context-based adaptive binary arithmetic coder is used—more specifically, the MQ coder from the JBIG2 standard [23]. For each pass, all of the symbols are either arithmetically coded or raw coded (i.e., the binary symbols are emitted as raw bits with simple bit stuffing). The arithmetic and raw coding processes both ensure that certain bit patterns never occur in the output, allowing such patterns to be used for error resilience purposes.

Cleanup passes always employ arithmetic coding. In the case of the significance and refinement passes, two possibilities exist, depending on whether the so called arithmetic-coding bypass mode (also known as lazy mode) is enabled. If lazy mode is enabled, only the significance and refinement passes for the four most significant bit planes use arithmetic coding, while the remaining such passes are raw coded. Otherwise, all significance and refinement passes are arithmetically coded. The lazy mode allows the computational complexity of bit-plane coding to be significantly reduced, by decreasing the number of symbols that must be arithmetically coded. This comes, of course, at the cost of reduced coding efficiency.

As indicated above, coding pass data can be encoded using one of two schemes (i.e., arithmetic or raw coding). Consecutive coding passes that employ the same encoding scheme constitute what is known as a segment. All of the coding passes in a segment can collectively form a single codeword or each coding pass can form a separate codeword. Which of these is the case is determined by the termination mode in effect. Two termination modes are supported: per-pass termination and per-segment termination. In the first case, all coding passes are terminated. In the second case, only the last coding pass of a segment is terminated. Terminating all coding passes facilitates improved error resilience at the expense of decreased coding efficiency.

Since context-based arithmetic coding is employed, a means for context selection is necessary. Generally speaking, context selection is performed by examining state information for the 4-connected or 8-connected neighbors of a sample as shown in Figure A.9.

In our explanation of the coding passes that follows, we focus on the encoder side as this facilitates easier understanding. The decoder algorithms follow directly from those employed on the encoder side.

A.3.10.1 Significance Pass

The first coding pass for each bit plane is the significance pass. This pass is used to convey significance and (as necessary) sign information for samples that have not yet been found to be significant and are predicted to become

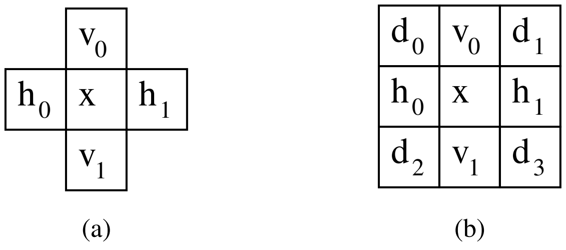


Figure A.9: Templates for context selection. The (a) 4-connected and (b) 8-connected neighbors.

significant during the processing of the current bit plane. The samples in the code block are scanned in the order shown previously in Figure A.10. If a sample has not yet been found to be significant, and is predicted to become significant, the significance of the sample is coded with a single binary symbol. If the sample also happens to be significant, its sign is coded using a single binary symbol. In pseudocode form, the significance pass is described by Algorithm 1.

Algorithm 1 Significance pass algorithm.

```
1: for each sample in code block do
2:   if sample previously insignificant and predicted to become significant during current bit plane then
3:     code significance of sample /* 1 binary symbol */
4:     if sample significant then
5:       code sign of sample /* 1 binary symbol */
6:     endif
7:   endif
8: endfor
```

If the most significant bit plane is being processed, all samples are predicted to remain insignificant. Otherwise, a sample is predicted to become significant if any 8-connected neighbor has already been found to be significant. As a consequence of this prediction policy, the significance and refinement passes for the most significant bit plane are always empty and not explicitly coded.

The symbols generated during the significance pass may or may not be arithmetically coded. If arithmetic coding is employed, the binary symbol conveying significance information is coded using one of nine contexts. The particular context used is selected based on the significance of the sample's 8-connected neighbors and the orientation of the subband with which the sample is associated (e.g., LL, LH, HL, HH). In the case that arithmetic coding is used, the sign of a sample is coded as the difference between the actual and predicted sign. Otherwise, the sign is coded directly. Sign prediction is performed using the significance and sign information for 4-connected neighbors.

A.3.10.2 Refinement Pass

The second coding pass for each bit plane is the refinement pass. This pass signals subsequent bits after the most significant bit for each sample. The samples of the code block are scanned using the order shown earlier in Figure A.10. If a sample was found to be significant in a previous bit plane, the next most significant bit of that sample is conveyed using a single binary symbol. This process is described in pseudocode form by Algorithm 2.

Algorithm 2 Refinement pass algorithm.

```
1: for each sample in code block do
2:   if sample found significant in previous bit plane then
3:     code next most significant bit in sample /* 1 binary symbol */
4:   endif
5: endfor
```

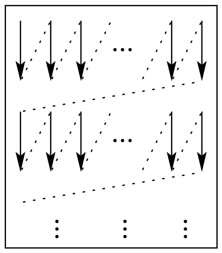


Figure A.10: Sample scan order within a code block.

Like the significance pass, the symbols of the refinement pass may or may not be arithmetically coded. If arithmetic coding is employed, each refinement symbol is coded using one of three contexts. The particular context employed is selected based on if the second MSB position is being refined and the significance of 8-connected neighbors.

A.3.10.3 Cleanup Pass

The third (and final) coding pass for each bit plane is the cleanup pass. This pass is used to convey significance and (as necessary) sign information for those samples that have not yet been found to be significant and are predicted to remain insignificant during the processing of the current bit plane.

Conceptually, the cleanup pass is not much different from the significance pass. The key difference is that the cleanup pass conveys information about samples that are predicted to remain insignificant, rather than those that are predicted to become significant. Algorithmically, however, there is one important difference between the cleanup and significance passes. In the case of the cleanup pass, samples are sometimes processed in groups, rather than individually as with the significance pass.

Recall the scan pattern for samples in a code block, shown earlier in Figure A.10. A code block is partitioned into stripes with a nominal height of four samples. Then, stripes are scanned from top to bottom, and the columns within a stripe are scanned from left to right. For convenience, we will refer to each column within a stripe as a vertical scan. That is, each vertical arrow in the diagram corresponds to a so called vertical scan. As will soon become evident, the cleanup pass is best explained as operating on vertical scans (and not simply individual samples).

The cleanup pass simply processes each of the vertical scans in order, with each vertical scan being processed as follows. If the vertical scan contains four samples (i.e., is a full scan), significance information is needed for all of these samples, and all of the samples are predicted to remain insignificant, a special mode, called aggregation mode, is entered. In this mode, the number of leading insignificant samples in the vertical scan is coded. Then, the samples whose significance information is conveyed by aggregation are skipped, and processing continues with the remaining samples of the vertical scan exactly as is done in the significance pass. In pseudocode form, this process is described by Algorithm 3.

When aggregation mode is entered, the four samples of the vertical scan are examined. If all four samples are insignificant, an all-insignificant aggregation symbol is coded, and processing of the vertical scan is complete. Otherwise, a some-significant aggregation symbol is coded, and two binary symbols are then used to code the number of leading insignificant samples in the vertical scan.

The symbols generated during the cleanup pass are always arithmetically coded. In the aggregation mode, the aggregation symbol is coded using a single context, and the two symbol run length is coded using a single context with a fixed uniform probability distribution. When aggregation mode is not employed, significance and sign coding function just as in the case of the significance pass.

A.3.11 Tier-2 Coding

In the encoder, tier-1 encoding is followed by tier-2 encoding. The input to the tier-2 encoding process is the set of bit-plane coding passes generated during tier-1 encoding. In tier-2 encoding, the coding pass information is packaged into data units called packets, in a process referred to as packetization. The resulting packets are then output to the

Algorithm 3 Cleanup pass algorithm.

```
 1: for each vertical scan in code block do
 2:   if four samples in vertical scan and all previously insignificant and unvisited and none have significant 8-
      connected neighbor then
 3:     code number of leading insignificant samples via aggregation
 4:     skip over any samples indicated as insignificant by aggregation
 5:   endif
 6:   while more samples to process in vertical scan do
 7:     if sample previously insignificant and unvisited then
 8:       code significance of sample if not already implied by run /* 1 binary symbol */
 9:       if sample significant then
10:         code sign of sample /* 1 binary symbol */
11:       endif
12:     endif
13:   endwhile
14: endfor
```

final code stream. The packetization process imposes a particular organization on coding pass data in the output code stream. This organization facilitates many of the desired codec features including rate scalability and progressive recovery by fidelity or resolution.

A packet is nothing more than a collection of coding pass data. Each packet is comprised of two parts: a header and body. The header indicates which coding passes are included in the packet, while the body contains the actual coding pass data itself. In the code stream, the header and body may appear together or separately, depending on the coding options in effect.

Rate scalability is achieved through (quality) layers. The coded data for each tile is organized into L layers, numbered from 0 to $L-1$, where $L \geq 1$. Each coding pass is either assigned to one of the L layers or discarded. The coding passes containing the most important data are included in the lower layers, while the coding passes associated with finer details are included in higher layers. During decoding, the reconstructed image quality improves incrementally with each successive layer processed. In the case of lossy compression, some coding passes may be discarded (i.e., not included in any layer) in which case rate control must decide which passes to include in the final code stream. In the lossless case, all coding passes must be included. If multiple layers are employed (i.e., $L > 1$), rate control must decide in which layer each coding pass is to be included. Since some coding passes may be discarded, tier-2 coding is the second primary source of information loss in the coding path.

Recall, from Section A.3.9, that each coding pass is associated with a particular component, resolution level, subband, and code block. In tier-2 coding, one packet is generated for each component, resolution level, layer, and precinct 4-tuple. A packet need not contain any coding pass data at all. That is, a packet can be empty. Empty packets are sometimes necessary since a packet must be generated for every component-resolution-layer-precinct combination even if the resulting packet conveys no new information.

As mentioned briefly in Section A.3.7, a precinct is essentially a grouping of code blocks within a subband. The precinct partitioning for a particular subband is derived from a partitioning of its parent LL band (i.e., the LL band at the next higher resolution level). Each resolution level has a nominal precinct size. The nominal width and height of a precinct must be a power of two, subject to certain constraints (e.g., the maximum width and height are both 2^{15}). The LL band associated with each resolution level is divided into precincts. This is accomplished by overlaying the LL band with a regular grid having horizontal and vertical spacings of 2^{PPx} and 2^{PPy}, respectively, as shown in Figure A.11, where the grid is aligned with the origin of the LL band's coordinate system. The precincts bordering on the edge of the subband may have dimensions smaller than the nominal size. Each of the resulting precinct regions is then mapped into its child subbands (if any) at the next lower resolution level. This is accomplished by using the coordinate transformation $(u,v) = (\lceil x/2 \rceil, \lceil y/2 \rceil)$ where (x,y) and (u,v) are the coordinates of a point in the LL band and child subband, respectively. Due to the manner in which the precinct partitioning is performed, precinct boundaries always align with code block boundaries. Some precincts may also be empty. Suppose the nominal code

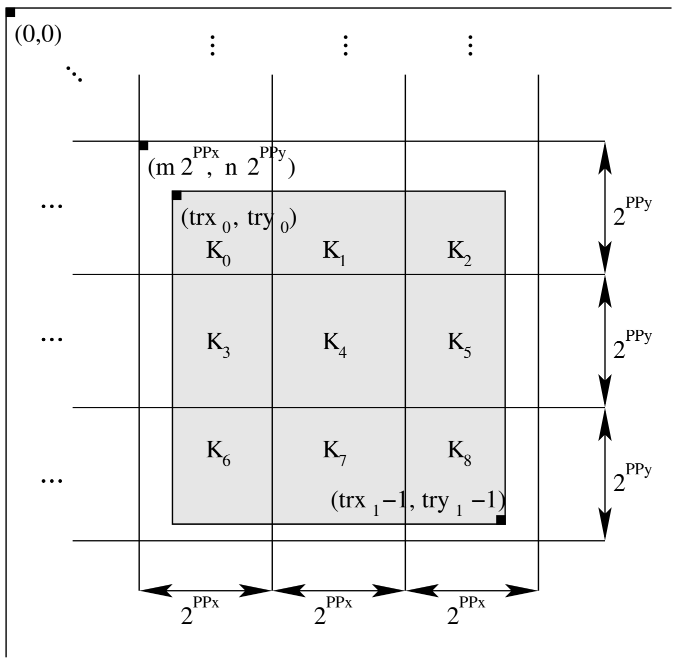


Figure A.11: Partitioning of a resolution into precincts.

block size is $2^{\text{xcb'}} \times 2^{\text{ycb'}}$. This results nominally in $2^{\text{PPx'}-\text{xcb'}} \times 2^{\text{PPy'}-\text{ycb'}}$ groups of code blocks in a precinct, where

$$\text{PPx'} = \begin{cases} \text{PPx} & \text{for } r = 0 \\ \text{PPx} - 1 & \text{for } r > 0, \end{cases} \tag{A.12}$$

$$\text{PPy'} = \begin{cases} \text{PPy} & \text{for } r = 0 \\ \text{PPy} - 1 & \text{for } r > 0, \end{cases} \tag{A.13}$$

and r is the resolution level.

Since coding pass data from different precincts are coded in separate packets, using smaller precincts reduces the amount of data contained in each packet. If less data is contained in a packet, a bit error is likely to result in less information loss (since, to some extent, bit errors in one packet do not affect the decoding of other packets). Thus, using a smaller precinct size leads to improved error resilience, while coding efficiency is degraded due to the increased overhead of having a larger number of packets.

More than one ordering of packets in the code stream is supported. Such orderings are called progressions. There are five built-in progressions defined: 1) layer-resolution-component-position ordering, 2) resolution-layer-component-position ordering, 3) resolution-position-component-layer ordering, 4) position-component-resolution-layer ordering, and 5) component-position-resolution-layer ordering. The sort order for the packets is given by the name of the ordering, where position refers to precinct number, and the sorting keys are listed from most significant to least significant. For example, in the case of the first ordering given above, packets are ordered first by layer, second by resolution, third by component, and last by precinct. This corresponds to a progressive recovery by fidelity scenario. The second ordering above is associated with progressive recovery by resolution. The three remaining orderings are somewhat more esoteric. It is also possible to specify additional user-defined progressions at the expense of increased coding overhead.

In the simplest scenario, all of the packets from a particular tile appear together in the code stream. Provisions exist, however, for interleaving packets from different tiles, allowing further flexibility on the ordering of data. If, for example, progressive recovery of a tiled image was desired, one would probably include all of the packets associated with the first layer of the various tiles, followed by those packets associated with the second layer, and so on.

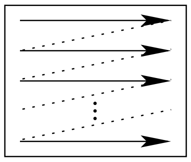


Figure A.12: Code block scan order within a precinct.

In the decoder, the tier-2 decoding process extracts the various coding passes from the code stream (i.e., depacketization) and associates each coding pass with its corresponding code block. In the lossy case, not all of the coding passes are guaranteed to be present since some may have been discarded by the encoder. In the lossless case, all of the coding passes must be present in the code stream.

In the sections that follow, we describe the packet coding process in more detail. For ease of understanding, we choose to explain this process from the point of view of the encoder. The decoder algorithms, however, can be trivially deduced from those of the encoder.

A.3.11.1 Packet Header Coding

The packet header corresponding to a particular component, resolution level, layer, and precinct, is encoded as follows. First, a single binary symbol is encoded to indicate if any coding pass data is included in the packet (i.e., if the packet is non-empty). If the packet is empty, no further processing is required and the algorithm terminates. Otherwise, we proceed to examine each subband in the resolution level in a fixed order. For each subband, we visit the code blocks belonging to the precinct of interest in raster scan order as shown in Figure A.12. To process a single code block, we begin by determining if any new coding pass data is to be included. If no coding pass data has yet been included for this code block, the inclusion information is conveyed via a quadtree-based coding procedure. Otherwise, a binary symbol is emitted indicating the presence or absence of new coding pass data for the code block. If no new coding passes are included, we proceed to the processing of the next code block in the precinct. Assuming that new coding pass data are to be included, we continue with our processing of the current code block. If this is the first time coding pass data have been included for the code block, we encode the number of leading insignificant bit planes for the code block using a quadtree-based coding algorithm. Then, the number of new coding passes, and the length of the data associated with these passes is encoded. A bit stuffing algorithm is applied to all packet header data to ensure that certain bit patterns never occur in the output, allowing such patterns to be used for error resilience purposes. The entire packet header coding process is summarized by Algorithm 4.

A.3.11.2 Packet Body Coding

The algorithm used to encode the packet body is relatively simple. The code blocks are examined in the same order as in the case of the packet header. If any new passes were specified in the corresponding packet header, the data for these coding passes are concatenated to the packet body. This process is summarized by Algorithm 5.

A.3.12 Rate Control

In the encoder, rate control can be achieved through two distinct mechanisms: 1) the choice of quantizer step sizes, and 2) the selection of the subset of coding passes to include in the code stream. When the integer coding mode is used (i.e., when only integer-to-integer transforms are employed) only the second mechanism may be used, since the quantizer step sizes must be fixed at one. When the real coding mode is used, then either or both of these rate control mechanisms may be employed.

When the first mechanism is employed, quantizer step sizes are adjusted in order to control rate. As the step sizes are increased, the rate decreases, at the cost of greater distortion. Although this rate control mechanism is conceptually simple, it does have one potential drawback. Every time the quantizer step sizes are changed, the quantizer indices change, and tier-1 encoding must be performed again. Since tier-1 coding requires a considerable amount of computation, this approach to rate control may not be practical in computationally-constrained encoders.

Algorithm 4 Packet header coding algorithm.

```
1: if packet not empty then
2:    code non-empty packet indicator /* 1 binary symbol */
3:    for each subband in resolution level do
4:       for each code block in subband precinct do
5:          code inclusion information /* 1 binary symbol or tag tree */
6:          if no new coding passes included then
7:             skip to next code block
8:          endif
9:          if first inclusion of code block then
10:            code number of leading insignificant bit planes /* tag tree */
11:         endif
12:         code number of new coding passes
13:         code length increment indicator
14:         code length of coding pass data
15:      endfor
16:   endfor
17: else
18:   code empty packet indicator /* 1 binary symbol */
19: endif
20: pad to byte boundary
```

Algorithm 5 Packet body coding algorithm.

```
1: for each subband in resolution level do
2:    for each code block in subband precinct do
3:       if new coding passes included for code block then
4:          output coding pass data
5:       endif
6:    endfor
7: endfor
```

When the second mechanism is used, the encoder can elect to discard coding passes in order to control the rate. The encoder knows the contribution that each coding pass makes to rate, and can also calculate the distortion reduction associated with each coding pass. Using this information, the encoder can then include the coding passes in order of decreasing distortion reduction per unit rate until the bit budget has been exhausted. This approach is very flexible in that different distortion metrics can be easily accommodated (e.g., mean squared error, visually weighted mean squared error, etc.).

For a more detailed treatment of rate control, the reader is referred to [25] and [42].

A.3.13 Region of Interest Coding

The codec allows different regions of an image to be coded with differing fidelity. This feature is known as region-of-interest (ROI) coding. In order to support ROI coding, a very simple yet flexible technique is employed as described below.

When an image is synthesized from its transform coefficients, each coefficient contributes only to a specific region in the reconstruction. Thus, one way to code a ROI with greater fidelity than the rest of the image would be to identify the coefficients contributing to the ROI, and then to encode some or all of these coefficients with greater precision than the others. This is, in fact, the basic premise behind the ROI coding technique employed in the JPEG-2000 codec.

When an image is to be coded with a ROI, some of the transform coefficients are identified as being more important than the others. The coefficients of greater importance are referred to as ROI coefficients, while the remaining coefficients are known as background coefficients. Noting that there is a one-to-one correspondence between transform coefficients and quantizer indices, we further define the quantizer indices for the ROI and background coefficients as the ROI and background quantizer indices, respectively. With this terminology introduced, we are now in a position to describe how ROI coding fits into the rest of the coding framework.

The ROI coding functionality affects the tier-1 coding process. In the encoder, before the quantizer indices for the various subbands are bit-plane coded, the ROI quantizer indices are scaled upwards by a power of two (i.e., by a left bit shift). This scaling is performed in such a way as to ensure that all bits of the ROI quantizer indices lie in more significant bit planes than the potentially nonzero bits of the background quantizer indices. As a consequence, all information about ROI quantizer indices will be signalled before information about background ROI indices. In this way, the ROI can be reconstructed at a higher fidelity than the background.

Before the quantizer indices are bit-plane coded, the encoder examines the background quantizer indices for all of the subbands looking for the index with the largest magnitude. Suppose that this index has its most significant bit in bit position $N-1$. All of the ROI indices are then shifted N bits to the left, and bit-plane coding proceeds as in the non-ROI case. The ROI shift value N is included in the code stream.

During decoding, any quantizer index with nonzero bits lying in bit plane N or above can be deduced to belong to the ROI set. After the reconstructed quantizer indices are obtained from the bit-plane decoding process, all indices in the ROI set are then scaled down by a right shift of N bits. This undoes the effect of the scaling on the encoder side.

The ROI set can be chosen to correspond to transform coefficients affecting a particular region in an image or subset of those affecting the region. This ROI coding technique has a number of desirable properties. First, the ROI can have any arbitrary shape and can consist of multiple disjoint regions. Second, there is no need to explicitly signal the ROI set, since it can be deduced by the decoder from the ROI shift value and the magnitude of the quantizer indices.

For more information on ROI coding, the reader is referred to [9, 15].

A.3.14 Code Stream

In order to specify the coded representation of an image, two different levels of syntax are employed by the codec. The lowest level syntax is associated with what is referred to as the code stream. The code stream is essentially a sequence of tagged records and their accompanying data.

The basic building block of the code stream is the marker segment. As shown in Figure A.13, a marker segment is comprised of three fields: the type, length, and parameters fields. The type (or marker) field identifies the particular kind of marker segment. The length field specifies the number of bytes in the marker segment. The parameters field provides additional information specific to the marker type. Not all types of marker segments have length and

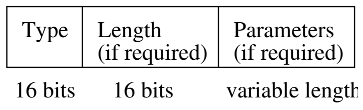


Figure A.13: Marker segment structure.

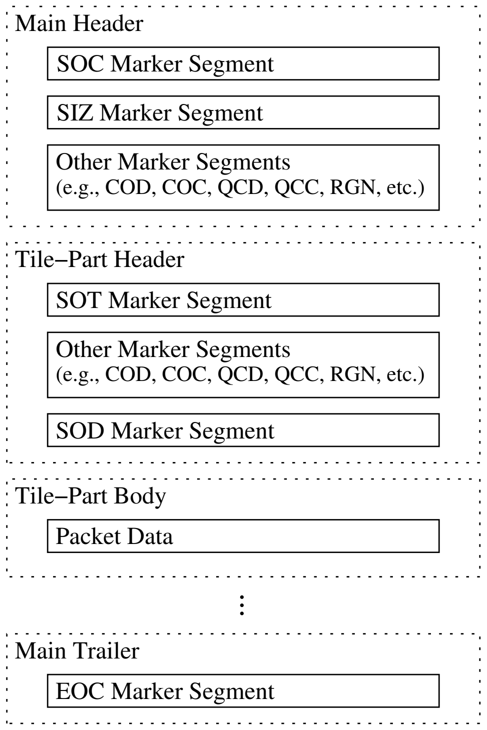


Figure A.14: Code stream structure.

parameters fields. The presence (or absence) of these fields is determined by the marker segment type. Each type of marker segment signals its own particular class of information.

A code stream is simply a sequence of marker segments and auxiliary data (i.e., packet data) organized as shown in Figure A.14. The code stream consists of a main header, followed by tile-part header and body pairs, followed by a main trailer. A list of some of the more important marker segments is given in Table A.2. Parameters specified in marker segments in the main header serve as defaults for the entire code stream. These default settings, however, may be overridden for a particular tile by specifying new values in a marker segment in the tile's header.

All marker segments, packet headers, and packet bodies are a multiple of 8 bits in length. As a consequence, all markers are byte-aligned, and the code stream itself is always an integral number of bytes.

A.3.15 File Format

A code stream provides only the most basic information required to decode an image (i.e., sufficient information to deduce the sample values of the decoded image). While in some simple applications this information is sufficient, in other applications additional data is required. To display a decoded image, for example, it is often necessary to know additional characteristics of an image, such as the color space of the image data and opacity attributes. Also, in some situations, it is beneficial to know other information about an image (e.g., ownership, origin, etc.) In order to allow the above types of data to be specified, an additional level of syntax is employed by the codec. This level of syntax is referred to as the file format. The file format is used to convey both coded image data and auxiliary information

Table A.2: Types of marker segments

Type	Description
Start of codestream (SOC)	Signals the start of a code stream. Always the first marker segment in the code stream (i.e., the first marker segment in the main header).
End of codestream (EOC)	Signals the end of the code stream. Always the last marker segment in the code stream.
Start of tile-part (SOT)	Indicates the start of a tile-part header. Always the first marker segment in a tile-part header.
Start of data (SOD)	Signal the end of the tile-part header. Always the last marker segment in the tile-part header. The tile body follows immediately after this marker segment.
Image and tile size (SIZ)	Conveys basic image characteristics (e.g., image size, number of components, precision of sample values), and tiling parameters. Always the second marker segment in the code stream.
Coding style default (COD)	Specifies coding parameters (e.g., multicomponent transform, wavelet/subband transform, tier-1/tier-2 coding parameters, etc.).
Coding style component (COC)	Specifies a subset of coding parameters for a single component.
Quantization default (QCD)	Specifies quantization parameters (i.e., quantizer type, quantizer parameters).
Quantization component (QCC)	Specifies quantization parameters for a single component.
Region of interest (RGN)	Specifies region-of-interest coding parameters.

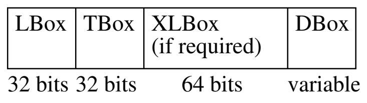


Figure A.15: Box structure.

about the image. Although this file format is optional, it undoubtedly will be used extensively by many applications, particularly computer-based software applications.

The basic building block of the file format is referred to as a box. As shown in Figure A.15, a box is nominally comprised of four fields: the LBox, TBox, XLBox, and DBox fields. The LBox field specifies the length of the box in bytes. The TBox field indicates the type of box (i.e., the nature of the information contained in the box). The XLBox field is an extended length indicator which provides a mechanism for specifying the length of a box whose size is too large to be encoded in the length field alone. If the LBox field is 1, then the XLBox field is present and contains the true length of the box. Otherwise, the XLBox field is not present. The DBox field contains data specific to the particular box type. Some types of boxes may contain other boxes as data. As a matter of terminology, a box that contains other boxes in its DBox field is referred to as a superbox. Several of the more important types of boxes are listed in Table A.3.

A file is a sequence of boxes. Since certain types of boxes are defined to contain others, there is a natural hierarchical structure to a file. The general structure of a file is shown in Figure A.16. The JPEG-2000 signature box is always first, providing an indication that the byte stream is, in fact, correctly formatted. The file type box is always second, indicating the version of the file format to which the byte stream conforms. Although some constraints exist on the ordering of the remaining boxes, some flexibility is also permitted. The header box simply contains a number of other boxes. The image header box specifies several basic characteristics of the image (including image size, number of components, etc.). The bits per component box indicates the precision and signedness of the component samples. The color specification box identifies the color space of image data (for display purposes) and indicates which components map to which type of spectral information (i.e., the correspondence between components and color/opacity planes). Every file must contain at least one contiguous code stream box. (Multiple contiguous code stream boxes are permit-

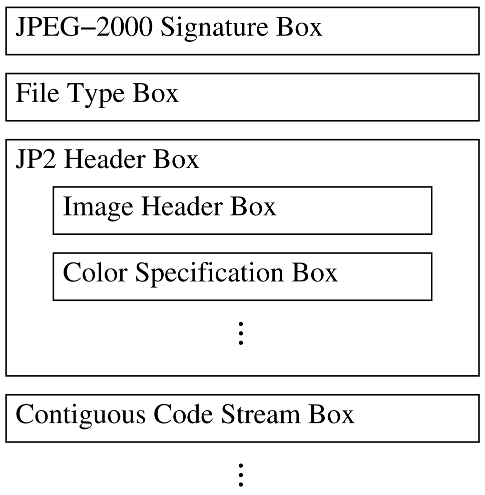


Figure A.16: File format structure.

Table A.3: Box types

Type	Description
JPEG-2000 Signature	Identifies the file as being in the JP2 format. Always the first box in a JP2 file.
File Type	Specifies the version of the format to which the file conforms. Always the second box in a JP2 file.
JP2 Header	Specifies information about the image aside from the coded image data itself. (A superbox.)
Image Header	Specifies the size and other basic characteristics of the image.
Color Specification	Specifies the colorspace to which the image sample data belongs.
Contiguous Code Stream	Contains a code stream.

ted in order to facilitate the specification of image sequences to support trivial animation effects.) Each contiguous code stream box contains a code stream as data. In this way, coded image data is embedded into a file. In addition to the types of boxes discussed so far, there are also box types for specifying the capture and display resolution for an image, palette information, intellectual property information, and vendor/application-specific data.

Although some of the information stored at the file format level is redundant (i.e., it is also specified at the code stream level), this redundancy allows trivial manipulation of files without any knowledge of the code stream syntax. The file name extension “jp2” is to be used to identify files containing data in the JP2 file format. For more information on the file format, the reader is referred to [20].

A.3.16 Extensions

Although the baseline codec is quite versatile, there may be some applications that could benefit from additional features not present in the baseline system. To this end, Part 2 of the standard [27] defines numerous extensions to the baseline codec. Some of these extensions include the following: 1) additional intercomponent transforms (e.g., multidimensional wavelet/subband transforms); 2) additional intracomponent transforms (e.g., subband transforms based on arbitrary filters and decomposition trees, different filters in the horizontal and vertical directions); 3) overlapped wavelet transforms; 4) additional quantization methods such as trellis coded quantization [32, 34]; 5) enhanced ROI support (e.g., a mechanism for explicitly signalling the shape of the ROI and an arbitrary shift value); and 6) extensions to the file format including support for additional color spaces and compound documents.

A.4 Conclusions

In this appendix, we commenced with a high-level introduction to the JPEG-2000 standard, and proceeded to study the JPEG-2000 codec in detail. With its excellent coding performance and many attractive features, the JPEG-2000 codec will no doubt prove to be extremely useful in many application areas.

A.5 JasPer

JasPer is a collection of software (i.e., a library and application programs) for the coding and manipulation of images. This software is written in the C programming language. Of particular interest here, the JasPer software provides an implementation of the JPEG-2000 Part-1 codec. The JasPer software was developed with the objective of providing a free JPEG-2000 codec implementation for anyone wishing to use the JPEG-2000 standard. This software has also been published in the JPEG-2000 Part-5 standard, as an official reference implementation of the JPEG-2000 Part-1 codec.

The JasPer software is available for download from the JasPer Project web site [4] and the JPEG web site (i.e., `http://www.jpeg.org/software`). For more information about JasPer, the reader is referred to [2, 4, 5].

A.6 Bibliography

[1] M. D. Adams. Reversible wavelet transforms and their application to embedded image compression. M.A.Sc. thesis, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, January 1998.

[2] M. D. Adams. JasPer software reference manual. ISO/IEC JTC 1/SC 29/WG 1 N 2415, December 2001.

[3] M. D. Adams. *Reversible Integer-to-Integer Wavelet Transforms for Image Coding*. PhD thesis, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, September 2002. Available online from `http://www.ece.uvic.ca/~mdadams`.

[4] M. D. Adams. The JasPer project home page. `http://www.ece.uvic.ca/~mdadams/jasper`, 2013.

[5] M. D. Adams and F. Kossentini. JasPer: A software-based JPEG-2000 codec implementation. In *Proc. of IEEE International Conference on Image Processing*, volume 2, pages 53–56, Vancouver, BC, Canada, October 2000.

[6] M. D. Adams and F. Kossentini. Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis. *IEEE Trans. on Image Processing*, 9(6):1010–1024, June 2000.

[7] M. D. Adams and R. K. Ward. Symmetric-extension-compatible reversible integer-to-integer wavelet transforms. *IEEE Trans. on Signal Processing*, 51(10):2624–2636, October 2003.

[8] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–220, April 1992.

[9] J. Askelof, M. Larsson Carlander, and C. Christopoulos. Region of interest coding in JPEG 2000. *Signal Processing: Image Communication*, 17(1):105–111, January 2002.

[10] C. M. Brislawn. Preservation of subband symmetry in multirate signal coding. *IEEE Trans. on Signal Processing*, 43(12):3046–3050, December 1995.

[11] C. M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Harmonic Analysis*, 3(4):337–357, October 1996.

[12] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5(3):332–369, July 1998.

[13] B. Carpentieri, M. J. Weinberger, and G. Seroussi. Lossless compression of continuous-tone images. *Proc. of IEEE*, 88(11):1797–1809, November 2000.

[14] H. Chao, P. Fisher, and Z. Hua. An approach to integer wavelet transforms for lossless for image compression. In *Proc. of International Symposium on Computational Mathematics*, pages 19–38, Guangzhou, China, August 1997.

[15] C. Christopoulos, J. Askelof, and M. Larsson. Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard. *IEEE Signal Processing Letters*, 7(9):247–249, September 2000.

[16] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG 2000 still image coding system: An overview. *IEEE Trans. on Consumer Electronics*, 46(4):1103–1127, November 2000.

[17] Federal Bureau of Investigation, Washington, DC, USA. *WSQ Gray-Scale Fingerprint Image Compression Specification, IAFIS-IC-0110(V3)*, December 1997.

[18] D. Le Gall and A. Tabatabai. Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 761–764, New York, NY, USA, April 1988.

[19] M. J. Gormish, E. L. Schwartz, A. F. Keith, M. P. Boliek, and A. Zandi. Lossless and nearly lossless compression of high-quality images. In *Proc. of SPIE*, volume 3025, pages 62–70, San Jose, CA, USA, March 1997.

[20] J. S. Houchin and D. W. Singer. File format technology in JPEG 2000 enables flexible use of still and motion sequences. *Signal Processing: Image Communication*, 17(1):131–144, January 2002.

[21] ISO/IEC call for contributions, JPEG 2000. ISO/IEC JTC 1/SC 29/WG 1 N505, March 1997.

[22] *ISO/IEC 10918-1:1994, Information technology—Digital compression and coding of continuous-tone still images: Requirements and guidelines*, 1994.

[23] *ISO/IEC 14492: Information technology—Lossy/lossless coding of bi-level images*, 2001.

[24] *ISO/IEC 14495-1: Lossless and near-lossless compression of continuous-tone still images: Baseline*, 2000.

[25] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.

[26] *ISO/IEC 15444-12: Information technology—JPEG 2000 image coding system—Part 12: ISO base media file format*, 2003.

[27] *ISO/IEC 15444-2: Information technology—JPEG 2000 image coding system—Part 2: Extensions*, 2002.

[28] *ISO/IEC 15444-3: Information technology—JPEG 2000 image coding system—Part 3: Motion JPEG 2000*, 2002.

[29] *ISO/IEC 15444-4: Information technology—JPEG 2000 image coding system—Part 4: Compliance testing*, 2002.

[30] *ISO/IEC 15444-5: Information technology—JPEG 2000 image coding system—Part 5: Reference software*, 2002.

[31] *ISO/IEC 15444-6: Information technology—JPEG 2000 image coding system—Part 6: Compound image file format*, 2003.

[32] J. H. Kasner, M. W. Marcellin, and B. R. Hunt. Universal trellis coded quantization. *IEEE Trans. on Image Processing*, 8(12):1677–1687, December 1999.

[33] A. S. Lewis and G. Knowles. Image compression using the 2-D wavelet transform. *IEEE Trans. on Image Processing*, 1(2):244–250, April 1992.

[34] M. W. Marcellin, M. A. Lepley, A. Bilgin, T. J. Flohr, T. T. Chinen, and J. H. Kasner. An overview of quanziation in JPEG 2000. *Signal Processing: Image Communication*, 17(1):73–84, January 2002.

[35] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Kluwer Academic, New York, NY, USA, 1992.

[36] M. Rabbani and R. Joshi. An overview of the JPEG2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3–48, January 2002.

[37] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[38] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi. JPEG 2000 performance evaluation and assessment. *Signal Processing: Image Communication*, 17(1):113–130, January 2002.

[39] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[40] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, 1996.

[41] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal of Mathematical Analysis*, 29(2):511–546, March 1998.

[42] D. Taubman. High performance scalable image compression with EBCOT. In *Proc. of IEEE International Conference on Image Processing*, volume 3, pages 344–348, Kobe, Japan, 1999.

[43] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Trans. on Image Processing*, 9(7):1158–1170, July 2000.

[44] D. Taubman, E. Ordentlich, M. Weinberger, and G. Seroussi. Embedded block coding in JPEG 2000. *Signal Processing: Image Communication*, 17(1):49–72, January 2002.

[45] D. S. Taubman and M. W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic, Boston, MA, USA, 2002.

[46] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[47] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: Principles and standarization into JPEG-LS. *IEEE Trans. on Image Processing*, 9(8):1309–1324, August 2000.

[48] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[49] W. Zeng, S. Daly, and S. Lei. An overview of the visual optimization tools in JPEG 2000. *Signal Processing: Image Communication*, 17(1):85–104, January 2002.

Appendix B

Signal Processing Library (SPL)

Overview

Some of the code examples associated with this book employ the Signal Processing Library (SPL). This appendix provides a brief introduction to this library.

B.1 Signal Processing Library (SPL)

The Signal Processing Library (SPL) is an open-source C++ library, developed by the author of this book, that provides various functionalities related to signal and geometry processing. The library was originally developed in order to provide various multirate signal-processing capabilities. Over time, some geometry-processing and other functionality has also found its way into the library. The SPL library provides support for such things as:

- one- and two-dimensional arrays;
- one- and two-dimensional sequences; downsampling, upsampling, convolution; polyphase decomposition/re-composition;
- lowpass, highpass, bandpass filter design;
- reading and writing audio data;
- reading and writing image data;
- measuring execution time and memory usage; and
- binary and multisymbol arithmetic coding.

The library also has various other miscellaneous functionality (e.g., arcball, math utilities, and CGAL utilities). To provide some of its functionality, the SPL library utilizes the libsndfile [3], CGAL [2], and OpenGL [5] libraries. Each release of the library is accompanied by a detailed reference manual (e.g., [1]). Since the library is still evolving significantly from one release to the next, many technical details of the library (such as interfaces) remain subject to change. Consequently, rather the present detailed information about the library here, which would inevitably become out of date rather quickly, we refer the reader to the reference manual for the most recent version of the library for more details.

B.2 SPL Example Programs

To give the reader a flavor for some of the functionality in the SPL library, it is worthwhile to consider some sample code. In the sections that follow, we provide a few examples of programs that utilize the SPL library:

1. The `makeAudio` program, which produces an audio file and demonstrates some of the one-dimensional signal-processing capabilities of the SPL library.
2. The `mandelbrot` program, which produces an image corresponding to the Mandelbrot set and demonstrates some of the two-dimensional signal-processing capabilities of the SPL library.

B.2.1 `makeAudio` Program

The `makeAudio` program generates an audio signal and encodes the signal in an output file in the WAV format. This program illustrates the use of some of the one-dimensional signal-processing capabilities of the SPL library. The program consists of the single source file shown in Listing B.1.

Listing B.1: `makeAudio.cpp`

```
// Copyright (c) 2011, 2012, 2013 Michael D. Adams
// All rights reserved.

// __START_OF_LICENSE__
//
// Copyright (c) 2011, 2012, 2013 Michael D. Adams
// All rights reserved.
//
// This file is part of the Signal Processing Library (SPL).
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License as
// published by the Free Software Foundation; either version 3,
// or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public
// License along with this program; see the file LICENSE.  If not,
// see <http://www.gnu.org/licenses/>.
//
// __END_OF_LICENSE__

// This program makes an audio file.

////////////////////////////////////////////////////////////////////////////////
// Header files
////////////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <string>
#include <SPL/Sequence1.hpp>
#include <SPL/audioFile.hpp>

////////////////////////////////////////////////////////////////////////////////
// Types
////////////////////////////////////////////////////////////////////////////////

typedef SPL::Sequence1<double> RealSequence1;

```

```
////////////////////////////////////////////////////////////////////////////////
// Sequence generation
////////////////////////////////////////////////////////////////////////////////

// A 1-periodic square wave function.
double squareWave(double t)
{
    int i = 2.0 * t;
    return (!(i % 2)) ? 1.0 : 0.0;
}

// A 1-periodic sinusoidal function.
double sinusoid(double t)
{
    return cos(2.0 * M_PI * t);
}

double signal(int id, double t)
{
    double x;
    switch (id) {
    default:
    case 0:
        x = sinusoid(440.0 * t);
        break;
    case 1:
        x = squareWave(440.0 * t);
        break;
    case 2:
        x = 0.5 * squareWave(220.0 * t) + sinusoid(440.0 * t);
        break;
    }
    return x;
}

// Construct a sequence corresponding the the specified function
// sampled at the given rate for a particular number of samples.
void makeSequence(int id, int sampRate, int numSamps, RealSequence1& seq)
{
    seq = RealSequence1(0, numSamps);
    for (RealSequence1::Iterator i = seq.begin(); i != seq.end(); ++i) {
        *i = signal(id, static_cast<double>(i - seq.begin()) / sampRate);
    }
}

////////////////////////////////////////////////////////////////////////////////
// Main program
////////////////////////////////////////////////////////////////////////////////

int main(int argc, char** argv)
{
    RealSequence1 seq;

    // Process the command line.
    if (argc < 5) {
        std::cerr << "usage: makeAudio signalId sampRate duration outputFile\n";
        exit(2);
    }
    int id = atoi(argv[1]);
    int sampRate = atoi(argv[2]);
    double duration = atof(argv[3]);
    std::string outFile(argv[4]);

    // Generate the sequence.
    makeSequence(id, sampRate, duration * sampRate, seq);

    // Scale the range of the sequence to fit the allowable dynamic range.
    if (seq.getSize() > 0) {
        double minVal = *std::min_element(seq.begin(), seq.end());
        double maxVal = *std::max_element(seq.begin(), seq.end());
        double maxMag = std::max(SPL::absVal(minVal), SPL::absVal(maxVal));
        double alpha = 0.95;
        double beta = alpha * 1.0 / maxMag;
        for (RealSequence1::Iterator i = seq.begin(); i != seq.end(); ++i) {
            *i *= beta;
        }
    }

    // Save the sequence in an audio file.
    if (SPL::saveAudioFile(outFile, sampRate, seq.getArray())) {
        std::cerr << "cannot write audio file\n";
    }

    return 0;
}
```

B.2.2 `mandelbrot` Program

The `mandelbrot` program generates an image corresponding to the Mandelbrot set [4] and writes the image in the PNM format to standard output. This program illustrates the use of some of the two-dimensional signal-processing capabilities of the SPL library. The program consists of the single source file shown in Listing B.2. An example of an image generated by the program is given in Figure B.1.

Listing B.2: `mandelbrot.cpp`

```
// Copyright (c) 2013 Michael D. Adams
// All rights reserved.

// __START_OF_LICENSE__
//
// Copyright (c) 2011, 2012, 2013 Michael D. Adams
// All rights reserved.
//
// This file is part of the Signal Processing Library (SPL).
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License as
// published by the Free Software Foundation; either version 3,
// or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
```

```
//
// You should have received a copy of the GNU General Public
// License along with this program; see the file LICENSE.  If not,
// see <http://www.gnu.org/licenses/>.
//
// __END_OF_LICENSE__

// Generate a Mandelbrot dataset in PNM format.

////////////////////////////////////////////////////////////////////////////////
// Header files
////////////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <complex>
#include <algorithm>
#include <cassert>
#include <cstdlib>
#include <SPL/Array2.hpp>
#include <SPL/Sequence2.hpp>
#include <SPL/Timer.hpp>

////////////////////////////////////////////////////////////////////////////////
// Types
////////////////////////////////////////////////////////////////////////////////

typedef SPL::Array2<double> RealArray2;
typedef SPL::Array2<int> IntArray2;
typedef std::complex<double> Complex;

////////////////////////////////////////////////////////////////////////////////
// Mandelbrot dataset generation
////////////////////////////////////////////////////////////////////////////////

// Compute function representing Mandelbrot dataset.
RealArray2 mandelbrot(int width, int height, const Complex& botLeft,
  const Complex& topRight)
{
    const int maxIters = 128;

    RealArray2 result(width, height);
    result.fill(0);

    double stepX = (topRight.real() - botLeft.real()) / (width - 1);
    double stepY = (topRight.imag() - botLeft.imag()) / (height - 1);

    for (int y = 0; y < height; ++y) {
        for (RealArray2::XIterator i = result.rowBegin(y);
          i != result.rowEnd(y); ++i) {
            Complex c = botLeft + Complex((i - result.rowBegin(y)) *
              stepX, y * stepY);
            int n = 0;
            Complex z(0.0);
            while (std::abs(z) < 2.0 && n < maxIters) {
                z = z * z + c;
                ++n;
            }
            *i = static_cast<double>(n) / maxIters;
        }
    }
    return result;
}

// Randomly permute the gray levels in an image.
void permuteGrayLevels(IntArray2& image, int numGrayLevels)
{
    // Create a lookup table for randomly permuting sample values.
    std::vector<int> lookupTable(numGrayLevels);
    for (std::vector<int>::iterator i = lookupTable.begin();
      i != lookupTable.end(); ++i) {
        *i = i - lookupTable.begin();
    }
    std::random_shuffle(lookupTable.begin(), lookupTable.end());

    // Remap the gray levels in the image.
    for (IntArray2::Iterator i = image.begin();
      i != image.end(); ++i) {
        assert(*i >= 0 && *i < numGrayLevels);
        *i = lookupTable[*i];
    }
}

////////////////////////////////////////////////////////////////////////////////
// Main program.
////////////////////////////////////////////////////////////////////////////////

int main(int argc, char **argv)
{
    Complex botLeft(-2.05, -1.2);
    Complex topRight(0.55, 1.2);
    int width = 1024;
    int height = 1024;
    int maxValue = 255;

    // Process the command line.
    if (argc > 1) {
        if (argc != 5) {
            std::cerr << "usage: mandelbrot [xmin ymin xmax ymax]\n";
            exit(2);
        }
        botLeft = Complex(atof(argv[1]), atof(argv[2]));
        topRight = Complex(atof(argv[3]), atof(argv[4]));
    }

    // Compute the function representing the mandelbrot dataset.
    SPL::Timer timer;
    timer.start();
    RealArray2 func = mandelbrot(width, height, botLeft, topRight);
    timer.stop();

    // Convert the real data to integer data.
    func *= static_cast<double>(maxValue);
    IntArray2 image(func);

```

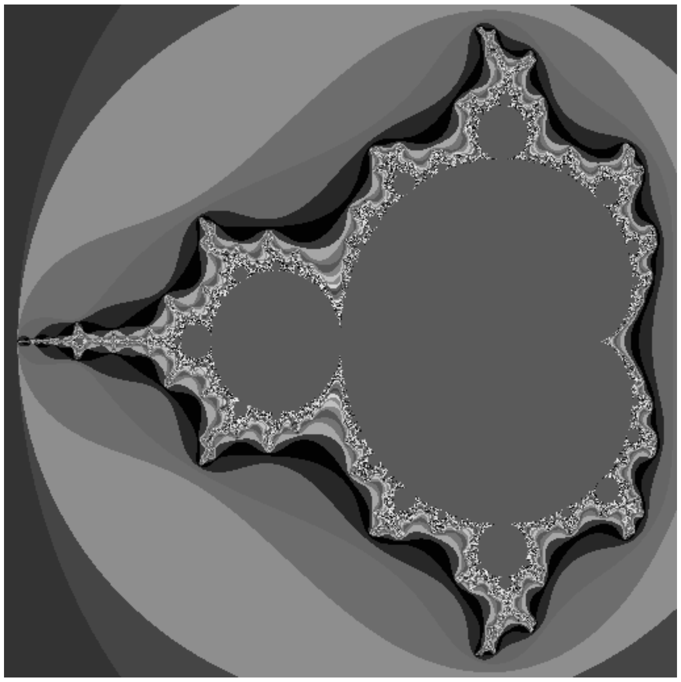


Figure B.1: An output image produced by the `mandelbrot` program.

```
    // Permute the gray levels (for aesthetic reasons).
    permuteGrayLevels(image, maxValue + 1);

    // Output image in PNM format.
    if (SPL::encodePgm(std::cout, image, maxValue, false)) {
        std::cerr << "cannot write output image\n";
        exit(1);
    }

    std::cerr << "Mandelbrot computation time (seconds): " << timer.get() << "\n";

    exit(0);
}
```

B.3 Bibliography

[1] M. D. Adams. *Signal Processing Library Reference Manual*, 2013. Available online from `http://www.ece.uvic.ca/~mdadams`.

[2] *CGAL User and Reference Manual: All Parts (Release 4.0.2)*, July 2012. Available online from `http://www.cgal.org`.

[3] E. de Castro Lopo. The official libsndfile web site. `http://www.mega-nerd.com/libsndfile/`, 2013.

[4] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Company, New York, NY, USA, 1983.

[5] M. Segal and K. Akeley, editors. *The OpenGL Graphics System: A Specification (Version 4.4 (Core Profile))*. July 2013.

Appendix C

Computational Geometry Algorithms Library (CGAL)

Overview

Some of the code examples associated with this book make use of the Computational Geometry Algorithms Library (CGAL). This appendix provides a brief introduction to this library.

C.1 Computational Geometry Algorithms Library (CGAL)

The Computational Geometry Algorithms Library (CGAL) [1, 2] is a very powerful open-source C++ library for geometric computation. The library provides a very extensive collection of data types and algorithms for use in geometry processing. Due to its heavy use of templates, the library is also quite flexible. Although the library is quite large and offers a great wealth of functionality, we will only focus on a relatively small subset of the library herein. In particular, we will consider the subset that is most relevant to the topics covered in this book. Of particular interest, the CGAL library provides:

- data types for representing various geometric objects (e.g., points, line segments, rays, lines, planes, and spheres) and algorithms for manipulating these data types;
- data types for representing polygon meshes (e.g., triangle/quadrilateral meshes) and algorithms for manipulating these data types;
- support for several subdivision schemes (e.g., Catmull-Clark, Loop, Kobbelt $\sqrt{3}$, and Doo-Sabin); and
- the ability to read and write polygon mesh data in various common formats (such as the OFF format).

By using CGAL, we can greatly reduce the amount of effort required to implement geometry-processing methods such as those associated with subdivision surfaces or subdivision wavelets.

The CGAL software supports a wide variety of platforms, including those based on UNIX/Linux and Microsoft Windows. Some Linux distributions already have packages defined for CGAL. For example, Fedora 17 has the yum packages: `CGAL`, `CGAL-devel`, and `CGAL-demos-source`. CGAL has a very large user base, and is used by many open-source projects as well as commercial products. Some examples of companies using CGAL include: British Telecom, Boeing, France Telecom, GE Health Care, and The MathWorks.

The CGAL software is very well documented, with an extensive manual [3] that is over 4000 pages in length. Since the manual is very detailed, we only provide a brief introduction to CGAL here. For additional information, the reader is referred to the CGAL manual. The material in the manual that is likely to be of most interest to the reader includes the following sections (in version 4.2 of the manual):

- Part IV “Geometry Kernels” Chapter 11 “2D and 3D Geometry Kernels”, which introduces the concept of a geometry kernel along with examples of several geometry kernels provided by the library;
- Part VII “Cell Complexes and Polyhedra” Chapter 25 “3D Polyhedral Surfaces”, which introduces the class used for representing polygon meshes;
- Part XII “Geometry Processing” Chapter 52 “3D Surface Subdivision Methods”, which introduces the subdivision methods provided by the library; and
- Part XVII “Support Library” Chapter 76 “Handles, Ranges, and Circulators”, which discusses some basic concepts such as circulators.

The part and chapter numbering in the manual tends to vary somewhat from one version of CGAL to another. The part and chapter titles, however, tend to change less frequently. Consequently, the part and chapter titles given above may be more helpful than the part and chapter numbers for locating the relevant material in the manual. Since the CGAL manual assumes some level of familiarity with concepts from computational geometry, the reader may also find the following references on computational geometry to be helpful: [4], [5], and [6].

The CGAL library consists of three major parts:

1. geometry kernels, which provide primitive geometric objects (e.g., points, lines, and planes) and operations on these objects;
2. geometric data structures (e.g., polygon meshes) and algorithms (e.g., subdivision schemes); and
3. non-geometric support facilities (e.g., support for debugging and interfacing CGAL to various visualization tools).

C.2 Handles and Circulators

Two important concepts in the CGAL library are handles and circulators. These two concepts are introduced below.

A **handle** is an object that can be used to reference another object (i.e., provides a dereferencing operator). Handles are frequently employed in order to provide access to objects stored inside a data structure. For example, for a data structure storing elements of type `T`, examples of handles include a simple pointer of type `T*` and an iterator with value type `T`. In the CGAL library, handle types are often used to refer to objects stored in data structures. For example, handles are used to refer to the constituent components of a polygon mesh (i.e., the vertices, facets, and halfedges).

Many libraries (including the C++ standard library) utilize iterators. While iterators are extremely useful, they are intended for use with linear sequences of elements (i.e., sequences with a well-defined first and last element). Iterators, however, are not well suited to circular sequences of elements. As it turns out, in geometry processing, circular sequences arise frequently. For this reason, CGAL introduces the notion of a circulator. A **circulator** is a type that allows iteration over elements in circular sequence of elements. For example, a circulator could be used to allow iteration over all 1-ring neighbours of vertex in polygon mesh. Just like iterators, circulators come in const (i.e., nonmutable) and non-const (i.e., mutable) versions. A non-const circulator can be used to modify the object that the circulator references, while a const circulator cannot.

C.3 Geometry Kernels

A fundamental building block in the CGAL library is what is known as a geometry kernel (which is sometimes simply referred to as a kernel). A geometry kernel provides data types for primitive geometric objects (such as points, lines, and planes) and operations on these objects. For example, a kernel specifies how a point is represented (e.g., Cartesian coordinates or homogeneous coordinates). This, in turn, effects how other primitive geometric objects are represented or specified. In addition to representing primitive geometric objects, kernels also provide various operations in relation to these objects. Two types of operations of particular interest are what are referred to as constructions and predicates. A **construction** (as the name suggests) is an operation that creates a new geometric object. For example, a construction might be used to create a line from two points, where the line is such that it passes through both points. A **predicate** is an operation that is used to test a particular condition and returns either a boolean or an enumerated type, where the possible values of the return type correspond to the different potential outcomes of the test. For example, a predicate might be used to test if three points are collinear. In this case, a boolean return value could be used with true and false corresponding to the points being or not being collinear, respectively.

Arithmetic with floating-point types is not exact, due to roundoff error. Unfortunately, this can lead to many complications when implementing geometric algorithms. For some types of calculations in geometric algorithms, any arbitrarily-small roundoff error could cause the algorithm to make an incorrect decision and fail to behave correctly. For example, due to roundoff error, an algorithm may incorrectly decide that three points are collinear when, in reality, they are not. In situations such as these, special techniques (e.g., arbitrary-precision arithmetic) must be employed in order to ensure correct code behavior. In effect, the calculation must be done as if no roundoff error occurred. In this regard, the geometry kernels provided by CGAL can be classified into one of three categories:

1. constructions and predicates are both inexact;
2. constructions are inexact but predicates are exact; and
3. constructions and predicates are both exact.

For our purposes, we are only interested in the first two categories. In situations where we need to be able guarantee that the results produced by predicates (such as a collinearity test) are correct in spite of roundoff error, we must use a kernel with exact predicates. Of course, the disadvantage of using exact predicates (relative to inexact ones) is greater computational cost. Therefore, exact predicates (or constructions) should only be used when required.

Although many geometric kernels are provided in the CGAL library, the only two that we consider are

`CGAL::Cartesian<double>` and `CGAL::Filtered_kernel<Cartesian<double>>`.

Both of these kernels represent points using Cartesian coordinates with the coordinates being of type `double`. The first of these kernels has inexact constructions and inexact predicates. The second of these kernels has inexact constructions and exact predicates. Which of these kernels should be selected depends on whether we need predicates to be exact.

C.4 Points and Vectors

Often, we tend to think of vectors and points as being essentially the same type of entity. The CGAL library, however, makes a clear distinction between vectors and points, treating them as distinct entities. As a consequence, in some contexts, CGAL may require a vector, while in other contexts, a point may be required. One can easily convert from a vector to a point and vice versa. The difference between two points is a vector. The origin plus a vector is point. So, one can convert between a vector and a point by either adding or subtracting `ORIGIN`, as shown in the following code fragment:

```
// Note: Kernel is some geometry kernel type
CGAL::Point_3<Kernel> p(1.0, 2.0, 3.0);
CGAL::Point_3<Kernel> q;
CGAL::Vector_3<Kernel> v;
v = p - CGAL::ORIGIN; // convert Point_3 to Vector_3
q = CGAL::ORIGIN + v; // convert Vector_3 to Point_3
```

C.5 `Polyhedron_3` Class

With the CGAL library, a polyhedral surface (i.e., polygon mesh), which consists of vertices, edges, and facets, and incidence relationship amongst them, is represented using the `Polyhedron_3` template class. This class is based on the half-edge data structure, described in detail in Section 8.17.3 (on page 358).

The `Polyhedron_3` class is declared as follows:

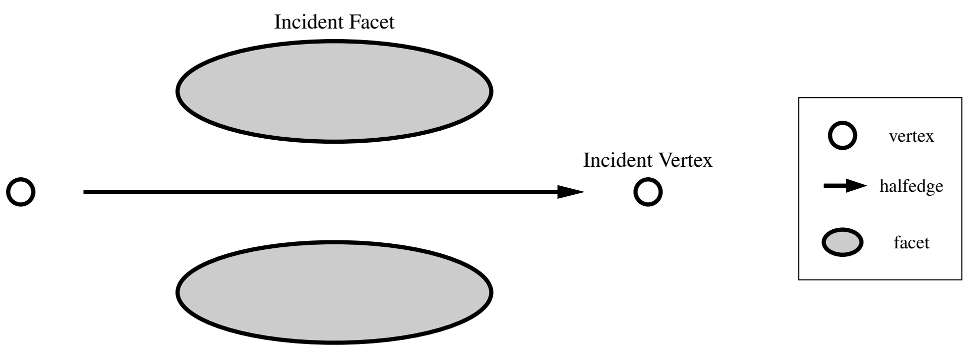


Figure C.1: A halfedge and its incident vertex and incident facet.

```
template <
  class Kernel,
  class PolyhedronItems = CGAL::Polyhedron_items_3,
  template <class T, class I> class HalfedgeDS = CGAL::HalfedgeDS_default,
  class Alloc = CGAL_ALLOCATOR(int)
> class Polyhedron_3;
```

The parameter `Kernel` specifies the geometric kernel to be used (such as `CGAL::Cartesian<double>`). The parameter `PolyhedronItems` identifies the data types to be used for representing vertices and facets. In many cases, the default value for this parameter will suffice. The parameter `HalfedgeDS` specifies the particular halfedge data structure used to represent the mesh. The default value for this parameter should always suffice for our purposes. The parameter `Alloc` is used to control how memory allocation is performed. The default value for this parameter should always suffice for our purposes.

Each halfedge is associated with one vertex and nominally one facet, referred to as the incident vertex and incident facet, respectively. The **incident vertex** of a halfedge is the vertex at the terminal end of the halfedge. The **incident facet** of a halfedge is the facet to the left of the halfedge. The relationship between a halfedge and its incident vertex and incident facet is illustrated in Figure C.1. If a halfedge belongs to an edge on the boundary of the mesh, the halfedge may not have an incident facet, as the halfedge may have no facet to its left. A halfedge that has no left facet is called a **boundary halfedge**. For a given edge on the boundary of a mesh, only one of the edge's two halfedges will be a boundary halfedge.

The `Polyhedron_3` class provides a number of type members, which are listed in Table C.1. The function members provided by the class are listed in Table C.2. The input and output operators (i.e., `operator<<` and `operator>>`) are overloaded for `Polyhedron_3` objects so that meshes can be easily read from and written to streams (e.g., in the OFF format).

C.5.1 `Polyhedron_3::Facet` Class

The `Facet` class is used to represent a facet (i.e., face) in the polygon mesh. This class may optionally store a plane equation (for the facet) and a reference to a halfedge incident to the facet. The class provides a circulator that can be used to visit all halfedges incident on the facet. The circulator can be either of the forward or bidirectional variety. The function members for the class are listed in Table C.3.

C.5.2 `Polyhedron_3::Vertex` Class

The `Vertex` class is used to represent a vertex in a polygon mesh. The vertex may optionally include a point (corresponding to the vertex position in space) and a reference to a halfedge that is incident on the vertex. The class provides a circulator that can be used to visit all of the halfedges incident on the vertex. The circulator can be either of the forward or bidirectional variety. The function members for the class are listed in Table C.4.

Table C.1: Type members for the `Polyhedron_3` class. (a) Basic types, (b) handle types, (c) iterator types, and (d) circulator types.

(a)

Name	Description
`Vertex`	vertex type
`Halfedge`	halfedge type
`Facet`	facet type
`Point_3`	point type (for vertices)

(b)

Name	Description
`Vertex_const_handle`	const handle to vertex
`Vertex_handle`	handle to vertex
`Halfedge_const_handle`	const handle to halfedge
`Halfedge_handle`	handle to halfedge
`Facet_const_handle`	const handle to facet
`Facet_handle`	handle to facet

(c)

Name	Description
`Vertex_const_iterator`	const iterator over all vertices
`Vertex_iterator`	iterator over all vertices
`Halfedge_const_iterator`	const iterator over all halfedges
`Halfedge_iterator`	iterator over all halfedges
`Facet_const_iterator`	const iterator over all facets
`Facet_iterator`	iterator over all facets
`Edge_const_iterator`	const iterator over all edges (every other halfedge)
`Edge_iterator`	iterator over all edges (every other halfedge)

(d)

Name	Description
`Halfedge_around_vertex_const_circulator`	const circulator of halfedges around vertex (CW)
`Halfedge_around_vertex_circulator`	circulator of halfedges around vertex (CW)
`Halfedge_around_facet_const_circulator`	const circulator of halfedges around facet (CCW)
`Halfedge_around_facet_circulator`	circulator of halfedges around facet (CCW)

Table C.2: Function members for the `Polyhedron_3` class. Function members related to (a) size, (b) iterators, (c) combinatorial predicates, and (d) border halfedges.

(a)

Name	Description
`size_of_vertices`	get number of vertices
`size_of_halfedges`	get number of halfedges
`size_of_facets`	get number of facets

(b)

Name	Description
`vertices_begin`	iterator over all vertices
`vertices_end`	past-the-end iterator
`halfedges_begin`	iterator over all halfedges
`halfedges_end`	past-the-end iterator
`facets_begin`	iterator over all facets
`facets_end`	past-the-end iterator
`edges_begin`	iterator over all edges
`edges_end`	past-the-end iterator

(c)

Name	Description
`is_closed`	true if no border edges (no boundary)
`is_pure_triangle`	true if all facets are triangles
`is_pure_quad`	true if all facets are quadrilaterals

(d)

Name	Description
`normalized_border_is_valid`	true if border is normalized
`normalize_border`	sort halfedges such that non-border edges precede border edges (i.e., normalize border)
`size_of_border_halfedges`	get number of border halfedges (border must be normalized)
`size_of_border_edges`	get number of border edges (border must be normalized)
`border_halfedges_begin`	halfedge iterator starting with border edges (border must be normalized)
`border_edges_begin`	edge iterator starting with border edges (border must be normalized)

Table C.3: Function members for `Facet` class.

Name	Description
`halfedge`	get incident halfedge that points to facet
`facet_begin`	get circulator of halfedges around facet (CCW)
`facet_degree`	get degree of facet (i.e., number of edges on boundary of facet)
`is_triangle`	true if facet is triangle
`is_quad`	true if facet is quadrilateral

Table C.4: Function members for the `Vertex` class. (a) Required and (b) optional.

(a)

Name	Description
`vertex_begin`	circulator of halfedges around vertex (CW)
`vertex_degree`	get valence of vertex
`is_bivalent`	true if vertex has valence two
`is_trivalent`	true if vertex has valence three

(b)

Name	Description
`point`	get point associated with vertex
`halfedge`	get incident halfedge that points to vertex

C.5.3 `Polyhedron_3::Halfedge` Class

The `Halfedge` class is used to represent a halfedge in a polygon mesh. At the implementation level, a halfedge contains a handle for the next halfedge around its left facet in the CCW direction and a handle for its opposite halfedge. This allows for efficient iteration around the halfedges of a facet in the CCW direction and around the halfedges of a vertex in the CW direction. Additional information may optionally be stored in halfedges in order to permit efficient iteration in the opposite direction. The function members for the `Halfedge` class are listed in Table C.5.

Example C.1 (Adjacency example)**.** Figure C.2 shows a simple triangle mesh. In the figure, one halfedge is labelled as “`h`”. In addition, various other parts of the mesh have been labelled with expressions showing how the various halfedge member functions can be used to navigate around the mesh.

C.6 Support for Subdivision Methods

The CGAL library provides support for several subdivision methods. The code for subdivision is contained in the namespace `Subdivision_method_3` (in the `CGAL` namespace). Several subdivision methods are available through the functions listed in Table C.6. Some generic subdivision methods (for which arbitrary geometric refinement rules can be defined) are associated with the functions listed in Table C.7.

C.7 Other Functionality

A few other data types, constants, and functions in CGAL that may be useful are listed in Tables C.8, C.9, and C.10, respectively.

C.8 Common Problems

Below, we identify a few problems commonly encountered by new CGAL users. In so doing, we hope to help new users to avoid such problems.

When using CGAL, it is quite important that code be const correct. A common source of problems is using a handle, iterator, or circulator type that has the wrong constness properties. For example, if one is looping over all of the vertices in the mesh using a `Vertex_const_iterator`, then one cannot try to obtain a `Halfedge_around_vertex_circulator` from the vertex obtained by dereferencing the iterator (as this would provide a means by which to change a const object). That is, code like the following is incorrect:

```
typedef CGAL::Cartesian<double> Kernel;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
Polyhedron mesh;
```

Table C.5: Function members for the `Halfedge` class. Function members related to (a) adjacency queries, (b) circulators, (c) valence/degree/border queries, and (d) the incident vertex/facet.

(a)

Name	Description
`opposite`	get opposite halfedge
`next`	next halfedge around facet (CCW)
`prev`	previous halfedge around facet (CCW)
`next_on_vertex`	next halfedge around vertex (CW)
`prev_on_vertex`	previous halfedge around vertex (CW)

(b)

Name	Description
`vertex_begin`	circulator of halfedges around vertex (CW)
`facet_begin`	circulator of halfedges around facet (CCW)

(c)

Name	Description
`is_border`	true if border halfedge
`vertex_degree`	get valence of incident vertex
`is_bivalent`	true if incident vertex has valence two
`is_trivalent`	true if incident vertex has valence three
`facet_degree`	get degree of incident facet
`is_triangle`	true if incident facet is triangle
`is_quad`	true if incident facet is quadrilateral

(d)

Name	Description
`vertex`	get handle for incident vertex of halfedge
`facet`	get handle for incident facet of halfedge

Table C.6: Functions for various subdivision methods

Name	Description
`CatmullClark_subdivision`	perform Catmull-Clark subdivision
`Loop_subdivision`	perform Loop subdivision
`DooSabin_subdivision`	perform Doo-Sabin subdivision
`Sqrt3_subdivision`	perform Kobbelt $\sqrt{3}$ subdivision

Table C.7: Functions for generic subdivision methods

Name	Description
`PQQ`	primal quadrilateral quadrisection with user-defined geometric refinement rule
`PTQ`	primal triangle quadrisection with user-defined geometric refinement rule
`DQQ`	dual quadrilateral quadrisection with user-defined geometric refinement rule
`Sqrt3`	$\sqrt{3}$ topologic refinement with user-defined geometric refinement rule

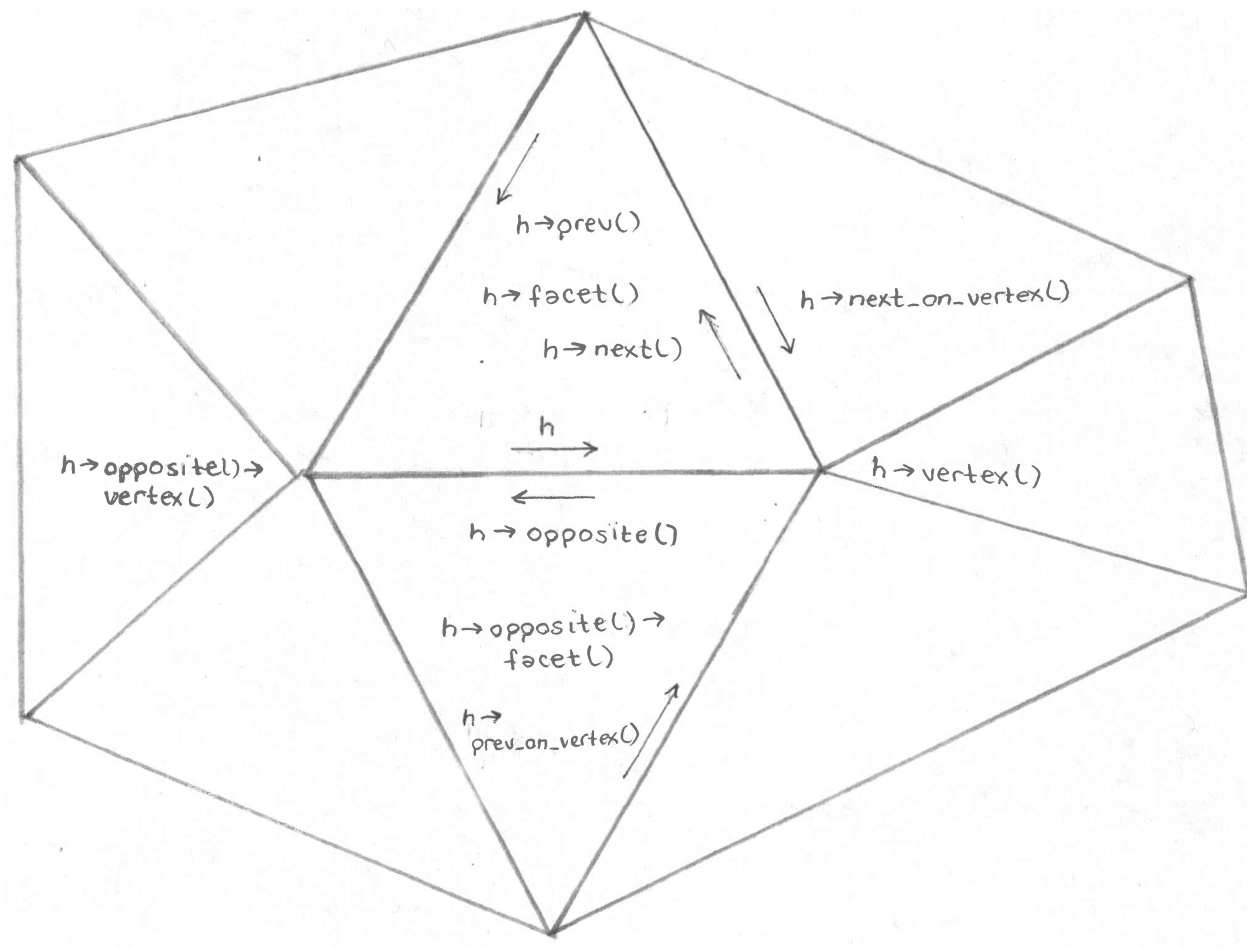


Figure C.2: Mesh for adjacency example.

Table C.8: Miscellaneous types

Name	Description
Aff_transformation_3	three-dimensional affine transformation
Bbox_3	bounding box in three-dimensional space
Direction_3	direction in three-dimensional space
Line_3	directed straight line in three-dimensional space
Point_3	point in three-dimensional space
Plane_3	oriented plane in three-dimensional space
Ray_3	directed straight ray in three-dimensional space
Segment_3	directed straight line segment in three-dimensional space
Sphere_3	oriented sphere in three-dimensional space
Triangle_3	triangle (excluding interior) in three-dimensional space
Vector_3	vector in three-dimensional space

Table C.9: Miscellaneous constants

Name	Description
ORIGIN	point at origin

Table C.10: Miscellaneous functions

Name	Description
`collinear`	test if three points are collinear
`coplanar`	test if four points are coplanar
`cross_product`	compute cross product of two vectors
`left_turn`	test if three points form left turn (i.e., form a CCW loop)
`parallel`	test if segments/rays/lines/planes are parallel
`right_turn`	test if three points form right turn (i.e., form a CW loop)

```
// ...
for (Polyhedron::Vertex_const_iterator i = mesh.vertices_begin();
  i != mesh.vertices_end(); ++i) {
    Polyhedron::Halfedge_around_vertex_circulator circ =
      i->vertex_begin(); // ERROR: violates const correctness
        // should use Halfedge_around_vertex_const_circulator
    // ...
}
```

Be careful to take into account which operations on `Polyhedron_3` objects can invalidate handles, iterators, or circulators. For example, when one adds new vertices/facets to a mesh, this has the potential to invalidate references (through handles, iterators, and circulators) to elements in the mesh. Not taking issues like this into consideration can lead to some very painful times debugging code.

Some functions for the `Polyhedron_3` class will only work correctly if the border is normalized. For example, the following member functions are only guaranteed to have the correct behavior if the border is normalized:

- `size_of_border_halfedges`,
- `size_of_border_edges`,
- `border_halfedges_begin`, and
- `border_edges_begin`.

The member function `normalized_border_is_valid` can be used to determine if the border is normalized. To normalize the border, the member function `normalize_border` should be used.

In the `Polyhedron_3` class, some circulators move in the CCW direction while others move in the CW direction. Depending on how a circulator is being used, this order may be important.

In the `Polyhedron_3` class, each edge is associated with two halfedges. If an edge is on the border of the mesh, only *one* of the edge's two halfedges will be a border halfedge. Note the emphasis on the word "one" in the previous sentence. The common mistake here is to assume that because a halfedge belongs to a border edge that the halfedge must be a border halfedge. This assumption is incorrect, however.

C.9 CGAL Example Programs

Now, we consider a few examples of programs that utilize the CGAL library. In particular, the source code listings for the following programs can be found in the sections that follow:

1. The `meshMake` program, which creates a polygon mesh.
2. The `meshInfo` program, which prints some information about a mesh.
3. The `meshSubdivide` program, which applies subdivision to a mesh.

C.9.1 `meshMake` Program

The `meshMake` program creates a polygon mesh corresponding to a tetrahedron and then outputs the resulting mesh in OFF format to standard output. The program consists of the single source code file shown in Listing C.1.

Listing C.1: `meshMake.cpp`

```
// Construct a simple mesh and write the result to standard output
// in OFF format.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <CGAL/Simple_cartesian.h>
#include <CGAL/Polyhedron_3.h>
#include <CGAL/IO/Polyhedron_iostream.h>
#include <CGAL/Polyhedron_incremental_builder_3.h>

typedef double Real;
typedef CGAL::Simple_cartesian<Real> Kernel;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
typedef Polyhedron::HalfedgeDS HalfedgeDS;

// A modifier class that creates a tetrahedron with the polyhedron
// incremental builder class.
template <class HDS>
class MeshBuilder : public CGAL::Modifier_base<HDS> {
public:

    MeshBuilder() {}

    void operator()(HDS& hds)
    {
        typedef typename HDS::Vertex Vertex;
        typedef typename Vertex::Point Point;

        // Start with an empty mesh.
        hds.clear();

        CGAL::Polyhedron_incremental_builder_3<HDS> b(hds, true);

        // Start the construction of a new surface with the specified
        // number of vertices and faces.
        b.begin_surface(4, 4);

        // Add four vertices to mesh.
        b.add_vertex(Point( 1.0,  1.0,  1.0)); // Add vertex 0
        b.add_vertex(Point(-1.0,  1.0,  1.0)); // Add vertex 1
        b.add_vertex(Point(-1.0, -1.0,  1.0)); // Add vertex 2
        b.add_vertex(Point( 1.0, -1.0, -1.0)); // Add vertex 3

        // Add facet with vertices 0,1,2 (in CCW order).
        b.begin_facet();
        b.add_vertex_to_facet(0);
        b.add_vertex_to_facet(1);
        b.add_vertex_to_facet(2);
        b.end_facet();

        // Add facet with vertices 0,2,3 (in CCW order).
        b.begin_facet();
        b.add_vertex_to_facet(0);
        b.add_vertex_to_facet(2);
        b.add_vertex_to_facet(3);
        b.end_facet();

        // Add facet with vertices 0,3,1 (in CCW order).
        b.begin_facet();
        b.add_vertex_to_facet(0);
        b.add_vertex_to_facet(3);
        b.add_vertex_to_facet(1);
        b.end_facet();

        // Add facet with vertices 3,2,1 (in CCW order).
        b.begin_facet();
        b.add_vertex_to_facet(3);
        b.add_vertex_to_facet(2);
        b.add_vertex_to_facet(1);
        b.end_facet();

        // End the construction of a surface.
        b.end_surface();
    }

};

int main()
{
    Polyhedron mesh;

    // Create a mesh builder object.
    MeshBuilder<HalfedgeDS> meshBuilder;

    // Replace the existing mesh with a new mesh produced by the
    // mesh builder object.
    mesh.delegate(meshBuilder);

    // Output the mesh to standard output (in OFF format).
    std::cout << mesh;
    if (!std::cout) {
        return 1;
    }

    return 0;
}
```

C.9.2 `meshInfo` Program

The `meshInfo` program reads a polygon mesh in OFF format from standard input and then calculates and prints some information about the mesh (e.g., triangle or quad mesh; the number of vertices, edges, facets, and halfedges; and the minimum, maximum, and average vertex valence). The program consists of the single source code file shown in Listing C.2.

Listing C.2: `meshInfo.cpp`

```
// Read a mesh from standard input in OFF format and print various information
// about the mesh to standard output.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <string>
#include <CGAL/Cartesian.h>
#include <CGAL/Filtered_kernel.h>
#include <CGAL/Polyhedron_3.h>
#include <CGAL/IO/Polyhedron_iostream.h>

typedef double Real;
typedef CGAL::Cartesian<Real> Kernel0;
// Use a filtered kernel so that all predicates are exact.
typedef CGAL::Filtered_kernel<Kernel0> Kernel;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
typedef Kernel::Point_3 Point;

int main(int argc, char **argv)
{
    Polyhedron mesh;

    // Read the input mesh from standard input in OFF format.
    if (!(std::cin >> mesh)) {
        std::cerr << "Cannot read input mesh\n";
        return 1;
    }

    // Determine the mesh type.
    std::string meshType;
    if (mesh.is_pure_triangle()) {
        meshType = std::string("triangle");
    } else if (mesh.is_pure_quad()) {
        meshType = std::string("quad");
    } else {
        meshType = std::string("general");
    }

    // Loop over all of the vertices in the mesh.
    // In the process of doing so, compute the minimum, maximum, and average
    // valences of the vertices in the mesh.
    Real valenceSum = 0;
    int minValence = -1;
    int maxValence = -1;
    // For each vertex in the mesh...
    for (Polyhedron::Vertex_const_iterator vertexIter = mesh.vertices_begin();
      vertexIter != mesh.vertices_end(); ++vertexIter) {

        // Get the valence of the current vertex.
        int valence = vertexIter->degree();

        // Update the minimum valence value.
        if (minValence < 0 || valence < minValence) {
            minValence = valence;
        }

        // Update the maximum valence value.
        if (maxValence < 0 || valence > maxValence) {
            maxValence = valence;
        }

        valenceSum += valence;

    }
    Real meanValence = valenceSum / mesh.size_of_vertices();

    // Check for nonplanar faces.
    int numNonplanarFaces = 0;
    Real degreeSum = 0;
    int minDegree = -1;
    int maxDegree = -1;
    // For each face in the mesh...
    for (Polyhedron::Facet_const_iterator faceIter = mesh.facets_begin();
      faceIter != mesh.facets_end(); ++faceIter) {

        // Get the degree of the face.
        int degree = faceIter->facet_degree();
        // The face can only be nonplanar if its degree exceeds three.
        if (degree >= 4) {

            // Get a circulator that can be used to visit all of the
            // halfedges around the face in CCW order (where the halfedge
            // has the face on its left side).
            Polyhedron::Facet::Halfedge_around_facet_const_circulator
              halfEdgeCirc = faceIter->facet_begin();

            // Get the first vertex of the face.
            const Point& v0 = halfEdgeCirc->vertex()->point();
            ++halfEdgeCirc;

            // Get the second vertex of the face.
            const Point& v1 = halfEdgeCirc->vertex()->point();
            ++halfEdgeCirc;

            // Get the third vertex of the face.
            const Point& v2 = halfEdgeCirc->vertex()->point();
            ++halfEdgeCirc;

            // Check that each remaining vertex is coplanar with the first
            // three vertices.
            for (int i = 3; i < degree; ++i) {

                // Get the next vertex of the face.
                const Point& v = halfEdgeCirc->vertex()->point();
                ++halfEdgeCirc;

                // Check if the vertex is coplanar with the first three.
                if (!CGAL::coplanar(v0, v1, v2, v)) {
                    ++numNonplanarFaces;
                    std::cout << "nonplanar face detected: "
                      << "(" << v0.x() << "," << v0.y() << "," << v0.z() << ") "
                      << "(" << v1.x() << "," << v1.y() << "," << v1.z() << ") "
                      << "(" << v2.x() << "," << v2.y() << "," << v2.z() << ") "
                      << "(" << v.x() << "," << v.y() << "," << v.z() << ")\n";
                }
            }
        }

        // Update the minimum degree.
        if (minDegree < 0 || degree < minDegree) {
            minDegree = degree;
        }

        // Update the maximum degree.
        if (maxDegree < 0 || degree > maxDegree) {
            maxDegree = degree;
        }

        degreeSum += degree;

    }
    Real meanDegree = degreeSum / mesh.size_of_facets();

    // Normalize the halfedges of the mesh.
    // That is, sort the halfedges so that the non-border edges precede the
    // border edges.
    // This is necessary since the member function size_of_border_edges
    // (which is called below) requires that the halfedges be normalized.
    mesh.normalize_border();

    // Output the mesh information.
    std::cout
      << "mesh type: " << meshType << "\n"
      << "number of vertices: " << mesh.size_of_vertices() << "\n"
      << "number of edges: " << mesh.size_of_halfedges() / 2 << "\n"
      << "number of border edges: " << mesh.size_of_border_edges() << "\n"
      << "number of faces: " << mesh.size_of_facets() << "\n"
      << "number of halfedges: " << mesh.size_of_halfedges() << "\n"
      << "mean vertex valence: " << meanValence << "\n"
      << "minimum vertex valence: " << minValence << "\n"
      << "maximum vertex valence: " << maxValence << "\n"
      << "mean face degree: " << meanDegree << "\n"
      << "minimum face degree: " << minDegree << "\n"
      << "maximum face degree: " << maxDegree << "\n"
      << "number of nonplanar faces: " << numNonplanarFaces << "\n"
      ;

    return 0;
}
```

C.9.3 `meshSubdivide` Program

The `meshSubdivide` program reads a polygon mesh from standard input, applies subdivision to the mesh, and then writes the refined mesh to standard output. Mesh data is read and written in the OFF format. The program allows the subdivision method and number of levels of subdivision to be specified. Several subdivision methods are supported, including the Catmull-Clark, Loop, and Kobbelt $\sqrt{3}$ schemes. The program consists of the single source code file shown in Listing C.3.

Listing C.3: `meshSubdivide.cpp`

```
// Read a mesh from standard input in OFF format, subdivide the mesh,
// and write the subdivided mesh to standard output in OFF format.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <string>
#include <CGAL/Cartesian.h>
#include <CGAL/Subdivision_method_3.h>
#include <CGAL/Polyhedron_3.h>
#include <CGAL/IO/Polyhedron_iostream.h>

typedef double Real;
typedef CGAL::Cartesian<Real> Kernel;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;

int main(int argc, char **argv)
{
    // If the number of command line arguments provided is incorrect,
    // print usage information and exit.
    if (argc != 3) {
        std::cout
          << "Usage:\n"
          << argv[0] << " method numLevels\n"
          << "method ...... The subdivision method.\n"
          << "numLevels ... The number of levels of subdivision.\n"
          << "Valid subdivision methods include:\n"
          << "    loop, sqrt3, catmull_clark, doo_sabin\n";
        std::cout
          << "The mesh is read from standard input in OFF format.\n"
          << "The refined mesh is written to standard output in OFF format.\n";
        return 2;
    }

    // Extract the input parameters from the command line.
    std::string method = argv[1];
    int numLevels = atoi(argv[2]);

    Polyhedron mesh;

    // Read the input mesh from standard input in OFF format.
    if (!(std::cin >> mesh)) {
        std::cerr << "Cannot read input mesh\n";
        return 1;
    }

    // Apply the specified subdivision method to the mesh.
    if (method == "loop") {
        if (!mesh.is_pure_triangle()) {
            std::cerr << "Loop subdivision requires a triangle mesh.\n";
            return 1;
        }
        CGAL::Subdivision_method_3::Loop_subdivision(mesh, numLevels);
    } else if (method == "sqrt3") {
        if (!mesh.is_pure_triangle()) {
            std::cerr <<
              "Kobbelt sqrt(3) subdivision requires a triangle mesh.\n";
            return 1;
        }
        CGAL::Subdivision_method_3::Sqrt3_subdivision(mesh, numLevels);
    } else if (method == "catmull_clark") {
        CGAL::Subdivision_method_3::CatmullClark_subdivision(mesh, numLevels);
    } else if (method == "doo_sabin") {
        if (!mesh.is_pure_quad()) {
            std::cerr << "Doo-Sabin subdivision requires a quad mesh.\n";
            return 1;
        }
        CGAL::Subdivision_method_3::DooSabin_subdivision(mesh, numLevels);
    } else {
        std::cerr << "The specified subdivision scheme is invalid.\n";
        return 1;
    }

    // Write the refined mesh to standard output in OFF format.
    std::cout << mesh;
    if (!std::cout) {
        return 1;
    }

    return 0;
}
```

C.10 Bibliography

[1] Computational geometry algorithms library (CGAL) home page, 2012. `http://www.cgal.org`.

[2] *CGAL User and Reference Manual: All Parts (Release 4.0.2)*, July 2012. Available online from `http://www.cgal.org`.

[3] *CGAL User and Reference Manual: All Parts (Release 4.2)*, April 2013. Available online from `http://www.cgal.org`.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, New York, NY, USA, 2nd edition, 2000.

[5] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1998.

[6] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, USA, 1985.

Appendix D

Open Graphics Library (OpenGL)

Overview

Some of the code examples associated with this book employ the Open Graphics Library (OpenGL). This appendix provides a brief introduction to this library.

D.1 Open Graphics Library (OpenGL)

The **Open Graphics Library** (OpenGL) [1, 6] is an application programming interface (API) for high-performance high-quality 2-D and 3-D graphics. This API has been widely adopted by industry and has become the de-facto standard for 2-D and 3-D graphics. OpenGL has bindings for numerous programming languages, including C, Java, and Fortran. Herein, however, we will consider only the binding for C. The OpenGL API is both operating-system and windowing-system independent, and implementations of the library are available for all mainstream computing platforms, including UNIX, Linux, Mac OS X, and Microsoft Windows. OpenGL is vendor neutral, controlled by an independent consortium with many organizations as members, including (not surprisingly) companies like Intel, NVIDIA, and AMD. (As an aside, we note that there is another specification of an API for graphics called OpenGL ES that is intended for use in embedded systems, such as mobile phones, game consoles, personal navigation devices, personal media players, automotive systems, and set-top boxes. This is a separate API specification from OpenGL, however, and is not discussed here.)

The OpenGL API is comprised of several hundred functions. OpenGL provides a number of geometric primitives such as points, lines, polygons, images, and bitmaps. Using the library, one can render a scene composed of the arrangement of geometric primitives in 3-D space, viewed from a particular vantage point. The rendering process is quite complicated. Fortunately, many of the complexities and technical details are hidden inside the library so that the application programmer need not worry about them. For example, the library assumes responsibility for calculating the colors of objects (e.g., by explicit assignment, lighting, texture mapping, or a combination thereof); converting the mathematical description of the objects to pixels on the screen (i.e., rasterization); eliminating hidden parts of objects (via depth buffering), performing antialiasing, and so on. It is important to note, however, that OpenGL only concerns itself with rendering. For example, OpenGL does not provide any capabilities for window management (e.g., creation and destruction of windows) or device management (e.g., for obtaining user input via a keyboard or mouse). Another auxiliary library must be used in conjunction with OpenGL in order to manage windows, handle user input, and so on.

D.2 OpenGL Utility Toolkit (GLUT)

As mentioned above, since OpenGL only concerns itself with rendering, another auxiliary library must be used along with OpenGL to provide other key functionalities required by graphics applications, such as window management.

Although a number of such auxiliary libraries are available, we will consider only one herein, namely, the OpenGL Utility Toolkit (GLUT).

The **OpenGL Utility Toolkit** (GLUT) [5] is a C library that provides a simple windowing API for use with OpenGL. Since GLUT is fairly basic in its capabilities, it is mainly intended for use with small to medium sized OpenGL programs. The GLUT library is operating-system and windowing-system independent, and supports most mainstream operating systems (e.g., UNIX, Linux, and Microsoft Windows). Some of the functionality provided by the library includes:

- window management: creating and destroying windows, displaying and resizing windows, and setting and querying window attributes;
- event management: provides an event-processing engine with a dispatcher for callback functions to handle various types of events (e.g., display, reshape, keyboard, and mouse events);
- device management: allows for interfacing to input devices such as keyboards and mice;
- graphics support to supplement OpenGL functionality: routines for drawing common wireframe/solid 3-D objects such as a sphere, torus, and the well-known teapot model.

An open-source implementation of GLUT called Freeglut is available from

```
http://sourceforge.net/projects/freeglut
```

All of the header files for GLUT (as well as OpenGL) are located in the directory `GL`. To use GLUT, one need only include the single header file `glut.h`, with a directive like:

```
#include <GL/glut.h>
```

The header file `glut.h` also includes all of the necessary OpenGL header files (e.g., `gl.h`, `glu.h`, and `glext.h`). So, when writing OpenGL applications with GLUT, the only header file from the OpenGL and GLUT libraries that need be included is `glut.h`.

D.2.1 Structure of GLUT Applications

Although programs can be structured in many different ways, one way in which a program can be structured is to use what is called an event-driven model. With an event-driven model, the flow of the program is completely determined by events (such as key presses or mouse clicks). A program that makes use of an event-driven model typically performs some initialization and then enters an event-processing loop for the duration of execution. Each iteration of the event-processing loop simply does the following: 1) wait for an event; and 2) process the event. An event might, for example, correspond to a key press or mouse button press/release. Many libraries for building graphical user interfaces (GUIs) employ an event-driven model.

Not surprisingly, the GLUT library uses an event-driven model. The types of events supported by the library include those listed in Table D.1. From this table, one can see that many of the event types are either directly or indirectly related to interactions between the user and the windowing system (e.g., pressing a mouse button, pressing a key on the keyboard, or resizing a window). A list of some of the functions provided by the library is provided in Table D.2. Generally, a GLUT program consists of the following steps:

1. Initialize the GLUT library by calling the `glutInit` function.
2. Set the display mode using the `glutInitDisplay` function.
3. Perform any additional initialization such as:
   - creating windows via the `glutCreateWindow` function;
   - registering callback functions for handling various types of events (e.g., via `glutDisplayFunc`, `glutReshapeFunc`, and `glutKeyboardFunc`);

Table D.1: GLUT event types

Event Type	Description
display	window contents needs to be displayed
overlay display	overlay plane contents needs to be displayed
reshape	window has been resized
keyboard	key has been pressed
mouse	mouse button has been pressed or released
motion	mouse moved within window while one or more buttons pressed
passive motion	mouse moved within window while no buttons pressed
visibility	visibility of window has changed (covered versus uncovered)
entry	mouse has left or entered window
special keyboard	special key has been pressed (e.g., arrow keys, function keys)
spaceball motion	spaceball translation has occurred
spaceball rotate	spaceball rotation has occurred
button box	button box activity has occurred
dials	dial activity has occurred
tablet motion	tablet motion has occurred
tablet button	table button has been pressed or released
menu status	menu status change
idle	no event activity has occurred
timer	timer has expired

- set the initial OpenGL state (e.g., depth buffering, shading, lighting, and clear color).

4. Enter the main event-processing loop by calling the `glutMainLoop` function. Note that the `glutMainLoop` function never returns.

The above steps typically constitute only a very small part of a GLUT application. This is because most of the work is performed in the callback functions of the application. For each event type of interest to an application, a callback function must be registered. As an absolute minimum, an application must register a display callback function (via `glutDisplayFunc`). This is because the display event is the event that requests an application to draw its graphics output. Therefore, if no display callback is registered, there is no way for an application to produce any graphics output. Since windows are often resized (and applications usually want to know when their window size changes), it is almost always the case that a reshape callback function will be registered. Whether other types of callback functions are registered depends on the particular application.

D.2.2 Example of a Minimalist GLUT Application

In this section, we provide an example of a minimalist GLUT application. The program simply has the effect of clearing the graphics output window to a particular color. Although this program does not do anything particularly exciting, it nevertheless well illustrates the basic structure of a GLUT application. The program consists of the single source code file shown in Listing D.1. The output produced by the program is shown in Figure D.1. In the program, the only callback function registered is for the handling of display events. Recall that, as a minimum, a GLUT application must register a display callback function; otherwise, the application cannot produce any graphics output.

The functions `glClearColor` and `glClear` are part of the OpenGL library (not GLUT) and have not yet been introduced. As we will see later, the `glClearColor` function sets the current clear color and the `glClear` function is used to clear the window to the current clear color.

Table D.2: GLUT functions relating to (a) initialization, (b) starting event processing, (c) window management, (d) callback registration, (e) state retrieval, and (f) geometric object rendering.

(a)

Function	Description
`glutInit`	initialize GLUT library
`glutInitWindowSize`	set initial window size for `glutCreateWindow`
`glutInitWindowPosition`	set initial window position for `glutCreateWindow`
`glutInitDisplayMode`	set initial display mode

(b)

Function	Description
`glutMainLoop`	enter GLUT event-processing loop

(c)

Function	Description
`glutCreateWindow`	create top-level window
`glutPostRedisplay`	mark current window as needing to be redisplayed
`glutSwapBuffers`	swaps buffers of current window if double buffered (flushes graphics output via `glFlush`)

(d)

Function	Description
`glutDisplayFunc`	sets display callback for current window
`glutReshapeFunc`	sets reshape callback for current window
`glutKeyboardFunc`	sets keyboard callback for current window
`glutSpecialFunc`	sets special keyboard callback for current window
`glutTimerFunc`	registers timer callback to be triggered in specified number of milliseconds

(e)

Function	Description
`glutGet`	retrieves simple GLUT state (e.g., size or position of current window)
`glutGetModifiers`	retrieve modifier key state when certain callbacks generated (i.e., state of shift, control, and alt keys)

(f)

Function	Description
`glutSolidSphere`	render solid sphere
`glutWireSphere`	render wireframe sphere
`glutSolidCube`	render solid cube
`glutWireCube`	render wireframe cube
`glutSolidCone`	render solid cone
`glutWireCone`	render wireframe cone
`glutSolidTorus`	render solid torus
`glutWireTorus`	render wireframe torus

Listing D.1: `trivial.cpp`

```
// A minimalist OpenGL/GLUT application.
//
// Draw a light green square.
//
// Copyright (c) 2013 Michael D. Adams

#include <GL/glut.h>

// The window display callback function.
void display()
{
    // Set the clear color to RGB value (0.0, 1.0, 1.0) (i.e., cyan).
    glClearColor(0.0, 1.0, 1.0, 0.0);

    // Clear the window.
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Flush the graphics output to the framebuffer.
    glutSwapBuffers();
}

int main(int argc, char** argv)
{
    // Initialize the GLUT library.
    // The function glutInit must be called before any other use of the
    // GLUT library is made.
    glutInit(&argc, argv);

    // Specify the type of display mode to be used for new windows.
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    // Set the nominal size for new windows.
    /// Note that this value does not have to be respected.
    glutInitWindowSize(512, 512);

    // Create a new window with the same name as the command name.
    // On many systems, the window name customarily appears in the
    // title bar window decoration added by the window manager.
    glutCreateWindow(argv[0]);

    // Register a display callback function.
    glutDisplayFunc(display);

    // Enter the GLUT event processing loop.
    // Note: The function glutMainLoop never returns.
    glutMainLoop();

    return 0;
}
```

D.3 Function Naming Conventions

In OpenGL, all function names begin with the prefix `gl` and all constant names begin with the prefix `GL`. This helps to reduce the likelihood of naming collisions with other code. OpenGL specifies several basic types as listed in

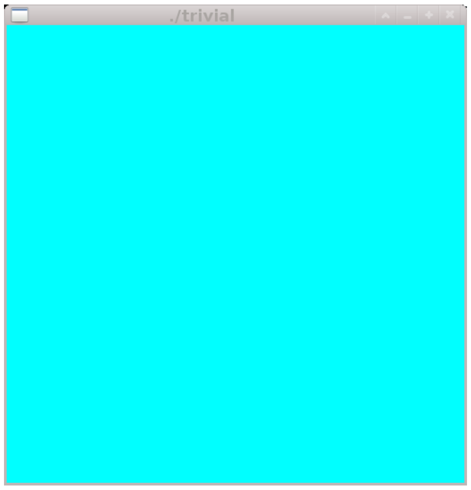


Figure D.1: Output of `trivial` program.

Table D.3: Basic types

Type Name	Typical Corresponding C Type
`GLbyte`	**`signed char`**
`GLshort`	**`short`**
`GLint`, `GLsizei`	**`int`** or **`long`**
`GLfloat`, `GLclampf`	**`float`**
`GLdouble`, `GLclampd`	**`double`**
`GLubyte`, `GLboolean`	**`unsigned char`**
`GLushort`	**`unsigned short`**
`GLuint`, `GLenum`, `GLbitfield`	**`unsigned int`** or **`unsigned long`**

Table D.3.

Some OpenGL commands allow their parameters to be specified in a number of different formats. Since C does not support function overloading, this means that each command is associated with a number of different functions, with each function taking its parameters using a different format. In this regard, the functions for a particular command are named according to the following pattern:

generic_name N T V

where *generic_name* is the generic name of the command, *N* is a digit (i.e., 2, 3, 4) indicating the number of components, *T* is one or two letters indicating the data type of the components, and *V* is either nothing or the letter `v` to indicate the component data is specified as individual values or as a vector (i.e., a pointer to an array), respectively. The component data could correspond to coordinates or color values. In the case that the components correspond to coordinates, the interpretation of *N* is as shown in Table D.4(a). The meaning of *T* is as indicated in Table D.4(b). For example, consider the process of specifying a vertex. This is performed via the `glVertex` command. The generic version of this function is denoted `glVertex*`. The specific version of `glVertex*` that takes three `GLfloat` parameters is named `glVertex3f`. In the name `glVertex3f`, the `3` indicates that the function takes three parameters and the `f` specifies that the parameters are of type `GLfloat`. Next, consider the process of specifying a color. This is accomplished via the `glColor` command. The generic version of this function is denoted `glColor*`. The specific version of `glColor*` that takes a single pointer to an array containing three `GLfloat` values is named `glColor3fv`. In the name `glColor3fv`, the `v` indicates that the function takes a pointer to an array containing the component data, the `3` indicates the presence of three components, and the `f` specifies that the components are of type `GLfloat`.

Table D.4: Function naming conventions. (a) The interpretation of N in the case of coordinate data. (b) The type of the components.

(a)

N	Coordinate Specifier
2	(x,y)
3	(x,y,z)
4	(x,y,z,w)

(b)

T	Data Type
b	GLbyte
s	GLshort
i	GLint, GLsizei
f	GLfloat, GLclampf
d	GLdouble, GLclampd
ub	GLubyte, GLboolean
us	GLushort
ui	GLuint, GLenum, GLbitfield

D.4 Drawing Geometric Primitives

In OpenGL, vertices are the basic building block for geometric primitives, such as line segments, triangles, and quadrilaterals. A vertex is specified with the `glVertex*` command. The most basic property of a vertex is its position in space. The most general way in which to specify a vertex position is using homogeneous coordinates (e.g., using `glVertex4f`). Some functions allow the specification of a vertex position using fewer than four coordinates (e.g., `glVertex2f` and `glVertex3f`). When a position is specified with two coordinates x and y (e.g., via `glVertex2f`), this is deemed equivalent to the homogeneous coordinates $(x,y,0,1)$. Similarly, when a position is specified with three coordinates x, y, and z (e.g., via `glVertex3f`), this is deemed equivalent to the homogeneous coordinates $(x,y,z,1)$. So, each of following function calls specifies the same point, namely $(1,2,0)$:

```
glVertex2f(1.0, 2.0);
glVertex3f(1.0, 2.0, 0.0);
glVertex4f(2.0, 4.0, 0.0, 2.0);
```

As explained above, all geometric primitives are specified by vertices, and each vertex is specified using `glVertex*`. When vertices are provided, the library must be told the type of geometric primitive to which the vertices belong. This is accomplished with the `glBegin` and `glEnd` functions. These function affect how vertices are interpreted. For example, vertices might be interpreted as individual points, pairs of vertices specifying line segments, triples of vertices specifying triangles, and so on. Supported geometric primitives include those listed in Table D.5.

To illustrate how `glBegin` and `glEnd` are used to assemble vertices into geometric primitives, we now provide a few examples. A single point could be specified by code like:

```
glBegin(GL_POINTS);
glVertex3f(1.0, 2.0, 3.0);
glEnd();
```

A single triangle could be specified using code like:

```
glBegin(GL_TRIANGLES);
glVertex3f(0.0, 0.0, 0.0); // first vertex
glVertex3f(1.0, 0.0, 0.0); // second vertex
glVertex3f(1.0, 1.0, 0.0); // third vertex
glEnd();
```

Two triangles could be specified by code like:

```
glBegin(GL_TRIANGLES);
glVertex3f(0.0, 0.0, 0.0); // first triangle, first vertex
glVertex3f(1.0, 0.0, 0.0); // first triangle, second vertex
glVertex3f(1.0, 1.0, 0.0); // first triangle, third vertex
glVertex3f(0.0, 0.0, 0.0); // second triangle, first vertex
glVertex3f(1.0, 1.0, 0.0); // second triangle, second vertex
```

Table D.5: Geometric primitives supported by `glBegin`

Value	Meaning
`GL_POINTS`	individual points
`GL_LINES`	pair of vertices interpreted as line segments
`GL_LINE_STRIP`	series of connected line segments
`GL_LINE_LOOP`	series of connected line segments with segment added between last and first vertices
`GL_TRIANGLES`	triples of vertices interpreted as triangles
`GL_TRIANGLE_STRIP`	linked strip of triangles
`GL_TRIANGLE_FAN`	linked fan of triangles
`GL_QUADS`	quadruples of vertices interpreted as quadrilaterals
`GL_QUAD_STRIP`	linked strip of quadrilaterals
`GL_POLYGON`	boundary of simple convex polygon

```
glVertex3f(0.0, 1.0, 0.0); // second triangle, third vertex
glEnd();
```

A single quadrilateral could be specified using code like:

```
glBegin(GL_QUADS);
glVertex3f(0.0, 0.0, 0.0); // first vertex
glVertex3f(1.0, 0.0, 0.0); // second vertex
glVertex3f(1.0, 1.0, 0.0); // third vertex
glVertex3f(0.0, 1.0, 0.0); // fourth vertex
glEnd();
```

Each vertex has several properties in addition to a position, including: a color, normal vector, texture coordinates, and material properties. To specify the color for a vertex, the `glColor*` function is used. To specify a normal vector for a vertex, the `glNormal*` function is employed. Calls to functions other than `glColor*` and `glNormal*` should not be placed inside a `glBegin`/`glEnd` block. Now, let us consider an example of specifying a triangle with specific colors and normals for its vertices. This can be done with code like the following:

```
glBegin(GL_TRIANGLES);
glNormal3f(0.0, 0.0, 1.0); // set current normal to (0,0,1)
glColor3f(1.0, 0.0, 0.0); // set current color to red
glVertex3f(0.0, 0.0, 0.0);
glColor3f(0.0, 1.0, 0.0); // set current color to green
glVertex3f(1.0, 0.0, 0.0);
glColor3f(0.0, 0.0, 1.0); // set current color to blue
glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

For a point, one can specify the point size (in pixels) with the `glPointSize` function. For example, code like the following will draw two points, each with a size of 10 pixels.:

```
glPointSize(10);
glBegin(GL_POINTS);
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);
glEnd();
```

For a line segment, one can specify the line width (in pixels) with the `glLineWidth` function. For example, code like the following will draw a line segment with a thickness of 10 pixels:

```
glLineWidth(10);
glBegin(GL_LINES);
```

```
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);
glEnd();
```

D.5 Coordinate Systems and Transformations

A computer-graphics system employs several coordinate systems, which are typically as follows:

- The **object coordinate system**, which is the coordinate system used to define a particular geometric object.
- The **world coordinate system**, which is the coordinate system for the scene in which various geometric objects are placed.
- The **camera coordinate system** (also known as the **eye coordinate system**), which is the coordinate system relative to which the scene is viewed (by the camera/eye).
- The **clip coordinate system**, which is used for clipping (i.e., eliminating parts of objects that fall outside the viewing volume).
- The **normalized device coordinate system**, which provides a coordinate system that is independent of the graphics-window size.
- The **window coordinate system**, which is the coordinate system used to address pixels in the graphics window.

Most operations require that points be specified with respect to the object coordinate system. Then, as each point passes through the various stages of the graphics pipeline, a sequence of transformations is applied to the point, as shown in Figure D.2. The **modelling transformation** is used to convert from object coordinates to world coordinates; the **viewing transformation** is used to convert from world coordinates to camera coordinates; the **projection transformation** is used to convert from camera coordinates to clip coordinates; and so on. Each coordinate system transformation is represented by a homogeneous-coordinate-transformation matrix. Although one can simply choose the modelling transformation as the identity matrix (in which case the object coordinate system and world coordinate system would be the same), it is frequently more convenient to make other choices.

In the interest of efficiency, in OpenGL, the world coordinate system is eliminated, and the modelling and viewing transformations are combined to form what is called a **modelview transformation**. In other words, with OpenGL, the sequence of transformations shown in Figure D.2 becomes the slightly simpler sequence shown in Figure D.3. The eye is always positioned at the origin and oriented in such a way as to be looking in the direction of the negative z axis with the positive y axis pointing upwards.

In OpenGL, the three key coordinate system transformations are the:

1. modelview transformation,
2. projection transformation, and
3. viewport transformation.

Points (i.e., vertices) are specified in object coordinates. First, the object coordinates are converted to eye coordinates using the equation

$$p_{\mathsf{eye}} = M_{\mathsf{mview}} p_{\mathsf{obj}}.$$

Next, the eye coordinates are mapped to clip coordinates as given by

$$p_{\mathsf{clip}} = M_{\mathsf{proj}} p_{\mathsf{eye}}.$$

Finally, the clip coordinates are converted internally to normalized device coordinates and then the viewport transformation is used to convert the normalized device coordinates to window coordinates.

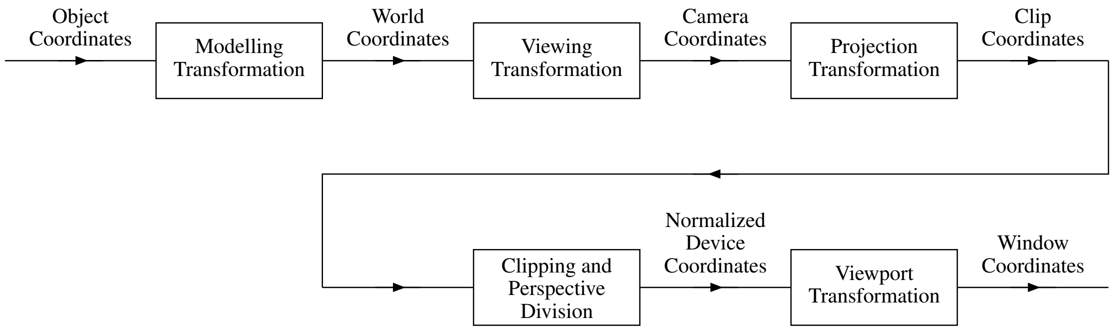


Figure D.2: Transformations in the graphics pipeline.

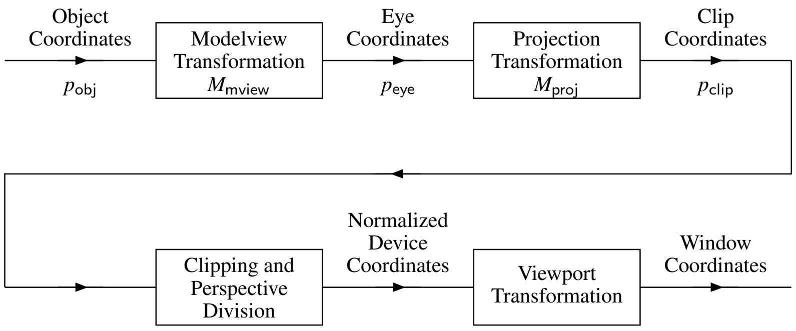

Object Coordinates
p_{obj}
Modelview Transformation M_{mview}
Eye Coordinates
p_{eye}
Projection Transformation M_{proj}
Clip Coordinates
p_{clip}
Clipping and Perspective Division
Normalized Device Coordinates
Viewport Transformation
Window Coordinates

Figure D.3: Transformations in the OpenGL graphics pipeline.

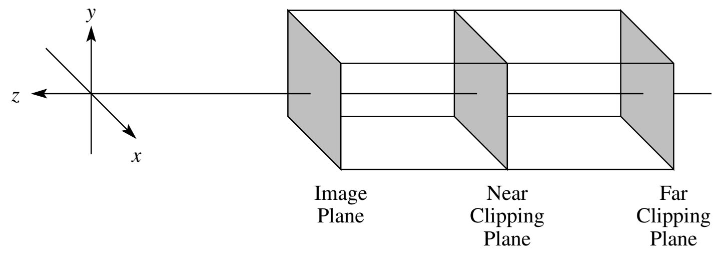


Figure D.4: Orthographic projection.

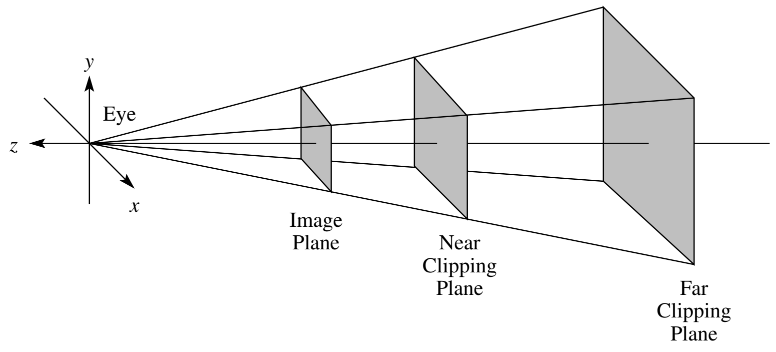


Figure D.5: Perspective projection.

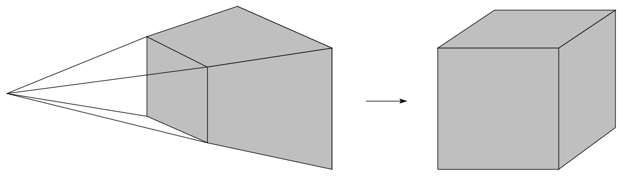


Figure D.6: Mapping the viewing frustum into a cube.

D.6 Projection Transformations

Recall that orthographic and perspective projections are associated with the diagrams shown in Figures D.4 and D.5. Although we speak of a projection transformation, in a practical computer-graphics systems (such as OpenGL), this transformation does not actually perform a projection. This is because a projection would discard depth information, which is needed for clipping and hidden object removal. The projection matrix used in practice simply consists of transformation to position and orient the viewing plane appropriately along with, in the case of perspective projection, a warping (as shown in Figure D.6). The actual projection itself, which would "flatten" the 3-D viewing volume onto the viewing plane is omitted.

A commonly used orthographic projection that maps the viewing volume $[l,r]\times[b,t]\times[n,f]$ to the cube $[-1,1]\times[-1,1]\times[-1,1]$ is given by the (homogeneous-coordinate-transformation) matrix

$$P(l,r,t,b,n,f)=\begin{bmatrix}\frac{2}{r-l} & 0 & 0 & \frac{r+l}{l-r}\\ 0 & \frac{2}{t-b} & 0 & \frac{t+b}{b-t}\\ 0 & 0 & \frac{2}{n-f} & \frac{f+n}{n-f}\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

The preceding matrix is the one used by the `glOrtho` function in OpenGL.

Suppose that the eye is positioned at the origin and is oriented in such a way as to be looking in the negative z direction with the positive y axis pointing upwards. In this case, we can specify a perspective projection by the parameters:

- θ, the field of view in the y direction;
- a, the aspect ratio (which determines the field of view in the x direction);
- n, the z coordinate of the near clipping plane; and
- f, the z coordinate of the far clipping plane.

The matrix associated with the above transformation is given by

$$P = \begin{bmatrix} \frac{c}{a} & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

where $c = \cot\theta/2$. The preceding matrix is the one used by the `gluPerspective` function in OpenGL.

D.7 Controlling Transformations in OpenGL

In order to function properly, an application must correctly set the modelview, projection, and viewport transformations. The modelview and projection transformations are controlled using similar mechanisms. So, we will consider them first.

The functions used for controlling the modelview and projection transformations include those listed in Table D.6. In OpenGL, there is a notion of a current transformation, which is either the modelview or projection transformation. Any operations related to transformations are performed on the current transformation. The current transformation is selected with the `glMatrixMode` function. Often, when working with the modelview and projection matrices, it is convenient to set the matrix to a particular value, use the new value, and then restore its old value. In OpenGL, a separate stack is maintained for each of the modelview and projection matrices. One can then push the current matrix onto the stack to save its value and then later pop the value from the stack in order to restore the old value. The push and pop operations are performed by the `glPushMatrix` and `glPopMatrix` functions. Since stack size is limited, one must be careful not to overflow the stack.

To initialize a transformation matrix, its value can be loaded with the identity matrix via the `glLoadIdentity` function. Once a matrix has been initialized, its value can be manipulated by multiplying it by other matrices. Various functions are provided that postmultiply the current transformation matrix by another matrix. As mentioned earlier, the eye position and orientation is always fixed in OpenGL. Sometimes, this fixed position is not what is desired. In this case, one can use the `gluLookAt` function to postmultiply the current matrix by a matrix that simulates a given eye position and orientation. One can also apply translation, rotation, and scaling transformations using the `glTranslatef`, `glRotatef`, and `glScalef` functions, respectively. The `gluPerspective` and `gluOrtho` functions can be used to more easily establish perspective and orthographic projections, respectively. (The `gluOrtho2D` function is helpful for establishing an orthographic projection for 2-D graphics applications.)

Now, let us consider the viewport transformation. The viewport transformation is much simpler to manipulate than the modelview and projection transformation. The functions associated with the viewport transformation are listed in Table D.7. In particular, the `glViewport` function is used to set the viewport transformation.

Example D.1 (Modelview matrix)**.** Suppose that we want to initialize the modelview matrix to the transformation

$$T = S(2,2,2)R_z(45)T(1,2,3).$$

The transformation T corresponds to the following translations/rotations/scalings in order:

1. a translation by $(1,2,3)$,
2. a rotation about the z axis by 45°; and
3. a scaling by $(2,2,2)$.

Table D.6: Functions related to modelview and projection transformations

Function	Description
`glMatrixMode`	set current matrix
`glLoadIdentity`	load current matrix with identity matrix
`gluPerspective`	multiply current matrix by perspective projection matrix
`gluOrtho`	multiply current matrix by orthographic projection matrix
`gluOrtho2D`	multiply current matrix by 2-D orthographic projection matrix
`gluLookAt`	multiply current matrix by viewing transformation matrix
`glRotatef`	multiply current matrix by rotation matrix
`glScalef`	multiply current matrix by scaling matrix
`glTranslatef`	multiply current matrix by translation matrix
`glPushMatrix`	push current matrix on current matrix stack
`glPopMatrix`	pop current matrix from current matrix stack

Table D.7: Functions related to viewport transformation

Function	Description
`glViewport`	set viewport (i.e., drawable part of window)

To initialize the modelview matrix with the value T, we could use the following code:

```
glMatrixMode(GL_MODELVIEW);      // set current matrix to modelview
glLoadIdentity();                // set to identity matrix
glScalef(2.0, 2.0, 2.0);         // postmultiply by scaling matrix
glRotatef(45.0, 0.0, 0.0, 1.0); // postmultiply by rotation matrix
glTranslatef(1.0, 2.0, 3.0);     // postmultiply by translation matrix
```

Note that the calls to `glScalef`, `glRotatef`, and `glTranslatef` are made in the exact reverse order from the order in which the transformations are applied to a point. This is due to that fact that functions like `glScalef`, `glRotatef`, and `glTranslatef`, postmultiply (i.e., multiply on the right) the current transformation matrix. As the above code executes, the modelview matrix M undergoes the following changes:

$$M = I \quad \longrightarrow \quad M = IS(2,2,2) = S(2,2,2) \quad \longrightarrow \quad M = S(2,2,2)R_{\mathsf{z}}(45) \quad \longrightarrow \quad M = S(2,2,2)R_{\mathsf{z}}(45)T(1,2,3).$$

Let p_{obj} and p_{eye} denote points in object and eye coordinates, respectively. So, we have

$$p_{\mathsf{eye}} = S(2,2,2)R_{\mathsf{z}}(45)T(1,2,3)p_{\mathsf{obj}}.$$

Thus, p_{eye} is obtained by applying the following sequence of transformations to p_{obj}:

1. a translation by $(1,2,3)$;
2. a rotation about the z axis by 45°; and
3. a scaling by $(2,2,2)$.

D.8 State Management

OpenGL has a considerable amount of state, and part of this state involves which capabilities within the library are enabled. In OpenGL, the `glEnable` and `glDisable` functions can be used to enable and disable certain functionality. Some capabilities that can be enabled or disabled include those listed in Table D.8. Some features may incur a

Table D.8: Capabilities controlled via `glEnable` and `glDisable`

Value	Meaning
`GL_CULL_FACE`	if enabled, cull polygons based on their winding in window coordinates
`GL_DEPTH_TEST`	if enabled, do depth comparisons and update depth buffer
`GL_LIGHT`_*i*_	include light *i* in evaluation of lighting equation
`GL_LIGHTING`	if enabled, use current lighting parameters to compute vertex color
`GL_LINE_SMOOTH`	if enabled, draw lines with antialiasing
`GL_NORMALIZE`	if enabled, normal vectors specified with `glNormal` scaled to unit length after transformation
`GL_POINT_SMOOTH`	if enabled, draw points with antialiasing
`GL_RESCALE_NORMAL`	if enabled, normal vectors specified with `glNormal` (assumed to be of unit length) scaled to unit length after transformation

Table D.9: Other functions

Function	Description
`glClear`	clear buffer to preset values
`glClearColor`	specify clear values for color buffers
`glShadeModel`	select flat or smooth shading
`glFrontFace`	define front- and back-facing polygons
`glLight*`	set light source parameters
`glLightModel*`	set lighting model parameters
`glColorMaterial`	specify how material color should track current color

significant computational cost when enabled. Therefore, it is advantageous to enable such features only if they are truly needed.

D.9 Miscellany

Several other OpenGL functions that may be useful are listed in Table D.9.

D.10 OpenGL/GLUT Example Programs

Now, we consider a few examples of programs that utilize the OpenGL and GLUT libraries. In particular, in the sections that follow, the source code listing for each of the following programs can be found:

1. The `simple_2d` program, which draws several geometric primitives in 2-D.
2. The `simple_3d` program, which draws and animates several simple polyhedra.
3. The `cube` program, which draws a cube with lighting.

D.10.1 `simple_2d` Program

The `simple_2d` program is a 2-D graphics application that draws a point, line, triangle, and square. The program terminates when the letter “q” is typed. A reshape callback function is used to track changes to the window size so that the proper aspect ratio can be maintained (e.g., the square remains square in spite of changes to the shape of the window). The program consists of the single source code file given in Listing D.2. The output generated by the program is shown in Figure D.7.

Listing D.2: `simple_2d.cpp`

```
// A simple 2-D graphics program.
//
// Create a window and draw some geometric primitives in it; and then
// wait until a "q" is typed before exiting.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <cstdlib>
#include <GL/glut.h>

// The window display callback function.
// This function is responsible for drawing the contents of a window.
// One can safely assume that the reshape callback for a window has
// been called at least once before the display callback is invoked.
void display()
{
    // Clear the window.
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Draw a point.
    glPointSize(5.0);
    glBegin(GL_POINTS);
    glColor3f(1.0, 1.0, 1.0);
    glVertex2f(0.25, 0.75);
    glEnd();

    // Draw a line.
    glLineWidth(2.0);
    glBegin(GL_LINES);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(0.6, 0.25);
    glVertex2f(0.9, 0.25);
    glEnd();

    // Draw a triangle.
    glBegin(GL_TRIANGLES);
    glColor3f(1.0, 0.0, 1.0);
    glVertex2f(0.6, 0.6);
    glVertex2f(0.9, 0.6);
    glVertex2f(0.9, 0.9);
    glEnd();

    // Draw a quad.
    glBegin(GL_QUADS);
    glColor3f(0.0, 1.0, 1.0);
    glVertex2f(0.1, 0.1);
    glVertex2f(0.4, 0.1);
    glVertex2f(0.4, 0.4);
    glVertex2f(0.1, 0.4);
    glEnd();

    // Flush the graphics output to the framebuffer.
    glutSwapBuffers();
}

// The window reshape callback function.
// This function is responsible for handling the resizing of a window.
// The reshape callback is always invoked immediately before the first
// call of the display callback (after a window is created).
void reshape(GLint width, GLint height)
{
    // Compute the aspect ratio, avoiding the possibility of division by zero.
    GLfloat aspectRatio = static_cast<GLfloat>(width) /
      ((height) ? height : 1.0);

    // Set the viewport to the entire window.
    glViewport(0, 0, width, height);

    // Initialize the projection matrix.
    // This is done in such a way to maintain the aspect ratio in case
    // the window shape is not square.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (width >= height) {
        gluOrtho2D(0.0, 1.0 * aspectRatio, 0.0, 1.0);
    } else {
        gluOrtho2D(0.0, 1.0, 0.0, 1.0 / aspectRatio);
    }

    // Initialize the modelview matrix.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

// The keyboard callback function.
// This function is responsible for processing keyboard input.
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'q':
        // Terminate the program, indicating success.
        exit(0);
        break;
    }
}

// The main program.
int main(int argc, char **argv)
{
    const int winWidth = 1024; // The nominal window width.
    const int winHeight = 1024; // The nominal window height.

    // Initialize the GLUT library.
    // The function glutInit must be called before any other use of the
    // GLUT library is made.
    glutInit(&argc, argv);

    // Specify the type of display mode to be used for new windows.
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    // Set the nominal size for new windows.
    // Note that this value does not have to be respected.
    glutInitWindowSize(winWidth, winHeight);

    // Create a new window with the same name as the command name.
    // On many systems, the window name customarily appears in the
    // title bar window decoration added by the window manager.
    glutCreateWindow(argv[0]);

    // Register a display callback function.
    glutDisplayFunc(display);

    // Register a reshape callback function.
    glutReshapeFunc(reshape);

    // Register a keyboard callback function.
    glutKeyboardFunc(keyboard);

    // Set the color to be used for clear operations.
    glClearColor(0.0, 0.0, 0.0, 0.0);

    // Enter the GLUT event processing loop.
    // Note: The function glutMainLoop never returns.
    glutMainLoop();

    // This line is never reached.
    return 0;
}
```

D.10.2 `simple_3d` Program

The `simple_3d` program draws several animated polyhedra. The program terminates when the letter “q” is typed. By typing “F” or “f”, the frame rate for the animation can be decreased or increased, respectively. The arrow keys can be used to change the vantage point of the camera. A reshape callback function is used to track changes to the window size so that the proper aspect ratio can be maintained in spite of changes to the window shape. A timer callback is used in order to precisely control that rate at which frames are drawn in the animation sequence. The program consists of the single source code file given in Listing D.3. The output generated by the program resembles that shown in Figure D.8.

Listing D.3: `simple_3d.cpp`

```
// A simple 3-D graphics program.
//
// Create a window and draw some animated polyhedra in it.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <cmath>
#include <cstdlib>
#include <GL/glut.h>

// The frame update period (in milliseconds).
```

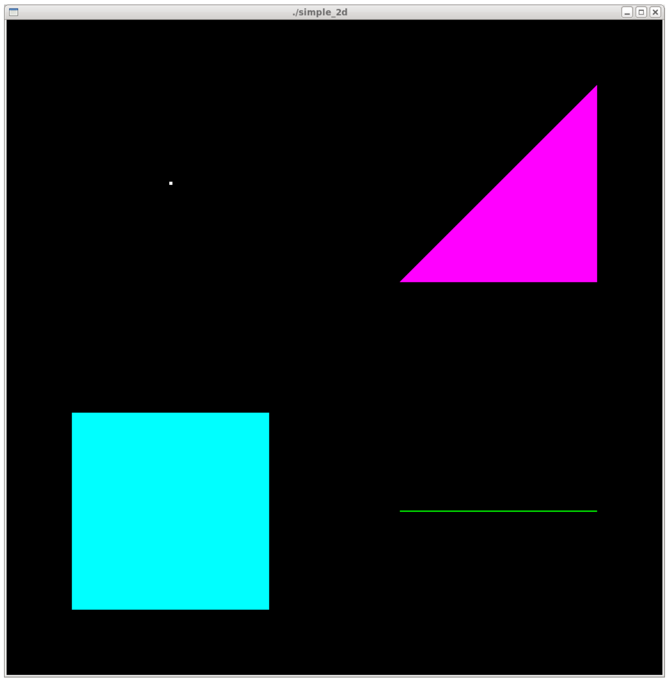


Figure D.7: Output of `simple_2d` program.

```
int framePeriod = 33;

// The parameters used to animate the polyhedra.
float theta = 0.0;

// The parameters used to specify the eye position.
GLfloat eye0 = 5.0;
GLfloat eye1 = 45.0;
GLfloat eye2 = 5.0;

// Convert from degrees to radians.
inline double degToRad(double x)
{
    return x * M_PI / 180.0;
}

// Draw the x, y, and z axes.
void drawAxes()
{
    const GLfloat length = 100.0;

    glLineWidth(2.0);
    glBegin(GL_LINES);

    // Draw the positive x axis (in bright red).
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(length, 0.0, 0.0);

    // Draw the negative x axis (in dark red).
    glColor3f(0.5, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(-length, 0.0, 0.0);

    // Draw the positive y axis (in bright green).
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, length, 0.0);

    // Draw the negative y axis (in dark green).
    glColor3f(0.0, 0.5, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, -length, 0.0);

    // Draw the positive z axis (in bright blue).
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, length);

    // Draw the negative z axis (in dark blue).
    glColor3f(0.0, 0.0, 0.5);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, -length);

    glEnd();
}

void drawTetrahedron()
{
    // The vertices of the tetrahedron.
    static const GLfloat vertices[][3] =
    {
        { 1.0,  1.0,  1.0},
        {-1.0,  1.0, -1.0},
        {-1.0, -1.0,  1.0},
        { 1.0, -1.0, -1.0}
    };

    // The faces of the tetrahedron.
    // Each triplet is a set of vertex indices for a face (specified
    // in CCW order).
    static const int faces[][3] =
    {
        {0, 1, 2},
        {0, 2, 3},
        {0, 3, 1},
        {3, 2, 1}
    };

    // The color for each vertex of the tetrahedron.
    static const GLfloat colors[][3] =
    {
        {1.0, 1.0, 1.0},
        {0.0, 0.0, 1.0},
        {0.0, 1.0, 0.0},
        {1.0, 0.0, 0.0},
    };

    // Draw the tetrahedron.
    glBegin(GL_TRIANGLES);
    // For each face...
    for (int i = 0; i < 4; ++i) {
        // For each vertex of the face...
        for (int j = 0; j < 3; ++j) {
            int v = faces[i][j];
            // Specify the color of the vertex.
            glColor3f(colors[v][0], colors[v][1], colors[v][2]);
            // Specify the position of the vertex.
            glVertex3f(vertices[v][0], vertices[v][1], vertices[v][2]);
        }
    }
    glEnd();
}

// Draw a star-shaped polyhedron via two overlapping tetrahedra.
// The polyhedron is drawn with its center at the origin.
void drawStar()
{
    glMatrixMode(GL_MODELVIEW);

    // Save the modelview matrix.
    glPushMatrix();

    // Draw the first tetrahedron.
    drawTetrahedron();

    // Draw the second tetrahedron rotated with respect to the first.
    glRotatef(90.0, 1.0, 0.0, 0.0);
    drawTetrahedron();

    // Restore the modelview matrix.
    glPopMatrix();
}

// The window display callback function.
// This function is responsible for drawing the contents of a window.
// One can safely assume that the reshape callback for a window has
// been called at least once before the display callback is invoked.
void display()
{
    // Clear the window.
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Set the current matrix to the modelview matrix.
    glMatrixMode(GL_MODELVIEW);

    // Save the current modelview matrix.
    glPushMatrix();

    // Set the eye position.
    // The eye is always oriented to look towards the origin.
    GLfloat eyeX = eye0 * cos(degToRad(eye1));
    GLfloat eyeY = eye0 * sin(degToRad(eye1));
    GLfloat eyeZ = eye2;
    gluLookAt(eyeX, eyeY, eyeZ, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    // Draw the coordinate axes.
    drawAxes();

    // Draw a star-shaped polyhedron at the origin.
    glPushMatrix();
    glScalef(0.5, 0.5, 0.5);
    glRotatef(theta, 1.0, 0.0, 0.0);
    glRotatef(2.0 * theta, 0.0, 1.0, 0.0);
    drawStar();
    glPopMatrix();

    // Draw a tetrahedron on the negative x axis.
    glPushMatrix();
    glTranslatef(-1.5, 0.0, 0.0);
    glScalef(0.5, 0.5, 0.5);
    glRotatef(0.0 - theta, 1.0, 0.0, 0.0);
    drawTetrahedron();
    glPopMatrix();

    // Draw a tetrahedron on the positive x axis.
    glPushMatrix();
    glTranslatef(1.5, 0.0, 0.0);
    glScalef(0.5, 0.5, 0.5);
    glRotatef(0.0 + theta, 1.0, 0.0, 0.0);
    drawTetrahedron();
    glPopMatrix();

    // Draw a tetrahedron on the negative y axis.
    glPushMatrix();
    glTranslatef(0.0, -1.5, 0.0);
    glScalef(0.5, 0.5, 0.5);
    glRotatef(90.0 - theta, 0.0, 1.0, 0.0);
    drawTetrahedron();
    glPopMatrix();

    // Draw a tetrahedron on the positive y axis.
    glPushMatrix();
    glTranslatef(0.0, 1.5, 0.0);
    glScalef(0.5, 0.5, 0.5);
    glRotatef(90.0 + theta, 0.0, 1.0, 0.0);
    drawTetrahedron();
    glPopMatrix();

    // Restore the old modelview matrix.
    glPopMatrix();

    // Flush the graphics output to the framebuffer.
    glutSwapBuffers();
}

// The window reshape callback function.
// This function is responsible for handling the resizing of a window.
// The reshape callback is always invoked immediately before the first
// call of the display callback (after a window is created).
```

```
void reshape(GLint width, GLint height)
{
    // Compute the aspect ratio, avoiding the possibility of division by zero.
    GLfloat aspectRatio = static_cast<GLfloat>(width) /
      ((height) ? height : 1.0);

    // Set the viewport to the entire window.
    glViewport(0, 0, width, height);

    // Initialize the projection matrix.
    // This is done in such a way to maintain the aspect ratio in case
    // the window size is not square.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // Setup a perspective projection.
    gluPerspective(45.0, aspectRatio, 1.0, 1000.0);

    // Initialize the modelview matrix.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

// The keyboard callback function.
// This function is responsible for processing keyboard input.
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'F':
        // Decrease the frame rate.
        ++framePeriod;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case 'f':
        // Increase the frame rate.
        framePeriod = std::max(framePeriod - 1, 1);
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case 'q':
        // Terminate the program, indicating success.
        exit(0);
        break;;
    }
}

// The timer callback function.
// This function is responsible for handling timer timeout events.
void timer(int value)
{
    // Update the parameter used to animate the polyhedra.
    theta += 3.0;

    // Force the window to be redisplayed.
    glutPostRedisplay();

    // Restart the timer.
    glutTimerFunc(framePeriod, timer, 0);
}

// The special keyboard callback function.
// This function is responsible for processing special keyboard keys
// (e.g., arrow keys, page up/down).
void special(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_LEFT:
        eye1 -= 1.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_RIGHT:
        eye1 += 1.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_DOWN:
        eye0 -= 0.05;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_UP:
        eye0 += 0.05;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_PAGE_DOWN:
        eye2 -= 0.1;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_PAGE_UP:
        eye2 += 0.1;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    }
}

// The main program.
int main(int argc, char **argv)
{
    const int winWidth = 1024; // The nominal window width.
    const int winHeight = 1024; // The nominal window height.

    // Initialize the GLUT library.
    // The function glutInit must be called before any other use of the
    // GLUT library is made.
    glutInit(&argc, argv);

    // Specify the type of display mode to be used for new windows.
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

    // Set the nominal size for new windows.
    // Note that this value does not have to be respected.
    glutInitWindowSize(winWidth, winHeight);

    // Create a new window with the same name as the command name.
    // On many systems, the window name customarily appears in the
    // title bar window decoration added by the window manager.
    glutCreateWindow(argv[0]);

    // Register a display callback function.
    glutDisplayFunc(display);

    // Register a reshape callback function.
    glutReshapeFunc(reshape);

    // Register a keyboard callback function.
    glutKeyboardFunc(keyboard);

    // Register a special keyboard callback function.
    glutSpecialFunc(special);

    // Set the clear color.
    glClearColor(0.0, 0.0, 0.0, 0.0);

    // Enable hidden object removal.
    glEnable(GL_DEPTH_TEST);

    // Do not render faces that are not visible.
    // For example, the back side of faces will not be rendered.
    glEnable(GL_CULL_FACE);

    // Interpolate color/normal values across faces.
    glShadeModel(GL_SMOOTH);

    // Specify the orientation of front-facing faces.
    glFrontFace(GL_CCW);

    // Register a timer callback function.
    glutTimerFunc(framePeriod, timer, 0);

    // Enter the GLUT event processing loop.
    // Note: The function glutMainLoop never returns.
    glutMainLoop();

    // This line is never reached.
    return 0;
}
```

D.10.3 `cube` Program

The `cube` program draws a cube with lighting, with the light source shown as a small sphere. The cube can be rotated using the arrow keys. The program terminates when the letter "q" is typed. A reshape callback function is used to track changes to the window size so that the proper aspect ratio can be maintained. The program consists of the single source code file given in Listing D.4. The output generated by the program resembles that shown in Figure D.9.

Listing D.4: `cube.cpp`

```
// A simple 2-D graphics program.
```

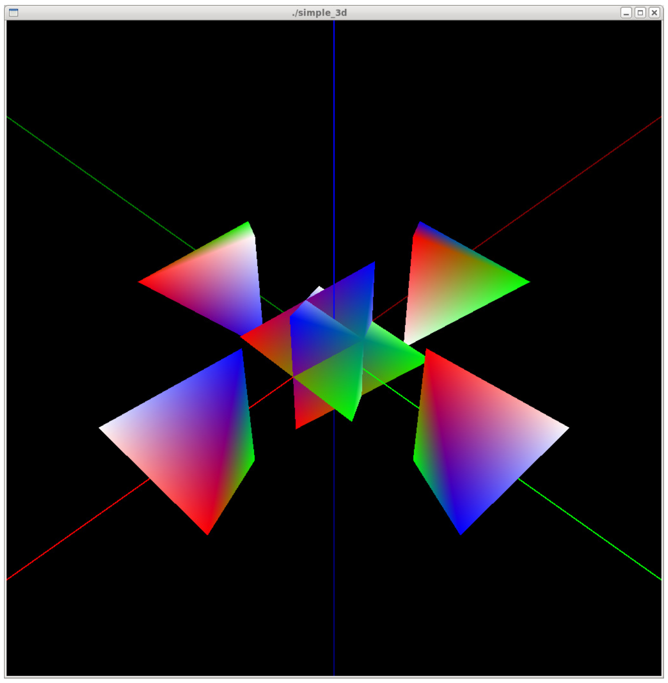


Figure D.8: Output of `simple_3d` program.

```
//
// Create a window and draw a cube in it.
//
// Copyright (c) 2013 Michael D. Adams

#include <iostream>
#include <cmath>
#include <cstdlib>
#include <GL/glut.h>

// The RGBA color of ambient light.
const GLfloat ambientLight[] = {
    0.25, 0.25, 0.25, 1.0 // whitish
};

// The RGBA color of diffuse light.
const GLfloat diffuseLight[] = {
    0.9, 0.9, 0.9, 1.0 // whitish
};

// The position of the light source (in homogeneous coordinates).
const GLfloat lightPosition[] = {
    0.0, 1.0, 1.0, 1.0
};

// The parameters used to animate the cube.
GLfloat thetaX = 0.0;
GLfloat thetaY = 0.0;

```

```
// Use a perspective projection (if true).
bool perspective = true;

// Draw the positive x, y, and z coordinate axes.
void drawAxes()
{
    glBegin(GL_LINES);
    glColor3f(1.0, 1.0, 1.0);

    // Draw the positive x axis.
    glColor3f(1.0, 0.0, 0.0); // red
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(2.0, 0.0, 0.0);

    // Draw the positive y axis.
    glColor3f(0.0, 1.0, 0.0); // green
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 2.0, 0.0);

    // Draw the positive z axis.
    glColor3f(0.0, 0.0, 1.0); // blue
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 2.0);

    glEnd();
}

void drawCube()
{
    // The vertices of the cube.
    static const GLfloat vertices[][3] =
    {
        { 0.5,  0.5,  0.5},
        { 0.5,  0.5, -0.5},
        { 0.5, -0.5,  0.5},
        { 0.5, -0.5, -0.5},
        {-0.5,  0.5,  0.5},
        {-0.5,  0.5, -0.5},
        {-0.5, -0.5,  0.5},
        {-0.5, -0.5, -0.5}
    };

    // The faces of the cube.
    // Each 4-tuple is a set of vertex indices for a face (specified
    // in CCW order).
    static const int faces[][4] =
    {
        {3, 1, 0, 2},
        {6, 4, 5, 7},
        {1, 5, 4, 0},
        {7, 3, 2, 6},
        {2, 0, 4, 6},
        {3, 7, 5, 1}
    };

    // The normals for the faces of the cube.
    static const GLfloat normals[][3] =
    {
        { 1.0,  0.0,  0.0},
        {-1.0,  0.0,  0.0},
        { 0.0,  1.0,  0.0},
        { 0.0, -1.0,  0.0},
        { 0.0,  0.0,  1.0},
        { 0.0,  0.0, -1.0},
    };

    // The colors for the faces of the cube.
    static const GLfloat colors[][3] =
    {
        {0.75, 0.55, 0.55}, // reddish
        {0.75, 0.55, 0.55}, // reddish
        {0.55, 0.75, 0.55}, // greenish
        {0.55, 0.75, 0.55}, // greenish
        {0.55, 0.55, 0.75}, // blueish
        {0.55, 0.55, 0.75}  // blueish
    };

    // Draw the quad faces of the cube.
    glBegin(GL_QUADS);
    // For each face...
    for (int i = 0; i < 6; ++i) {

        // Specify the normal for all vertices of the face.
        glNormal3f(normals[i][0], normals[i][1], normals[i][2]);

        // Specify the color for all vertices of the face.
        glColor3f(colors[i][0], colors[i][1], colors[i][2]);

        // Specify the vertices of the face.
        // For each vertex of face...
        for (int j = 0; j < 4; ++j) {
            glVertex3f(vertices[faces[i][j]][0],
              vertices[faces[i][j]][1],
              vertices[faces[i][j]][2]);
        }

    }
    glEnd();
}

// The window display callback function.
// This function is responsible for drawing the contents of a window.
// One can safely assume that the reshape callback for a window has
// been called at least once before the display callback is invoked.
void display()
{
    // Clear the window.
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Set the position of light 0.
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);

    // Set the current matrix to the modelview matrix.
    glMatrixMode(GL_MODELVIEW);

    // Draw the coordinate axes (with lighting disabled).
    glDisable(GL_LIGHTING);
    drawAxes();
    glEnable(GL_LIGHTING);

    ////////////////////////////////////////////////////////////
    // Draw a sphere to indicate the light source (with lighting disabled).
    ////////////////////////////////////////////////////////////

    // Save the modelview matrix.
    glPushMatrix();
    // Establish a new coordinate system with the position of light 0
    // as the origin.
    glTranslatef(lightPosition[0], lightPosition[1], lightPosition[2]);
    // Disable lighting.
    glDisable(GL_LIGHTING);
    glColor3f(1.0, 1.0, 1.0); // white
    // Draw a sphere at origin of the coordinate system.
    glutSolidSphere(0.01, 15, 15);
    // Enable lighting.
    glEnable(GL_LIGHTING);
    // Restore the modelview matrix.
    glPopMatrix();

    ////////////////////////////////////////////////////////////
    // Draw a cube (with lighting enabled).
    ////////////////////////////////////////////////////////////

    // Save the modelview matrix.
    glPushMatrix();
    // Establish a new coordinate system that is rotated with respect
    // to the original coordinate system.
    glRotatef(thetaX, 1.0, 0.0, 0.0);
    glRotatef(thetaY, 0.0, 1.0, 0.0);
    GLfloat thetaZ = thetaX + thetaY;
    glRotatef(thetaZ, 0.0, 1.0, 1.0);
    // Draw a cube at the origin.
    drawCube();
    // Restore the modelview matrix.
    glPopMatrix();

    // Flush the graphics output to the framebuffer.
    glutSwapBuffers();
}

// The window reshape callback function.
// This function is responsible for handling the resizing of a window.
// The reshape callback is always invoked immediately before the first
// call of the display callback (after a window is created).
void reshape(GLint width, GLint height)
{
    // Compute the aspect ratio, avoiding the possibility of division by zero.
    GLfloat aspectRatio = static_cast<GLfloat>(width) /
      ((height) ? height : 1.0);

    // Set the viewport to the entire window.
    glViewport(0, 0, width, height);

    // Initialize the projection matrix.
    // This is done in such a way to maintain the aspect ratio in case
    // the window size is not square.
    // We allow for either a perspective or orthographic projection
    // to be used.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (perspective) {
        // Establish a perspective projection.
        gluPerspective(45.0, aspectRatio, 1.0, 1000.0);
    } else {
        // Establish an orthographic projection.
        GLfloat left = -1.5;
        GLfloat right = 1.5;
        GLfloat bottom = -1.5;
        GLfloat top = 1.5;
        GLfloat zNear = 1;
        GLfloat zFar = 4;
        if (aspectRatio >= 0) {
            glOrtho(left * aspectRatio, right * aspectRatio, bottom, top,
              zNear, zFar);
        } else {
            glOrtho(left, right, bottom / aspectRatio, top / aspectRatio,
              zNear, zFar);
        }
    }

    // Initialize the modelview matrix.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 10.0);
}

// The keyboard callback function.
// This function is responsible for processing keyboard input.
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'q':
        // Terminate the program, indicating success.
        exit(0);
        break;;
    }
}

// The special keyboard callback function.
// This function is responsible for processing special keyboard keys
// (e.g., arrow keys, page up/down).
void special(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_LEFT:
        thetaX -= 5.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_RIGHT:
        thetaX += 5.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_DOWN:
        thetaY -= 5.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    case GLUT_KEY_UP:
        thetaY += 5.0;
        // Force the window to be redisplayed.
        glutPostRedisplay();
        break;
    }
}

// The main program.
int main(int argc, char **argv)
{
    const int winWidth = 1024; // The nominal window width.
    const int winHeight = 1024; // The nominal window height.

    // Initialize the GLUT library.
    // The function glutInit must be called before any other use of the
    // GLUT library is made.
    glutInit(&argc, argv);

    // If any non-GLUT command line arguments were specified, disable
    // perspective viewing.
    if (argc > 1) {
        perspective = false;
    }

    // Specify the type of display mode to be used for new windows.
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);

    // Set the nominal size for new windows.
    // Note that this value does not have to be respected.
    glutInitWindowSize(winWidth, winHeight);

    // Create a new window with the same name as the command name.
    // On many systems, the window name customarily appears in the
    // title bar window decoration added by the window manager.
    glutCreateWindow(argv[0]);

    // Register a display callback function.
    glutDisplayFunc(display);

    // Register a reshape callback function.
    glutReshapeFunc(reshape);

    // Register a keyboard callback function.
    glutKeyboardFunc(keyboard);

    // Register a special keyboard callback function.
    glutSpecialFunc(special);

    // Set the clear color.
    glClearColor(0.0, 0.0, 0.0, 0.0);

    // Set the color of the ambient light.
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);

    // Set the color of diffuse light for light 0.
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);

    // Enable light 0.
    glEnable(GL_LIGHT0);

    // Enable lighting calculations.
    glEnable(GL_LIGHTING);

    // Have front material parameters track the current color.
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);

    // Enable hidden object removal.
    glEnable(GL_DEPTH_TEST);

    // Do not render faces that are not visible.
    // For example, the back side of faces will not be rendered.
    glEnable(GL_CULL_FACE);

    // Specify the orientation of front-facing faces.
    glFrontFace(GL_CCW);

    // Automatically adjust normals to have unit norm.
    glEnable(GL_NORMALIZE);

    // Enter the GLUT event processing loop.
    // Note: The function glutMainLoop never returns.
    glutMainLoop();

    // This line is never reached.
    return 0;
}
```

D.11 OpenGL and CGAL Example Programs

Now, we consider an example of a program that utilizes both OpenGL and CGAL. In particular, in the section that follows, the source code listing for the following program can be found:

1. The `wireframe` program, which is a basic polygon mesh viewer.

D.11.1 `wireframe` Program

The `wireframe` program is a basic polygon mesh viewer, which draws the specified mesh as a wireframe model. If no command line parameters are given, a built-in mesh will be used; otherwise, the mesh to be used is read from standard input in OFF format. The mouse can be used to rotate, scale, and translate the mesh. The program terminates when

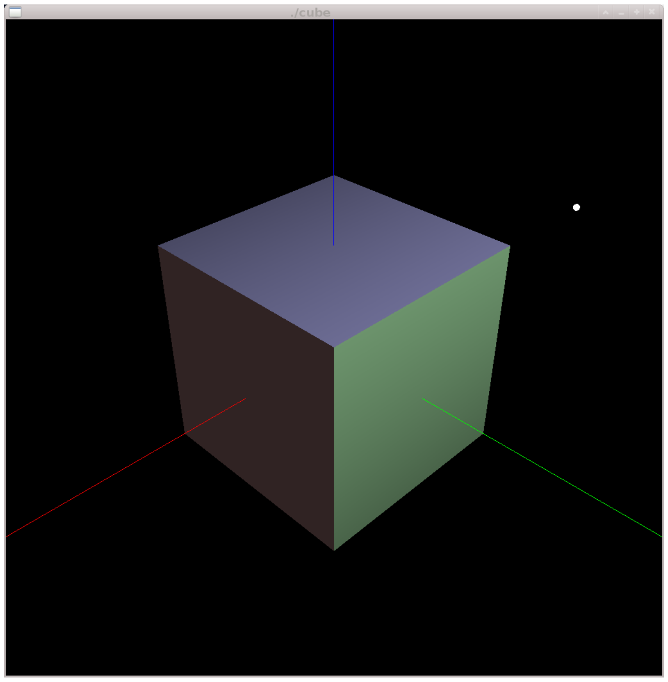


Figure D.9: Output of `cube` program.

the letter "q" is typed. The program consists of the single source code file given in Listing D.5. An example of the output produced by the program is shown in Figure D.10.

Listing D.5: `wireframe.cpp`

```
// Copyright (c) 2011, 2012, 2013 Michael D. Adams
// All rights reserved.

// __START_OF_LICENSE__
//
// Copyright (c) 2011, 2012, 2013 Michael D. Adams
// All rights reserved.
//
// This file is part of the Signal Processing Library (SPL).
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License as
// published by the Free Software Foundation; either version 3,
// or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public
// License along with this program; see the file LICENSE.  If not,
// see <http://www.gnu.org/licenses/>.
//
// __END_OF_LICENSE__

// A simple 3-D wireframe mesh viewer that uses both OpenGL and CGAL.

////////////////////////////////////////////////////////////////////////////////
// Header files
////////////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <string>
#include <GL/glut.h>
#include <CGAL/Cartesian.h>
#include <CGAL/Polyhedron_3.h>
#include <CGAL/IO/Polyhedron_iostream.h>
#include <CGAL/Bbox_3.h>
#include <SPL/cgalUtil.hpp>
#include <SPL/math.hpp>
#include <SPL/Arcball.hpp>

using SPL::norm;
using SPL::radToDeg;
using SPL::degToRad;
using SPL::normalize;
using SPL::angleBetweenVectors;

////////////////////////////////////////////////////////////////////////////////
// Types
////////////////////////////////////////////////////////////////////////////////

// Basic types.
typedef double Real;
typedef CGAL::Cartesian<Real> Kernel;
typedef Kernel::Point_3 Point3;
typedef Kernel::Point_2 Point2;
typedef Kernel::Vector_3 Vector3;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
typedef CGAL::Bbox_3 Bbox_3;
typedef SPL::Arcball<Kernel> ArcBall;
typedef SPL::Rotation_3<Kernel> Rotation3;

// The global state information for the program.
// In order to avoid a proliferation of global variables, all global
// state information for the program is kept in this structure.
struct Info
{
    static const Real sphereScale = 0.1;
    Info() : eyePos(0, 0, 10), sceneCenter(0, 0, 0), eyeUpDir(0, 1, 0),
      mode(0), trans(0, 0, 0), scale(1.0), rot(Vector3(0, 0, 1), 0),
      rotateMethod(0), arcBallRadius(1.0), displayAxes(false),
      displayArcBall(true) {}
    Polyhedron mesh;  // The polyhedral mesh.
    Bbox_3 boundBox;  // The bounding box for the mesh.
    Real smallEdge;  // The length of the shortest edge in the mesh.
    int viewportWidth;  // The viewport width.
    int viewportHeight;  // The viewport height.
    Point3 eyePos;  // The eye position.
    Point3 sceneCenter;  // The center of the scene.
    Vector3 eyeUpDir;  // The eye's up direction.
    Real scale;  // The scaling used in drawing the mesh object.
    Rotation3 rot;  // The rotation used in drawing the mesh object.
    Vector3 trans;  // The translation used in drawing the mesh object.
    int mode;  // The mode for mouse-based operations.
    // The following modes are valid:
    static const int rotateMode = 1;
    static const int translateMode = 2;
    static const int scaleMode = 4;
    int rotateMethod; // The rotation method.
    Real arcBallRadius; // The arcball radius (for rotation).
    Point2 mousePos;  // The window coordinates of the last mouse event.
    bool displayArcBall; // Display the arcball?
    bool displayAxes; // Display the axes for the world frame of reference?
    ArcBall arcBall; // The arcball used for rotation.
};

////////////////////////////////////////////////////////////////////////////////
// OpenGL utility functions
////////////////////////////////////////////////////////////////////////////////

// Convert from window coordinates to world coordinates.
// If no viewing matrix is provided, the current modelview matrix is
// assumed to be the viewing matrix.
// Optionally, the depth buffer information may be used to add a
// z dimension to the window coordinates before conversion.
Point3 winToWorld(const Point2& p, bool useDepth = false,
  const GLdouble* viewMatrix = 0)
{
    GLint viewport[4];
    GLdouble projMatrix[16];
    GLdouble modelviewMatrix[16];

    // Get the current viewport matrix.
    glGetIntegerv(GL_VIEWPORT, viewport);

    // Get the current projection matrix.
    glGetDoublev(GL_PROJECTION_MATRIX, projMatrix);

    // Get the current modelview matrix (if needed).
    if (!viewMatrix) {
        glGetDoublev(GL_MODELVIEW_MATRIX, modelviewMatrix);
    }

    GLdouble worldX;
    GLdouble worldY;
    GLdouble worldZ;
    GLdouble z;

    // Initialize the z component of the window coordinates.
    if (useDepth) {
        // Use the depth buffer information for the z value.
        glReadPixels(p.x(), p.y(), 1, 1, GL_DEPTH_COMPONENT, GL_DOUBLE, &z);
    } else {
        z = 0;
    }

    // Convert from window coordinates to object/world coordinates.
    gluUnProject(p.x(), p.y(), z, viewMatrix ? viewMatrix : modelviewMatrix,
      projMatrix, viewport, &worldX, &worldY, &worldZ);

    return Point3(worldX, worldY, worldZ);
}

// Draw a circle in the xy-plane centered at the origin and with unit radius.
void drawCircle()
{
    glBegin(GL_LINE_LOOP);
    const Real twoPi = 2.0 * M_PI;
    const Real angleInc = degToRad(1.0);
    for (Real angle = 0.0; angle < twoPi; angle += angleInc) {
        glVertex3f(cos(angle), sin(angle), 0.0);
    }
    glEnd();
}

// Draw the x, y, and z axes.
void drawAxes()
{
    glLineWidth(4.0);
    glBegin(GL_LINES);

    // Draw the positive x axis (in bright red).
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);

    // Draw the negative x axis (in dark red).
    glColor3f(0.5, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(-1.0, 0.0, 0.0);

    // Draw the positive y axis (in bright green).
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);

    // Draw the negative y axis (in dark green).
    glColor3f(0.0, 0.5, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, -1.0, 0.0);

    // Draw the positive z axis (in bright blue).
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 1.0);

    // Draw the negative z axis (in dark blue).
    glColor3f(0.0, 0.0, 0.5);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, -1.0);

    glEnd();
}

```

```
// Draw the arc ball.
void drawArcBall()
{
    glLineWidth(1.0);

    // Draw a circle in the yz-plane.
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glColor3f(1.0, 0.0, 0.0);
    drawCircle();
    glPopMatrix();

    // Draw a circle in the xz-plane.
    glPushMatrix();
    glRotatef(90.0, 0.0, 1.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    drawCircle();
    glPopMatrix();

    // Draw a circle in the xy-plane.
    glPushMatrix();
    glRotatef(90.0, 0.0, 0.0, 1.0);
    glColor3f(0.0, 0.0, 1.0);
    drawCircle();
    glPopMatrix();
}

// Draw a sphere.
void drawSphere()
{
    glutSolidSphere(1.0, 32, 32);
}

// Draw a rectangular prism with the specified center of gravity, axis,
// axis length, and "radius".
void drawRectPrism(const Point3& center, Real length, Real radius,
  const Vector3& axis)
{
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();

    Vector3 dir = axis / norm(axis);
    Real theta = angleBetweenVectors(dir, Vector3(0.0, 0.0, 1.0));
    glTranslatef(center.x(), center.y(), center.z());
    if (fabs(theta) > 1e-6) {
        Vector3 rotAxis = CGAL::cross_product(Vector3(0.0, 0.0, 1.0), dir);
        glRotatef(radToDeg(theta), rotAxis.x(), rotAxis.y(), rotAxis.z());
    }
    glScalef(radius, radius, length);
    glutSolidCube(1.0);

    glPopMatrix();
}

////////////////////////////////////////////////////////////////////////////////
// Global data
////////////////////////////////////////////////////////////////////////////////
```

```

// The default builtin mesh (in OFF format).
const std::string defaultMesh(
    "OFF\n"
    "5 4 0\n"
    "-1 -1  0\n"
    " 1 -1  0\n"
    " 1  1  0\n"
    "-1  1  0\n"
    " 0  0  1.5\n"
    "3 0 1 4\n"
    "3 1 2 4\n"
    "3 2 3 4\n"
    "3 0 4 3\n"
);

// The global state information for the program.
Info info;

////////////////////////////////////////////////////////////////////////////////
// Transformation functions
////////////////////////////////////////////////////////////////////////////////

// Perform scaling.
void scale(const Point2& pos)
{
    const Real minScale = 1e-6;

    // Calculate the amount by which to scale.
    Real refDist = (1.0 / sqrt(2.0)) * norm(winToWorld(Point2(
      info.viewportWidth - 1, info.viewportHeight - 1)) -
      winToWorld(Point2(0, 0)));
    Vector3 upDir = normalize(info.eyeUpDir);
    Vector3 delta = winToWorld(pos) - winToWorld(info.mousePos);
    Real dist = norm((upDir * delta) * upDir);
    Real scale;
    if (upDir * delta > 0) {
        scale = 1.0 + 2.0 * dist / refDist;
    } else if (upDir * delta < 0) {
        scale = 1.0 / (1.0 + 2.0 * dist / refDist);
    } else {
        scale = 1.0;
    }
    scale = std::max(scale, minScale / info.scale);

    // Apply the scaling.
    info.scale *= scale;

    // Update the most recent mouse position.
    info.mousePos = pos;

    // Redraw the window with the new scaling applied.
    glutPostRedisplay();
}

// Perform rotation.
void rotate(const Point2& pos)
```

```
{
    // Select the arcball rotation mode.
    info.arcBall.setMode(info.rotateMethod);

    // Specify the starting position for the arcball movement.
    info.arcBall.start(winToWorld(info.mousePos));

    // Specify the current position for the arcball movement.
    info.arcBall.move(winToWorld(pos));

    // Add the arcball rotation to the current rotation.
    info.rot = ArcBall::combineRotations(info.rot, info.arcBall.getRotation());

    // Update the most recent mouse position.
    info.mousePos = pos;

    // Redraw the window with the new rotation applied.
    glutPostRedisplay();
}

// Perform translation.
void translate(const Point2& pos)
{
    // Calculate the amount by which to translate.
    Real refDist = (1.0 / sqrt(2.0)) * norm(winToWorld(Point2(
      info.viewportWidth - 1, info.viewportHeight - 1)) -
      winToWorld(Point2(0, 0)));
    Vector3 upDir = normalize(info.eyeUpDir);
    Vector3 rightDir = normalize(CGAL::cross_product(info.sceneCenter -
      info.eyePos, info.eyeUpDir));
    Vector3 delta = winToWorld(pos) - winToWorld(info.mousePos);
    Vector3 trans = (2.0 / refDist) * ((upDir * delta) * upDir + (rightDir *
      delta) * rightDir);

    // Apply the translation.
    info.trans = info.trans + trans;

    // Update the most recent mouse position.
    info.mousePos = pos;

    // Redraw the window with the new translation applied.
    glutPostRedisplay();
}

////////////////////////////////////////////////////////////////////////////////
// GLUT callback functions
////////////////////////////////////////////////////////////////////////////////

// The window display callback function.
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);

    if (info.displayAxes) {
        // Draw axes for world frame of reference.
        glDisable(GL_LIGHTING);
        drawAxes();
        glEnable(GL_LIGHTING);
    }

    if (info.displayArcBall) {
        glDisable(GL_LIGHTING);
        glPushMatrix();
        if (!info.rotateMethod) {
            // Draw three circles as a visual cue for rotation.
            glTranslatef(info.sceneCenter.x(), info.sceneCenter.y(),
              info.sceneCenter.z());
            glRotatef(radToDeg(info.rot.angle), info.rot.axis.x(),
              info.rot.axis.y(), info.rot.axis.z());
            glScalef(info.arcBallRadius, info.arcBallRadius,
              info.arcBallRadius);
            drawArcBall();
        } else {
            // Draw single circle as a visual cue for rotation.
            glTranslatef(info.sceneCenter.x(), info.sceneCenter.y(),
              info.sceneCenter.z());
            glColor3f(0.5, 0.5, 0.5);
            glLineWidth(1.0);
            drawCircle();
        }
        glPopMatrix();
        glEnable(GL_LIGHTING);
    }

    glPushMatrix();

    // Establish the frame of reference (position and orientation)
    // for the mesh object.
    glTranslatef(info.trans.x(), info.trans.y(), info.trans.z());
    glRotatef(radToDeg(info.rot.angle), info.rot.axis.x(),
      info.rot.axis.y(), info.rot.axis.z());
    glScalef(info.scale, info.scale, info.scale);

    // Draw axes showing the frame of reference for the mesh object.
    glDisable(GL_LIGHTING);
    glPushMatrix();
    glScalef(100.0, 100.0, 100.0);
    drawAxes();
    glPopMatrix();
    glEnable(GL_LIGHTING);

    // For each vertex in the mesh...
    for (Polyhedron::Vertex_const_iterator vertexIter =
      info.mesh.vertices_begin(); vertexIter != info.mesh.vertices_end();
      ++vertexIter) {
        // Draw a sphere at the vertex position.
        Point3 v = vertexIter->point();
        glPushMatrix();
        glTranslatef(v.x(), v.y(), v.z());
        glScalef(Info::sphereScale * info.smallEdge, Info::sphereScale *
          info.smallEdge, Info::sphereScale * info.smallEdge);
        drawSphere();
        glPopMatrix();
    }

    // For each edge in the mesh...
    for (Polyhedron::Edge_const_iterator edgeIter = info.mesh.edges_begin();
      edgeIter != info.mesh.edges_end(); ++edgeIter) {
        // Draw a rectangular prism along the extent of the edge.
        Point3 v0 = edgeIter->vertex()->point();
        Point3 v1 = edgeIter->opposite()->vertex()->point();
        Point3 midpoint = CGAL::midpoint(v0, v1);
        Vector3 axis = v1 - v0;
        drawRectPrism(midpoint, norm(axis), 0.25 * Info::sphereScale *
          info.smallEdge, axis);
    }

    glPopMatrix();
    glutSwapBuffers();
}

// The window reshape callback function.
void reshape(GLint width, GLint height)
{
    info.viewportWidth = width;
    info.viewportHeight = height;

    // Compute the aspect ratio being careful to avoid the possibility of
    // division by zero (when the window width is zero).
    GLfloat aspectRatio = static_cast<GLfloat>(width) /
      ((height) ? height : 1.0);

    // Set the viewport to the entire window.
    glViewport(0, 0, width, height);

    // Establish a perspective projection.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, aspectRatio, 1.0, 1000.0);

    // Establish a viewing transform.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(info.eyePos.x(), info.eyePos.y(), info.eyePos.z(),
      info.sceneCenter.x(), info.sceneCenter.y(), info.sceneCenter.z(),
      info.eyeUpDir.x(), info.eyeUpDir.y(), info.eyeUpDir.z());
    info.arcBall.initialize(info.arcBallRadius, info.eyePos,
      info.sceneCenter - info.eyePos, info.eyeUpDir, info.sceneCenter);
}

// The keyboard callback function.
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'z':
        info.trans = info.trans + Vector3(0, 0, 0.5);
        glutPostRedisplay();
        break;
    case 'r':
        // Change the rotation method.
        info.rotateMethod ^= 1;
        glutPostRedisplay();
        break;
    case 'a':
        info.displayAxes ^= 1;
        glutPostRedisplay();
        break;
    case 'b':
        info.displayArcBall ^= 1;
        glutPostRedisplay();
        break;
    case 'q':
        // Quit the program.
        exit(0);
        break;
    }
}

// The mouse callback function.
// This function is called for each press and each release of a mouse button.
void mouse(int button, int state, int x, int y)
{
    // Compute the mouse position with the origin at the bottom left of the
    // window.
    Point2 mousePos(x, info.viewportHeight - 1 - y);

    switch (button) {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN) {
            // Enter rotation mode and save the initial mouse position.
            info.mode = Info::rotateMode;
            info.mousePos = mousePos;
        } else if (state == GLUT_UP) {
            // Perform any remaining rotation and exit rotation mode.
            rotate(mousePos);
            info.mode = 0;
        }
        glutPostRedisplay();
        break;
    case GLUT_MIDDLE_BUTTON:
        if (state == GLUT_DOWN) {
            // Enter scaling mode and save the initial mouse position.
            info.mode = Info::scaleMode;
            info.mousePos = mousePos;
        } else if (state == GLUT_UP) {
            // Perform any remaining scaling and exit scaling mode.
            scale(mousePos);
            info.mode = 0;
        }
        break;
    case GLUT_RIGHT_BUTTON:
        if (state == GLUT_DOWN) {
            // Enter translation mode and save the initial mouse position.
            info.mode = Info::translateMode;
            info.mousePos = mousePos;
        } else if (state == GLUT_UP) {
            // Perform any remaining translation and exit translation mode.
            translate(mousePos);
            info.mode = 0;
        }
        break;
    default:
        info.mode = 0;
        break;
    }
}

// The motion callback function.
// This function is called when the mouse moves within the window while
// one or more mouse buttons are pressed.
void motion(int x, int y)
{
    // Compute the mouse position with the origin at the bottom left of the
    // window.
    Point2 mousePos(x, info.viewportHeight - 1 - y);

    // Perform the appropriate processing for the current transformation mode.
    switch (info.mode) {
    case Info::rotateMode:
        // We are in rotation mode.  Perform the necessary rotation.
        rotate(mousePos);
        break;
    case Info::scaleMode:
        // We are in scale mode.  Perform the necessary scaling.
        scale(mousePos);
        break;
    case Info::translateMode:
        // We are in translation mode.  Perform the necessary translation.
        translate(mousePos);
        break;
    }
}

////////////////////////////////////////////////////////////////////////////////
// Main program
////////////////////////////////////////////////////////////////////////////////

int main(int argc, char **argv)
{
    const int winWidth = 1024;
    const int winHeight = 1024;
    bool useBuiltinMesh = (argc <= 1);

    // Load the mesh.
    if (!useBuiltinMesh) {
        // Read the mesh in OFF format from standard input.
        if (!(std::cin >> info.mesh)) {
            std::cerr << "cannot read mesh from standard input\n";
            exit(1);
        }
    } else {
        // Use the default builtin mesh.
        std::stringstream inStream(defaultMesh);
        inStream >> info.mesh;
    }

    // Compute the bounding box of the mesh.
    if (info.mesh.size_of_vertices() > 0) {
        Point3 v = info.mesh.vertices_begin()->point();
        info.boundBox = Bbox_3(v.x(), v.y(), v.z(), v.x(), v.y(), v.z());
    } else {
        info.boundBox = Bbox_3(0, 0, 0, 0, 0, 0);
    }
    for (Polyhedron::Vertex_const_iterator vertexIter =
      info.mesh.vertices_begin(); vertexIter != info.mesh.vertices_end();
      ++vertexIter) {
        const Point3& v = vertexIter->point();
        info.boundBox = info.boundBox + Bbox_3(v.x(), v.y(), v.z(), v.x(),
          v.y(), v.z());
    }

    // Compute the length of the shortest edge in the mesh.
    info.smallEdge = -1.0;
    for (Polyhedron::Edge_const_iterator edgeIter = info.mesh.edges_begin();
      edgeIter != info.mesh.edges_end(); ++edgeIter) {
        Point3 v0 = edgeIter->vertex()->point();
        Point3 v1 = edgeIter->opposite()->vertex()->point();
        Real length = norm(v1 - v0);
        if (info.smallEdge < 0.0 || length < info.smallEdge) {
            info.smallEdge = length;
        }
    }

    // Print some information about the mesh.
    std::cout << "bounding box: "
      << "[" << info.boundBox.xmin() << "," << info.boundBox.xmax() << "] x "
      << "[" << info.boundBox.ymin() << "," << info.boundBox.ymax() << "] x "
      << "[" << info.boundBox.zmin() << "," << info.boundBox.zmax() << "]"
      << "\n";
    std::cout << "length of shortest edge: " << info.smallEdge << "\n";

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow(argv[0]);

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);
    glEnable(GL_NORMALIZE);
    glEnable(GL_LIGHT0);

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glutMainLoop();

    return 0;
```

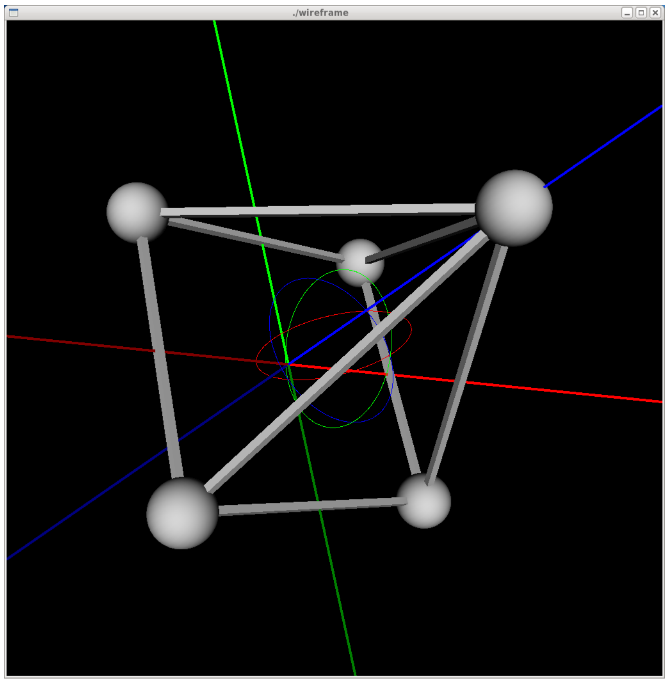


Figure D.10: Output of `wireframe` program.

```
}
```

D.12 Bibliography

[1] The OpenGL web site, 2013. `http://www.opengl.org`.

[2] M. K. Agoston. *Computer Graphics and Geometric Modeling: Implementation and Algorithms*. Springer, London, UK, 2005.

[3] M. K. Agoston. *Computer Graphics and Geometric Modeling: Mathematics*. Springer, London, UK, 2005.

[4] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, New York, NY, USA, 2nd edition, 1996.

[5] M. J. Kilgard. *The OpenGL Utility Toolkit (GLUT): Programming Interface (API Version 3)*, November 1996. Available from `http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf`.

[6] M. Segal and K. Akeley, editors. *The OpenGL Graphics System: A Specification (Version 4.4 (Core Profile))*. July 2013.

[7] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Computer Graphics*, 19:245–254, July 1985.

[8] K. Shoemake. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. In *Proc. of Conference on Graphics Interface*, pages 151–156, 1992.

[9] K. Shoemake. Arcball rotation control. In *Graphics Gems IV*, pages 175–192, San Diego, CA, USA, 1994. Academic Press.

[10] R. S. Wright, B. Lipchak, and N. Haemel. *OpenGL SuperBible*. Addison-Wesley, Upper Saddle River, NJ, USA, 4th edition, 2007.

Appendix E

Miscellany

E.1 Definitions

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{E.1}$$

E.2 Determinants

$$\det \boldsymbol{A}^T = \det \boldsymbol{A}$$
$$\det(\boldsymbol{AB}) = \det \boldsymbol{A} \det \boldsymbol{B}$$

E.3 Series

The sum of an arithmetic series is given by

$$\sum_{k=0}^{n-1} (a+kd) = \frac{n(2a+d(n-1))}{2}.$$

The sum of a geometric series is given by

$$\sum_{k=0}^{n-1} ar^k = a\frac{r^n-1}{r-1}$$

where $r \neq 1$.

E.4 Trigonometric Formulae

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$
$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\tan(a \pm b) = \frac{\tan a \pm \tan b}{1 \mp \tan a \tan b}$$

E.5 Derivatives

$$\tfrac{d}{dt}\cos t = -\sin t$$
$$\tfrac{d}{dt}\sin t = \cos t$$

Lemma E.1 (Leibniz rule)**.** *If $f(t)$ and $g(t)$ are n times continuously differentiable, then*

$$D^n[f(t)g(t)] = \sum_{k=0}^{n} \binom{n}{k} D^k f(t) D^{n-k} g(t).$$

E.6 Integrals

$$\int t \sin t\, dt = \sin t - t\cos t$$
$$\int t \cos t\, dt = \cos t + t\sin t$$
$$\int t^2 \sin t\, dt = -t^2\cos t + 2\cos t + 2t\sin t$$
$$\int t^2 \cos t\, dt = t^2\sin t - 2\sin t + 2t\cos t$$

E.7 Miscellaneous Theorems

Theorem E.1 (Bezout's identity)**.** *If a and b are nonzero integers with greatest common divisor d, then there exist integers x and y such that $ax+by=d$. Furthermore, d is the least positive integer for which there exist integers x and y satisfying the preceding equation.*

Theorem E.2 (Binomial theorem)**.** *For $n \in \mathbb{N}$, the following relationship holds:*

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}.$$

Theorem E.3 (Binomial theorem (multivariate case))**.** *Let a denote a multi-index. Let $x = (x_1, x_2, \ldots, x_n)$. Let $k \in \mathbb{N}$.*

$$(x_1 + x_2 + \ldots + x_n)^k = k! \sum_{a:|a|=k} \frac{x^a}{a!}.$$

The conditions under which the order of integration can be interchanged are given by the following theorem.

Theorem E.4 (Fubini theorem)**.** *Let f be a (measurable) function defined on $A \times B$. If $f \in L^1(A \times B)$ or $f \geq 0$, then*

$$\int_A \left(\int_B f(x,y)\,dy \right) dx = \int_B \left(\int_A f(x,y)\,dx \right) dy$$

Theorem E.5 (Holder inequality)**.** *Let $p > 1$, $q > 1$, and $1/p + 1/q = 1$. If $a \in l^p(\mathbb{N})$ and $b \in l^q(\mathbb{N})$, then*

$$\sum_{k\in\mathbb{N}} |a_k b_k| \leq \left(\sum_{k\in\mathbb{N}} |a_k|^p \right)^{1/p} \left(\sum_{k\in\mathbb{N}} |b_k|^q \right)^{1/q}.$$

If $f \in L^p(\Omega)$ and $g \in L^q(\Omega)$, then

$$\int_\Omega |f(t)g(t)dt| \leq \left(\int_\Omega |f(t)|^p\,dt \right)^{1/p} \left(\int_\Omega |g(t)|^q\,dt \right)^{1/q}.$$

Proof. See [1, pp. 548–551] □

Theorem E.6 (Minkowski inequality)**.** *Let $1 \leq p < \infty$. If $a, b \in l^p(\mathbb{N})$, then*

$$\left(\sum_{k\in\mathbb{N}} |a_k + b_k|^p\right)^{1/p} \leq \left(\sum_{k\in\mathbb{N}} |a_k|^p\right)^{1/p} + \left(\sum_{k\in\mathbb{N}} |b_k|^p\right)^{1/p}. \tag{E.2}$$

If $f, g \in L^p(\Omega)$, then

$$\left(\int_\Omega |f(t)+g(t)|^p\,dt\right)^{1/p} \leq \left(\int_\Omega |f(t)|^p\,dt\right)^{1/p} + \left(\int_\Omega |g(t)|^p\,dt\right)^{1/p}. \tag{E.3}$$

Proof. See [1, pp. 548–551] □

Theorem E.7 (Monotone convergence)**.** *Let $\{f_n\}$ be an increasing sequence in L with $\int f_n dt \leq M < \infty$ for all n. Let $f = \lim f_n$, then $f \in L$ and*

$$\int f dt = \lim_{n\to\infty} \int f_n dt.$$

(That is, we can interchange the limit with the integral.)

Theorem E.8 (Dominated convergence)**.** *Let $\{f_n\}$ be a sequence in L and assume that $f = \lim f_n$ (a.e.). If there is a $g \in L$ such that (for every n) $|f_n| \leq g$ (a.e.), then $f \in L$ and*

$$\int f dt = \lim_{n\to\infty} \int f_n dt.$$

(That is, we can interchange the limit with the integral.)

Theorem E.9 (Levi)**.** *Let $\{f_n\}$ be a sequence in L with $\sum_{n=1}^{\infty} \int |f_n| dt < \infty$. Then, the series $\sum_n f_n$ converges almost everywhere. Moreover, if $f = \sum_n f_n$ (a.e.), then $f \in L$ and*

$$\int f dt = \sum_n \int f_n dt.$$

(That is, we can interchange the infinite sum with the integral.)

Theorem E.10 (Fatou)**.** *Let $\{f_n\}$ be a sequence of nonnegative functions in L with*

$$\liminf_{n\to\infty} \int f_n dt = \lim_{n\to\infty} \left[\inf_{n\leq k} \int f_k dt\right] < \infty$$

and assume that $f = \lim_{n\to\infty} f_n$ (a.e.). Then, $f \in L$ and

$$\int f dt \leq \liminf_{n\to\infty} \int f_n dt.$$

E.8 Convolution

Consider the convolution of the sequences x and h given by

$$y[n] = x * h[n] = \sum_{k\in\mathbb{Z}} x[k]h[n-k].$$

Let us define the sequence $a[n] = h[-n]$ and the following (infinite dimensional) vectors:

$$\boldsymbol{x} = \begin{bmatrix} \cdots & x[-1] & x[0] & x[1] & \cdots \end{bmatrix}^T, \quad \boldsymbol{y} = \begin{bmatrix} \cdots & y[-1] & y[0] & y[1] & \cdots \end{bmatrix}^T,$$
$$\boldsymbol{h} = \begin{bmatrix} \cdots & h[-1] & h[0] & h[1] & \cdots \end{bmatrix}^T, \quad \text{and} \quad \boldsymbol{a} = \begin{bmatrix} \cdots & h[1] & h[0] & h[-1] & \cdots \end{bmatrix}^T.$$

(Note that $\boldsymbol{a}$ is simply $\boldsymbol{h}$ with the order of the elements reversed.) In what follows, let $\mathcal{S}$ denote an operator that shifts the elements in a vector one row downward (or one column rightward). That is, $\mathcal{S}a[\cdot] = a[\cdot - 1]$. For example, $\mathcal{S}\begin{bmatrix} \cdots & 1 & 2 & 3 & \cdots \end{bmatrix}^T = \begin{bmatrix} \cdots & 0 & 1 & 2 & 3 & \cdots \end{bmatrix}^T$. Consider the convolution $y[n]$ evaluated at $n = 0$. We have

$$\begin{aligned} y[0] &= \sum_{k \in \mathbb{Z}} x[k]h[-k] \\ &= \sum_{k \in \mathbb{Z}} x[k]a[k] \\ &= \boldsymbol{a}^T \boldsymbol{x}. \end{aligned}$$

Consider the convolution $y[n]$ evaluated at $n = 1$. We have

$$\begin{aligned} y[1] &= \sum_{k \in \mathbb{Z}} x[k]h[1-k] \\ &= \sum_{k \in \mathbb{Z}} x[k]a[k-1] \\ &= (\mathcal{S}\boldsymbol{a})^T \boldsymbol{x}, \end{aligned}$$

For arbitrary n, we have $y[n]$ is given by

$$\begin{aligned} y[n] &= \sum_{k \in \mathbb{Z}} x[k]h[n-k] \\ &= \sum_{k \in \mathbb{Z}} x[k]a[k-n] \\ &= (\mathcal{S}^n\boldsymbol{a})^T \boldsymbol{x}. \end{aligned}$$

Using this result, we can rewrite the convolution in matrix form as

$$\begin{bmatrix} \vdots \\ y[-2] \\ y[-1] \\ y[0] \\ y[1] \\ y[2] \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ (\mathcal{S}^{-2}\boldsymbol{a})^T \\ (\mathcal{S}^{-1}\boldsymbol{a})^T \\ \boldsymbol{a}^T \\ (\mathcal{S}\boldsymbol{a})^T \\ (\mathcal{S}^2\boldsymbol{a})^T \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x[-2] \\ x[-1] \\ x[0] \\ x[1] \\ x[2] \\ \vdots \end{bmatrix}$$

which is equivalent to

$$\underbrace{\begin{bmatrix} \vdots \\ y[-2] \\ y[-1] \\ y[0] \\ y[1] \\ y[2] \\ \vdots \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & h[0] & h[-1] & h[-2] & h[-3] & h[-4] & \cdots \\ \cdots & h[1] & h[0] & h[-1] & h[-2] & h[-3] & \cdots \\ \cdots & h[2] & h[1] & h[0] & h[-1] & h[-2] & \cdots \\ \cdots & h[3] & h[2] & h[1] & h[0] & h[-1] & \cdots \\ \cdots & h[4] & h[3] & h[2] & h[1] & h[0] & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \vdots \\ x[-2] \\ x[-1] \\ x[0] \\ x[1] \\ x[2] \\ \vdots \end{bmatrix}}_{\boldsymbol{x}}$$

Observe that the matrix $\boldsymbol{A}$ is Toeplitz. That is, the operation of multiplying by an infinite-dimensional Toeplitz matrix is equivalent to convolution.

Consider the convolution of x and h given by

$$y[n] = x * h[n] = \sum_{k \in \mathbb{Z}} x[k]h[n-k].$$

Let us define the set of sequences

$$a_k[n] = h^*[k-n], \quad k \in \mathbb{Z}.$$

(This implies that $h[n-k] = a_n^*[k]$.) We can re-express the above convolution in terms of a_k as

$$\begin{aligned} y[n] &= \sum_{k\in\mathbb{Z}} x[k]a_n^*[k] \\ &= \langle x, a_n \rangle . \end{aligned}$$

Thus, a convolution can be viewed as a sequence of inner products.

E.9 Miscellaneous Examples

Example E.1. Let $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ be the matrices given by

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A}_2 = \begin{bmatrix} -1 & 2 & -1 \\ 3 & -1 & 2 \\ 2 & 3 & -1 \end{bmatrix}.$$

Find $\boldsymbol{A}_1^{-1}$ and $\boldsymbol{A}_2^{-1}$.

Solution. Consider the quantity $\boldsymbol{A}_1^{-1}$. First, we compute $\det\boldsymbol{A}_1$ and $\operatorname{Adj}\boldsymbol{A}_1$ to obtain

$$\begin{aligned} \det\boldsymbol{A}_1 &= (1)(4) - (2)(3) \\ &= 4 - 6 \\ &= -2, \quad \text{and} \end{aligned}$$

$$\begin{aligned} \operatorname{Adj}\boldsymbol{A}_1 &= \begin{bmatrix} (1)(4) & (-1)(3) \\ (-1)(2) & (1)(1) \end{bmatrix}^T \\ &= \begin{bmatrix} 4 & -3 \\ -2 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}. \end{aligned}$$

Then, we have

$$\begin{aligned} \boldsymbol{A}_1^{-1} &= [\det\boldsymbol{A}_1]^{-1} \operatorname{Adj}\boldsymbol{A}_1 \\ &= -\tfrac{1}{2} \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}. \end{aligned}$$

Consider the quantity $\boldsymbol{A}_2^{-1}$. First, we compute $\det\boldsymbol{A}_2$ and $\operatorname{Adj}\boldsymbol{A}_2$ to obtain

$$\begin{aligned} \det\boldsymbol{A}_2 &= (-1)[(-1)(-1) - (2)(3)] - (2)[(3)(-1) - (2)(2)] + (-1)[(3)(3) - (-1)(2)] \\ &= (-1)(-5) - 2(-7) + (-1)(11) \\ &= 5 + 14 - 11 \\ &= 8 \quad \text{and} \end{aligned}$$

$$
\begin{aligned}
\operatorname{Adj}\boldsymbol{A}_2 &= \begin{bmatrix} (1)(1-6) & (-1)(-3-4) & (1)(9+2) \\ (-1)(-2+3) & (1)(1+2) & (-1)(-3-4) \\ (1)(4-1) & (-1)(-2+3) & (1)(1-6) \end{bmatrix}^T \\
&= \begin{bmatrix} -5 & 7 & 11 \\ -1 & 3 & 7 \\ 3 & -1 & -5 \end{bmatrix}^T \\
&= \begin{bmatrix} -5 & -1 & 3 \\ 7 & 3 & -1 \\ 11 & 7 & -5 \end{bmatrix}.
\end{aligned}
$$

Then, we have

$$
\begin{aligned}
\boldsymbol{A}_2^{-1} &= [\det\boldsymbol{A}_2]^{-1}\operatorname{Adj}\boldsymbol{A}_2 \\
&= \tfrac{1}{8}\begin{bmatrix} -5 & -1 & 3 \\ 7 & 3 & -1 \\ 11 & 7 & -5 \end{bmatrix} \\
&= \begin{bmatrix} -\frac{5}{8} & -\frac{1}{8} & \frac{3}{8} \\ \frac{7}{8} & \frac{3}{8} & -\frac{1}{8} \\ \frac{11}{8} & \frac{7}{8} & -\frac{5}{8} \end{bmatrix}.
\end{aligned}
$$

□

E.10 Elementary Matrix Operations

E.10.1 Inverse of Elementary Matrix

- $\mathcal{A}^{-1}(k,l,a) = \mathcal{A}(k,l,-a)$ (sign of one element changes)

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & a & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -a & 1 \end{bmatrix}
$$

- $\mathcal{S}^{-1}(k,a) = \mathcal{S}(k,1/a)$ (one element replaced by its reciprocal)

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/a & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

E.10.2 S-Type Elementary Matrix Operations

- premultiply by $\mathcal{S}(2,2,b)$; row 2 multiplied by b

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ ba_{2,1} & ba_{2,2} & ba_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}
$$

- premultiply by $\mathcal{S}(3,3,b)$; row 3 multiplied by b

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & b \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ ba_{3,1} & ba_{3,2} & ba_{3,3} \end{bmatrix}
$$

- postmultiply by $\mathcal{S}(2,2,b)$; column 2 multiplied by b

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & ba_{1,2} & a_{1,3} \\ a_{2,1} & ba_{2,2} & a_{2,3} \\ a_{3,1} & ba_{3,2} & a_{3,3} \end{bmatrix}$$

- postmultiply by $\mathcal{S}(3,3,b)$; column 3 multiplied by b

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & b \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & ba_{1,3} \\ a_{2,1} & a_{2,2} & ba_{2,3} \\ a_{3,1} & a_{3,2} & ba_{3,3} \end{bmatrix}$$

E.10.3 A-Type Elementary Matrix Operations

- premultiply by $\mathcal{A}(3,2,b)$; $(\text{row } 3) + b(\text{row } 2) \to \text{row } 3$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1}+ba_{2,1} & a_{3,2}+ba_{2,2} & a_{3,3}+ba_{2,3} \end{bmatrix}$$

- premultiply by $\mathcal{A}(1,2,b)$; $(\text{row } 1) + b(\text{row } 2) \to \text{row } 1$

$$\begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a_{1,1}+ba_{2,1} & a_{1,2}+ba_{2,2} & a_{1,3}+ba_{2,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

- postmultiply by $\mathcal{A}(3,2,b)$; $(\text{column } 2) + b(\text{column } 3) \to \text{column } 2$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2}+ba_{1,3} & a_{1,3} \\ a_{2,1} & a_{2,2}+ba_{2,3} & a_{2,3} \\ a_{3,1} & a_{3,2}+ba_{3,3} & a_{3,3} \end{bmatrix}$$

- postmultiply by $\mathcal{A}(1,2,b)$; $(\text{column } 2) + b(\text{column } 1) \to \text{column } 2$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2}+ba_{1,1} & a_{1,3} \\ a_{2,1} & a_{2,2}+ba_{2,1} & a_{2,3} \\ a_{3,1} & a_{3,2}+ba_{3,1} & a_{3,3} \end{bmatrix}$$

E.11 Distribution Theory

Definition E.1 (Test function)**.** A smooth (i.e., infinitely differentiable) compactly-supported function on $\mathbb{R}^n$ is called a **test function**.

The space of test functions is denoted as $\mathcal{D}(\mathbb{R}^n)$ or simply $\mathcal{D}$ (i.e., $\mathcal{D} = C_0^\infty(\mathbb{R}^n)$).

Example E.2. An example of test function ϕ is given by

$$\phi(t) = \begin{cases} e^{1/(t^2-a^2)} & \text{for } |t| < a \\ 0 & \text{otherwise} \end{cases}$$

$\operatorname{supp}\phi = [-a, a]$.

Definition E.2 (Distribution)**.** A continuous linear functional on $\mathcal{D}(\mathbb{R}^n)$ is called a **distribution**.

Table E.1: Fourier transform pairs

Pair	$f(t)$	$\hat{f}(\omega)$
1	$e^{-a\lvert t\rvert}$	$\frac{2a}{\omega^2+a^2}$
2	$\operatorname{rect}\frac{t}{T}$	$T\operatorname{sinc}\frac{T\omega}{2}$
3	$\frac{B}{\pi}\operatorname{sinc}Bt$	$\operatorname{rect}\frac{\omega}{2B}$
4	$\operatorname{tri}\frac{t}{T}$	$\frac{T}{2}\operatorname{sinc}^2\frac{T\omega}{4}$
5	$\frac{B}{2\pi}\operatorname{sinc}^2\frac{Bt}{2}$	$\operatorname{tri}\frac{\omega}{2B}$
6	$e^{-t^2/(2\sigma^2)}$	$\lvert\sigma\rvert\sqrt{2\pi}e^{-\sigma^2\omega^2/2}$

The space of distributions is denoted $\mathcal{D}'(\mathbb{R}^n)$ or simply $\mathcal{D}'$. It is the dual space to $\mathcal{D} = C_0^\infty(\mathbb{R}^n)$.

Definition E.3 (Regular and singular distributions)**.** A distribution $F \in \mathcal{D}'$ is called **regular** if there exists a locally integrable function f such that

$$(F,\phi) = \int_{\mathbb{R}} f(t)\phi(t)dt$$

for every $\phi \in \mathcal{D}$. A distribution that is not regular is said to be **singular**.

Example E.3 (Dirac distribution)**.** The Dirac distribution given by $(\delta,\phi) = \phi(0)$ is an example of a singular distribution.

Definition E.4 (Schwartz function)**.** A C^∞ complex-valued function f on $\mathbb{R}^n$ is called a **Schwartz function** if for all α,β there exist positive constants $C_{\alpha,\beta}$ such that

$$\rho_{\alpha,\beta}(f) = \sup_{x\in\mathbb{R}^n}\left|x^\alpha\partial^\beta f(x)\right| = C_{\alpha,\beta} < \infty$$

The space of Schwartz functions is denoted $\mathcal{S}(\mathbb{R}^n)$ or simply $\mathcal{S}$.

Example E.4. Examples of Schwartz functions include the following: $f(t) = e^{-t^2}$, any $f \in C_0^\infty(\mathbb{R})$

Definition E.5 (Tempered distribution)**.** A continuous linear functional on $\mathcal{S}(\mathbb{R}^n)$ is called a **tempered distribution**.

The space of tempered distributions is denoted as $\mathcal{S}'(\mathbb{R}^n)$ or simply $\mathcal{S}'$.

E.12 Tables

Some common Fourier transform pairs are given in Table E.1.

E.13 Bibliography

[1] A. W. Naylor and G. R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag, New York, NY, USA, 1982.

Index