

Eastern
Economy
Edition

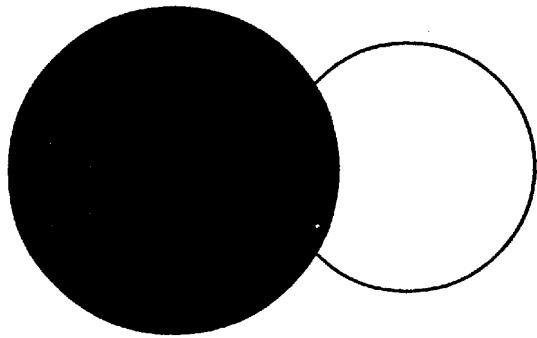
INSIGHT INTO
WAVELETS

From Theory to Practice



K.P. Soman
K.I. Ramachandran





Contents

<i>Preface</i>	<i>xi</i>
<i>Acknowledgments</i>	<i>xiii</i>

1. The Age of Wavelets 1–13

<i>Introduction</i>	<i>1</i>
1.1	The Origins of Wavelets—Are They Fundamentally New? <i>1</i>
1.2	Wavelets and Other Reality Transforms <i>3</i>
1.3	Managing Heisenberg's Uncertainty Ghost <i>5</i>
1.4	History of Wavelet from Morlet to Daubechies Via Mallat <i>6</i>
1.4.1	Different Communities of Wavelets <i>9</i>
1.4.2	Different Families of Wavelets within Wavelet Communities <i>10</i>
1.4.3	Interesting Recent Developments <i>11</i>
1.5	Wavelets in the Future <i>12</i>
<i>Summary</i>	<i>13</i>
<i>References</i>	<i>13</i>

2. Fourier Series and Geometry 14–30

<i>Introduction</i>	<i>14</i>
2.1	Vector Space <i>15</i>
2.1.1	Bases <i>15</i>
2.1.2	Orthonormality <i>15</i>
2.1.3	Projection <i>15</i>
2.2	Functions and Function Spaces <i>16</i>
2.2.1	Orthogonal Functions <i>16</i>
2.2.2	Orthonormal Functions <i>17</i>
2.2.3	Function Spaces <i>17</i>

2.2.4	Orthogonal Basis Functions	20
2.2.5	Orthonormality and the Method of Finding the Coefficients	20
2.2.6	Complex Fourier Series	24
2.2.7	Orthogonality of Complex Exponential Bases	25
<i>Summary</i>	27	
<i>Exercises</i>	28	
<i>References</i>	30	

3. Continuous Wavelet and Short Time Fourier Transform 31–47

<i>Introduction</i>	31	
3.1	Wavelet Transform—A First Level Introduction	31
3.2	Mathematical Preliminaries—Fourier Transform	35
3.2.1	Continuous Time-Frequency Representation of Signals	37
3.2.2	The Windowed Fourier Transform (Short Time Fourier Transform)	38
3.2.3	The Uncertainty Principle and Time Frequency Tiling	39
3.3	Properties of Wavelets Used in Continuous Wavelet Transform	43
3.4	Continuous Versus Discrete Wavelet Transform	43
<i>Summary</i>	45	
<i>Exercises</i>	46	
<i>References</i>	47	

4. Discrete Wavelet Transform 48–72

<i>Introduction</i>	48	
4.1	Haar Scaling Functions and Function Spaces	48
4.1.1	Translation and Scaling of $\phi(t)$	49
4.1.2	Orthogonality of Translates of $\phi(t)$	50
4.1.3	Function Space V_0	51
4.1.4	Finer Haar Scaling Functions	53
4.2	Nested Spaces	54
4.3	Haar Wavelet Function	55
4.3.1	Scaled Haar Wavelet Functions	57
4.4	Orthogonality of $\phi(t)$ and $\psi(t)$	62
4.5	Normalization of Haar Bases at Different Scales	63
4.6	Standardizing the Notations	65
4.7	Refinement Relation with Respect to Normalized Bases	65
4.8	Support of a Wavelet System	66
4.8.1	Triangle Scaling Function	67
4.9	Daubechies Wavelets	68
4.10	Seeing the Hidden—Plotting the Daubechies Wavelets	69
<i>Summary</i>	71	
<i>Exercises</i>	71	
<i>References</i>	72	

5. Designing Orthogonal Wavelet Systems—A Direct Approach	73–85
<i>Introduction</i> 73	
5.1 Refinement Relation for Orthogonal Wavelet Systems 73	
5.2 Restrictions on Filter Coefficients 74	
5.2.1 Condition 1: Unit Area Under Scaling Function 74	
5.2.2 Condition 2: Orthonormality of Translates of Scaling Functions 75	
5.2.3 Condition 3: Orthonormality of Scaling and Wavelet Functions 76	
5.2.4 Condition 4: Approximation Conditions (Smoothness Conditions) 77	
5.3 Designing Daubechies Orthogonal Wavelet System Coefficients 79	
5.3.1 Constraints for Daubechies' 6 Tap Scaling Function 80	
5.4 Design of Coiflet wavelets 81	
5.5 Symlets 82	
<i>Summary</i> 83	
<i>Exercises</i> 83	
<i>References</i> 85	
6. Discrete Wavelet Transform and Relation to Filter Banks	86–101
<i>Introduction</i> 86	
6.1 Signal Decomposition (Analysis) 86	
6.2 Relation with Filter Banks 89	
6.3 Frequency Response 94	
6.4 Signal Reconstruction: Synthesis from Coarse Scale to Fine Scale 96	
6.4.1 Upsampling and Filtering 97	
6.5 Perfect Matching Filters 98	
6.6 Computing Initial s_{j+1} Coefficients 100	
<i>Summary</i> 100	
<i>Exercises</i> 100	
<i>References</i> 101	
7. Generating and Plotting of Parametric Wavelets	102–124
<i>Introduction</i> 102	
7.1 Orthogonality Conditions and Parameterization 102	
7.2 Polyphase Matrix and Recurrence Relation 104	
7.2.1 Maple Code 106	
7.2.2 Parameterization of Length-6 Scaling Filter 107	
7.2.3 Parameterization of Length-8 Scaling Filter 107	
7.2.4 Parameterization of Length-10 Scaling Filter 108	
7.3 Pollen-type Parameterizations of Wavelet Bases 109	
7.4 Precise Numerical Evaluation of ϕ and ψ 110	
7.4.1 Method 1. Cascade Algorithm: Direct Evaluation at Dyadic Rational Points 110	
7.4.2 Method 2. Successive Approximation 114	
7.4.3 Method 3. Daubechies-Lagarias Algorithm 117	
7.4.4 Method 4. Subdivision Scheme 120	
<i>Summary</i> 123	
<i>Exercises</i> 123	
<i>References</i> 124	

8. Biorthogonal Wavelets	125–140
<i>Introduction</i>	125
8.1 Biorthogonality in Vector Space	125
8.2 Biorthogonal Wavelet Systems	127
8.3 Signal Representation Using Biorthogonal Wavelet System	129
8.4 Biorthogonal Analysis	129
8.5 Biorthogonal Synthesis—From Coarse Scale to Fine Scale	131
8.6 Construction of Biorthogonal Wavelet Systems	132
8.6.1 B-splines	132
8.6.2 B-spline Biorthogonal Wavelet System or Cohen-Daubechies-Feauveau Wavelets (CDF)	134
<i>Summary</i>	139
<i>Exercises</i>	139
<i>References</i>	140
9. Designing Wavelets—Frequency Domain Approach	141–152
<i>Introduction</i>	141
9.1 Basic Properties of Filter Coefficients	141
9.2 Choice of Wavelet Function Coefficients $\{g(k)\}$	144
9.3 Vanishing Moment Conditions in Fourier Domain	146
9.4 Derivation of Daubechies Wavelets	147
9.4.1 Daubechies Wavelets with One Vanishing Moment	151
9.4.2 Daubechies Wavelets with Two Vanishing Moment	151
<i>Summary</i>	152
<i>Exercises</i>	152
<i>References</i>	152
10. Lifting Scheme	153–195
<i>Introduction</i>	153
10.1 Wavelet Transform Using Polyphase Matrix Factorization	154
10.1.1 Inverse Lifting	158
10.1.2 Example: Forward Wavelet Transform	159
10.2 Geometrical Foundations of Lifting Scheme	161
10.2.1 Haar and Lifting	163
10.2.2 The Lifting Scheme	163
10.2.3 The Linear Wavelet Transform Using Lifting	167
10.2.4 Higher Order Wavelet Transform	170
10.3 Lifting Scheme in the Z-Domain	171
10.3.1 Design Example 1	176
10.3.2 Example 2: Lifting Haar Wavelet	180
10.4 Mathematical Preliminaries for Polyphase Factorization	182
10.4.1 Laurent Polynomial	182
10.4.2 The Euclidean Algorithm	183
10.4.3 Factoring Wavelet Transform into Lifting Steps—A Z-domain Approach	186

10.5 Dealing with Signal Boundary 191	
10.5.1 Circular Convolution 191	
10.5.2 Padding Policies 192	
10.5.3 Iteration Behaviour 193	
Summary 193	
Exercises 194	
References 195	
11. Image Compression	196–220
<i>Introduction</i> 196	
11.1 Overview of Image Compression Techniques 197	
11.1.1 The JPEG-Standard (ITU T.81) 199	
11.2 Wavelet Transform of an Image 200	
11.3 Quantization 203	
11.3.1 Uniform Quantization 203	
11.3.2 Subband Uniform Quantization 204	
11.3.3 Uniform Dead-Zone Quantization 205	
11.3.4 Non-Uniform Quantization 205	
11.4 Entropy Encoding 206	
11.4.1 Huffman Encoding 206	
11.4.2 Run Length Encoding 207	
11.5 EZW Coding (Embedded Zero-tree Wavelet Coding) 208	
11.5.1 EZW Performance 217	
11.6 SPIHT (Set Partitioning in Hierarchical Tree) 218	
11.7 EBCOT (Embedded Block Coding with Optimized Truncation) 218	
Summary 218	
<i>Weblems (Web based problems)</i> 219	
References 219	
12. Denoising	221–237
<i>Introduction</i> 221	
12.1 A Simple Explanation and a 1-D Example 221	
12.2 Denoising Using Wavelet Shrinkage—Statistical Modelling and Estimation 222	
12.3 Noise Estimation 223	
12.4 Shrinkage Functions 225	
12.5 Shrinkage Rules 227	
12.5.1 Universal 227	
12.5.2 Minimizing the False Discovery Rate 227	
12.5.3 Top 228	
12.5.4 Sure 228	
12.5.5 Translation Invariant Thresholding 229	
12.5.6 BayesShrink 229	
12.6 Denoising Images with Matlab 230	

12.7 Matlab Programs for Denoising	232
12.8 Simulation for Finding Effectiveness of Thresholding Method	233
<i>Summary</i>	235
<i>Exercises</i>	236
<i>References</i>	237

13. Spline Wavelets: Introduction and Applications to Computer Graphics

238–283

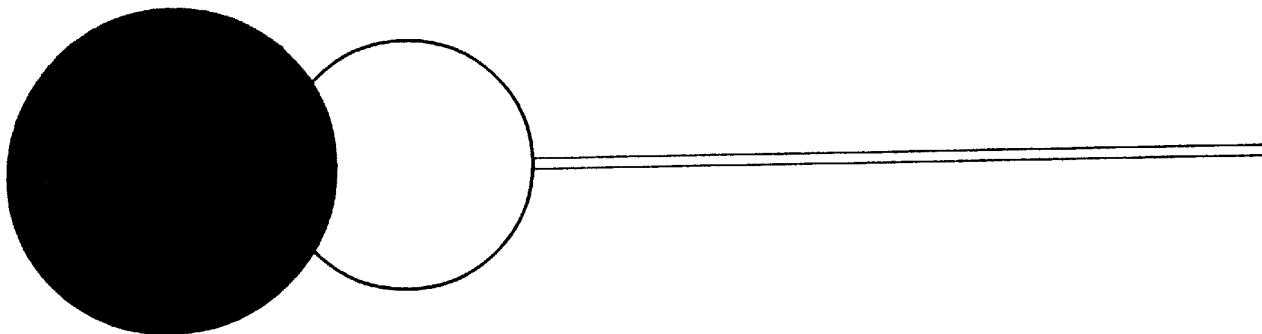
13.1 Introduction To Curves and Surfaces	238
13.1.1 Spline Curves and Surfaces	241
13.1.2 Cubic Spline Interpolation Methods	245
13.1.3 Hermite Spline Interpolation	245
13.1.4 Cubic Splines	247
13.1.5 Splines in Signal and Image Processing	249
13.1.6 Bezier Curves and Surfaces	250
13.1.7 Properties of Bezier Curves	252
13.1.8 Quadratic and Cubic Bezier Curves	253
13.1.9 Parametric Cubic Surfaces	255
13.1.10 Bezier Surfaces	257
13.1.11 B-Spline Curves	258
13.1.12 Cubic Periodic B-Splines	260
13.1.13 Conversion between Hermite, Bezier and B-Spline Representations	261
13.1.14 Non-Uniform B-Splines	262
13.1.15 Relation between Spline, Bezier and B-Spline Curves	264
13.1.16 B-Spline Surfaces	265
13.1.17 Beta-Splines and Rational Splines	265
13.2 Multiresolution Methods and Wavelet Analysis	266
13.3 The Filter Bank	268
13.4 Orthogonal and Semi-Orthogonal Wavelets	269
13.5 Spline Wavelets	270
13.6 Properties of Spline Wavelets	273
13.7 Advantages of B-Spline Based Wavelets in Signal Processing Applications	275
13.8 B-Spline Filter Bank	276
13.9 Multiresolution Curves and Faces	276
13.10 Wavelet Based Curve and Surface Editing	277
13.11 Variational Modelling and the Finite Element Method	278
13.12 Adaptive Variational Modelling Using Wavelets	281
<i>Summary</i>	282
<i>References</i>	282

Appendix

285–289

Index

291–293



Preface

In the past few years, the study of wavelets and the exploration of the principles governing their behaviour have brought about sweeping changes in the disciplines of pure and applied mathematics and sciences. One of the most significant development is the realization that, in addition to the canonical tool of representing a function by its Fourier series, there is a different representation more adapted to certain problems in data compression, noise removal, pattern classification and fast scientific computation.

Many books are available on wavelets but most of them are written at such a level that only research mathematicians can avail them. The purpose of this book is to make wavelets accessible to anyone (for example, graduate and undergraduate students) with a modest background in basic linear algebra and to serve as an introduction for the non-specialist. The level of the applications and the format of this book are such as to make this suitable as a textbook for an introductory course on wavelets.

Chapter 1 begins with a brief note on the origin of wavelets, mentioning the main early contributors who laid the foundations of the theory and on the recent developments and applications. Chapter 2 introduces the basic concepts in Fourier series and orients the reader to look at everything found in the Fourier kingdom from a geometrical point of view. In Chapter 3, the focus is on the continuous wavelet transform and its relation with short time Fourier transform. Readers who have not had much exposure to Fourier transforms earlier may skip this chapter, which is included only for the purpose of completeness.

Chapter 4 places the wavelet theory in a concrete setting using the Haar scaling and wavelet function. The rest of the book builds on this material. To understand the concepts in this chapter fully, the reader need to have only an understanding of the basic concepts in linear algebra: addition and multiplication of vectors by scalars, linear independence and dependence, orthogonal bases, basis set, vector spaces and function spaces and projection of vector/function on to the bases. The chapter introduces the concept of nested spaces, which is the corner stone of multiresolution analysis. Daubechies' wavelets are also introduced. The chapter concludes with a note on the fact that most of the wavelets are fractal in nature and that iterative methods are required to display wavelets.

Designing wavelets is traditionally carried out in the Fourier domain. Readers who are not experts in Fourier analysis usually find the theoretical arguments and terminology used quite

baffling and totally out of the world. This book, therefore, adopts a time domain approach to designing. The orthogonality and smoothness/regularity constraints are directly mapped on to constraints on the scaling and wavelet filter coefficients, which can then be solved using solvers available in Microsoft Excel—a spreadsheet package or a scientific computation package like MATLAB or Mathematica.

Engineers often view signal processing in terms of filtering by appropriate filters. Thus, Chapter 6 is devoted to establish the relationship between ‘signal expansion in terms of wavelet bases’ and the ‘filter bank’ approach to signal analysis/synthesis.

Chapter 7 discusses the theory behind parametric wavelets in an intuitive way rather than by using rigorous mathematical approach. The chapter also discusses various methods of plotting scaling and wavelet functions. The focus of Chapter 8 is on biorthogonal wavelets which is relatively a new concept. To drive home this concept to the readers, biorthogonality is explained using linear algebra. The chapter then goes on to discuss the design of elementary B-spline biorthogonal wavelets.

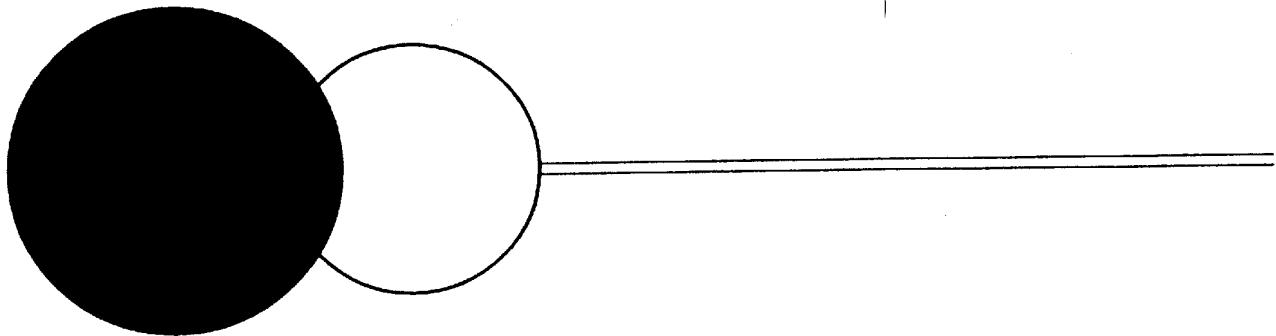
Chapter 9 addresses orthogonal wavelet design using the Fourier domain approach. Chapter 10 is devoted to the lifting scheme which provides a simple means to design wavelets with the desirable properties. The chapter also shows how the lifting scheme allows faster implementation of wavelet decomposition/reconstruction.

Chapter 11 to 13 describle applications of wavelets in Image Compression, Signal Denoising and Computer Graphics. The notations used in Chapter 13 are that used by researchers in this particular area and could be slightly different from those in the rest of the chapters.

To make the book more useful to the readers, we propose to post the teaching material (mainly PowerPoint slides for each chapter, and MATLAB/Excel demonstration programs) at the companion website of the book: www.amrita.edu/cen/publications/wavelets.

We earnestly hope that this book will initiate several persons to this exciting and vigorously growing area. Though we have spared no pains to make this book free from mistakes, some errors may still have survived our scrutiny. We gladly welcome all corrections, recommendations, suggestions and constructive criticism from our readers.

**K.P. SOMAN
K.I. RAMACHANDRAN**



Acknowledgments

First and foremost, we would like to express our gratitude to Brahmachari Abhayamrita Chaitanya, who persuaded us to take this project, and never ceased to lend his encouragement and support. We thank Dr. P. Venkat Rangan, Vice Chancellor of the university and Dr. K.B.M. Nambudiripad, Dean (Research)—our guiding stars—who continually showed us what perfection means and demanded perfection in everything that we did. We would like to thank, especially, Dr. P. Murali Krishna, a scientist at NPOL, Cochin, for his endearing support during the summer school on ‘Wavelets Fractals and Chaos’ that we conducted in 1998. It was then that we learned wavelets seriously. We take this opportunity to thank our research students C.R. Nitya, Shyam Divakar, Ajith Peter, Santhana Krishnan and V. Ajay for their help in simplifying the concepts. G. Sreenivasan and S. Sooraj, who helped us in drawing the various figures in the textbook, deserve a special thanks. Finally, we express our sincere gratitude to the editors of Prentice-Hall of India.

K.P. SOMAN
K.I. RAMACHANDRAN



CHAPTER

1

The Age of Wavelets

INTRODUCTION

Wavelet analysis is a new development in the area of applied mathematics. They were first introduced in seismology to provide a time dimension to seismic analysis that Fourier analysis lacked. Fourier analysis is ideal for studying stationary data (data whose statistical properties are invariant over time) but is not well suited for studying data with transient events that cannot be statistically predicted from the data's past. Wavelets were designed with such non-stationary data in mind, and with their generality and strong results have quickly become useful to a number of disciplines.

1.1 THE ORIGINS OF WAVELETS—ARE THEY FUNDAMENTALLY NEW?

Research can be thought of as a continuous growing fractal (see Figure 1.1) which often folds back onto itself. This folding back definitely occurred several times in the wavelet field. Even though as an organized research topic wavelets is less than two decades old, it arises from a constellation of related concepts developed over a period of nearly two centuries, repeatedly rediscovered by scientists who wanted to solve technical problems in their various disciplines. Signal processors were seeking a way to transmit clear messages over telephone wires. Oil prospectors wanted a better way to interpret seismic traces. Yet “wavelets” did not become a household word among scientists until the theory was liberated from the diverse applications in which it arose and was synthesized into a purely mathematical theory. This synthesis, in turn, opened scientists’ eyes to new applications. Today, for example, wavelets are not only the workhorse in computer imaging and animation; they also are used by the FBI to encode its data base of 30 million fingerprints (see Figure 1.2). In the future, scientists may put wavelet analysis for diagnosing breast cancer, looking for heart abnormalities (look at Figure 1.3) or predicting the weather.



FIGURE 1.1 Fractals that fold back on itself.



FIGURE 1.2 An FBI-digitized left thumb fingerprint. (The image on the left is the original; the one on the right is reconstructed from a 26:1 compression.)

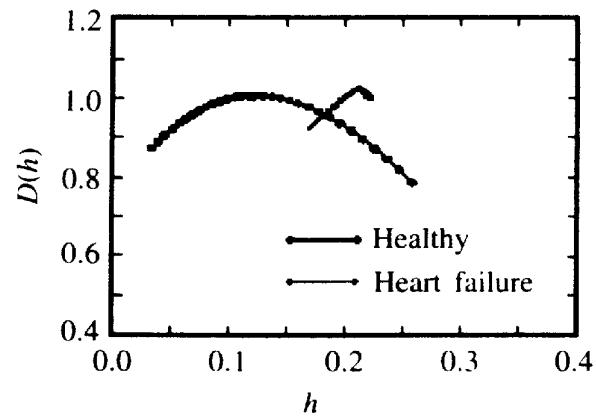


FIGURE 1.3 Multifractal spectrum of heart beat oscillations. It is a graph of singularity measure versus fractal dimension. This spectrum captures a different kind of information that cannot be captured by a frequency spectrum. Wavelets are used for multifractal spectrum estimation.

1.2 WAVELETS AND OTHER REALITY TRANSFORMS

Wavelet analysis allows researchers to isolate and manipulate specific types of patterns hidden in masses of data, in much the same way our eyes can pick out the trees in a forest, or our ears can pick out the flute in a symphony. One approach to understanding how wavelets do this is to start with the difference between two kinds of sounds—a tuning fork and the human voice (see Figures 1.4 and 1.5). Strike a tuning fork and you get a pure tone that lasts for a very long time. In mathematical theory, such a tone is said to be “localized” in frequency, that is, it consists of a single note with no higher-frequency overtones. A spoken word, by contrast, lasts for only a second, and thus is “localized” in time. It is not localized in frequency because the word is not a single tone but a combination of many different frequencies.

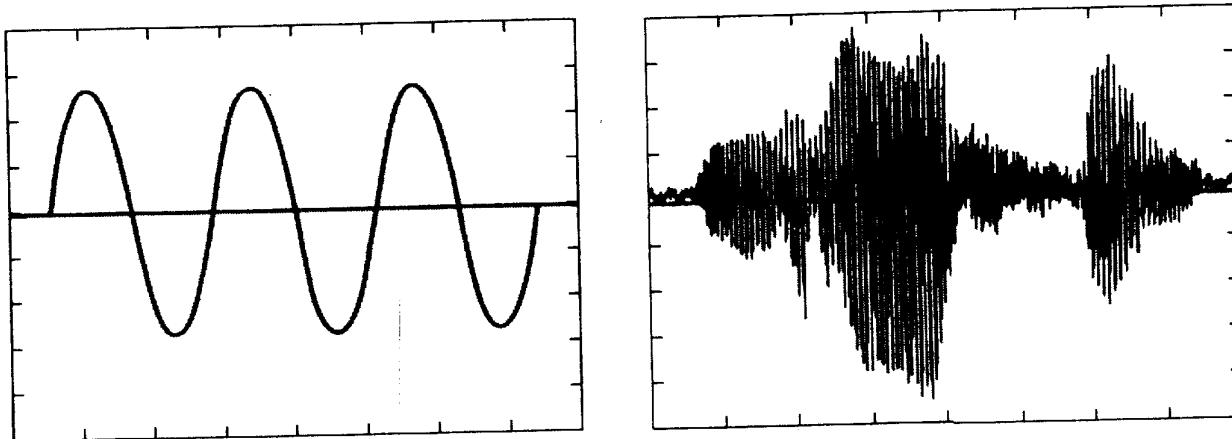


FIGURE 1.4 Graphs of the sound waves produced by a tuning fork (top) and the spoken word “greasy” (bottom) illustrate the difference between a tone localized in frequency and one localized in time. The tuning fork produces a simple “sine wave”.

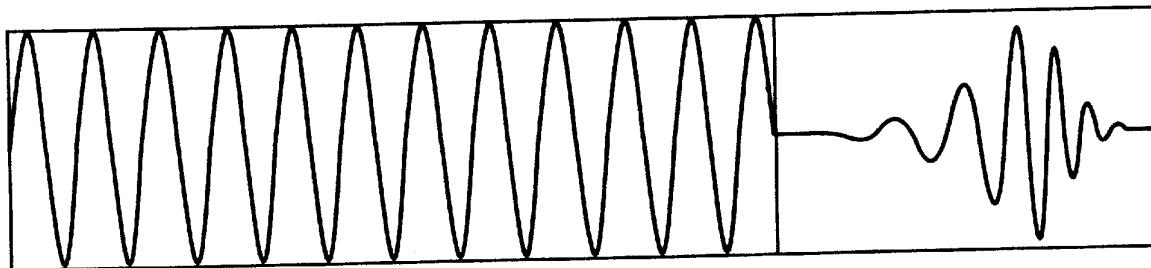


FIGURE 1.5 A wave and a wavelet.

Graphs of the sound waves produced by the tuning fork and human voice highlight the difference, as illustrated here. The vibrations of the tuning fork trace out what mathematicians call a sine wave, a smoothly undulating curve that, in theory, could repeat forever. In contrast, the graph of the word “greasy” contains a series of sharp spikes; there are no oscillations.

In the nineteenth century, mathematicians perfected what might be called the **tuning fork** version of reality, a theory known as *Fourier analysis*. Jean Baptiste Joseph Fourier, a French mathematician, claimed in 1807 that any repeating waveform (or periodic function), like the tuning fork sound wave, can be expressed as an infinite sum of sine waves and cosine waves of various frequencies. (A cosine wave is a sine wave shifted forward a quarter cycle.)

A familiar demonstration of Fourier's theory occurs in music. When a musician plays a note, he or she creates an irregularly shaped sound wave. The same shape repeats itself for as long as the musician holds the note. Therefore, according to Fourier, the note can be separated into a sum of sine and cosine waves. The lowest frequency wave is called the **fundamental frequency** of the note, and the higher frequency ones are called **overtones**. For example, the note A, played on a violin or a flute, has a fundamental frequency of 440 cycles per second and overtones with frequencies of 880, 1320, and so on. Even if a violin and a flute are playing the same note, they will sound different because their overtones have different strengths or "amplitudes". As music synthesizers demonstrated in the 1960s, a very convincing imitation of a violin or a flute can be obtained by recombining pure sine waves with the appropriate amplitudes. That, of course, is exactly what Fourier predicted back in 1807.

Mathematicians later extended Fourier's idea to non-periodic functions (or waves) that change over time, rather than repeating in the same shape forever. Most real-world waves are of this type: say, the sound of a motor that speeds up, slows down, and hiccups now and then. In images, too, the distinction between repeating and non-repeating patterns is important.

A repeating pattern may be seen as a texture or background while a non-repeating one is picked out by the eye as an object. Periodic or repeating waves composed of a discrete series of overtones can be used to represent repeating (background) patterns in an image. Non-periodic features can be resolved into a much more complex spectrum of frequencies, called the **Fourier transform**, just as sunlight can be separated into a spectrum of colours. The Fourier transform portrays the structure of a periodic wave in a much more revealing and concentrated form than a traditional graph of a wave would. For example, a rattle in a motor will show up as a peak at an unusual frequency in the Fourier transform.

Fourier transforms have been a hit. During the nineteenth century they solved many problems in physics and engineering. This prominence led scientists and engineers to think of them as the preferred way to analyze phenomena of all kinds. This ubiquity forced a close examination of the method. As a result, throughout the twentieth century, mathematicians, physicists, and engineers came to realize a drawback of the Fourier transform: they have trouble reproducing transient signals or signals with abrupt changes, such as the spoken word or the rap of a snare drum. Music synthesizers, as good as they are, still do not match the sound of concert violinists because the playing of a violinist contains transient features—such as the contact of the bow on the string—that are poorly imitated by representations based on sine waves.

The principle underlying this problem can be illustrated by what is known as the Heisenberg Indeterminacy Principle. In 1927, the physicist Werner Heisenberg stated that the position and the velocity of an object cannot be measured exactly at the same time even in theory. In signal processing terms, this means it is impossible to know simultaneously the exact frequency and the exact time of occurrence of this frequency in a signal. In order to know its frequency, the signal must be spread in time or vice versa. In musical terms, the trade-off means that any signal with a short duration must have a complicated frequency spectrum made of a

rich variety of sine waves whereas any signal made from a simple combination of a few sine waves must have a complicated appearance in the time domain. Thus, we can't expect to reproduce the sound of a drum with an orchestra of tuning forks.

1.3 MANAGING HEISENBERG'S UNCERTAINTY GHOST

Over the course of the twentieth century, scientists in different fields struggled to get around these limitations, in order to allow representations of the data to adapt to the nature of the information. In essence, they wanted to capture both the low-resolution forest—the repeating background signal—and the high-resolution trees—the individual, localized variations in the background. Although the scientists were trying to solve the problems particular to their respective fields, they began to arrive at the same conclusion—namely, that Fourier transforms themselves were to blame. They also arrived at essentially the same solution. Perhaps by splitting a signal into components that were not pure sine waves, it would be possible to condense the information in both the time and frequency domains. This is the idea that would ultimately be known as **wavelets**.

Wavelet transforms allow time-frequency localisation

The first entrant in the wavelet derby was a Hungarian mathematician named Alfred Haar, who introduced in 1909 the functions that are now called **Haar wavelets**. These functions consist simply of a short positive pulse followed by a short negative pulse. Although the short pulses of Haar wavelets are excellent for teaching wavelet theory, they are less useful for most applications because they yield jagged lines instead of smooth curves. For example, an image reconstructed with Haar wavelets looks like a cheap calculator display and a Haar wavelet reconstruction of the sound of a flute is too harsh.

From time to time over the next several decades, other precursors of wavelet theory arose. In the 1930s, the English mathematicians John Littlewood and R.E.A.C. Paley developed a method of grouping frequencies by octaves thereby creating a signal that is well localized in frequency (its spectrum lies within one octave) and also relatively well localized in time. In 1946, Dennis Gabor, a British-Hungarian physicist, introduced the Gabor transform, analogous to the Fourier transform, which separates a wave into “time-frequency packets” or “coherent states” (see Figure 1.6) that have the greatest possible simultaneous localization in both time and frequency.

And in the 1970s and 1980s, the signal processing and image processing communities introduced their own versions of wavelet analysis, going by such names as “subband coding,” “quadrature mirror filters” and the “pyramidal algorithm”.

While not precisely identical, all of these techniques had similar features. They decomposed or transformed signals into pieces that could be localized to any time interval and could also be dilated or contracted to analyze the signal at different scales of resolution. These precursors of wavelets had one other thing in common: no one knew about them beyond individual specialized communities. But in 1984, wavelet theory finally came into its own.

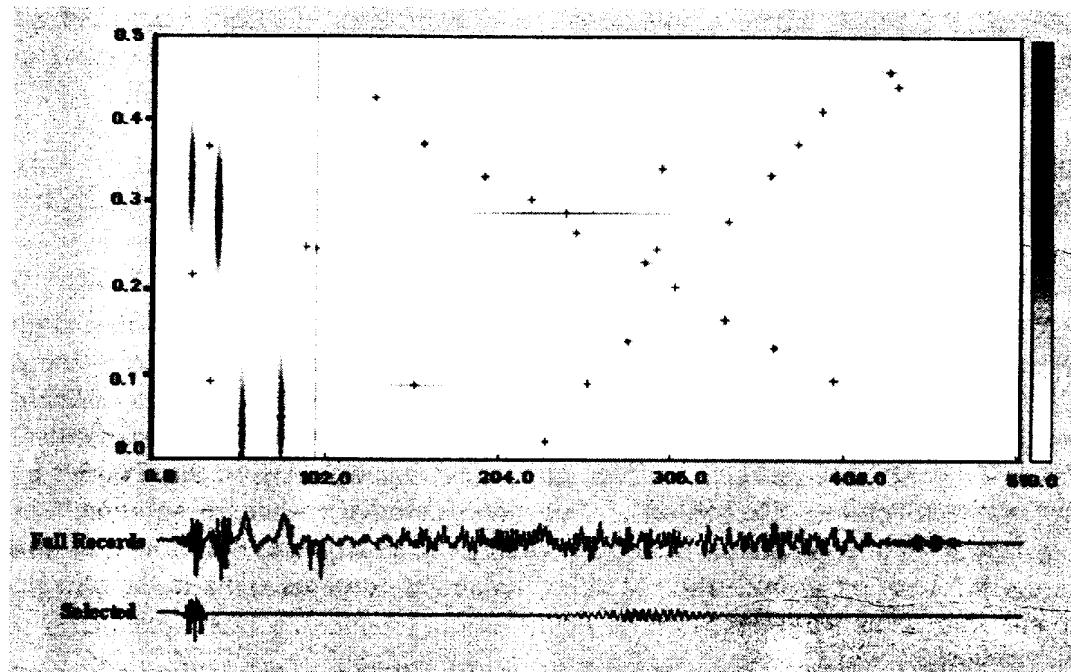


FIGURE 1.6 Decomposing signal into time-frequency atoms. Bottom of the picture shows two time frequency atoms. The signal and the time-frequency map is shown above that.

1.4 HISTORY OF WAVELET FROM MORLET TO DAUBECHIES VIA MALLAT

Jean Morlet didn't plan to start a scientific revolution. He was merely trying to give geologists a better way to search for oil.

Petroleum geologists usually locate underground oil deposits by making loud noises. Because sound waves travel through different materials at different speeds, geologists can infer what kind of material lies under the surface by sending seismic waves into the ground and measuring how quickly they rebound. If the waves propagate especially quickly through one layer, it may be a salt dome, which can trap a layer of oil underneath.

Figuring out just how the geology translates into a sound wave (or vice versa) is a tricky mathematical problem, and one that engineers traditionally solve with Fourier analysis. Unfortunately, seismic signals contain lots of transients—abrupt changes in the wave as it passes from one rock layer to another. Fourier analysis spreads that spatial information out all over the place.

Morlet, an engineer for Elf-Aquitaine, developed his own way of analyzing the seismic signals to create components that were localized in space, which he called **wavelets of constant shape**. Later, they would be known as **Morlet wavelets**. Whether the components are dilated, compressed or shifted in time, they maintain the same shape. Other families of wavelets can be built by taking a different shape, called a **mother wavelet**, and dilating, compressing or shifting it in time. Researchers would find that the exact shape of the mother wavelet strongly affects the accuracy and compression properties of the approximation. Many of the differences between earlier versions of wavelets simply amounted to different choices for the mother wavelet.

Morlet's method wasn't in the books but it seemed to work. On his personal computer, he could separate a wave into its wavelet components and then reassemble them into the original wave. But he wasn't satisfied with this empirical proof and began asking other scientists if the method was mathematically sound.

Morlet found the answer he wanted from Alex Grossmann, a physicist at the Centre de Physique Théorique in Marseilles. Grossmann worked with Morlet for a year to confirm that waves could be reconstructed from their wavelet decompositions. In fact, wavelet transforms turned out to work better than Fourier transforms because they are much less sensitive to small errors in the computation. An error or an unwise truncation of the Fourier coefficients can turn a smooth signal into a jumpy one or vice versa; wavelets avoid such disastrous consequences.

Morlet and Grossmann's paper, the first to use the word "wavelet", was published in 1984. Yves Meyer, currently at the École Normale Supérieure de Cachan, widely acknowledged as one of the founders of wavelet theory, heard about their work in the fall of the same year. He was the first to realize the connection between Morlet's wavelets and earlier mathematical wavelets, such as those in the work of Littlewood and Paley. (Indeed, Meyer has counted 16 separate rediscoveries of the wavelet concept before Morlet and Grossmann's paper.)

Meyer went on to discover a new kind of wavelet, with a mathematical property called **orthogonality** that made the wavelet transform as easy to work with and manipulate as a Fourier transform. ("Orthogonality" means that the information captured by one wavelet is completely independent of the information captured by another.) Perhaps most importantly, he became the nexus of the emerging wavelet community.

In 1986, Stéphane Mallat (see Figure 1.7), a former student of Meyer's who was working on a doctorate in computer vision, linked the theory of wavelets to the existing literature on



FIGURE 1.7 Stéphane Mallat (CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex, France).

subband coding and quadrature mirror filters which are the image processing community's versions of wavelets. The idea of multiresolution analysis—that is, looking at signals at different scales of resolution—was already familiar to experts in image processing. Mallat, collaborating with Meyer, showed that wavelets are implicit in the process of multiresolution analysis.

Thanks to Mallat's work, wavelets became much easier. One could now do a wavelet analysis without knowing the formula for a mother wavelet. The process was reduced to simple operations of averaging groups of pixels together and taking their differences, over and over. The language of wavelets also became more comfortable to electrical engineers, who embraced familiar terms such as "filters", "high frequencies" and "low frequencies".

Morlet's method wasn't in the books but it seemed to work. On his personal computer, he could separate a wave into its wavelet components and then reassemble them into the original wave. But he wasn't satisfied with this empirical proof and began asking other scientists if the method was mathematically sound.

Morlet found the answer he wanted from Alex Grossmann, a physicist at the Centre de Physique Théorique in Marseilles. Grossmann worked with Morlet for a year to confirm that waves could be reconstructed from their wavelet decompositions. In fact, wavelet transforms turned out to work better than Fourier transforms because they are much less sensitive to small errors in the computation. An error or an unwise truncation of the Fourier coefficients can turn a smooth signal into a jumpy one or vice versa; wavelets avoid such disastrous consequences.

Morlet and Grossmann's paper, the first to use the word "wavelet", was published in 1984. Yves Meyer, currently at the École Normale Supérieure de Cachan, widely acknowledged as one of the founders of wavelet theory, heard about their work in the fall of the same year. He was the first to realize the connection between Morlet's wavelets and earlier mathematical wavelets, such as those in the work of Littlewood and Paley. (Indeed, Meyer has counted 16 separate rediscoveries of the wavelet concept before Morlet and Grossmann's paper.)

Meyer went on to discover a new kind of wavelet, with a mathematical property called **orthogonality** that made the wavelet transform as easy to work with and manipulate as a Fourier transform. ("Orthogonality" means that the information captured by one wavelet is completely independent of the information captured by another.) Perhaps most importantly, he became the nexus of the emerging wavelet community.

In 1986, Stéphane Mallat (see Figure 1.7), a former student of Meyer's who was working on a doctorate in computer vision, linked the theory of wavelets to the existing literature on

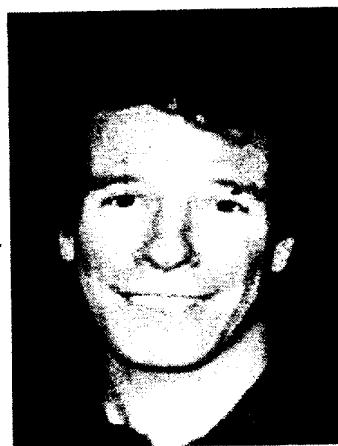


FIGURE 1.7 Stéphane Mallat (CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex, France).

subband coding and quadrature mirror filters which are the image processing community's versions of wavelets. The idea of multiresolution analysis—that is, looking at signals at different scales of resolution—was already familiar to experts in image processing. Mallat, collaborating with Meyer, showed that wavelets are implicit in the process of multiresolution analysis.

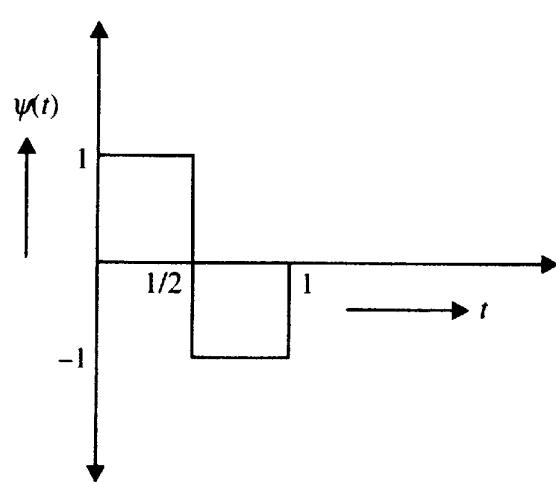
Thanks to Mallat's work, wavelets became much easier. One could now do a wavelet analysis without knowing the formula for a mother wavelet. The process was reduced to simple operations of averaging groups of pixels together and taking their differences, over and over. The language of wavelets also became more comfortable to electrical engineers, who embraced familiar terms such as "filters", "high frequencies" and "low frequencies".

The final great salvo in the wavelet revolution was fired in 1987, when Ingrid Daubechies (see Figure 1.8), while visiting the Courant Institute at New York University and later during

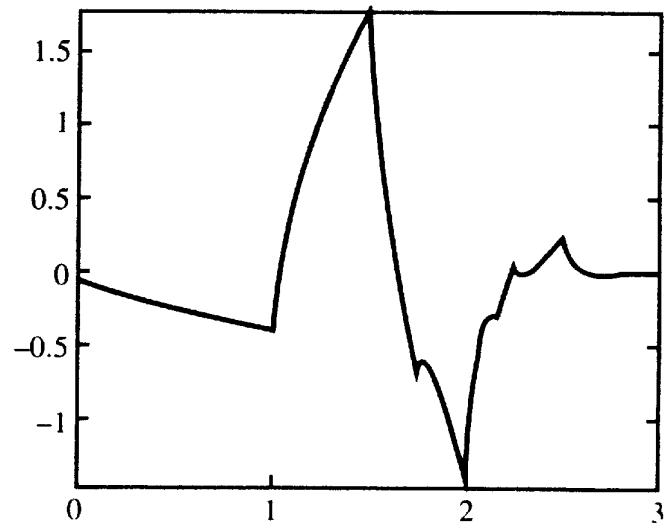


FIGURE 1.8 Ingrid Daubechies (Professor, Department of Mathematics, Princeton University).

her appointment at AT&T Bell Laboratories, discovered a whole new class of wavelets (see Figure 1.9b) which were not only orthogonal (like Meyer's) but which could be implemented using simple digital filtering ideas, in fact, using short digital filters. The new wavelets were almost as simple to program and use as Haar wavelets but they were smooth, without the jumps of Haar wavelets. Signal processors now had a dream tool: a way to break up digital data into contributions of various scales. Combining Daubechies and Mallat's ideas, there was a simple, orthogonal transform that could be rapidly computed on modern digital computers.

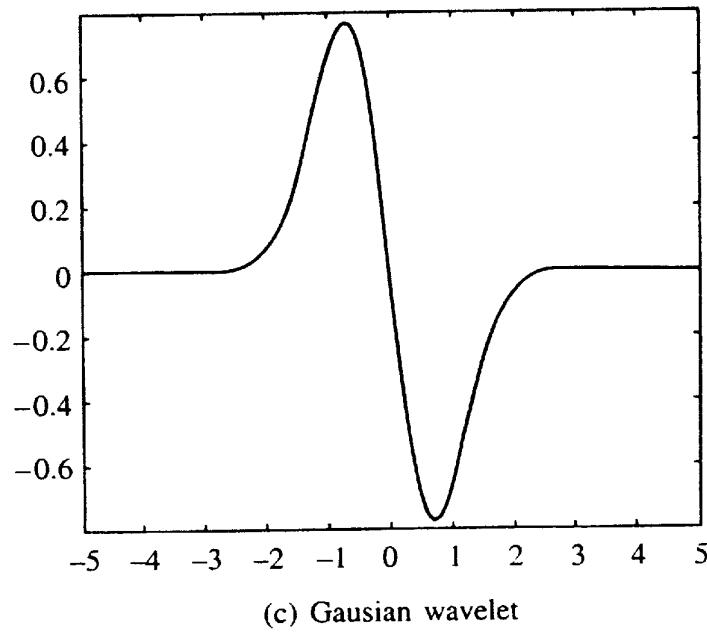


(a) Haar wavelet



(b) Daubechies' 4-tap Maxflat wavelet

FIGURE 1.9 (Cont.).



(c) Gausian wavelet

FIGURE 1.9 Graphs of several different types of wavelets.

The Daubechies wavelets have surprising features—such as intimate connections with the theory of fractals. If their graph is viewed under magnification, characteristic jagged wiggles can be seen, no matter how strong the magnification is. This exquisite complexity of detail means, there is no simple formula for these wavelets. They are ungainly and asymmetric; nineteenth-century mathematicians would have recoiled from them in horror. But like the Model-T Ford, they are beautiful because they work. The Daubechies wavelets turn the theory into a practical tool that can be easily programmed and used by any scientist with a minimum of mathematical training.

1.4.1 Different Communities of Wavelets

There are several instances of functions (see Figures 1.9 and 1.10) that can be used for multiresolution analysis of data. All of them are referred to as wavelets. Some of these instances are:

- *Dyadic translates and dilates of one function:* These are classical wavelets.
- *Wavelet packets:* This is an extension of the classical wavelets which yields basis functions with better frequency localization at the cost of slightly more expensive transform.
- *Local trigonometric bases:* The main idea is to work with cosines and sines defined on finite intervals combined with a simple but very powerful way to smoothly join the basis functions at the end points.
- *Multiwavelets:* Instead of using one fixed function to translate and dilate for making basis functions, we use a finite number of wavelet functions.

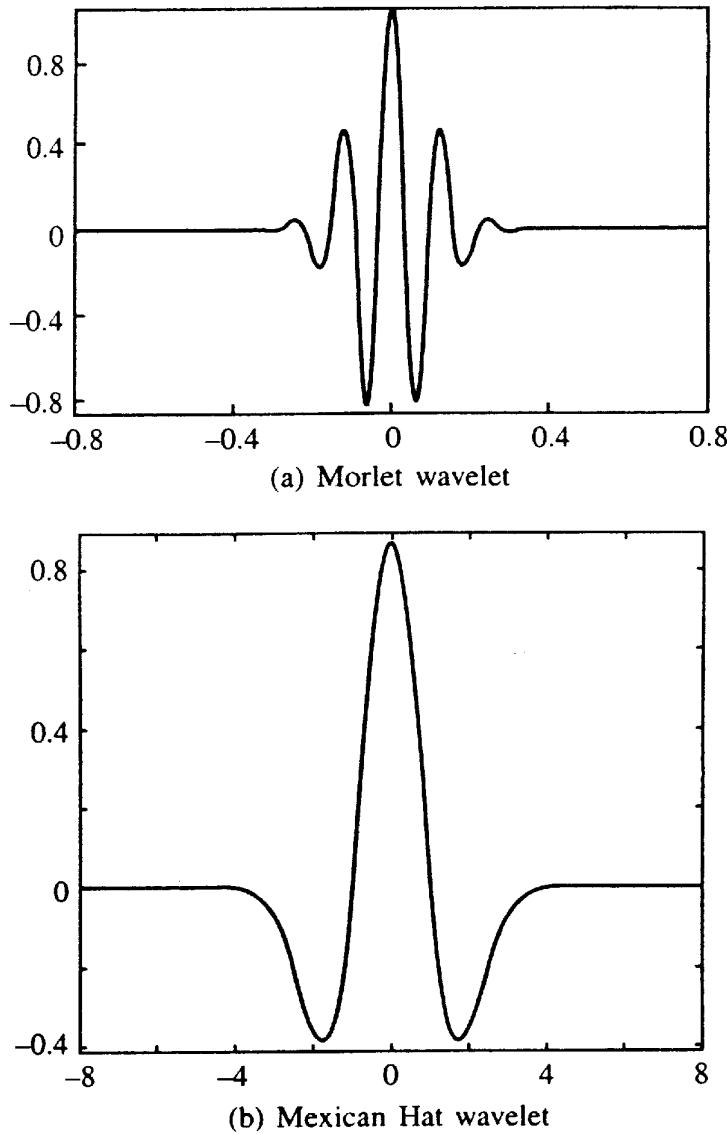


FIGURE 1.10 Wavelets used in continuous wavelet transform.

- *Second generation wavelets:* Here one entirely abandons the idea of translation and dilation. This gives extra flexibility which can be used to construct wavelets adapted to irregular samples.

1.4.2 Different Families of Wavelets within Wavelet Communities

Like humans, wavelets also live in families. Each member of a family has certain common features that distinguish each member of a family. Some wavelets are for continuous wavelet transform and others are for discrete wavelet transform. Some of the families that belong to ‘classical’ community of wavelets are (see Figures 1.10 and 1.11):

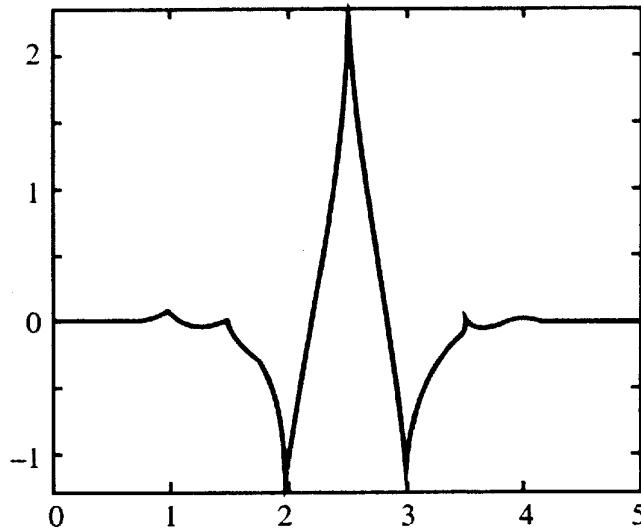


FIGURE 1.11 Coiflet wavelet.

- Wavelets for continuous wavelet transform (Gaussian, Morlet, Mexican Hat)
- Daubechies Maxflat wavelets
- Symlets
- Coiflets
- Biorthogonal spline wavelets
- Complex wavelets

1.4.3 Interesting Recent Developments

- Wavelet based denoising has opened up other fields and important techniques such as dictionaries and non-linear approximation, smoothing and reduction to small optimization problems are real achievements.
- Wavelets have had a big psychological impact. People from many different areas became interested in time-frequency and time-scale transforms. There has been a revolution in signal processing. There is less specialization and the subject is now opened to new problems. More than just a simple tool, wavelet ideas prompt new points of view. Some of the best ideas aren't written down. The big difference will come from new generation researchers now growing up amidst wavelet ideas.
- Wavelets have advanced our understanding of singularities. The singularity spectrum completely characterizes the complexity of the data. Now we must go to an understanding of the underlying phenomenon to get an equation from the solution. Wavelets don't give all the answers but they force us to ask right questions.
- Wavelets can be used to distinguish coherent versus incoherent parts of turbulence in fluid flow. They give some information but don't entirely solve the problem.
- The results on regularity, approximation power and wavelet design techniques have led to significant developments in signal and image processing.

1.5 WAVELETS IN THE FUTURE

With the foundations of wavelet theory securely in place, the field has grown rapidly over the last decade. A distribution list on wavelets that began with 40 names in 1990 is now an online newsletter with more than 17,000 subscribers. Moreover, it has continued to evolve through a healthy mix of theory and practice. Engineers are always trying new applications, and for mathematicians, there are still important theoretical questions to be answered.

Although wavelets are best known for image compression, many researchers are interested in using wavelets for pattern recognition. In weather forecasting, for example, they might slim down the data-bloated computer models that are now in use. Traditionally, such models sample the barometric pressure (for instance) at an enormous number of grid points and use this information to predict how the data will evolve. However, this approach uses a lot of computer memory. A model of the atmosphere that uses a 1000-by-1000-by-1000 grid requires a billion data points—and it's still a fairly crude model.

However, most of the data in the grid are redundant. The barometric pressure in your town is probably about the same as the barometric pressure a mile down the road. If the weather models used wavelets, they could view the data the same way weather forecasters do, concentrating on the places where abrupt changes occur—warm fronts, cold fronts and the like. Other problems in fluid dynamics have been tackled the same way. At Los Alamos National Laboratory, for example, wavelets are used to study the shock waves produced by a bomb explosion.

As demonstrated by the recent spate of full-length computer-animated films, wavelets also have a promising future in the movies. Because the wavelet transform is a reversible process, it is just as easy to synthesize an image (build it up out of wavelets) as it is to analyze it (break it down into wavelet components). This idea is related to a new computer animation method called **subdivision surfaces**, basically a multiresolution analysis run in reverse. To draw a cartoon character, the animator only has to specify where a few key points go, creating a low-resolution version of the character. The computer can then do a reverse multiresolution analysis, making the character look like a real person and not a stick figure.

Subdivision surfaces debuted in the 1998 movie *A Bug's Life*, replacing a more clumsy method called **NURBS** (for non-uniform rational B splines) that had been used in the first Toy Story movie in 1995. Interestingly, NURBS and subdivision methods coexisted in 1999's Toy Story 2 where the characters that appeared in the first Toy Story continued to be NURBS while the new characters were based on the subdivision method. The next frontier for subdivision surfaces may be the video game industry where they could eliminate the blocky look of today's graphics.

Meanwhile, on the theoretical side, mathematicians are still looking for better kinds of wavelets for two- and three-dimensional images. Although the standard wavelet methods are good at picking up edges, they do it one pixel at a time—an inefficient way of representing something that may be a very simple curve or line. David Donoho (see Figure 1.12) and Emmanuel Candès of Stanford University have proposed a new class of wavelets called **ridgelets**, which are specifically designed to detect discontinuities along a line. Other researchers are studying "multiwavelets" which can be used to encode multiple signals travelling through the same line, such as colour images in which three colour values (red, green and blue) have to be transmitted at once.



FIGURE 1.12 David Donoho (Statistics Department, Stanford University, Stanford, CA 94305, USA).

When asked to justify the value of mathematics, mathematicians often point out that ideas developed to solve a pure mathematical problem can lead to unexpected applications years later. But the story of wavelets paints a more complicated and somewhat more interesting picture. In this case, specific applied research led to a new theoretical synthesis which in turn opened scientists' eyes to new applications. Perhaps the broader lesson of wavelets is that we should not view basic and applied sciences as separate endeavours: good science requires us to see both the theoretical forest and the practical trees.

SUMMARY

Though wavelet as an organised research topic, is only two decades old, it has been in use for a long time in various disciplines under different names. Morlet and Grossman were the first to use the word '*wavelet*'. Stephen Mallat brought out the relation between wavelet methodology used by Morlet and filter bank theory used in image processing applications. The greatest contribution came from Ingrid Daubechies who put the whole theory on a strong mathematical foundation. Wavelets are now emerging as one of the fastest growing field with application ranging from seismology to astrophysics.

REFERENCES

- [1] Daubechies, I., Where do wavelet come from? A personal point of view, *Proceedings of the IEEE*, Special issue on wavelets, Vol. 84, no. 4, pp. 510–513, April 1996.
- [2] Sweldons, W., Wavelets what next? *Proceedings of the IEEE*, Vol. 84, no. 4, April 1996.
- [3] An Introduction to wavelets:
“<http://www.amara.com/IEEEwave/IEEEwavelet.html>”
- [4] Amara's wavelet page:
“<http://www.amara.com/current/wavelet.html>”



Fourier Series and Geometry

INTRODUCTION

Deemed one of the crowning achievements of the 20th Century, the Fourier series has applications that are far reaching in various fields of science and mathematics. The Discrete Fourier transform is one particular tool widely used in today's age of computers and solid state electronics. From graphic equalizers in stereos to the most advanced scientific sampling software, the usefulness of this mathematical feat is astounding. Most of the readers of this book might already know this fact but many of them may not be knowing that Fourier series, as a mathematical tool in analyzing signals, has strong connection with geometry. In this chapter our aim is to understand the theory of Fourier series from a geometrical viewpoint.

We assume that you are familiar with vector algebra and co-ordinate geometry. If you are really comfortable in using these topics, you can very well understand what Fourier series is, computation and interpretation of Fourier series coefficients, Fourier transform, discrete Fourier transform, fast Fourier transform, etc. There exists a strong analogy between what you do in vector algebra and what you do in signal processing. This analogy will help you to visualize and give interpretation to the processes and output of the processes that you do on signals. Development of this geometrical mental picture about signal processing is the main aim of this chapter. Einstein once said "Imagination is more important than knowledge". Signal processing, which many students consider as dry and non-intuitive, demands imagination and some form of abstract thinking from the part of students. In fact, it requires only that. Once you develop a conceptual link between geometry (vector algebra) and signal processing, you need not remember any formula and every formula that you come across will become transparent to your mind. Here we will refresh concepts from *vector space*. The detailed exposition of vector space is given in Appendix A. In this introductory chapter, we won't try to be 100% mathematically precise in our statements regarding vector space. Instead our aim is to relate geometry and Fourier series in an intuitive way.

2.1 VECTOR SPACE

Any vector in 3-dimensional space can be represented as $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$. $\vec{i}, \vec{j}, \vec{k}$ are unit vectors in three orthogonal directions. This orthogonality is expressed by requiring the condition that $\vec{i} \cdot \vec{j} = \vec{k} \cdot \vec{j} = \vec{i} \cdot \vec{k} = 0$. The orthogonality condition ensures that the representation of any vector using $\vec{i}, \vec{j}, \vec{k}$ is unique.

2.1.1 Bases

We call $\vec{i}, \vec{j}, \vec{k}$ as bases of space \mathbb{R}^3 . By this we mean that any vector in \mathbb{R}^3 can be represented using $\vec{i}, \vec{j}, \vec{k}$ vectors. We also say $\vec{i}, \vec{j}, \vec{k}$ span the space \mathbb{R}^3 . Let $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$ be a vector in \mathbb{R}^3 . Continuously changing scalar a, b, c ; we will go on get new vectors in \mathbb{R}^3 . We imagine that, set of all such vectors constitute the vector space \mathbb{R}^3 . We express this truth by

$$\overline{\text{Span}}_{a,b,c}(a\vec{i} + b\vec{j} + c\vec{k}) \equiv \mathbb{R}^3$$

2.1.2 Orthonormality

Norm of a vector $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$ is conventionally defined as $\sqrt{a^2 + b^2 + c^2}$ and denoted as $\|\vec{V}\|$. $\|\vec{V}\|$ is equal to $\sqrt{\vec{V} \cdot \vec{V}}$. We interpret this quantity as the magnitude of the vector. It must be noted that other definitions are also possible for norm. What is interest to us is that our basis vectors of \mathbb{R}^3 , i.e., $\vec{i}, \vec{j}, \vec{k}$ are having unity norm. Hence the name unit vectors. They are also orthogonal. Therefore, they are called **orthonormal vectors** or **orthonormal bases**. How does it help us? Given a vector $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$, we can easily tell to which direction $\vec{i}, \vec{j}, \vec{k}$, the vector \vec{V} , is more inclined by noting the value of a, b, c .

2.1.3 Projection

Given a vector \vec{V} how shall we find its normal component vectors or in other words how shall we find the scalar coefficients a, b and c ? We project \vec{V} on to bases $\vec{i}, \vec{j}, \vec{k}$ to get a, b and c . This is the direct result of orthogonality of our basis vectors. Since $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$,

$$\vec{V} \cdot \vec{i} = (a\vec{i} + b\vec{j} + c\vec{k}) \cdot \vec{i} = a$$

$$\vec{V} \cdot \vec{j} = (a\vec{i} + b\vec{j} + c\vec{k}) \cdot \vec{j} = b \quad \text{and}$$

$$\vec{V} \cdot \vec{k} = (a\vec{i} + b\vec{j} + c\vec{k}) \cdot \vec{k} = c$$

So projecting vector \vec{V} on to a base of space \mathfrak{R}^3 (spanned by a set orthonormal vectors) gives the corresponding coefficient (component) of the base. There is a parallel for this in Fourier series analysis.

With these ideas firmly engraved in your mind, you are ready to understand and visualize Fourier series.

2.2 FUNCTIONS AND FUNCTION SPACES

In vector space, we represent a vector in terms of orthogonal unit vectors called **bases**. Fourier series is an extension of this idea for functions (electronics engineers call them as signals). A function is expressed in terms of a set of orthogonal functions. To correlate with ideas in geometry, we now need to introduce the concept of orthogonality and norm with respect to functions. Note that, function is quantity which varies with respect to one or more running parameter, usually time and space. Orthogonality of functions depends on multiplication and integration of functions. Multiplication and integration of function must be thought of as equivalent to projection of a vector on to another. Here comes the requirement for abstract thinking, mentioned at the start of the chapter.

2.2.1 Orthogonal Functions

Two real functions $f_1(t)$ and $f_2(t)$ are said to be orthogonal if and only if

$$\int_{-\infty}^{\infty} f_1(t) f_2(t) dt = 0 \quad (2.1)$$

Ponder over this equation. What we are doing? We are first doing a point by point multiplication of two function. Then we are summing up the area under the resulting function (obtained after multiplication). If this area is zero, we say, the two functions are orthogonal. We also interpret it as similarity of functions $f_1(t)$ and $f_2(t)$. If $f_1(t)$ and $f_2(t)$ vary synchronously, that is, if $f_1(t)$ goes up $f_2(t)$ also goes up and if $f_1(t)$ goes down $f_2(t)$ also goes down then we say the two functions have high similarity. Otherwise, they are dissimilar. Magnitude or absolute value of $\int_{-\infty}^{\infty} f_1(t) f_2(t) dt$ will be high if they vary synchronously and zero if they vary perfectly asynchronous away.

For example consider two functions $f_1(t) = \sin t$ and $f_2(t) = \cos t$ for t in the interval $(0, 2\pi)$. Figure 2.1 shows the two functions and their point wise product. Observe that the net area under the curve $\sin t \cos t$ is zero. That is, the area above the t axis is same as the area below the t axis. For these two functions, it can be easily shown that, for t in the range of multiples of 2π , area under the product curve is always zero. Therefore, we say that $\sin t$ and $\cos t$ are orthogonal in the range $(0, 2\pi)$. Specification of range is very important when you say orthogonality of functions. Because the deciding factor is area under the curve which in turn

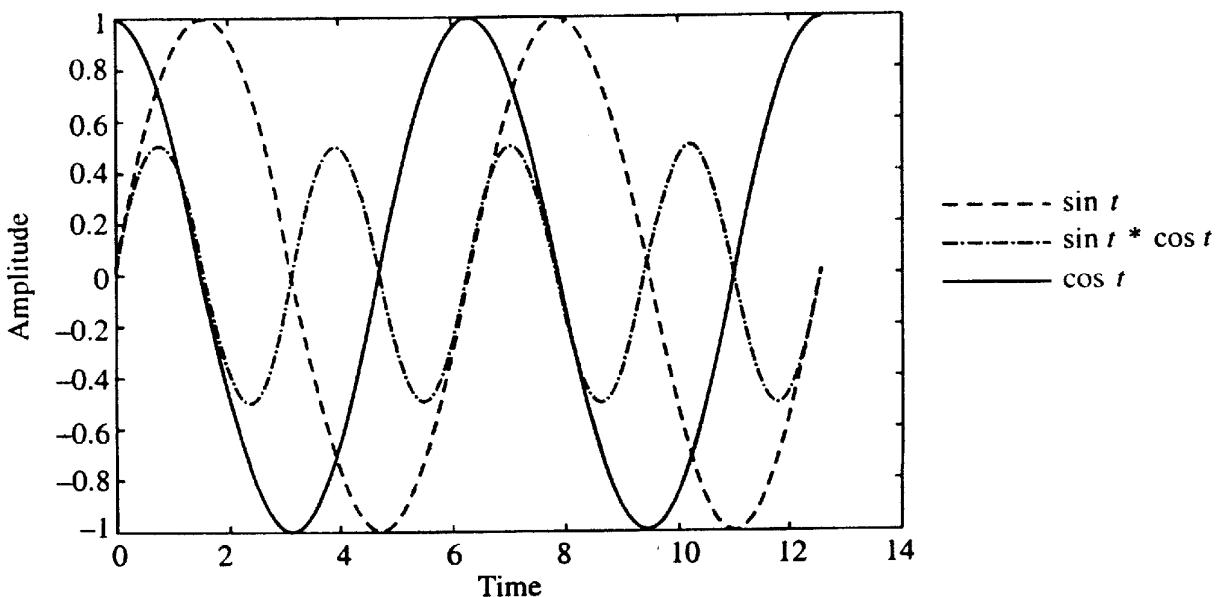


FIGURE 2.1 Orthogonal functions.

depends on the range of integration. Now here observe the variation of $\sin t$ and $\cos t$. When $\sin t$ is increasing, $\cos t$ is decreasing and vice versa. For range of t in multiples of 2π , the two functions considered are said to be orthogonal.

The geometrical analog of $\int f_1(t) f_2(t) dt = 0$ is the dot product, $\vec{i} \cdot \vec{j} = 0$. For the given interval, when $f_1(t)$ is projected on to $f_2(t)$ (projection here means multiplying and integrating), if the result is zero, we say the two functions are orthogonal in the given interval. Clearly look at the analogy. Taking a dot product in vector space (multiplying corresponding terms and adding) is equivalent to multiplying and integrating in the function space (point wise multiplication and integration). The dot product is maximum when two vectors are collinear and zero when they are at 90° . This analogy will help you to easily visualize and interpret various results in signal processing.

2.2.2 Orthonormal Functions

Two real functions $f_1(t)$ and $f_2(t)$ are said to be orthonormal if and only if $\int_{-\infty}^{\infty} f_1(t) f_2(t) dt = 0$ and $\int_{-\infty}^{\infty} f_i(t) f_i(t) dt = 1$, for $i = 1, 2$.

2.2.3 Function Spaces

Just like unit orthogonal set of vectors span vector spaces, orthogonal set of functions can span function spaces. Consider the following functions defined over a period T :

Let us assume that $\omega = 2\pi/T$. Then (2.2)

- (i) $u(t) = 1$ for $0 \leq t \leq T$
- (ii) $\cos n\omega t$, $n = 1, 2, \dots$; $0 \leq t \leq T$
- (iii) $\sin n\omega t$, $n = 1, 2, \dots$; $0 \leq t \leq T$

Here we have infinite number of functions of sines and cosines. These set of functions are mutually orthogonal in the interval $(0, T)$. Let us verify the truth through some representative examples.

Let us take functions $u(t) = 1$ and $\sin \omega t$ for $0 \leq t \leq T$. These functions and their product function is shown in Figure 2.2. Point wise multiplication gives the resulting curve as $\sin \omega t$.

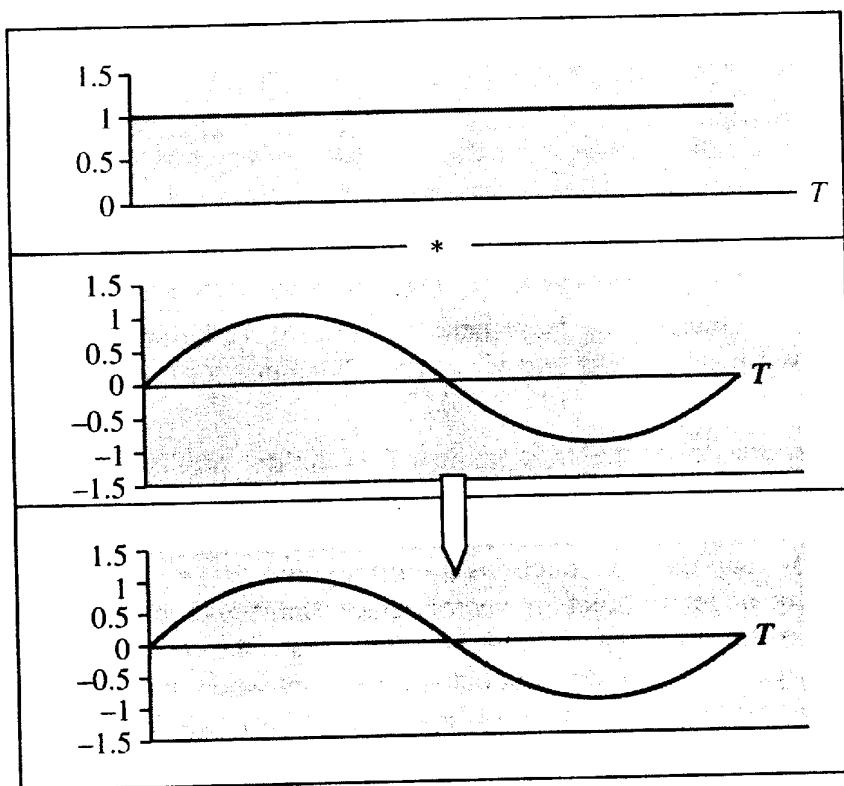


FIGURE 2.2 Orthogonality of functions.

What is the area under the curve $\sin \omega t$ in the interval $(0, T)$? Obviously zero. Mathematically,

$$\int_{t=0}^T u(t) \sin \omega t \, dt = \int_{t=0}^T 1 \cdot \sin(2\pi t/T) \, dt = 0$$

We say $u(t) = 1$ is orthogonal to $\sin \omega t$ in $(0, T)$. Do not forget the relation between ω and T . Similarly we can show that $u(t) = 1$ is orthogonal to $\cos \omega t$ in $(0, T)$.

In general,

$$\int_{t=0}^T u(t) \sin n\omega t \, dt = 0 \text{ also } \int_{t=0}^T u(t) \cos n\omega t \, dt = 0 \text{ for all } n = 1, 2, \dots$$

Consider $\cos \omega t$ and $\sin \omega t$ in $(0, T)$. Figure 2.3 shows the two functions and corresponding function obtained after point wise multiplication. Area above and below of the

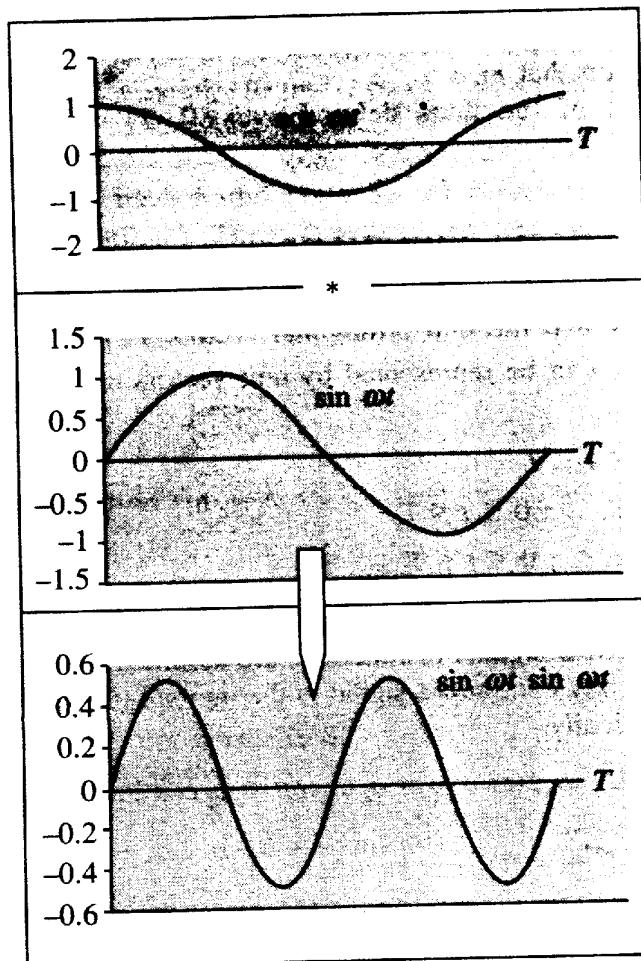


FIGURE 2.3 Plots of $\sin \omega t$, $\cos \omega t$ and $\sin \omega t \cos \omega t$.

resulting curve is same implying that the net area is zero. This in turn imply that $\cos \omega t$ and $\sin \omega t$ in $(0, T)$ are orthogonal.

Generalizing,

$$\int_{t=0}^T \sin n\omega t \cos m\omega t dt = 0, \text{ for all } m, n = 1, 2, 3, \dots,$$

Similarly,

$$\int_{t=0}^T \sin n\omega t \sin m\omega t dt = 0$$

(for all integer m and n such that $m \neq n$).

Also

$$\int_{t=0}^T \cos n\omega t \cos m\omega t dt = 0$$

(for all integer m and n such that $m \neq n$).

Thus, we have a set of functions defined over $(0, T)$ such that they are mutually orthogonal.

2.2.4 Orthogonal Basis Functions

In vector algebra, we know that the unit orthogonal vectors $\vec{i}, \vec{j}, \vec{k}$ span the space \mathbb{R}^3 . This imply that any vector in \mathbb{R}^3 can be represented by unit vectors or bases $\vec{i}, \vec{j}, \vec{k}$. Similarly the set of functions,

- (i) $u(t) = 1$ for $0 \leq t \leq T$
- (ii) $\cos n\omega t, n = 1, 2, \dots; 0 \leq t \leq T$
- (iii) $\sin n\omega t, n = 1, 2, \dots; 0 \leq t \leq T$

defined over $0 \leq t \leq T$ span a function space and we denote it as L^2 (set of square integrable functions). Most of the practical signals or functions are assumed to belong in the space of L^2 . This means that, practically any continuous signal in the interval $(0, T)$ can be represented using the above bases. Mathematically,

$$f(t) \in L^2 \Rightarrow f(t) = a_0 + a_1 \cos \omega t + a_2 \cos 2\omega t + \dots + b_1 \sin \omega t + b_2 \sin 2\omega t + \dots \quad (2.3)$$

Let us look at the concept from another angle. Consider the sum $f(t)$ for each t in the range $(0, T)$ given in Eq. (2.3). Choose a set of coefficients a_0, a_1, \dots , and b_1, b_2, \dots , and make the function (or signal) $f(t)$ in the range $(0, T)$. Now go on choose different set of coefficients to get a different signal. The set of all such possible signal generated is our signal space L^2 in the range $(0, T)$. Practically all smoothly varying functions that you can imagine in the interval $(0, T)$ belongs to L^2 . Or in other words, practically all smoothly varying functions that you can imagine in the interval $(0, T)$ can be represented by the equation:

$$f(t) = a_0 + a_1 \cos \omega t + a_2 \cos 2\omega t + \dots + b_1 \sin \omega t + b_2 \sin 2\omega t + \dots$$

The number of coefficients required to represent a signal depends on how smooth and how fast your signal is varying in the range $(0, T)$.

2.2.5 Orthonormality and the Method of Finding the Coefficients

The problem that we are addressing is finding the coefficient set a_0, a_1, b_1, a_2, b_2 , etc. In vector algebra, orthonormality of basis vectors allowed us to find the component of a vector along a

particular base by projecting the vector on to that base. The property that $\|i\| = \|j\| = \|k\| = 1$ where $\|i\| = \sqrt{i \cdot i}$ and i, j, k are orthogonal implies that for a vector $\vec{V} \in \mathbb{R}^3$, if $\vec{V} \cdot i = a, \vec{V} \cdot j = b, \vec{V} \cdot k = c$ then $\vec{V} = ai + bj + ck$. It is a unique representation using the bases i, j, k . To reiterate the principle, to find a particular coefficient, project the vector on to the corresponding basis. The same mental picture is applicable to Fourier series representation of signals.

In the last section, we have found out a set of functions that are orthogonal among themselves in a given range $(0, T)$. These functions are orthogonal but we haven't checked whether they form an orthonormal set. In function space, orthonormality of two functions $f_1(t)$ and $f_2(t)$ in $(0, T)$ requires that

$$\int_0^T f_1(t) f_2(t) dt = 0, \quad \int_0^T f_1(t) f_1(t) dt = 1 \text{ and } \int_0^T f_2(t) f_2(t) dt = 1$$

Consider the basis function $\sin n\omega t$.

$$\int_0^T \sin n\omega t \sin n\omega t dt = \int_0^T \sin\left(\frac{2\pi nt}{T}\right) \sin\left(\frac{2\pi nt}{T}\right) dt = \frac{T}{2}$$

The result is a function of T and is not equal to 1. For any integer $n > 0$, the integral is $T/2$. So, we say our sine bases are not normalized.

Similarly it can be easily shown that,

$$\int_0^T \cos n\omega t \cos n\omega t dt = \int_0^T \cos\left(\frac{2\pi nt}{T}\right) \cos\left(\frac{2\pi nt}{T}\right) dt = \frac{T}{2} \text{ for } n = 1, 2, \dots,$$

So our cosine bases are also not normalized bases. What about our first basis function (which is also called DC base) $u(t) = 1$, in $(0, T)$?

$$\text{As, } \int_0^T 1^2 dt = T, \quad u(t) \text{ is also not normalized.}$$

Therefore, none of our bases are normalized. So how shall we normalize our bases? Multiply all the sine and cosine bases with $\sqrt{2/T}$ and $u(t) = 1$ with $\sqrt{1/T}$.

Now,

$$\int_0^T \sqrt{2/T} \sin n\omega t \sqrt{2/T} \sin n\omega t dt = \frac{2}{T} \int_0^T \sin\left(\frac{2\pi nt}{T}\right) \sin\left(\frac{2\pi nt}{T}\right) dt = \frac{2}{T} \cdot \frac{T}{2} = 1$$

$$\int_0^T \sqrt{2/T} \cos n\omega t \sqrt{2/T} \cos n\omega t dt = \frac{2}{T} \int_0^T \cos\left(\frac{2\pi nt}{T}\right) \cos\left(\frac{2\pi nt}{T}\right) dt = \frac{2}{T} \cdot \frac{T}{2} = 1$$

$$\int_0^T \sqrt{1/T} \cdot \sqrt{1/T} dt = \frac{1}{T} \int_0^T 1 dt = \frac{1}{T} \cdot T = 1$$

The set of normalized bases of Fourier series are:

$$\left\{ \sqrt{\frac{1}{T}}, \sqrt{\frac{2}{T}} \cos \omega t, \sqrt{\frac{2}{T}} \sin \omega t, \sqrt{\frac{2}{T}} \cos 2\omega t, \sqrt{\frac{2}{T}} \sin 2\omega t, \dots, \sqrt{\frac{2}{T}} \cos n\omega t, \sqrt{\frac{2}{T}} \sin n\omega t, \dots \right\}$$

All these functions are defined in the interval $(0, T)$ and $\omega = 2\pi/T$.

Let us now take a function $f(t) \in L^2$ defined in the interval $(0, T)$ and express it using the normalized sine and cosine bases defined over the same period $(0, T)$. Note that the length of bases are same as the length of the signal. Also note that the fundamental frequency of the sinusoidal (also 'cosinusoidal') series is given by $\omega = 2\pi/T$. (This is something which students tend to ignore or forget and hence the repetition of the fact in many places.)

$$\begin{aligned} f(t) &= a'_0 \frac{1}{\sqrt{T}} + a'_1 \sqrt{\frac{2}{T}} \cos \omega t + b'_1 \sqrt{\frac{2}{T}} \sin \omega t + \dots \\ &\quad + a'_n \sqrt{\frac{2}{T}} \cos n\omega t + b'_n \sqrt{\frac{2}{T}} \sin n\omega t + \dots \end{aligned} \quad (2.4)$$

Here $a'_0, a'_1, a'_2, \dots, b'_1, b'_2, \dots$ are called **normalized Fourier series coefficients**.

Now, given that, such a representation is possible, how shall we find out the normalized coefficients. The answer is, project $f(t)$ on to the corresponding bases. Suppose we want to find a'_n . Then project $f(t)$ on to $\sqrt{2/T} \cos n\omega t$, that is,

$$a'_n = \int_0^T f(t) \sqrt{\frac{2}{T}} \cos n\omega t dt$$

Note the analogy. In vector algebra, to find a component coefficient, we project the vector on to the corresponding unit vector (base). Let us verify the truth through mathematical microscope. Multiplying both sides of Eq. (2.4) by $\sqrt{2/T} \cos n\omega t$ and integrating over $(0, T)$, we obtain

$$\begin{aligned} \int_0^T f(t) \sqrt{2/T} \cos n\omega t dt &= a'_0 \frac{1}{\sqrt{T}} \cdot \int_0^T \sqrt{2/T} \cos n\omega t dt \\ &\quad + a'_1 \int_0^T \sqrt{2/T} \cos \omega t \sqrt{2/T} \cos n\omega t dt \\ &\quad + b'_1 \int_0^T \sqrt{2/T} \sin \omega t \sqrt{2/T} \cos n\omega t dt + \dots \end{aligned}$$

$$\begin{aligned}
 & + a'_n \int_0^T \sqrt{2/T} \cos n\omega t \sqrt{2/T} \cos n\omega t dt \\
 & + b'_n \int_0^T \sqrt{2/T} \sin n\omega t \sqrt{2/T} \sin n\omega t dt + \dots \\
 & = a'_n
 \end{aligned}$$

Thus, $\int_0^T f(t) \sqrt{2/T} \cos n\omega t dt = a'_n$ since $\int_0^T \sqrt{2/T} \cos n\omega t \sqrt{2/T} \cos n\omega t dt = 1$ and all other integral vanishes because of orthogonality property of the bases.

Similarly $b'_n = \int_0^T f(t) \sqrt{2/T} \sin n\omega t dt$. That is, project $f(t)$ on to the corresponding normalized base.

What about a'_0 ? Project $f(t)$ on to the base $u(t) = \sqrt{1/T}$ defined over $(0, T)$, giving

$$a'_0 = \int_0^T f(t) \sqrt{1/T} dt$$

Well, we understood how to get normalized coefficients in Fourier representation of signals. To get a particular coefficient simply project $f(t)$ on to the corresponding normalized base. Projection in function space means point wise multiplication and integration of the functions concerned over the defined period.

After obtaining the normalized coefficients, we substitute in Eq. (2.4) to get Fourier series representation of the signal. Let us simplify Eq. (2.4) to avoid scalars $\sqrt{2/T}$ and $\sqrt{1/T}$.

Let $a'_0 1/\sqrt{T} = a_0$ so that

$$a_0 = \sqrt{1/T} a'_0 = \sqrt{1/T} \int_0^T \sqrt{1/T} f(t) dt = \frac{1}{T} \int_0^T f(t) dt$$

(Note that we substituted the expression for a'_0 to get a_0 .)

Similarly let $a'_n \sqrt{2/T} = a_n$. Then

$$a_n = \sqrt{2/T} a'_n = \sqrt{2/T} \int_0^T f(t) \sqrt{2/T} \cos n\omega t dt = \frac{2}{T} \int_0^T f(t) \cos n\omega t dt$$

Let $b'_n \sqrt{2/T} = b_n$. Then

$$b_n = \sqrt{2/T} b'_n = \sqrt{2/T} \int_0^T f(t) \sqrt{2/T} \sin n\omega t dt = \frac{2}{T} \int_0^T f(t) \sin n\omega t dt$$

Thus, we can rewrite Eq. (2.4) as:

$$f(t) = a_0 + a_1 \cos \omega t + b_1 \sin \omega t + \cdots + a_n \cos n\omega t + b_n \sin n\omega t + \cdots \quad (2.5)$$

where

$$a_0 = \frac{1}{T} \int_0^T f(t) dt \quad (2.6)$$

$$a_n = \frac{2}{T} \int_0^T f(t) \cos n\omega t dt, \quad n = 1, 2, 3, \dots, \quad (2.7)$$

$$b_n = \frac{2}{T} \int_0^T f(t) \sin n\omega t dt, \quad n = 1, 2, 3, \dots, \quad (2.8)$$

These are the formulae given in most of the textbooks.

2.2.6 Complex Fourier Series

We know that

$$\sin \omega t = \frac{e^{j\omega t} - e^{-j\omega t}}{2j}$$

and

$$\cos \omega t = \frac{e^{j\omega t} + e^{-j\omega t}}{2}$$

Now,

$$a_1 \cos \omega t + b_1 \sin \omega t = \frac{(a_1 - jb_1)}{2} e^{j\omega t} + \frac{(a_1 + jb_1)}{2} e^{-j\omega t}$$

$$\text{Let us denote } C_1 = \left(\frac{a_1 - jb_1}{2} \right) \text{ and } C_{-1} = \left(\frac{a_1 + jb_1}{2} \right)$$

The Fourier series can now be rewritten as:

$$\begin{aligned} f(t) &= C_0 + C_1 e^{j\omega t} + C_{-1} e^{-j\omega t} + \cdots + C_n e^{jn\omega t} + C_{-n} e^{-jn\omega t} + \cdots \\ &= \sum_{n=-\infty}^{\infty} C_n e^{jn\omega t} \end{aligned} \quad (2.9)$$

Note that the formulation involves complex exponentials and the concept of negative angular frequency $-\omega$. There is nothing imaginary in any signal and there is no negative frequency. Complex exponentials facilitate easy manipulation of multiplication and integration involved in the calculation of Fourier series coefficients than multiplication and integration using plain sine and cosine functions. Let us first try to visualize $e^{j\omega t}$, where $\omega = 2\pi/T$ and T is the period of our signal.

By Demovies formula,

$$e^{j\omega t} = \cos \omega t + j \sin \omega t$$

We can think of this function as a 'bundle' of sine and cosine function kept together side by side such that they do not 'mix' with each other. The function of j can be thought of as a separator indicating that the two parts, viz. $\cos \omega t$ and $\sin \omega t$ are orthogonal.

2.2.7 Orthogonality of Complex Exponential Bases

The set of bases $(1, e^{j\omega t}, e^{-j\omega t}, \dots, e^{jn\omega t}, e^{-jn\omega t}, \dots)$ are orthogonal over the interval $[0, T]$, where $\omega = 2\pi/T$ and T is the duration of the signal which we are representing using the bases. However, there is a twist with respect to interpretation of orthogonality and projection of a function on to a complex function. To make you prepared for that twist and to accept that twist, let us consider $\vec{V} = a\vec{i} + b\vec{j} + c\vec{k}$, an element of \Re^3 . The norm of this vector (or magnitude) is obtained by projecting \vec{V} on to itself and then taking the square root of resulting quantity. That is

$$\|V\| = \sqrt{\vec{V} \cdot \vec{V}} = \sqrt{a^2 + b^2 + c^2}$$

But in case of a complex number,

$$Z = a + jb$$

to find the magnitude, we have to multiply Z with its complex conjugate \bar{Z} and then take the square root. In case of real function, say $f(t)$, we find norm by

$$\int_t f(t) f(t) dt$$

In case our function is complex, say $Z(t)$, its norm is given by

$$\int_t Z(t) \bar{Z}(t) dt$$

This has implication in the formula for projection of a function on to a complex function. We must multiply the first function with complex conjugate of the second function and then integrate. With this twist in mind, we will prove that the set of functions $(1, e^{j\omega t}, e^{-j\omega t}, \dots, e^{jn\omega t}, e^{-jn\omega t}, \dots)$ are orthogonal. We denote the projection of $f_1(t)$ on to $f_2(t)$ as $\langle f_1(t), f_2(t) \rangle$.

Case 1: $u(t) = 1$ and $e^{j\omega t}$, for $0 \leq t \leq T$ and $\omega = 2\pi/T$

$$\langle 1, e^{j\omega t} \rangle = \int_{t=0}^T 1 \cdot \overline{e^{j\omega t}} dt = \int_0^T e^{-j\omega t} dt = \int_0^T e^{-j(2\pi t/T)} dt = 0$$

26 • Insight into Wavelets—From Theory to Practice

Case 2: $e^{jn\omega t}$ and $e^{-jm\omega t}$ where n and m are positive integers.

$$\langle e^{jn\omega t}, e^{-jm\omega t} \rangle = \int_{t=0}^T e^{jn\omega t} \overline{e^{-jm\omega t}} dt = \int_0^T e^{j(n+m)\omega t} dt = \int_0^T e^{j(n+m)(2\pi/T)t} dt = 0$$

Case 3: $e^{jn\omega t}$ and $e^{jm\omega t}$ where n and m are either positive or negative integers with $m \neq n$.

$$\langle e^{jn\omega t}, e^{jm\omega t} \rangle = \int_{t=0}^T e^{jn\omega t} \overline{e^{jm\omega t}} dt = \int_0^T e^{j(n-m)\omega t} dt = \int_0^T e^{j(n-m)(2\pi/T)t} dt = 0$$

Cases 1, 2 and 3 prove that the functions are orthogonal. What about orthonormality of complex exponential functions?

Since,

$$\langle e^{jn\omega t}, e^{jn\omega t} \rangle = \int_{t=0}^T e^{jn\omega t} \overline{e^{jn\omega t}} dt = \int_0^T e^{j(n-n)\omega t} dt = \int_0^T 1 \cdot dt = T$$

and

$$\langle u(t), u(t) \rangle = \int_0^T 1 \cdot dt = T, \text{ the functions are not orthonormal}$$

The functions though orthogonal do not satisfy the requirement of orthonormality. So we scale the functions to make them orthonormal among themselves.

For both unit function and complex exponentials, scaling constant is $\sqrt{1/T}$. Therefore, orthonormal bases are:

$$(\sqrt{1/T}, \sqrt{1/T} e^{j\omega t}, \sqrt{1/T} e^{-j\omega t}, \dots, \sqrt{1/T} e^{jn\omega t}, \sqrt{1/T} e^{-jn\omega t}, \dots)$$

We denote the signal space generated by these bases as L^2 . Now, any signal which is an element of L^2 can be written as:

$$f(t) = C'_0 \sqrt{1/T} u(t) + C'_1 \sqrt{1/T} e^{j\omega t} + C'_{-1} \sqrt{1/T} e^{-j\omega t} + \dots + C'_n \sqrt{1/T} e^{jn\omega t} + C'_{-n} \sqrt{1/T} e^{-jn\omega t} + \dots \quad (2.10)$$

To get a particular coefficient, we project $f(t)$ on to the corresponding base.

$$\text{i.e., } C'_0 = \int_0^T f(t) \sqrt{1/T} u(t) dt \quad (2.11)$$

$$C'_n = \int_0^T f(t) \sqrt{1/T} e^{-jn\omega t} dt \quad (2.12)$$

$$C'_{-n} = \int_0^T f(t) \sqrt{1/T} e^{-jn\omega t} dt \quad (2.13)$$

This result follows from the orthonormality of bases used in (2.10).

Now, we simplify the expression given in Eq. (2.10) by introducing new constants.

Let $C_0 = C'_0 \sqrt{1/T}$

Since

$$C'_0 = \int_0^T f(t) \sqrt{1/T} u(t) dt$$

Therefore,

$$C_0 = \sqrt{1/T} \int_0^T f(t) \sqrt{1/T} u(t) dt = \frac{1}{T} \int_0^T f(t) u(t) dt = \frac{1}{T} \int_0^T f(t) dt \quad (2.14)$$

Similarly

$$C_n = C'_n \sqrt{1/T} = \frac{1}{T} \int_0^T f(t) e^{-jn\omega t} dt \quad (2.15)$$

and

$$C_{-n} = C'_{-n} \sqrt{1/T} = \frac{1}{T} \int_0^T f(t) e^{jn\omega t} dt \quad (2.16)$$

Thus, we obtain complex fourier series in standard form:

$$f(t) = C_0 + C_1 e^{j\omega t} + C_{-1} e^{-j\omega t} + \dots + C_n e^{jn\omega t} + C_{-n} e^{-jn\omega t} + \dots \quad (2.17)$$

with the coefficients defined in Eqs. (2.14) to (2.16).

This completes formal introduction to Fourier series. The reader who is not familiar with advanced Fourier methods can skip Chapter 3 and proceed reading Chapter 4.

SUMMARY

It is no understatement that the Fourier series especially discrete Fourier transform is considered one of the crowning achievements of the 20th Century. The theory of Fourier series has strong connection with geometry or at least we can understand the Fourier transform theory from a geometrical viewpoint. We had the honour and a privilege to bring it to the reader in this user friendly, low fat and no cholesterol form. In the forthcoming chapters on wavelets, this geometrical viewpoint will help you to digest the theory in a better way. It will surely prevent intellectual diarrhea.

EXERCISES

2.1 Consider the interval $[-1, 1]$ and the basis functions $(1, t, t^2, t^3, \dots)$ for $L^2[-1, 1]$.

The inner product on this vector space is defined as $\langle f, g \rangle = \int_{-1}^1 f(t) \bar{g}(t) dt$. We can

show that these functions do not form an orthonormal basis. Given a finite or countably infinite set of linearly independent vectors x_i , we can construct an orthonormal set y_i with the same span as x_i as follows:

- Start with $y_1 = \frac{x_1}{\|x_1\|}$
- Then, recursively set $y_2 = \frac{x_2 - \langle x_2, y_1 \rangle y_1}{\|x_2 - \langle x_2, y_1 \rangle y_1\|}$
- Therefore, $y_k = \frac{x_k - v_k}{\|x_k - v_k\|}$, where $v_k = \sum_{i=1}^{k-1} \langle x_k, y_i \rangle y_i$

This procedure is known as **Gram-Schmidt orthogonalization** and it can be used to obtain an orthonormal basis from any other basis. Following is a MATLAB function that implements the Gram-Schmidt algorithm. Your input must be a set of linearly independent functions. The output will be a set of orthonormal functions that span the same space.

```

function z=gramschmidt(x,N,M);
for i=1:N;
    s(i,:)=x(i,:);
end;
e(1)=s(1,:)*conj(s(1,:).');
phi(1,:)=s(1,:)/sqrt(e(1));
for i=2:N;
    th(i,:)=zeros(1,M);
    for r=i-1:-1:1;
        th(i,:)=th(i,:)+(s(i,:)*conj(phi(r,:).'))*phi(r,:);
    end;
    th(i,:)=s(i,:)-th(i,:);
    e(i)=th(i,:)*conj(th(i,:).'); // note.' means
        transpose without conjugation
    phi(i,:)=th(i,:)/sqrt(e(i));
end;
z=phi(1:N,:);

```

Apply the algorithm on Legendre polynomials and plot the first six orthonormal basis functions.

- 2.2 Project the function $f(x) = x$ on to the space spanned by $\phi(x), \psi(x), \psi(2x), \psi(2x - 1) \in L^2[0, 1]$ where

$$\phi(x) = \begin{cases} 1, & 0 \leq x \leq 1/2 \\ 0, & \text{otherwise} \end{cases}$$

$$\psi(x) = \begin{cases} 1 & 0 \leq x \leq 1/2 \\ -1 & 1/2 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Sketch the projection of $f(x)$ to this space.

- 2.3 Show that the set of functions $f_m(x) = \sqrt{\frac{2}{\pi}} \sin mx, m = 1, 2, \dots$, is an orthonormal system in $L^2[0, \pi]$.

- 2.4 Given that

$$H_4 = \left\{ \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix} \right\}$$

is an orthonormal basis for \mathfrak{R}^4 and $z = [z]_E = \begin{bmatrix} 9 \\ 7 \\ 3 \\ 5 \end{bmatrix} \in \mathfrak{R}^4$, where E is the standard basis for \mathfrak{R}^4 , that is,

$$E = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Find z in terms of basis in H_4 .

- 2.5 (a) Find the Fourier series of $f(t) = t$ for $-\pi \leq t < \pi$.

- (b) Using Plancherel's formula and (a), evaluate the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$.

[Hint: According to Plancherel's formula $\|f\|^2 = \sum_{n \in \mathbb{Z}} |C_n|^2$, where the C_n s are complex Fourier series coefficients of the function $f(t)$. $\|f\|^2$ in the given problem

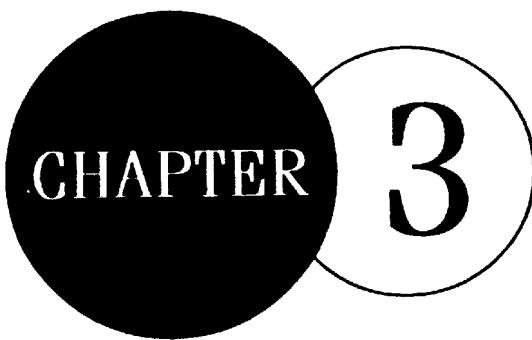
$$\text{is } \frac{1}{2\pi} \int_{-\pi}^{\pi} |f(t)|^2 dt]$$

2.6 Find the Fourier series of

$$f(x) = \begin{cases} x + \pi & \text{for } -\pi \leq x < 0 \\ \pi & \text{for } 0 \leq x < \pi \end{cases}$$

REFERENCES

- [1] Leon, Steven, *Linear Algebra with Applications*, Prentice Hall, New York, 1998.
- [2] Strang, Gilbert, *Introduction to Linear Algebra*, Cambridge Press, Massachusetts, 1998.
- [3] Burden, Richard L., *Numerical Analysis*, Brook/Cole, Albany, 1997.
- [4] Ramirez, Robert W., *The FFT: Fundamentals and Concepts*, Prentice Hall, New York, 1985.
- [5] Lyons, Richard, *Understanding Digital Signal Processing*, Addison-Wesley, Menlo Park, CA, 1998.
- [6] Arnold, Dave, *I am the Resource for all that is Math: A Contemporary Perspective*, C.R., CA, 1998.



Continuous Wavelet and Short Time Fourier Transform

INTRODUCTION

The aperiodic, noisy, intermittent, transient signals are the type of signals for which wavelet transforms are particularly useful. Wavelets have special ability to examine signals simultaneously in both time and frequency. This has resulted in the development of a variety of wavelet based methods for signal manipulation and interrogation. Current applications of wavelet include climate analysis, financial time series analysis, heart monitoring, condition monitoring of rotating machinery, seismic signal denoising, denoising of astronomical images, crack surface characterization, characterization of turbulent intermittency, audio and video compression, compression of medical and thump impression records, fast solution of partial differential equations, computer graphics and so on. Some of these applications require continuous wavelet transform, which we will explore in this chapter and find out how it differs from classical methods that deals with aperiodic noisy signals.

3.1 WAVELET TRANSFORM—A FIRST LEVEL INTRODUCTION

Wavelet mean ‘small wave’. So wavelet analysis is about analyzing signal with short duration finite energy functions. They transform the signal under investigation into another representation which presents the signal in a more useful form. This transformation of the signal is called **wavelet transform**. Unlike Fourier transform, we have a variety of wavelets that are used for signal analysis. Choice of a particular wavelet depends on the type of application in hand. Figures 3.1 to 3.4 show examples of some real and complex wavelets.

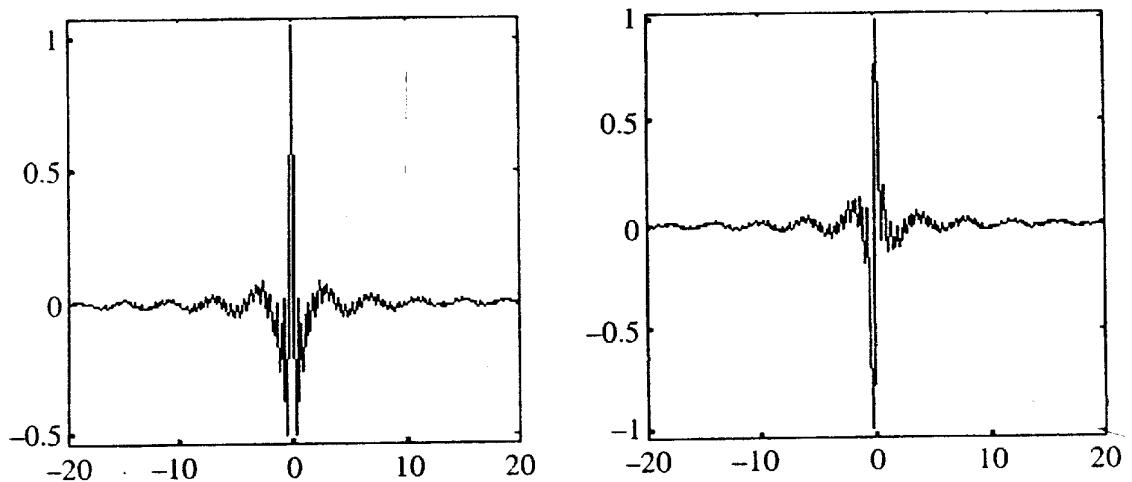


FIGURE 3.1 Real and imaginary part of Shan wavelet.

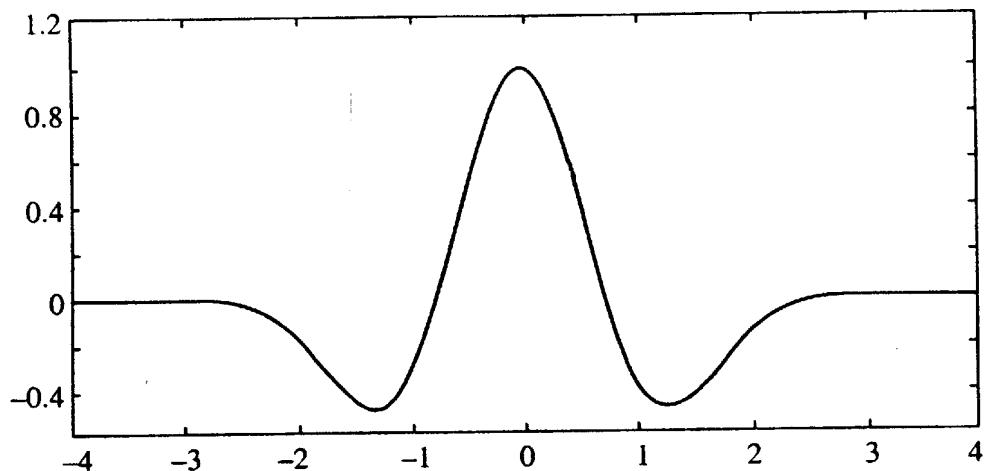


FIGURE 3.2 Quintic Spline wavelet.

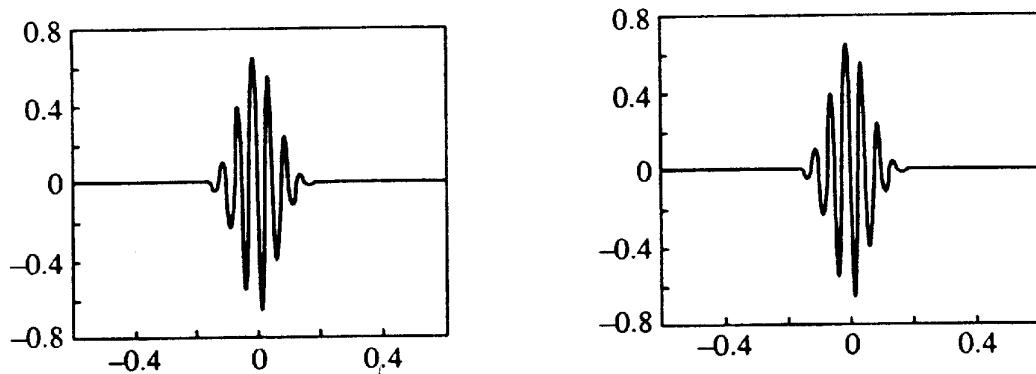


FIGURE 3.3 Real and imaginary part of Gabor wavelet.

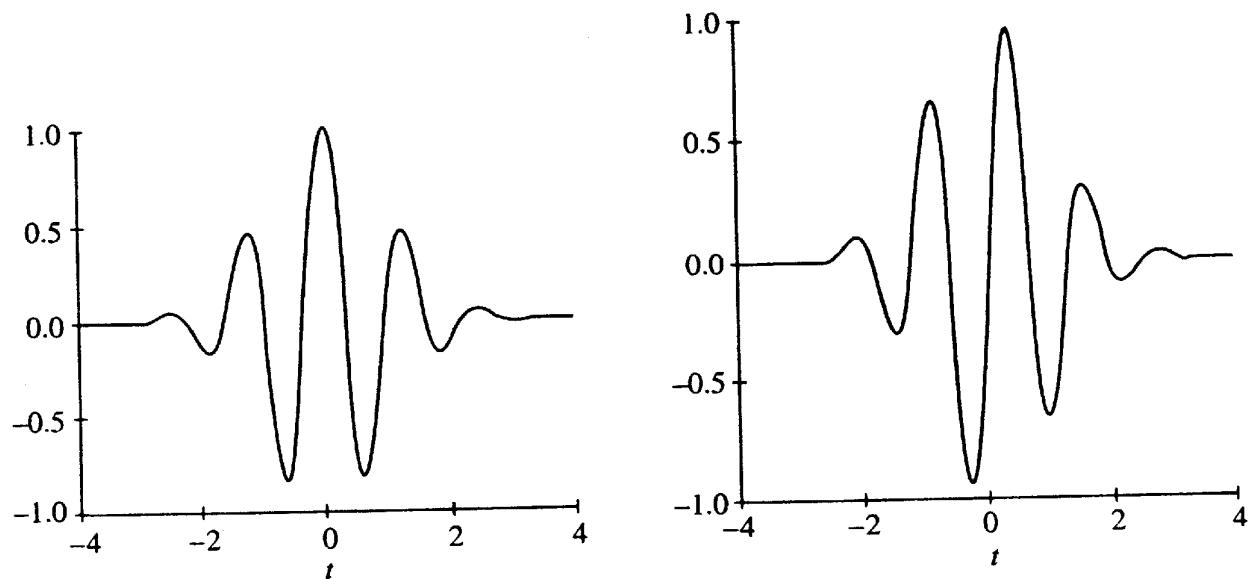


FIGURE 3.4 Real and imaginary part of Morlet wavelet.

We manipulate wavelet in two ways. The first one is translation. We change the central position of the wavelet along the time axis. The second one is scaling. Figures 3.5 and 3.6 show translated and scaled versions of wavelets.

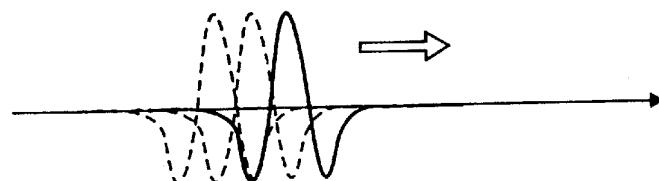


FIGURE 3.5 Translation (change in position) of wavelets.

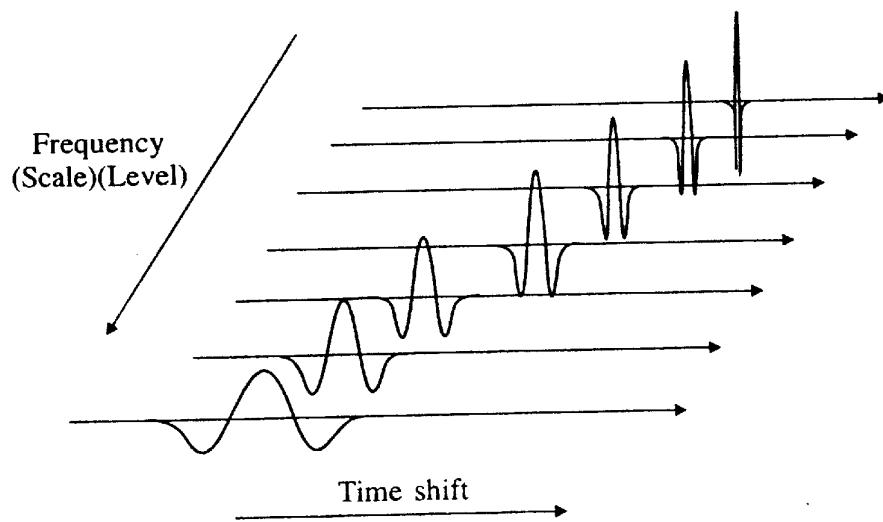


FIGURE 3.6 Change in scale (also called level) of wavelets.

Figure 3.7 shows a schematic of the wavelet transform which basically quantifies the local matching of the wavelet with the signal. If the wavelet matches the shape of the signal well at a specific scale and location, as it happens to do in the top plot of the Figure 3.7, then a large transform value is obtained. If, however, the wavelet and signal do not correlate well, a low value of transform is obtained. The transform value is then plotted in the two-dimensional transform plane shown at the bottom of the Figure 3.7. (indicated by a dot). The transform is computed at various locations of the signal and for various scales of the wavelet, thus filling up the transform plane. If the process is done in a smooth and continuous fashion (i.e., if scale and position is varied very smoothly) then the transform is called **continuous wavelet transform**. If the scale and position are changed in discrete steps, the transform is called **discrete wavelet transform**.

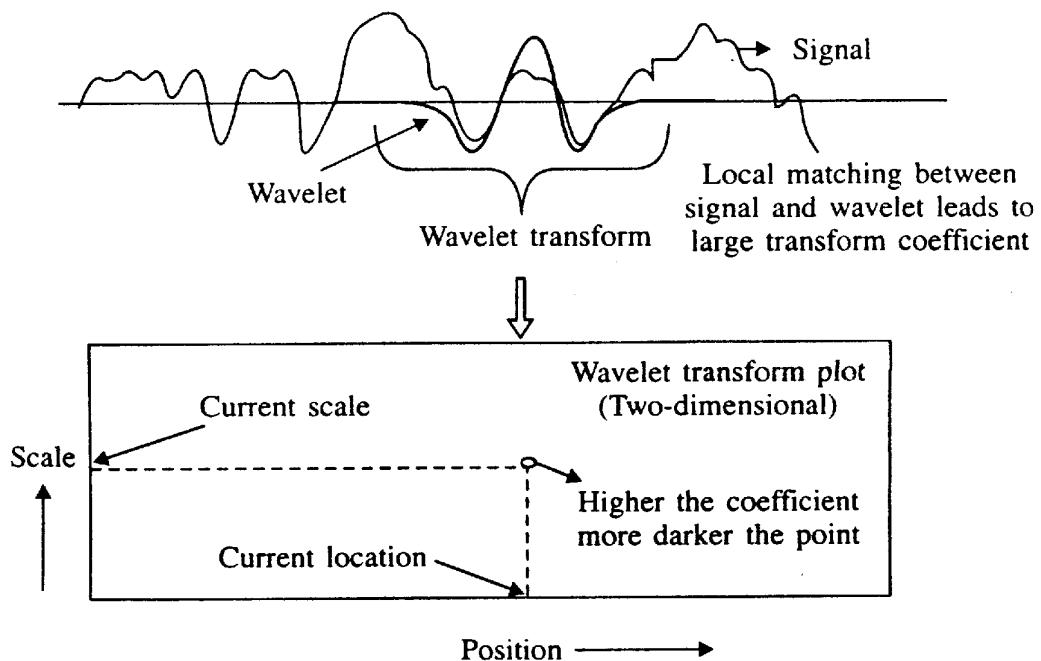


FIGURE 3.7 The signal, wavelet and transform.

Plotting the wavelet transform allows a picture to be built up of correlation between wavelet—at various scales and locations—and the signal. Figure 3.8 shows an example of a signal and corresponding spectrum. Commercial software packages use various colours and its gradations to show the transform values on the two-dimensional plot. Note that in case of Fourier transform, spectrum is one-dimensional array of values whereas in wavelet transform, we get a two-dimensional array of values. Also note that the spectrum depends on the type of wavelet used for analysis.

Mathematically, we denote a wavelet as:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right)$$

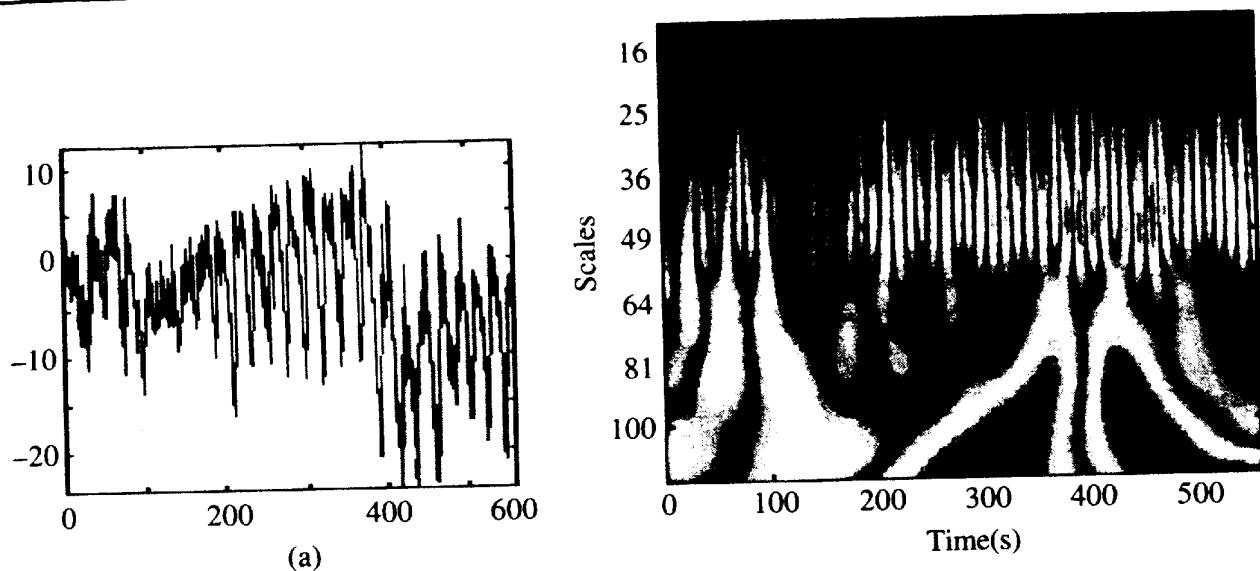


FIGURE 3.8 Signal and its wavelet transform spectrum.

where b is location parameter and a is scaling parameter. For the function to be a wavelet, it should be time limited. For a given scaling parameter a , we translate the wavelet by varying the parameter b .

We define wavelet transform as:

$$W(a, b) = \int_t f(t) \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right) dt \quad (3.1)$$

According to Eq. (3.1), for every (a, b) , we have a wavelet transform coefficient, representing how much the scaled wavelet is similar to the function at location $t = (b/a)$.

In the following section (Section 3.2) we will explore the classical time-frequency representation of signals, and associated uncertainties in frequency and time. Then we shall compare the same with that of wavelet transforms.

3.2 MATHEMATICAL PRELIMINARIES—FOURIER TRANSFORM

Continuous Fourier transform

Assume that f is a complex-valued function defined on \mathbb{R} . Its Fourier transform \hat{f} is defined as:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-it\omega} dt$$

If $f \in L^1$, the integral makes sense for every value of ω and \hat{f} is a continuous, bounded function which goes to zero at infinity (this last fact is called Riemann-Lebesgue Lemma).

The Fourier transform is also defined for $f \in L^2$. In this case, the integral may not be defined in the usual sense. One way to define it is:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \lim_{R \rightarrow \infty} \int_{-R}^R f(t) e^{-it\omega} dt$$

In this case \hat{f} is also in L^2 .

If $f \in L^1$ or $f \in L^2$ it can be written in terms of its Fourier transform as:

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{it\omega} d\omega$$

where the integral may need to be interpreted as a limit of finite integrals, as before. In general, every locally integrable function f has a Fourier transform but \hat{f} may not be a function any more, rather a *generalized function* or *distribution*. We will assume from now onwards that all functions have suitable integrability properties so that all Fourier transforms (and other operations that show up) are well defined. Some properties of the Fourier transform are:

- The Fourier transform preserves L^2 norms and inner products (this is called the **Parseval-Plancherel Theorem**). Thus, if $f, g \in L^2$ then

$$\langle f, g \rangle = \langle \hat{f}, \hat{g} \rangle$$

$$\|\hat{f}\|_2 = \|f\|_2$$

- The Fourier transform turns convolutions into products and vice versa. The *convolution* of f, g is defined as:

$$(f * g)(t) = \int f(y) g(t - y) dy = \int f(t - y) g(y) dy$$

We find

$$(f * g) \wedge (\omega) = \sqrt{2\pi} \hat{f}(\omega) \hat{g}(\omega)$$

$$(f \cdot g) \wedge (\omega) = \sqrt{2\pi} (\hat{f} * \hat{g})(\omega)$$

- The Fourier transform turns translation into modulation and vice versa. The *translation* of f by $a \in \mathbb{R}$ is defined as:

$$T_a f(t) = f(t - a)$$

The *modulation* of f by $a \in \mathbb{R}$ is defined as:

$$E_a f(t) = e^{iat} f(t)$$

Then

$$(T_a f) \wedge (\omega) = e^{-ia\omega} \hat{f}(\omega) = E_{-a} \hat{f}(\omega)$$

$$(E_a f) \wedge (\omega) = \hat{f}(\omega - a) = T_a \hat{f}(\omega)$$

- The Fourier transform turns dilation into inverse dilation. The *dilation* of f by $s \in \mathbb{R}$ is given by

$$D_s f(t) = |s|^{-1/2} f(t/s)$$

The factor in front is chosen so that $\|D_s f\|_2 = \|f\|_2$. The Fourier transform relationship is:

$$(D_s f) \wedge (\omega) = |s|^{1/2} \hat{f}(s\omega) = D_{1/s} \hat{f}(\omega)$$

- The Fourier transform turns differentiation into multiplication by $i\omega$ and vice versa.

$$(f') \wedge (\omega) = i\omega \hat{f}(\omega)$$

$$(tf(t)) \wedge (\omega) = i\hat{f}'(\omega)$$

3.2.1 Continuous Time-Frequency Representation of Signals

Assume that $f(t)$ is a complex valued function on \mathbb{R} which represents some signal (think of t as time).

The Fourier transform

$$\hat{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-it\omega} dt \quad (3.2)$$

is used to decompose f into its frequency components. The inversion formula is expressed as:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{it\omega} d\omega \quad (3.3)$$

Equation (3.3) can be interpreted as writing f as a superposition of time-harmonic waves $e^{it\omega}$. If \hat{f} is large near some frequency then f has a large component that is periodic with that frequency.

This approach works well for analyzing signals that are produced by some periodic process. However, in other applications, like speech analysis, we would like to localize the frequency components in time as well and Fourier transform is not suitable for that.

We will consider two methods that attempt to provide information on both time and frequency: the Windowed Fourier Transform (WFT), also called Short Time Fourier Transform (STFT) and the Continuous Wavelet Transform (CWT). Both of them map a function of one variable (time) into a function of two variables (time and frequency). A large

value of the transform near time t , and frequency ω is interpreted as: the signal f contains a large component with frequency ω near time t . There is a lot more theory to both of these transforms than we will cover in this chapter but our main interest lies elsewhere.

3.2.2 The Windowed Fourier Transform (Short Time Fourier Transform)

Fix a function $W \in L^2$, the window function. W should be localized in time near $t = 0$, with a spread σ (we will define “spread” more precisely later).

Typical choices are:

- (i) $W = \chi_{[-1,1]}$ with $\sigma = 1/\sqrt{3}$ (see Figure 3.9)

χ_s is the characteristic function of the set S , which has value 1 on the set, 0 otherwise.

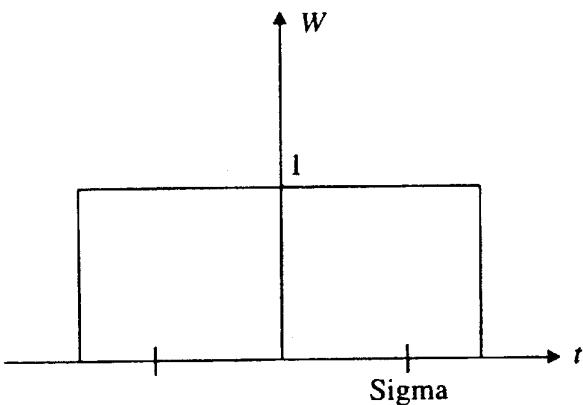


FIGURE 3.9 Characteristic function of $W = \chi_{[-1,1]}$.

- (ii) $W = (1 + \cos 2t)/2$ for $t \in [-\pi/2, \pi/2]$, $\sigma = 1/\sqrt{3}$ (see Figure 3.10)

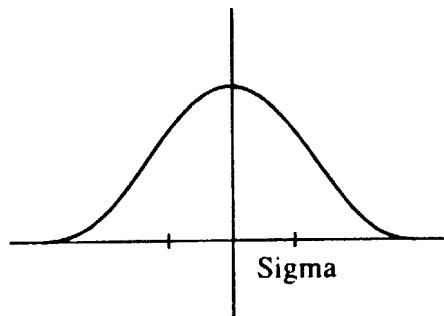


FIGURE 3.10 Plot of window function $W = (1 + \cos 2t)/2$.

- (iii) $W = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$ (Gabor window), $\sigma = 1$ (see Figure 3.11)

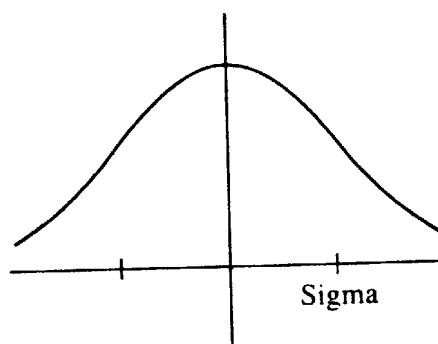


FIGURE 3.11 Plot of window function $W = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$.

The WFT with window W of f is defined as:

$$\psi_W f(a, b) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-ibt} \overline{W(t-a)} dt \quad (3.4)$$

Thus, a is the time parameter, b is the frequency.

$\psi_W f(a, b)$ can be interpreted as inner product of f with the test function $e^{ibt} W(t-a)$. A typical test function look like the one given in Figure 3.12.

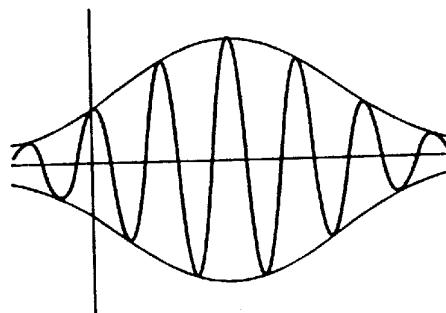


FIGURE 3.12 Shifted and modulated window.

Note how the time resolution σ (related to the window width) is constant, independent of the frequency.

3.2.3 The Uncertainty Principle and Time Frequency Tiling

Assume a function $f \in L^2$. The mean μ_f of f is defined as:

$$\mu_f = \frac{1}{\|f\|^2} \int_{-\infty}^{\infty} t |f(t)|^2 dt \quad (3.5)$$

The uncertainty σ_f of f is defined as:

$$\sigma_f = \frac{1}{\|f\|} \left(\int_{-\infty}^{\infty} (t - \mu_f)^2 |f(t)|^2 dt \right)^{1/2} \quad (3.6)$$

Remark: If $f \in L^2$ then $f^2/\|f\|^2$ is a probability distribution, i.e. a non-negative function with integral 1. μ_f and σ_f are simply the mean and standard deviation of this distribution in the statistical sense.

μ_f measures where f is localized in time and σ_f measures how spread out (or uncertain) this time measurement is. $\mu_{\hat{f}}$ localizes the frequency and $\sigma_{\hat{f}}$ measures the uncertainty in frequency.

Here

$$\mu_{\hat{f}} = \frac{1}{\|\hat{f}\|^2} \int_{-\infty}^{\infty} \omega |\hat{f}(\omega)|^2 d\omega \quad (3.7)$$

where

$$\hat{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

and

$$\sigma_{\hat{f}} = \frac{1}{\|\hat{f}(\omega)\|} \left(\int_{-\infty}^{\infty} (\omega - \mu_{\hat{f}})^2 |\hat{f}(\omega)|^2 d\omega \right)^{1/2} \quad (3.8)$$

The uncertainty principle states that for any f

$$\sigma_f \cdot \sigma_{\hat{f}} \geq \frac{1}{2} \quad (3.9)$$

If a function is localized in time, it must be spread out in the frequency domain and vice versa (see Figure 3.13). The optimal value of 1/2 is achieved if and only if f is a Gaussian distribution. To visualize this, consider a hypothetical function $F(t, \omega)$ over the time-frequency plane. $F(t, \omega)$ represents the component of f with frequency ω at time t .

The uncertainty principle says that it makes no sense to try to assign point-wise values to $F(t, \omega)$. All we can do is to assign meaning to averages of F over rectangles of area atleast 2. (The signal is localized in time to $[\mu - \sigma, \mu + \sigma]$ and likewise for frequency, so the rectangle has area $(2\sigma_f) \times (2\sigma_{\hat{f}})$).

The uncertainty principle applies to both WFT and CWT but in different ways. Let μ_w and σ_w represent ‘localization’ (mean) and ‘spread’ of the window function $W(t)$ in time domain which are formally defined as:

$$\mu_w = \frac{1}{\|W(t)\|^2} \int_{-\infty}^{\infty} t |W(t)|^2 dt \quad (3.10)$$

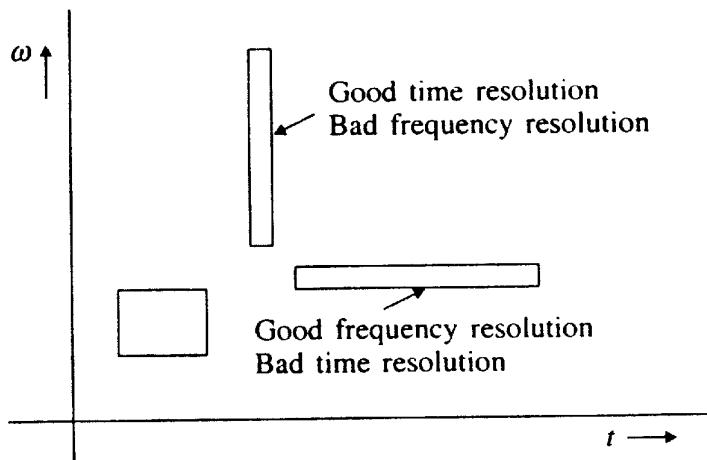


FIGURE 3.13 Possible time frequency windows.

$$\sigma_w = \frac{1}{\|W(t)\|} \left(\int_{-\infty}^{\infty} (t - \mu_w)^2 |W(t)|^2 dt \right)^{1/2} \quad (3.11)$$

Also let $\hat{\mu}_w$ and $\hat{\sigma}_w$ represent ‘localization’ (mean) and ‘spread’ of the function which is Fourier transform of window function $W(t)$. Then the inner product $\langle f, W \rangle$ contains information on f in $[\mu_w - \sigma_w, \mu_w + \sigma_w]$. Since $\langle f, W \rangle = \langle \hat{f}, \hat{W} \rangle$, it also contains information on $\hat{f}(\omega)$ in $[\hat{\mu}_w - \hat{\sigma}_w, \hat{\mu}_w + \hat{\sigma}_w]$. Thus, $\langle f, W \rangle$ represents the frequencies between $\hat{\mu}_w - \hat{\sigma}_w$ and $\hat{\mu}_w + \hat{\sigma}_w$ that are present between time $\mu_w - \sigma_w$ and $\mu_w + \sigma_w$.

For WFT, the test function is the shifted and modulated window.

$$e^{ibt} W(t - a) = E_b T_a W$$

which is localized in time near $\mu_w + a$ with uncertainty σ_w . Its Fourier transform is $T_b E_{-a} \hat{W}$ which is localized near $\hat{\mu}_w + b$ with uncertainty $\hat{\sigma}_w$.

Here E and T are respectively modulation and translation operators.

$$\psi_W f(a, b) = \frac{1}{\sqrt{2\pi}} \langle f, E_b T_a W \rangle = \frac{1}{\sqrt{2\pi}} \langle \hat{f}, T_b E_{-a} \hat{W} \rangle \quad (3.12)$$

contain information on f in $[\mu_w + a - \sigma_w, \mu_w + a + \sigma_w]$ and information on \hat{f} in $[\hat{\mu}_w + b - \hat{\sigma}_w, \hat{\mu}_w + b + \hat{\sigma}_w]$.

For a WFT with fixed window $W(t)$, the time resolution is fixed at σ_w , the frequency resolution is fixed at $\hat{\sigma}_w$. We can shift the window around in both time and frequency but the uncertainty box always has the same shape. The shape can only be changed by changing the window $W(t)$. Refer Figure 3.14.

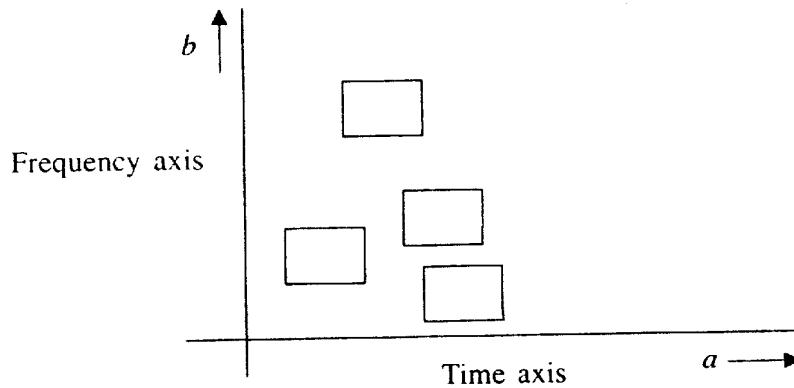


FIGURE 3.14 Some possible time frequency windows of windowed Fourier transform.

All time and frequencies are resolved equally well (or equally bad). A discrete WFT (with equally spaced time and frequency samples) gives a uniform tiling as in Figure 3.15.

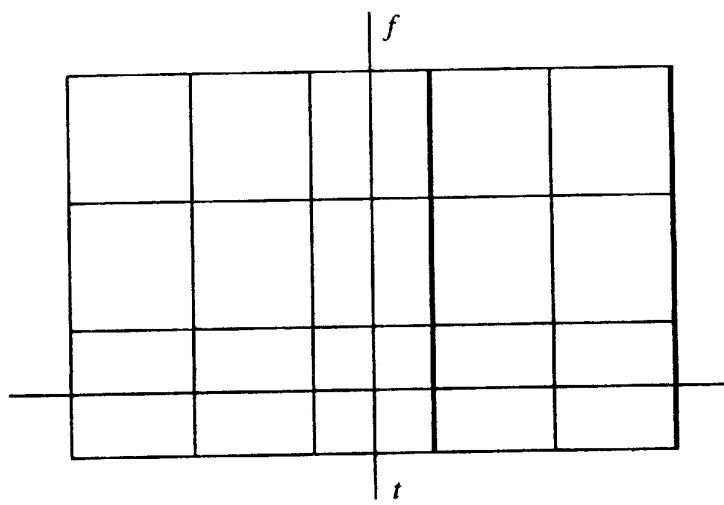


FIGURE 3.15 Time-frequency tiling of windowed Fourier transform.

For CWT, the test function is of the form:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right) = T_b D_a \psi \quad (3.13)$$

with Fourier transform $E_{-b} D_{1/a} \hat{\psi}$. Thus,

$$\langle f, \psi_{a,b} \rangle = \langle f, T_b D_a \psi \rangle = \langle \hat{f}, E_{-b} D_{1/a} \hat{\psi} \rangle \quad (3.14)$$

contains information on f in $[a\mu_\psi + b - a\sigma_\psi, a\mu_\psi + b + a\sigma_\psi]$ and information on \hat{f} in $[\hat{\mu}_\psi/a - \hat{\sigma}_\psi/a, \hat{\mu}_\psi/a + \hat{\sigma}_\psi/a]$. The uncertainty boxes have different shapes in different parts of the time-frequency plane as in Figure 3.16.

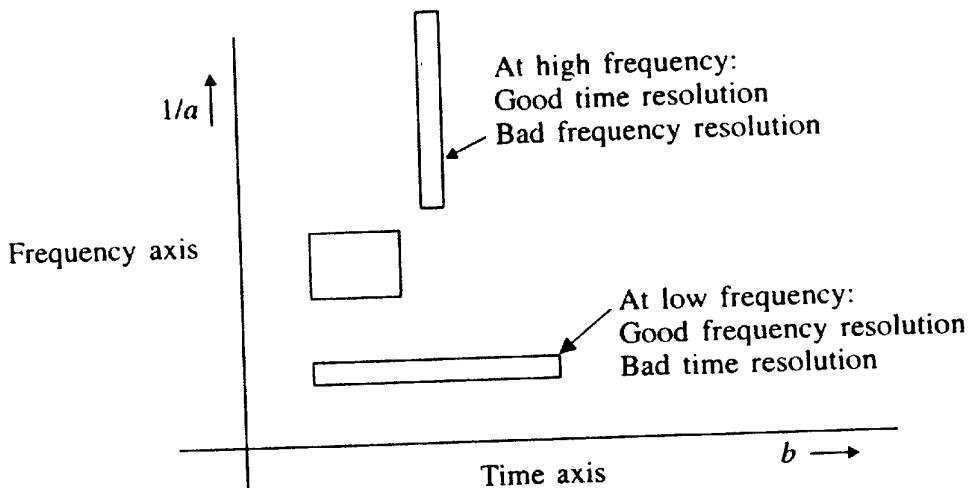


FIGURE 3.16 Possible time-frequency tilings in the case of continuous wavelet transform.

3.3 PROPERTIES OF WAVELETS USED IN CONTINUOUS WAVELET TRANSFORM

A wavelet $\psi(t)$ is simply a function of time t that obeys a basic rule, known as the wavelet admissibility condition:

$$C_\psi = \int_0^\infty \frac{|\hat{\psi}(\omega)|}{\omega} d\omega < \infty \quad (3.15)$$

where $\hat{\psi}(\omega)$ is the Fourier transform. This condition ensures that $\hat{\psi}(\omega)$ goes to zero quickly as $\omega \rightarrow 0$. In fact, to guarantee that $C_\psi < \infty$, we must impose $\hat{\psi}(0) = 0$ which is equivalent to

$$\int_{-\infty}^{\infty} \psi(t) dt = 0 \quad (3.16)$$

A secondary condition imposed on wavelet function is unit energy. That is

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt = 1 \quad (3.17)$$

3.4 CONTINUOUS VERSUS DISCRETE WAVELET TRANSFORM

As mentioned previously, CWT is a function of two parameters and, therefore, contains a high amount of extra(redundant) information when analyzing a function. Instead of continuously varying the parameters, we analyze the signal with a small number of scales with varying

number of translations at each scale. This is the discrete wavelet transform. Although discrete wavelet transform may be derived without referring to CWT, we may view it as a “discretization” of the CWT through sampling specific wavelet coefficients. A *critical sampling* of the CWT

$$W(a,b) = \int_a^b f(t) \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right) dt$$

is obtained via $a = 2^{-j}$, where j and k are integers representing the set of discrete translations and discrete dilations. Upon this substitution

$$\int_t^{\infty} f(t) \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right) dt$$

becomes

$$\int_{\mathbb{R}} f(t) 2^{j/2} \psi(2^j t - k) dt$$

which is a function of j and k . We denote it as $W(j, k)$.

A critical sampling defines the resolution of the DWT in both time and frequency. We use the term critical sampling to denote the minimum number of coefficients sampled from CWT to ensure that all the information present in the original function is retained by the wavelet coefficients. Figure 3.17 shows the time-frequency (scale) plane of the wavelet transform. In

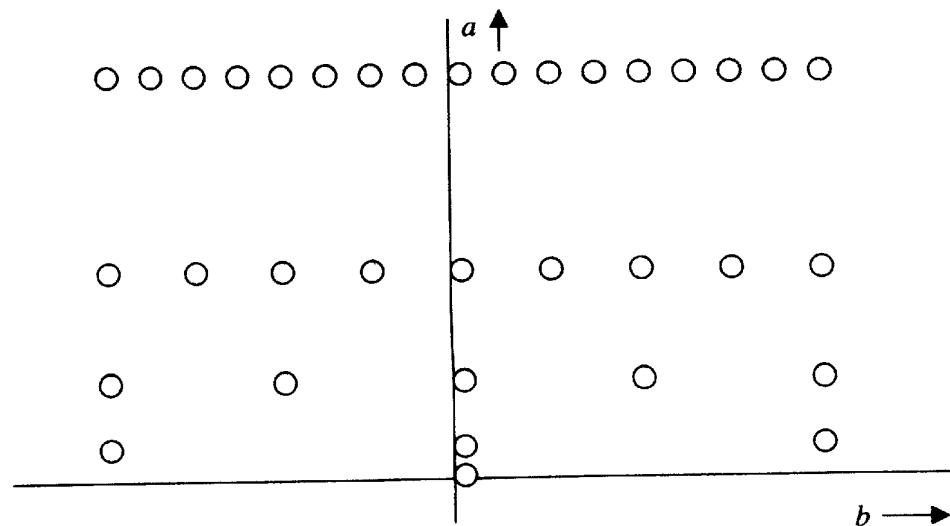


FIGURE 3.17 Critical sampling of the time-frequency (scale) plane by discretizing the CWT parameters via $a = 2^{-j}$ and $b = k2^{-j}$. The CWT is defined at all points in the plane and corresponds to a redundant representation of the information present in the function. As a increases, the number of coefficients sampled across time get doubled. By using only the points provided, the minimum number of wavelet coefficients are used so that the function may be perfectly reconstructed.

CWT, we find wavelet coefficients for every (a, b) combination whereas in discrete wavelet transform, we find wavelet coefficients only at very few points denoted by the dots and the wavelets that follow these values are given by

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k) \quad (3.18)$$

These wavelets for all integers j and k produce an orthogonal basis. We call $\psi_{0,0}(t) = \psi(t)$ as *mother wavelet*. Other wavelets are produced by translation and dilation of the mother wavelet. More details about the discrete wavelet transform will be dealt in forthcoming chapters.

The discrete wavelet transform results in a time-frequency tiling as shown in Figure 3.18.

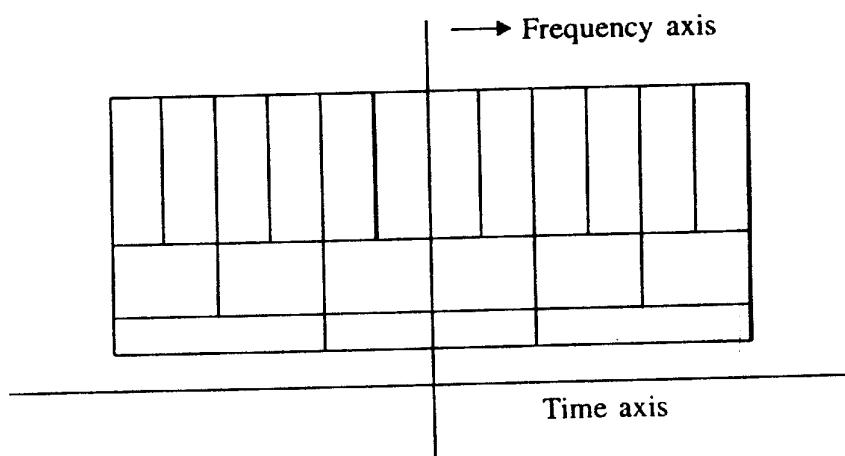


FIGURE 3.18 Time-frequency tilings in case of discrete wavelet transform.

In many applications (such as speech analysis), high frequencies are present very briefly at the onset of a sound while lower frequencies are present later for longer periods. To resolve all these frequencies well, the WFT has to be applied several times, with windows of varying widths. The CWT and DWT resolve all frequencies simultaneously, localized in time to a level proportional to their wavelength. That is what the wavelet people always claim.

SUMMARY

Fourier analysis is a global scheme. While we are able to analyze the frequencies that make up a signal (and do so quickly), the local properties of the signal cannot be easily detected from the Fourier coefficients. For example, if our signal was a recording of a drum being struck once and silence otherwise, we could look at the Fourier coefficients to analyze the frequencies, but without significant efforts, we would not be able to tell when the beat of the drum happened. It is this phenomenon that is known as **Heisenberg's Uncertainty Principle**. It says that a signal cannot be simultaneously localized in time and frequency. Wavelets, are an attempt to

overcome this shortcoming of Fourier analysis. They provide a way to do time-frequency analysis. The idea is that one chooses a “mother wavelet”, i.e. a function subject to some conditions like mean-value 0. Rather than using the pure harmonics as in Fourier analysis, one uses shifted and dilated versions of the mother wavelet. By using a 2-variable base (one for the amount of shift and one for the amount of dilation), you are able to introduce enough redundancy to maintain the local properties of the original function. While it may seem that such redundancy could only come at the cost of computational time, this is not entirely true. With the use of convolutions and filters, a fast discrete wavelet transform may be developed which enables wavelets to be used in many of the same places where Fourier analysis was once used. That is the topic of Chapter 4.

EXERCISES

- 3.1 Show that for $f(x)$ given by $f(x) = e^{-ax^2}$, $a \geq 0$ $\sigma_f^2 = \frac{1}{4a}$ and $\sigma_{\hat{f}}^2 = a$.

- 3.2 Find $\hat{f}(\omega)$ for

$$f(x) = \begin{cases} e^{-\alpha x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

- 3.3 Find $\hat{f}(\omega)$ for $f(t) = e^{-a|t|}$

[Hint: Show first that $\int_R^\infty e^{-\alpha|x|-i\omega x} dx = 2 \int_0^\infty e^{-\alpha x} \cos \omega x dx$]

- 3.4 Find $\hat{f}(\omega)$ for $f(t) = te^{-a|t|}$

[Hint: Show first that $\int_R^\infty xe^{-\alpha|x|-i\omega x} dx = -2i \int_0^\infty xe^{-\alpha x} \sin \omega x dx$]

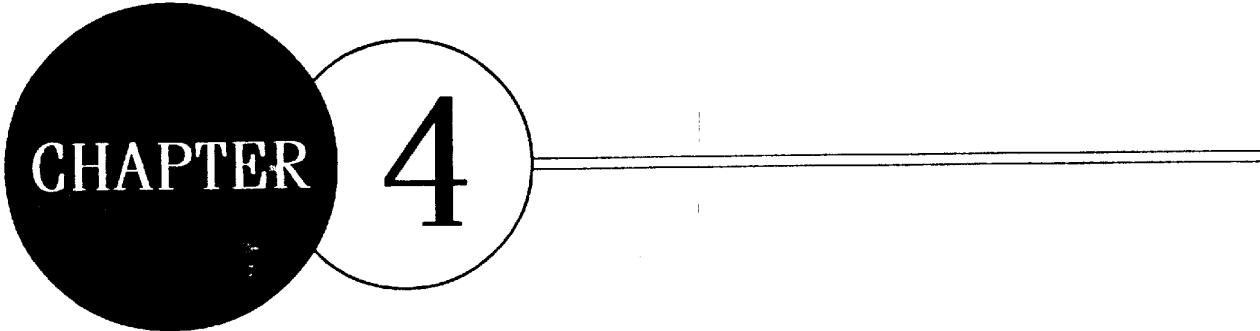
- 3.5 A graphical display of the magnitude of the STFT, $\psi_W f(a, b) =$

$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-ibt} \overline{W(t-a)} dt$, is called the **spectrogram** of the signal. Prove that the spectrogram is time and frequency shift invariant.

- 3.6 Compute the time-bandwidth product for the spectrogram of $x(t) = e^{j\omega_0 t}$, computed with the window function $W(t)$. The answer should be in terms of $W(t)$ and ω_0 .

REFERENCES

- [1] Cohen, L., *Time Frequency Analysis*, Prentice Hall, 1995.
- [2] Qian, S. and D. Chen, *Joint Time-frequency Analysis—Methods and Applications*, Prentice Hall, 1996.
- [3] Boashash, B., *Time-frequency Signal Analysis—Methods and Applications*, Longman-Cheshire, Melbourne, Australia, 1992.
- [4] Combes, J.M. (Ed.), *Wavelets, Time-frequency Methods and Phase Space*, Springer-Verlag, 1988.
- [5] Hlawatsch, Franz, *Time-frequency Analysis and Synthesis of Linear Signal Spaces*, Kluwer Academic Publishers, Boston, 1998.
- [6] Flandrin, P., *Time-frequency/Time-scale Analysis*, Academic Press, San Diego, California, 1999.
- [7] Strang, G. and T.Q. Nguyen, *Wavelets and Filter Banks*, Revised Edition, Wellesley-Cambridge Press, Wellesley, MA, 1998.
- [8] Vaidyanathan, P.P., *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [9] Vetterli, M. and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.



Discrete Wavelet Transform

INTRODUCTION

In the last chapter, we came across continuous wavelet transform and discrete wavelet transform was viewed as sampled version of the continuous parameter wavelet transform. However theory of discrete wavelet transform can be studied independent of its continuous counterpart and in this chapter we will develop the theory without any dependence to continuous wavelet transform.

Wavelets are a special kind of functions which exhibits oscillatory behaviour for a short period of time and then die out. Unlike Fourier series, in wavelets, we use a single function and its dilations and translations to generate a set of orthonormal basis functions to represent a signal. Number of such functions are infinite and we can choose one that suits to our application. Unfortunately, most of the wavelets used in discrete wavelet transform are fractal in nature. They are expressed in terms of a recurrence relation so that to see them we must do several iterations. For a newcomer in this field, it is the greatest hurdle. Fortunately we have two special functions called **Haar wavelet functions** and **scaling functions** which have explicit expression. To understand wavelet mathematics with geometrical insight, Haar wavelet system is our only hope. Scaling functions and wavelet functions are twins. Corresponding to a wavelet function, there is a scaling function. To understand this relationship, we must understand the concept of a set of *nested spaces* and their complementary spaces, which are discussed in forthcoming sections.

4.1 HAAR SCALING FUNCTIONS AND FUNCTION SPACES

In discrete wavelet transform, we have to deal with basically two sets of functions—scaling functions and wavelet functions. Understanding the relation between these two functions is the main aim of this chapter.

Consider Haar scaling function $\phi(t)$ defined in Eq. (4.1) and shown in Figure 4.1.

$$\phi(t) = \begin{cases} 1 & 0 \leq t \leq 1 \\ 0 & \text{elsewhere} \end{cases} \quad (4.1)$$

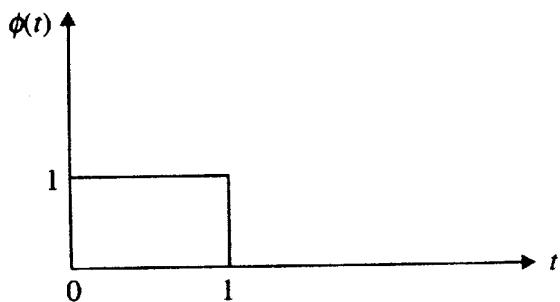


FIGURE 4.1(a) Haar scaling function.

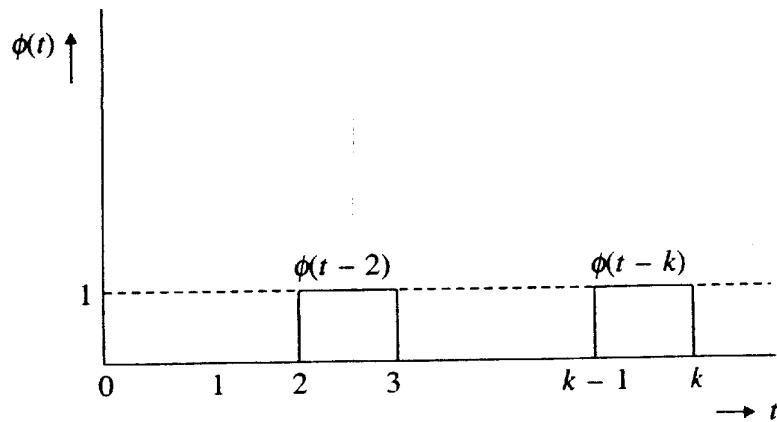


FIGURE 4.1(b) Translations of Haar scaling function $\phi(t)$.

The domain of the function is $[0, 1]$. Note that the function is time limited and have finite energy. That is, $\int_{-\infty}^{\infty} |f(t)|^2 dt$ exists and is finite. First we will make a set of functions that are translates of $\phi(t)$. Then we will show that, these functions are orthogonal and can form a basis set just like sine and cosines functions formed a basis set for Fourier series analysis.

4.1.1 Translation and Scaling of $\phi(t)$

Consider now functions of the type $\phi(t - 1)$, $\phi(t - 2)$, $\phi(t + 1)$ or in general $\phi(t - k)$. We call these functions as translates of $\phi(t)$. In function $\phi(t)$, the function exists practically for values of t in the range $[0, 1]$. Beyond this range, function value is zero. We say the domain of the function is $[0, 1]$. What about $\phi(t - 1)$? Where does it exist? To visualize this, consider $\phi(t - 1)$

again. According to Eq. (4.1), the quantity inside the bracket ‘()’ of ϕ may vary only in the closed interval $[0, 1]$ to get a non-zero value. Once the function is defined, this remain same irrespective of what you write inside the bracket. For example, once we define $\phi(t)$ as in Eq. (4.1), then in $\phi(t - 1)$, the quantity inside the bracket, that is, $t - 1$ may vary in the range $[0, 1]$ to have non-zero values for ϕ . Therefore t in $\phi(t - 1)$ when vary between 1 and 2, function value is 1 and for all other values of t , the function value is zero.

For economy of thought, if we want to find out where does $\phi(t - k)$ have non-zero value for the function given by Eq. (4.1), we set $(t - k) = 0$ to get $t = k$ and set $(t - k) = 1$ to get $t = k + 1$. So the function $\phi(t - k)$ has non-zero value for t in the range $[k, k + 1]$.

This argument can be utilized to find, say, where $\phi(10t)$ exists if $\phi(t)$ is as defined in Eq. (4.1). We set $10t = 0$, to obtain $t = 0$, and set $10t = 1$, to get $t = 1/10$. Therefore, the function $\phi(10t)$ has non-zero value in the interval $[0, 1/10]$. Thus, $\phi(10t)$ is a scaled version of $\phi(t)$.

What about $\phi(10t - k)$? We set $10t - k = 0$, to get $t = k/10$, and set $10t - k = 1$, to get $t = (k + 1)/10$. Therefore, the function $\phi(10t - k)$ has non-zero value in the interval $[k/10, (k + 1)/10]$. Thus, $\phi(10t - k)$ is a scaled and translated version of $\phi(t)$.

4.1.2 Orthogonality of Translates of $\phi(t)$

Consider $\int_{-\infty}^{\infty} \phi(t) \phi(t - 1) dt$. For Haar, $\phi(t) \phi(t - 1)$ is identically zero as shown in Figure 4.2.

Seeing the function, we are able to tell the location where the function exists. It can be easily seen that point-wise multiplication of these two functions results in zero for every value of t . Therefore, the integral is zero. Here it is quite obvious, since, there is no overlap between the two functions in the sense that wherever $\phi(t)$ has non-zero value $\phi(t - 1)$ is zero.

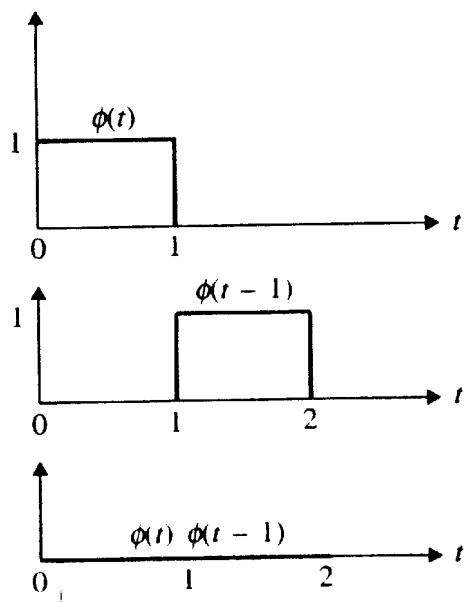


FIGURE 4.2 Orthogonality of translates of $\phi(t)$.

So, for Haar scaling function,

$$\int_{-\infty}^{\infty} \phi(t-m) \phi(t-n) dt = \delta_{m-n} \quad (4.2)$$

When m and n are equal, the functions are identical and perfectly overlap so that the area is equal to

$$\int_{-\infty}^{\infty} \phi(t-m) \phi(t-m) dt = \int_{-\infty}^{\infty} \phi(t) \phi(t) dt = \int_0^1 1^2 dt = 1$$

Equation (4.2) implies that the functions $\phi(t-m)$ and $\phi(t-n)$ are orthonormal.

4.1.3 Function Space V_0

We now have a set of orthonormal functions $\{\dots, \phi(t+1), \phi(t), \phi(t-1), \dots\}$, which are translates of a single function $\phi(t)$. These orthonormal functions can now represent certain type of signal. For example, consider Figure 4.3. There we have a sort of 'random staircase' type signal, which can be expressed using the bases we have just defined.

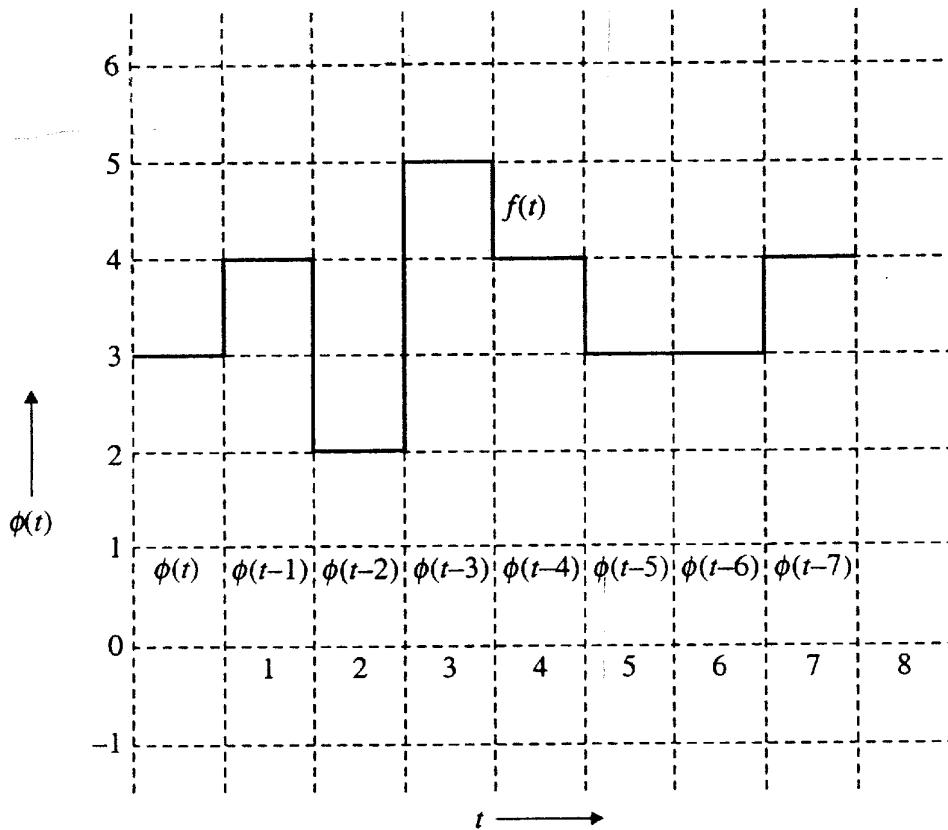


FIGURE 4.3 A representative signal in V_0 , spanned by Haar scaling functions.

Let V_0 be the space spanned by the set of bases $\{\dots, \phi(t+1), \phi(t), \phi(t-1), \dots\}$. We denote this as:

$$V_0 = \text{Span}_K \overline{\{\phi(t-k)\}} \quad (4.3)$$

Consider a function $f(t) = \sum_{k=-\infty}^{\infty} a_k \phi(t-k)$ where a_k s are real numbers (scalars) which we

call as coefficients of $\phi(t-k)$ s. For, one set of a_k s, we have one particular signal. But assume that we are continuously changing a_k s to generate continuously new functions or signals. The set of all such signals constitute the function space V_0 . Note that, all types of signals cannot be element of V_0 . For a signal $f(t)$ to be an element of V_0 , we must be able to express $f(t)$ using the bases of V_0 . What is the specialty of a signal in V_0 which is spanned by the Haar scaling functions? They are piecewise constant in the unit interval. Figure 4.3 shows such a signal.

Here $f(t)$ is given by

$$\begin{aligned} f(t) = & 3\phi(t) + 4\phi(t-1) + 2\phi(t-2) + 5\phi(t-3) + 4\phi(t-4) + 3\phi(t-5) \\ & + 3\phi(t-6) + 4\phi(t-7) \end{aligned} \quad (4.4)$$

Remember, the set of all such signals constitute our signal space V_0 . Can the signals given in Figure 4.4 belong to V_0 (spanned by Haar scaling function)?

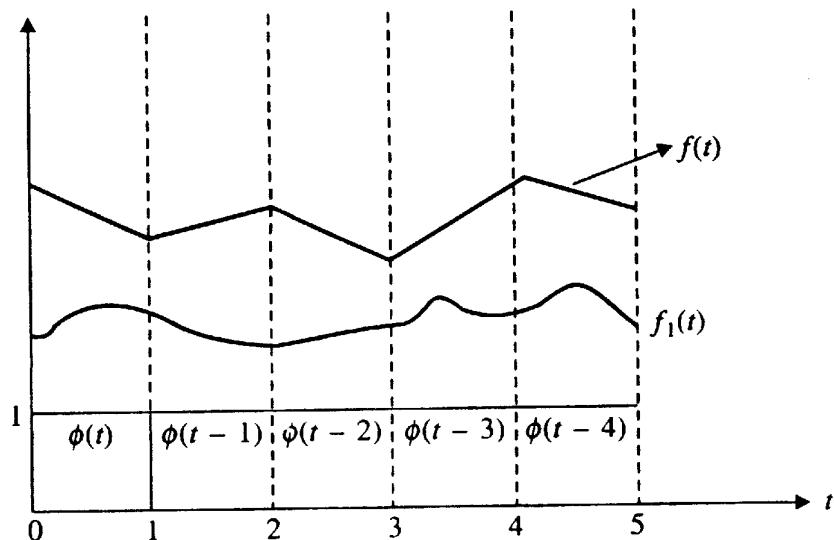


FIGURE 4.4 Signals which are not in V_0 (spanned by Haar scaling functions).

No. Both $f(t)$ and $f_1(t)$ cannot be represented by the bases of V_0 . The signal in V_0 space must be piecewise constant in each unit interval.

4.1.4 Finer Haar Scaling Functions

Let us now scale the Haar basis function and form a new basis set. We scale $\phi(t)$ by 2 and form functions of the type $\phi(2t+1)$, $\phi(2t)$, $\phi(2t-1)$, $\phi(2t-2)$ or in general $\phi(2t-k)$. These functions are again non-overlapping and are, therefore, orthogonal among themselves.

We call the space spanned by this set of function $\{\phi(2t-k), k \in N\}$ as V_1 . Figure 4.5 shows the new set of bases. Formally,

$$V_1 \equiv \text{Span}_K \overline{\{\phi(2t-k)\}}$$

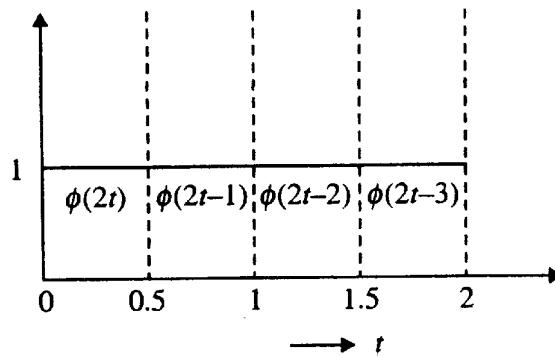


FIGURE 4.5 Haar scaling functions which form the basis for V_1 .

Any signal in such space can be written as:

$$f_1(t) = \sum_{k=-\infty}^{\infty} a_k \phi(2t-k) \quad (4.5)$$

By varying a_k s in Eq. (4.5), we can generate new functions and set of all such possible functions constitute the space V_1 . A signal in such a space is illustrated in Figure 4.6.

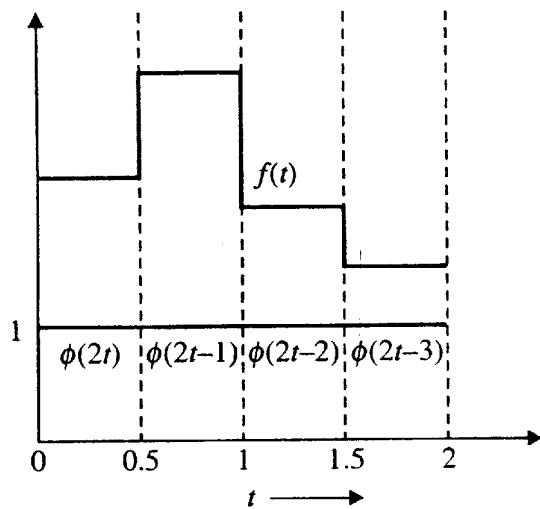


FIGURE 4.6 A signal which is element of space V_1 .

Similarly, V_2 is the space spanned by $\phi(2^2 t - k)$, that is,

$$V_2 \equiv \overline{\text{Span}_K \left\{ \phi(2^2 t - k) \right\}}$$

Generalizing, V_j is the space spanned by $\phi(2^j t - k)$.

$$V_j \equiv \overline{\text{Span}_K \left\{ \phi(2^j t - k) \right\}} \quad (4.6)$$

Let us now move on to the concept of nested spaces. This concept is one of the cornerstone of wavelet theory.

4.2 NESTED SPACES

Closely watch Figure 4.3 and Figure 4.5. We have a signal in V_0 space which basically means that the signal can be represented using the basis of V_0 . Our signal is given by Eq. (4.4), that is,

$$\begin{aligned} f(t) = & 3\phi(t) + 4\phi(t-1) + 2\phi(t-2) + 5\phi(t-3) + 4\phi(t-4) + 3\phi(t-5) \\ & + 3\phi(t-6) + 4\phi(t-7) \end{aligned}$$

The same signal can also be represented using the basis of V_1 , that is,

$$f(t) = 3\phi(2t) + 3\phi(2t-1) + 4\phi(2t-2) + 4\phi(2t-3) + 2\phi(2t-4) + 2\phi(2t-5) + \dots + \quad (4.7)$$

We have substituted $3\phi(t)$ by $3\phi(2t) + 3\phi(2t-1)$, $4\phi(t-1)$ by $4\phi(2t-2) + 4\phi(2t-3)$ and so on. This is true for any signal in space V_0 . Therefore we say that V_0 is contained in V_1 or $V_0 \subset V_1$. V_1 is a finer space than V_0 and contains all the signals in V_0 . V_1 is a bigger space than V_0 .

This is possible because bases of V_0 itself can be represented using bases of V_1 . For example,

$$\phi(t) = \phi(2t) + \phi(2t-1) \quad (4.8)$$

$$\phi(t-1) = \phi(2t-2) + \phi(2t-3)$$

The relation is called **scaling relation** or **refinement relation** or **dilation equation**.

Similarly, $V_1 \subset V_2$, i.e., any signal in V_1 can be represented using the basis of V_2 . Again this is possible because any basis of V_1 can be represented using the basis of V_2 , i.e.,

$$\phi(2t) = \phi(4t) + \phi(4t-1)$$

In general, we can write

$$\dots V_{-1} \subset V_0 \subset V_1 \subset \dots \subset V_\infty \quad (4.9)$$

Pictorially, we visualize this as exhibited in Figure 4.7.

Based on the relation $V_0 \subset V_1$, it is natural to ask, what is missing in V_0 that makes V_0 a subset of V_1 ? We pose another question to understand the meaning of above question.

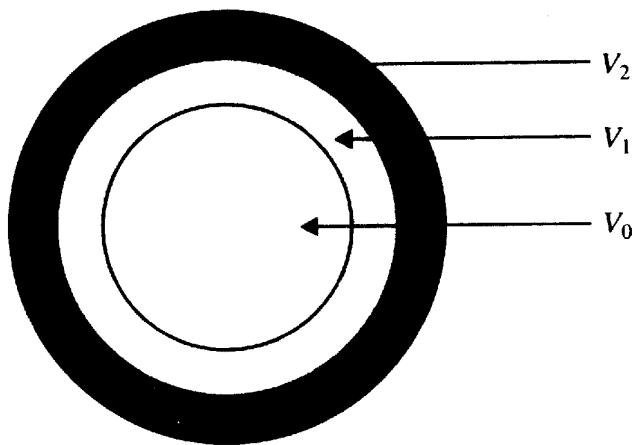


FIGURE 4.7 Nested spaces.

Consider the relation $\mathfrak{R}^2 \subset \mathfrak{R}^3$ where \mathfrak{R}^2 is the space spanned by the basis vectors \vec{i} and \vec{j} . \mathfrak{R}^3 is the space spanned by the basis vectors \vec{i} , \vec{j} and \vec{k} . We now ask, what is missing in \mathfrak{R}^2 that makes \mathfrak{R}^2 a subset of \mathfrak{R}^3 . Yes, you guessed it right. The basis vector \vec{k} . To answer the first question, let us explore another function called **Haar wavelet function**.

4.3 HAAR WAVELET FUNCTION

Haar wavelet function $\psi(t)$ (see Figure 4.8) is given by

$$\psi(t) = \begin{cases} 1 & 0 \leq t \leq 1/2 \\ -1 & 1/2 \leq t \leq 1 \\ 0 & \text{elsewhere} \end{cases} \quad (4.10)$$

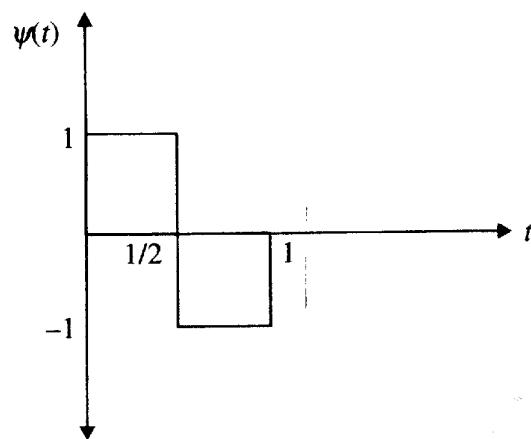


FIGURE 4.8 Haar wavelet function.

It is quite easy to see that the set of functions $\{\psi(t - k), k \in N\}$ form an orthonormal set of basis functions. Like Haar scaling function, there is no overlap between the translates of $\psi(t)$ (see Figure 4.9).

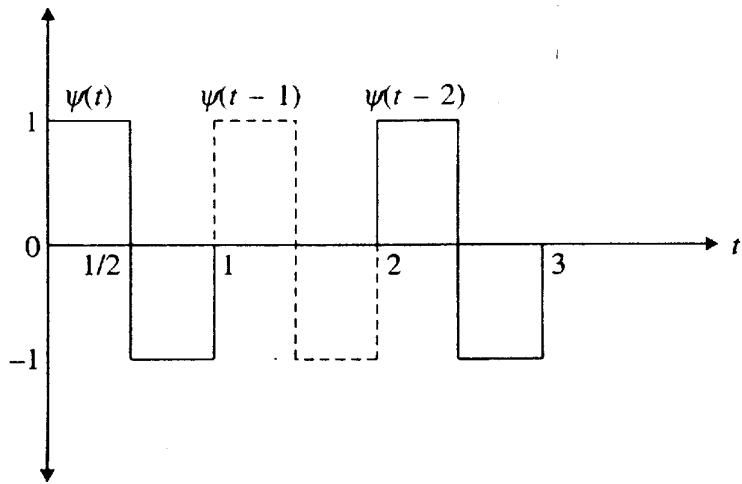


FIGURE 4.9 Haar wavelet function and its translates.

Therefore,

$$\int_{-\infty}^{\infty} \psi(t) \psi(t-1) dt = 0$$

In general,

$$\int_{-\infty}^{\infty} \psi(t-m) \psi(t-n) dt = \delta_{m-n} \quad (4.11)$$

That is, the integral is 1 if $m = n$ and 0 if $m \neq n$. Figure 4.10 shows that $\int_{-\infty}^{\infty} \psi(t) \psi(t) dt = 1$.

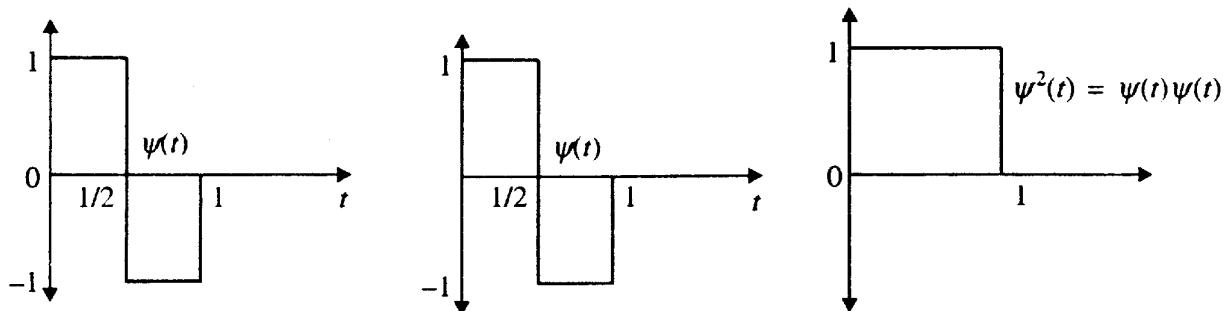


FIGURE 4.10 Finding norm of Haar wavelet function.

Let W_0 be the space spanned by the orthonormal set of bases $\{\psi(t - k), k \in N\}$.

Formally we define,

$$W_0 \equiv \text{Span}_k \left\{ \overline{\psi(t-k)} \right\} \quad (4.12)$$

How does a signal which is an element of W_0 look like? Figure 4.11 shows a signal in W_0 .

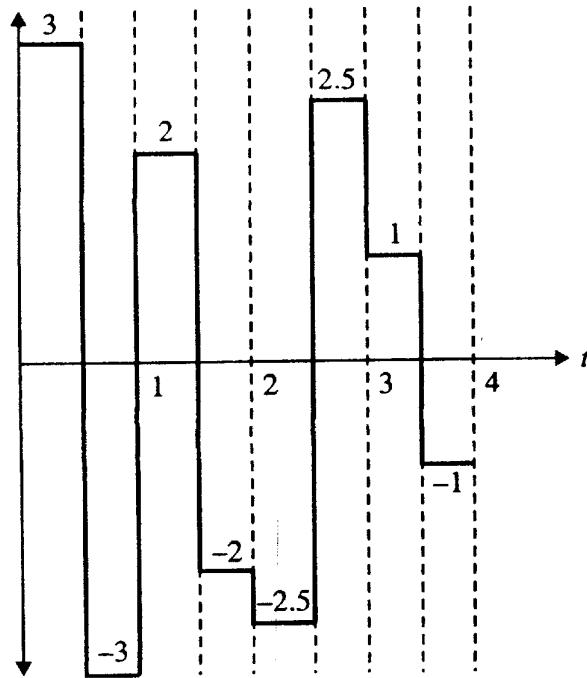


FIGURE 4.11 A signal in space W_0 .

What is the speciality of signal in W_0 (spanned by Haar wavelet functions)? In every unit interval, if the function value in first half is $\pm m$ units then in the next half it must necessarily be $\mp m$ units. Otherwise, we cannot express the signal using the basis of W_0 , that is, $\psi(t)$ and its translates. Thus, the space W_0 is a highly restricted space. The signal in Figure 4.11 is expressed using the bases of W_0 as:

$$f(t) = 3\psi(t) + 2\psi(t-1) - 2.5\psi(t-2) + \psi(t-3)$$

4.3.1 Scaled Haar Wavelet Functions

Consider a scaled version of $\psi(t)$. We will go, only for dyadic scaling. That is, scaling by the integer power of two. Consider at first $\psi(2t)$. Given $\psi(t)$ as defined in Eq. (4.10), $\psi(2t)$ can easily be shown to be

$$\psi(2t) = \begin{cases} 1 & 0 \leq t \leq 1/4 \\ -1 & 1/2 \leq t \leq 1/2 \\ 0 & \text{elsewhere} \end{cases}$$

We denote the space spanned by the set of orthogonal bases $\{\psi(2t - k), k \in N\}$ as W_1 . Formally we define,

$$W_1 \equiv \text{Span}_K \left\{ \overline{\psi(2t - k)} \right\}$$

Figure 4.12 shows some of the bases of W_1 .

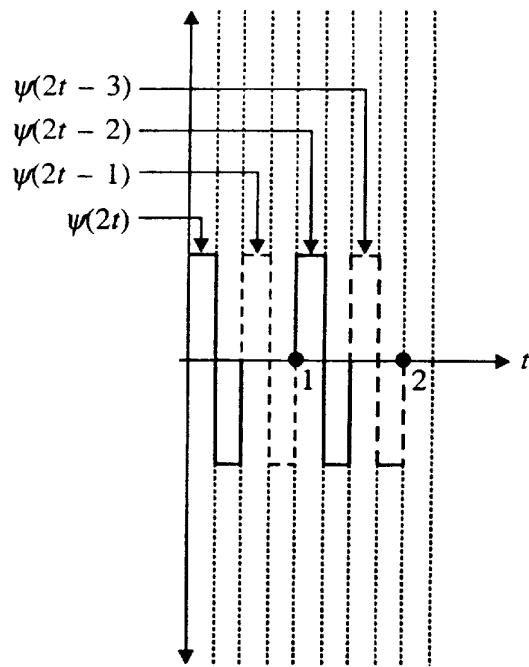


FIGURE 4.12 Some bases in W_1 .

Is $W_0 \subset W_1$? Or in other words, can we represent a signal in W_0 by using the bases of W_1 . The answer is ‘No’. See Figure 4.13.

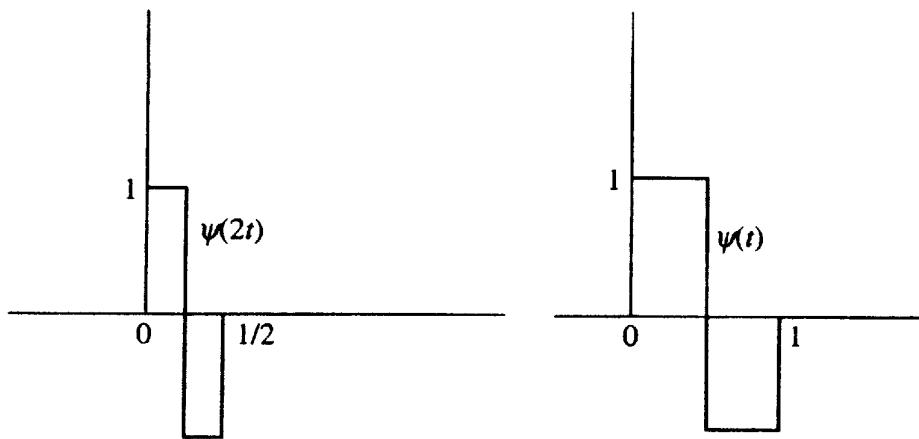


FIGURE 4.13 $\psi(2t)$ and its translates cannot express a $\psi(t)$.

Since linear combination of $\psi(2t)$ and its translates cannot make a $\psi(t)$, no signal in W_0 can be expressed using bases of W_1 . Therefore, $W_0 \not\subset W_1$. However we shall show that $W_0 \perp W_1$. This means that bases of W_0 are orthogonal to bases of W_1 . Figure 4.14 shows that $\int \psi(t) \psi(2t) dt = 0$ also $\int \psi(t) \psi(2t - 1) dt = 0$.

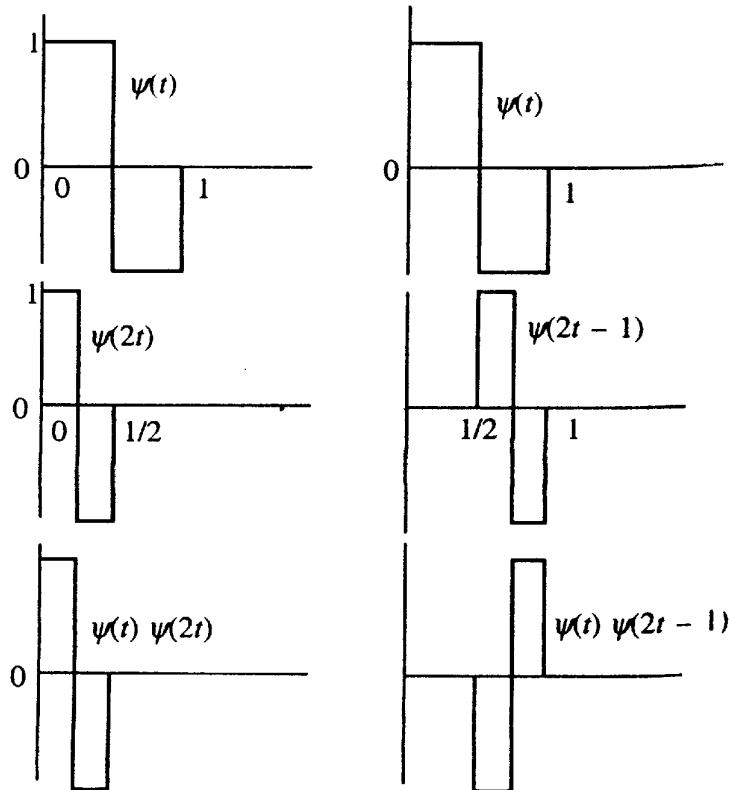


FIGURE 4.14 Product function $\psi(t) \psi(2t)$ and $\psi(t) \psi(2t - 1)$.

Note that the area under the function, which is the product of $\psi(t)$ and $\psi(2t)$, is zero. Similarly, the area under the function which is the product of $\psi(t)$ and $\psi(2t - 1)$ is also zero. This implies that space W_0 is orthogonal to the space W_1 . Again it can be proved that $W_1 \perp W_2$. Thus, we have another most important relation in wavelet theory.

$$\dots W_{-1} \perp W_0 \perp W_1 \perp W_2 \perp \dots \quad (4.13)$$

So far we have learned two very important concepts in wavelet theory with respect to function spaces. They are:

- Space spanned by scaling function bases are nested, i.e.,

$$\dots V_{-1} \subset V_0 \subset V_1 \subset \dots \subset V_\infty$$

- Space spanned by wavelet function bases are orthogonal among themselves. Thus,

$$\dots W_{-1} \perp W_0 \perp W_1 \perp W_2 \perp \dots$$

We are now ready to answer the question we have posed earlier. What is that missing in V_0 to make it a subset of V_1 . To answer this question we consider a signal in V_1 as shown in Figure 4.15.

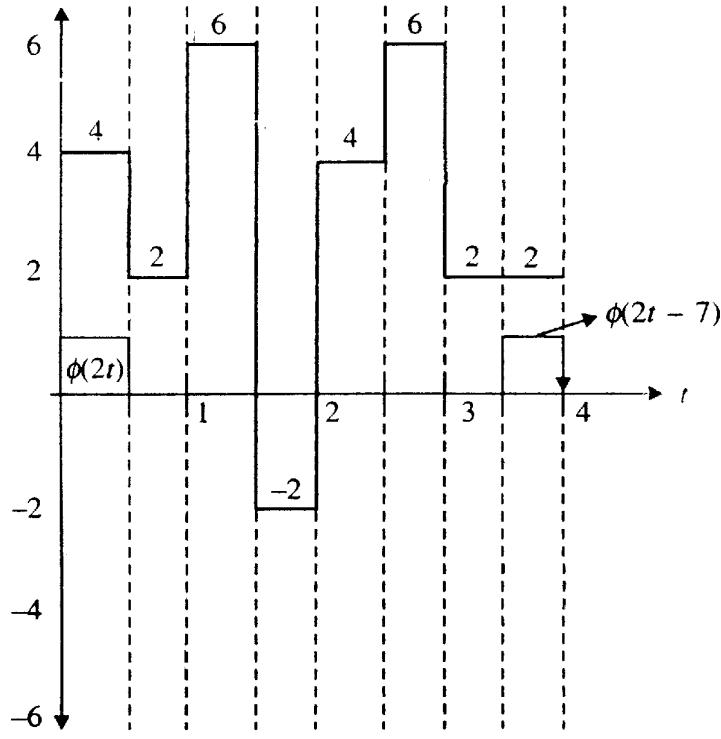


FIGURE 4.15 A signal in space V_1

Since it is a signal in V_1 , we can express it using the basis of V_1 . Thus,

$$\begin{aligned} f(t) = & 4\phi(2t) + 2\phi(2t-1) + 6\phi(2t-2) - 2\phi(2t-3) + 4\phi(2t-4) + 6\phi(2t-5) \\ & + 2\phi(2t-6) + 2\phi(2t-7) \end{aligned}$$

It may be observed that the signal segment in the first unit interval can be represented using the basis of V_0 and W_0 , i.e.,

$$4\phi(2t) + 2\phi(2t-1) = \frac{4+2}{2}\phi(t) + \frac{4-2}{2}\psi(t) = 3\phi(t) + \psi(t)$$

Pictorially this can be visualized as in Figure 4.16.

Similarly,

$$6\phi(2t-2) - 2\phi(2t-3) = \frac{6-2}{2}\phi(t-2) + \frac{6-2}{2}\psi(t) = 2\phi(t-1) + 4\psi(t-1)$$

Again

$$4\phi(2t-4) + 6\phi(2t-5) = \frac{4+6}{2}\phi(t-3) + \frac{4-6}{2}\psi(t-3) = 6\phi(t-2) - \psi(t-2)$$

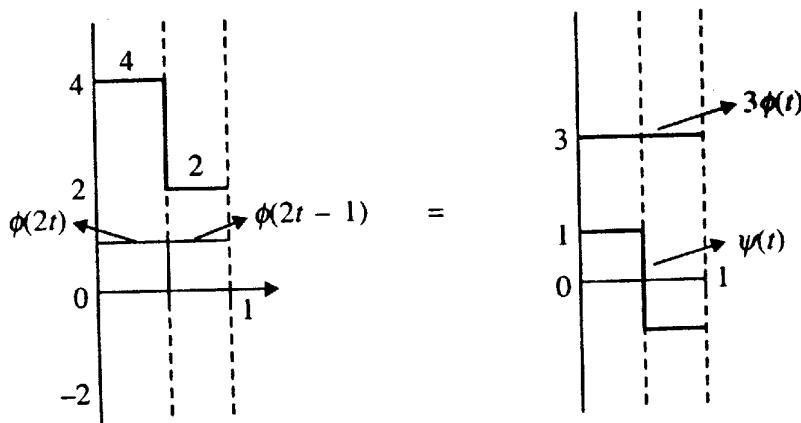


FIGURE 4.16 Decomposition of signal in V_1 into signal in V_0 and W_0 .

Lastly,

$$2\phi(2t-6) + 2\phi(2t-7) = \frac{2+2}{2} \phi(t-4) + \frac{2-2}{2} \psi(t-4) = 2\phi(t-3) + 0\psi(t-3)$$

Thus,

$$\begin{aligned} f(t) &= 4\phi(2t) + 2\phi(2t-1) + 6\phi(2t-2) - 2\phi(2t-3) + 4\phi(2t-4) + 6\phi(2t-5) \\ &\quad + 2\phi(2t-6) + 2\phi(2t-7) \end{aligned} \tag{4.14a}$$

$$= 3\phi(t) + 2\phi(t-1) + 6\phi(t-2) + 2\phi(t-3) + \psi(t) + 4\psi(t-1) - \psi(t-2) + 0\psi(t-3) \tag{4.14b}$$

We could express a signal in V_1 space, in terms of bases of V_0 space and W_0 space. If we combine the bases of V_0 and W_0 space, we can express any signal in V_1 space. We express this truth mathematically as:

$$V_1 = V_0 \oplus W_0$$

We call W_0 as the complementary space of V_0 . Now, we found out the ‘thing’ that was missing in V_0 , that made V_0 a proper subset of V_1 . We call V_0 and W_0 space as complementary space because V_0 and W_0 space are orthogonal and their bases together can represent any signal in the next ‘higher’ or finer space V_1 .

We call this as the decomposition of a finer signal into two coarser signals. Figure 4.17 shows the standard representation through diagram.

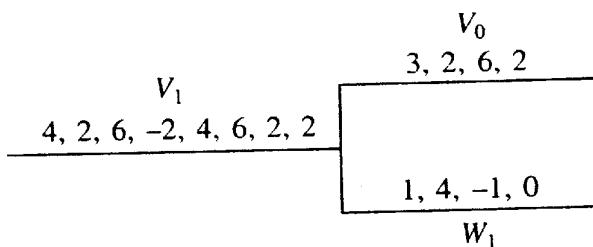


FIGURE 4.17 Visualization of Eqs. (4.14a) and (4.14b).

4.4 ORTHOGONALITY OF $\phi(t)$ AND $\psi(t)$

How shall we prove that $V_0 \perp W_0$? We must show that the bases of V_0 are orthogonal to bases of W_0 . We need to show the orthogonality only in the case where both the bases overlap (exist in the same interval). If the bases do not overlap, naturally they are orthogonal because the product of the two functions will identically be zero at every point. We know that $\phi(t)$ and $\psi(t)$ are defined in the same interval $[0, 1]$ and hence overlap. So we will try to prove that $\phi(t)$ is orthogonal to $\psi(t)$.

$$\langle \phi(t), \psi(t) \rangle = \int \phi(t) \psi(t) dt = \int_0^{0.5} 1 \cdot 1 dt + \int_{0.5}^1 1 \cdot (-1) dt = 0.5 - 0.5 = 0$$

Pictorially this is depicted in Figure 4.18. Product of $\phi(t)$ and $\psi(t)$ gives $\psi(t)$ whose net area is zero.

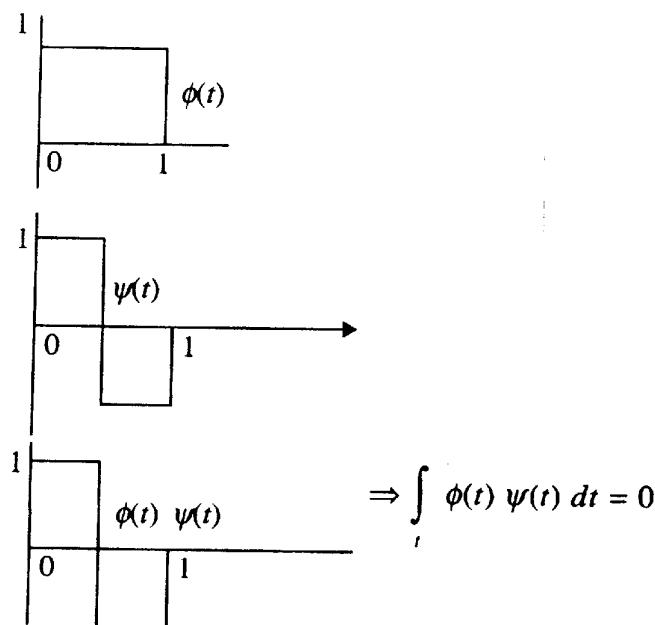


FIGURE 4.18 Orthogonality of $\phi(t)$ and $\psi(t)$.

Let us go back to our previous result. We have proved that the bases V_0 and W_0 together span V_1 . That is,

$$V_1 = V_0 \oplus W_0$$

Using the same argument, we may write

$$V_2 = V_1 \oplus W_1$$

In general, $V_j = V_{j-1} \oplus W_{j-1}$

Let us now express V_j using wavelet spaces alone. (Except the last term)

$$V_j = V_{j-1} \oplus W_{j-1}$$

But $V_{j-1} = V_{j-2} \oplus W_{j-2}$.

(4.15)

Therefore,

$$\begin{aligned} V_j &= W_{j-1} \oplus W_{j-2} \oplus V_{j-2} \\ &\vdots \quad \vdots \quad \vdots \quad \vdots \\ V_j &= W_{j-1} \oplus W_{j-2} \oplus W_{j-3} \oplus \dots \oplus W_0 \oplus V_0 \end{aligned} \quad (4.16)$$

Equation (4.16) tells that any signal in V_j can be expressed using the bases of W_{j-1} , W_{j-2} , ..., W_0 and V_0 . You will recognize later that Eq. (4.16) is the foundation of wavelet based decomposition of the signal. Schematically, this process is shown in Figure 4.19.

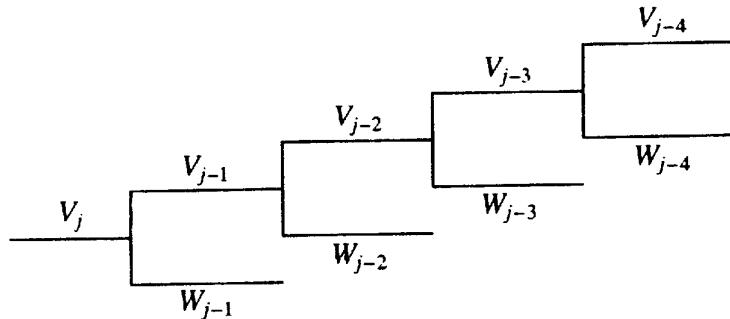


FIGURE 4.19 Schematic representation of wavelet decomposition of signals.

4.5 NORMALIZATION OF HAAR BASES AT DIFFERENT SCALES

Our basis vectors in \mathbb{R}^3 are orthonormal. In addition to orthogonality, they satisfy the relation $\vec{i} \cdot \vec{i} = \vec{j} \cdot \vec{j} = \vec{k} \cdot \vec{k} = 1$. Hence orthonormality. In function space, corresponding to dot product we have multiplication and integration.

Consider bases $\phi(t - k)$ s of V_0 . We know they are orthogonal. In addition, we have,

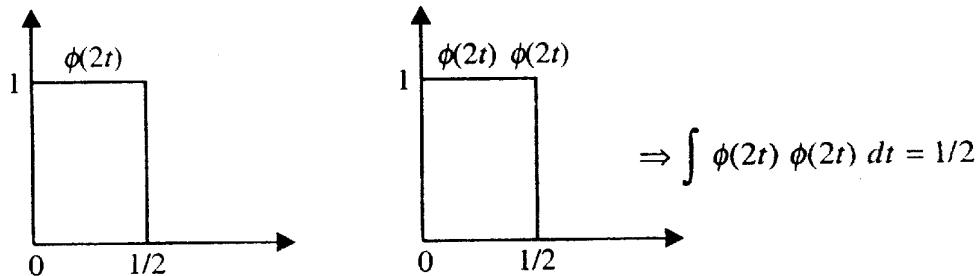
$$\int \phi(t)\phi(t) dt = 1$$

or

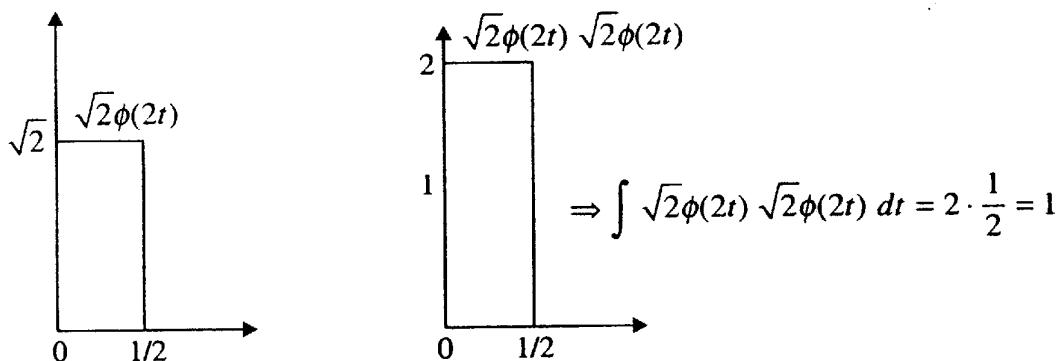
$$\int \phi(t - k) \phi(t - k) dt = 1$$

making the bases of V_0 orthonormal. We say, bases of Haar at scale level '0' are orthonormal. The actual value of scale is given by 2^j where j is the scale level.

What about orthonormality of bases of V_1 , where scale level is 1? The bases of space V_1 are translates of $\phi(2t)$ s. Let us check the value of integral $\int \phi(2t) \phi(2t) dt$. Its value is 1/2. Figure 4.20 depicts the process.

FIGURE 4.20 Finding the normalizing constant for $\phi(2t)$.

So to normalize $\phi(2t)$ and its translates, we multiply each by $1/\sqrt{2}$. Refer Figure 4.21 for visualization.

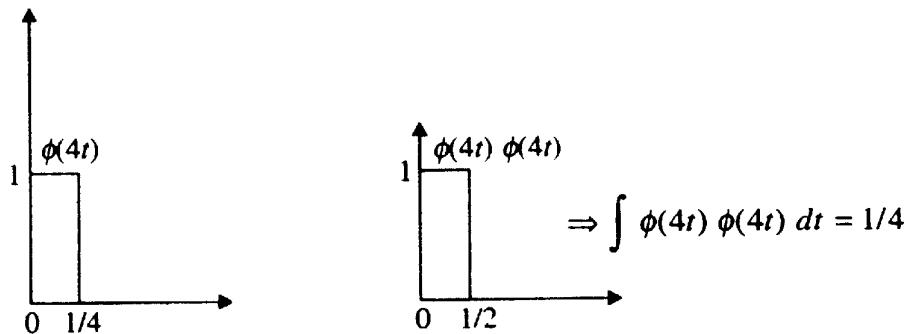
FIGURE 4.21 Normalized base for V_1 .

Now, we redefine V_1 as:

$$V_1 = \text{Span}_K \left\{ \sqrt{2}\phi(2t - k) \right\} \quad (4.17)$$

Consider bases for V_2 . Here the scale level is 2. Hence we call bases for V_2 as level 2 bases.

Since $\int \phi(4t) \phi(4t) dt = 1/4$, we should multiply each base by $\sqrt{4} = 2$. See Figure 4.22 for visualization.

FIGURE 4.22 Finding normalization constant for space V_2 .

We redefine V_2 as:

$$V_2 = \text{Span}_K \left\{ \overline{2\phi(4t - k)} \right\} = \text{Span}_K \left\{ \overline{2\phi(2^2 t - k)} \right\}$$

Let us now generalize. Consider un-normalized bases for V_j . The un-normalized bases for V_j are translates of $\phi(2^j t)$.

Since

$$\int \phi(2^j t) \phi(2^j t) dt = \frac{1}{2^j}$$

The normalizing constant is $\sqrt{2^j} = 2^{j/2}$. Therefore, V_j is redefined as:

$$V_j = \text{Span}_K \left\{ \overline{2^{j/2} \phi(2^j t - k)} \right\} \quad (4.18)$$

4.6 STANDARDIZING THE NOTATIONS

When we express a signal in terms of normalized bases, we have to repeat the normalizing constant in each term. To avoid this repetition, an abbreviated notation is introduced. We write $2^{j/2} \phi(2^j t - k)$ as $\phi_{j,k}(t)$. The first subscript denotes the scale level or simply level and the second subscript denotes translation in that scale level. Thus, the basis of V_0 are:

$$\{\dots \phi_{0,-1}(t), \phi_{0,0}(t), \phi_{0,1}(t), \dots, \phi_{0,k}(t), \dots\}$$

The basis of V_j are given by

$$\{\dots \phi_{j,-1}(t), \phi_{j,0}(t), \phi_{j,1}(t), \dots, \phi_{j,k}(t), \dots\}$$

Similarly, bases in W_j are normalized. The basis of W_j are expressed as:

$$\{\dots \psi_{j,-1}(t), \psi_{j,0}(t), \psi_{j,1}(t), \dots, \psi_{j,k}(t), \dots\}, \text{ where } \psi_{j,k}(t) \text{ are given by}$$

$$2^{j/2} \psi(2^j t - k)$$

4.7 REFINEMENT RELATION WITH RESPECT TO NORMALIZED BASES

With respect to un-normalized bases, we have the relation:

$$\phi(t) = \phi(2t) + \phi(2t - 1)$$

Similarly it is quite easy to see that

$$\psi(t) = \phi(2t) - \phi(2t - 1)$$

Both these relations follow from the fact that V_0 and W_0 are subset of V_1 . As a result of that, any base in V_0 and W_0 can be expressed using that bases of V_1 . With respect to normalized bases, this relation can be re-expressed as:

$$\phi(t) = \frac{1}{\sqrt{2}} \cdot \sqrt{2}\phi(2t) + \frac{1}{\sqrt{2}} \cdot \sqrt{2}\phi(2t-1) = \frac{1}{\sqrt{2}} \cdot \phi_{1,0}(t) + \frac{1}{\sqrt{2}} \cdot \phi_{1,1}(t) \quad (4.19)$$

$$\psi(t) = \frac{1}{\sqrt{2}} \cdot \sqrt{2}\phi(2t) - \frac{1}{\sqrt{2}} \cdot \sqrt{2}\phi(2t-1) = \frac{1}{\sqrt{2}} \cdot \phi_{1,0}(t) - \frac{1}{\sqrt{2}} \cdot \phi_{1,1}(t) \quad (4.20)$$

These two relations, which seems quite obvious with respect to Haar wavelet system, are the corner stone of wavelet theory. When we go on other wavelet systems, this relation may not be that obvious since most of other wavelet systems are fractal in nature. We cannot draw directly the functions for visualization.

Generally, we write the above relation as:

$$\phi(t) = \sum_k h(k) \phi_{1,k}(t) = \sum_k h(k) \cdot \sqrt{2}\phi(2t-k) \quad (4.21)$$

$$\psi(t) = \sum_k g(k) \phi_{1,k}(t) = \sum_k g(k) \cdot \sqrt{2}\phi(2t-k) \quad (4.22)$$

Specifically for Haar

$$\phi(t) = \sum_{k=0}^1 h(k) \phi_{1,k}(t) = \sum_{k=0}^1 h(k) \cdot \sqrt{2}\phi(2t-k)$$

$$\psi(t) = \sum_{k=0}^1 g(k) \phi_{1,k}(t) = \sum_{k=0}^1 g(k) \cdot \sqrt{2}\phi(2t-k)$$

$$\text{where } h(0) = \frac{1}{\sqrt{2}}, h(1) = \frac{1}{\sqrt{2}}, g(0) = \frac{1}{\sqrt{2}}, g(1) = -\frac{1}{\sqrt{2}}$$

Later we will interpret $\{h(k), k \in N\}$, $\{g(k), k \in N\}$ as low pass scaling filter and high pass wavelet filter coefficients. In conventional signal processing what really matters is the property of filters (frequency response, phase response, etc). We will study such properties of these filter coefficients in detail later.

Before proceeding further, we would like to introduce some other types of wavelet systems. By wavelet system we mean both the scaling function and wavelet function with their refinement relation defined through a known set of coefficients. To introduce other type of wavelet, at first we need to familiarize with another concept called **support** of the wavelet system.

4.8 SUPPORT OF A WAVELET SYSTEM

Support basically means the range of the interval over which the scaling function and wavelet function is defined. Beyond this interval, the functions should be identically zero.

Conventionally, we define the support for only the bases of V_0 and W_0 . In the case of Haar, we have defined $\phi(t)$ and $\psi(t)$ in the unit interval. Therefore, the support of $\phi(t)$ and $\psi(t)$ is $[0,1]$. There is an interesting relation between length of support and number of coefficients in the refinement relation. For orthogonal wavelet system, length of support is always one less than the number of coefficients in the refinement relation. This will become clear, as we proceed.

4.8.1 Triangle Scaling Function

This function is defined in $[0, 2]$. So the length of support is 2. It is defined as:

$$\phi(t) = \begin{cases} t & 0 \leq t \leq 1 \\ 2-t & 1 \leq t \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

It obeys a scaling relation given by

$$\phi(t) = \frac{1}{2}\phi(2t) + 1\phi(2t - 1) + \frac{1}{2}\phi(2t - 2)$$

Let us try to analyze the above relation. Since $\phi(t)$ is defined in $[0, 2]$, $\phi(2t)$ is defined over $[0, 1]$, $\phi(2t - 1)$ is defined over $[0.5, 1.5]$ and $\phi(2t - 2)$ over $[1, 2]$. Thus, three scaled and translated version of $\phi(t)$ cover the interval $[0, 2]$ and reproduce $\phi(t)$. Figure 4.23 shows the triangular scaling function and show how its translates of scaled version make up the original one.

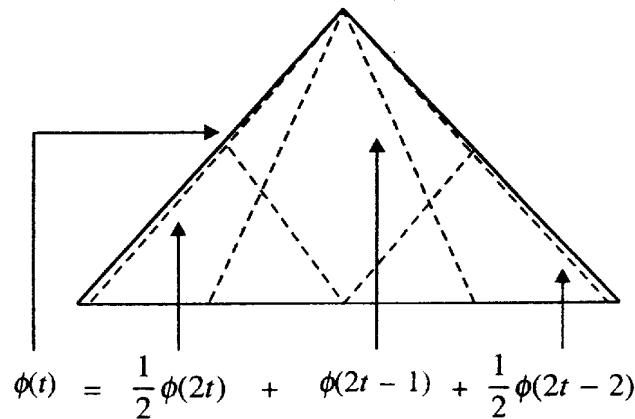
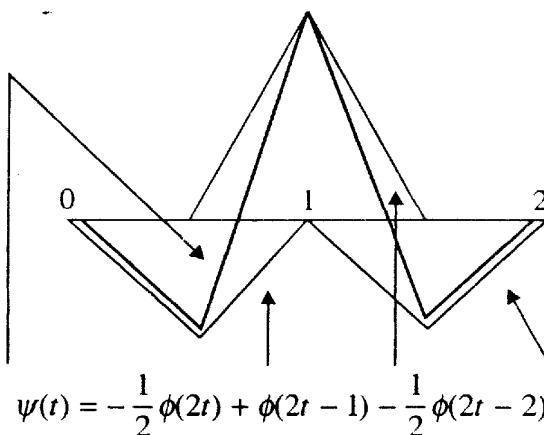


FIGURE 4.23 Triangular scaling function.

The associated wavelet function is shown in Figure 4.24.

In case of triangular scaling function, $\phi(2t)$ and its translates are not orthogonal among themselves. Therefore, it does not form an orthogonal wavelet system.

You may wonder how we are getting the coefficients $g(k)$ s for expressing $\psi(t)$ in terms of $\phi(2t)$ and its translates. There is a direct relation between $h(k)$ coefficients which appear in refinement relation (also called **dilation equation**) and $g(k)$ coefficients. The theoretical

**FIGURE 4.24** Triangular scaling function.

derivation of this relationship will be explained later. At present it is sufficient to know that, to get $g(k)$, we flip $h(k)$ and change the sign of alternate coefficients. Flipping means reversing the sequence of coefficients. In this example $\{1/2, 1, 1/2\}$ are the sequence in $h(k)$. Then $g(k)$ coefficients are either $\{-1/2, 1, -1/2\}$ or $\{1/2, -1, 1/2\}$.

4.9 DAUBECHIES WAVELETS

It is Daubechies, who gave solid foundation for wavelet theory. She developed many wavelet system with compact support (finite number of coefficients in the refinement relation). We will deal the design of Daubechies type wavelet system in a separate chapter. In this section, we will explore one wavelet system called **Daub-4 type wavelet system**. The peculiarity of this wavelet system is that, there is no explicit function, so we cannot draw it directly. What we are given is $h(k)$ s, the coefficients in refinement relation which connect $\phi(t)$ and translates of $\phi(2t)$. These coefficients for normalized Daub-4 are as follows:

$$h(0) = \frac{1}{4\sqrt{2}} (1 + \sqrt{3})$$

$$h(1) = \frac{1}{4\sqrt{2}} (3 + \sqrt{3})$$

$$h(2) = \frac{1}{4\sqrt{2}} (1 - \sqrt{3})$$

$$h(3) = \frac{1}{4\sqrt{2}} (3 - \sqrt{3})$$

Now,

$$\phi(t) = h(0)\sqrt{2}\phi(2t) + h(1)\sqrt{2}\phi(2t - 1) + h(2)\sqrt{2}\phi(2t - 2) + h(3)\sqrt{2}\phi(2t - 3) \quad (4.23)$$

where $\phi(t)$ is expressed in terms of $\phi(2t)$ and its translates.

How shall we see its shape then? We start with a ‘uniform’ $\phi(t)$, i.e. we assume $\phi(t)$ to be a square pulse of unit height and spanning the support of $\phi(t)$. Then we use the above refinement relation and build $\phi(t)$ iteratively. Section 4.10 describes the steps involved.

4.10 SEEING THE HIDDEN—PLOTTING THE DAUBECHIES WAVELETS

The problem here is that refinement relation tells us how bigger scaling function can be expressed using the smaller one. It does not tell us anyway how to draw the bigger one or for that matter how to draw the smaller one. Because of this, it has some parallel with fractals. Many fractals have this nature. The whole is made up of scaled and translated version of itself. For example, fern leaf as shown in Figure 4.25. This fern leaf has no explicit equation to draw itself. Here it is generated through an iterative scheme. The companion Website of this book (<http://www.amrita.edu/cen/publications/wavelets>) contains a spreadsheet implementation of this scheme. Open the spreadsheet ‘Fern.xls’ and press F9 to draw the picture in Figure 4.25.



FIGURE 4.25 Fern leaf.

To see Daubechies 4-tap wavelets, we will initially use a brute force approach. Note that $\phi(t)$ of Daub-4 is defined in the range $t = [0, 3]$ because number of coefficients in the refinement relation is 4. $\phi(2t)$ lies in the range $t = [0, 1.5]$. $\phi(2t - 1)$ lies in the range $t = [0.5, 2]$. Similarly, $\phi(2t - 2)$ in $[1, 2.5]$ and $\phi(2t - 3)$ in $[1.5, 3]$. Note again that four $\phi(2t)$ s span the range $[0, 3]$. We now sample $\phi(t)$. Sample size must be multiple of $2(N - 1)$ where N is the number of coefficients. This is to place $\phi(t)$ s properly in the given range. Let us take sample size as 180, i.e., we take 180 samples from $\phi(t)$. Perform the following steps:

- (i) Initialize arrays PHI20, PHI21, PHI22, PHI23 of size 180 with zeroes. Initialize array PHI of size 180 with ones. Initialize another array PHI2 of size 90 with zeroes.
- (ii) Take alternate values from PHI array and fill PHI2 array. (We are making $\phi(2t)$ from $\phi(t)$.)
- (iii) Take PHI2 array, multiply with $h(0)\sqrt{2}$ and store in PHI20 array from position 1 to 90.
- (iv) Take PHI2 array, multiply with $h(1)\sqrt{2}$ and store in PHI21 array from position 31 to 120.
- (v) Take PHI2 array, multiply with $h(2)\sqrt{2}$ and store in PHI22 array from position 61 to 150.
- (vi) Take PHI2 array, multiply with $h(3)\sqrt{2}$ and store in PHI23 array from position 91 to 120.
- (vii) Add PHI20, PHI21, PHI22, PHI23 array, point by point, and store in PHI array.
- (viii) Repeat steps from 2 to 7 several times (usually within 10 iterations PHI array will converge) and draw $\phi(t)$. Figure 4.26 shows the positioning of various array values.

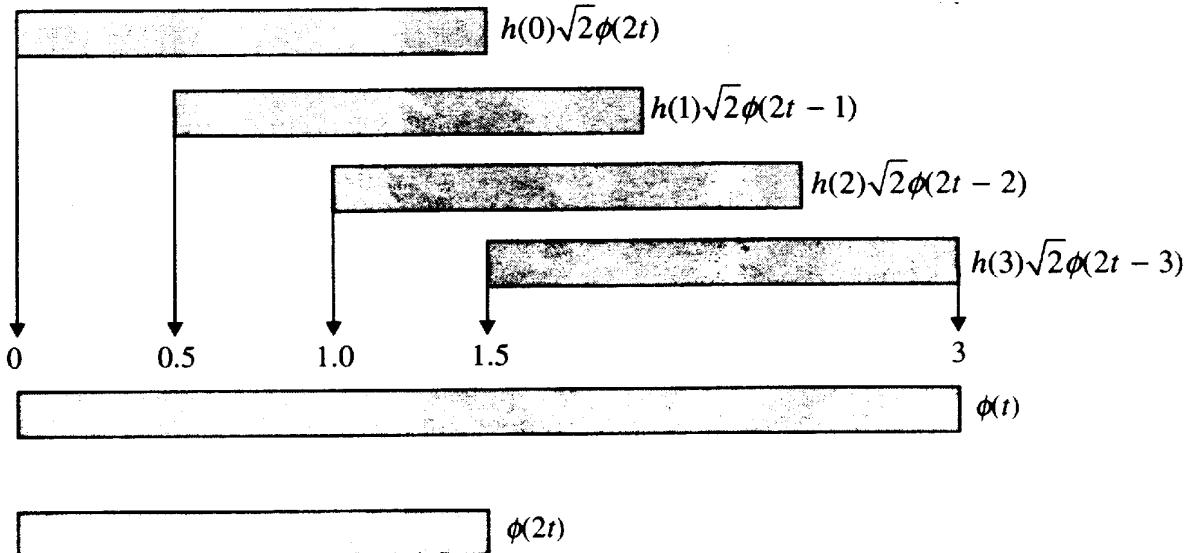


FIGURE 4.26 Visualizing the construction of $\phi(t)$.

- (ix) Use converged $\phi(t)$ values to draw $\psi(t)$ using the relation:

$$\psi(t) = g(0)\sqrt{2}\phi(2t) + g(1)\sqrt{2}\phi(2t - 1) + g(2)\sqrt{2}\phi(2t - 2) + g(3)\sqrt{2}\phi(2t - 3)$$

Figure 4.27 shows the output of the above procedure implemented in EXCEL spreadsheet.

The left figure is wavelet and the other is scaling function. The book's website contains the Excel sheet. Interested readers can download and use it.

Accurate methods of finding values of $\phi(t)$ and $\psi(t)$ is given in Chapter 7.

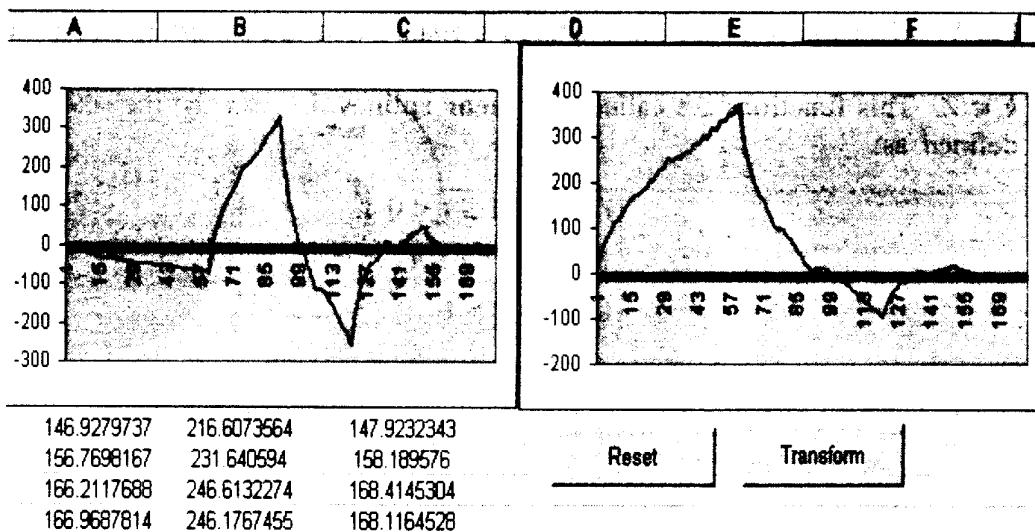


FIGURE 4.27 Seeing wavelets by iteration.

SUMMARY

Wavelet-based analysis of signals is an interesting, and relatively recent, new tool. Similar to Fourier series analysis, where sinusoids are chosen as the basis function, wavelet analysis is also based on a decomposition of a signal using an orthonormal (typically, although not necessarily) family of basis functions. Unlike a sine wave, a wavelet has its energy concentrated in time. Sinusoids are useful in analyzing periodic and time-invariant phenomena while wavelets are well suited for the analysis of transient, time-varying signals. Discrete wavelet transform theory requires two sets of related functions called **scaling function** and **wavelet function**. Basis sets of scaling functions span a nested space. Most of the scaling and wavelet functions are fractal in nature and iterative methods are required to see its shape.

EXERCISES

- 4.1 Let $x = [9 \ 5 \ 0 \ 1 \ -6 \ -5 \ 1 \ 4 \ 0 \ -5 \ 7 \ 9 \ 0 \ -1 \ 5 \ 3]$. Perform the Haar decomposition, using $(1/2, 1/2)$ and $(1/2, -1/2)$ as decomposition (analysis) filters,
- 4.2 Write a simple MATLAB program that will implement a two-stage analysis filter bank tree using the Haar filter.
 - (a) Apply it to a simple sinusoidal signal $x = \sin(\pi t/8)$. Generate a 128 point sampled version of this signal. Plot the outputs at each branch and show on a diagram of the filter bank tree where each is.
 - (b) Interpret your results, where is most of the signal?
 - (c) Show analytically why you get the result in part (a).

- 4.3 Let V_j be the space of all finite energy signals f that are continuous and piece-wise linear, with possible discontinuities occurring only at the dyadic points $k/2^j$, $k \in \mathbb{Z}$. These functions are called the **linear splines**. Let $\phi(x)$ be the scaling function defined as:

$$\phi(t) = \begin{cases} t+1 & -1 \leq t < 0 \\ 1-t & 0 < t \leq 1 \\ 0 & |t| > 1 \end{cases}$$

- (a) Find the scaling filter coefficients $h(n)$.
- (b) Construct the wavelet basis that is orthogonal to the scaling function. Plot the wavelet function.

REFERENCES

- [1] Daubechies, I., *Ten lectures on wavelets*, SIAM, Philadelphia, 1992.
- [2] Strang, G. and T.Q. Nguyen, *Wavelets and Filter Banks*, Revised Edition, Wellesley-Cambridge Press, Wellesley, MA, 1998.
- [3] Kaiser, G., *A Friendly Guide to Wavelets*, Birkhauser, San Diego, CA, 1994.
- [4] Mallat, S., *A Wavelet Tour of Signal Processing*, Second Edition, Academic Press, 1999.
- [5] Rao, R.M. and A.S. Bopardikar, *Wavelet Transforms: Introduction to Theory and Applications*, Addison-Wesley, MA, 1998.
- [6] Suter, B.W., *Multirate and Wavelet Signal Processing*, Academic Press, Boston, 1998.
- [7] Chui, C.K., *An Introduction to Wavelets*, Academic Press, Boston, 1992.

CHAPTER

5

Designing Orthogonal Wavelet Systems—A Direct Approach

INTRODUCTION

In the last chapter we used Daubechies' 4-tap wavelet system coefficients. We have two sets of coefficients $\{h(k)\}$ and $\{g(k)\}$ that define the refinement relation. These coefficients decide shape of the scaling function and wavelet function. This shape in turn decide the application where we can use the particular wavelet. The coefficients $\{h(k)\}$ and $\{g(k)\}$ act as signal filters, which we are going to discuss in due course of time. Vast amount of literature is available on filter design for specific application but it is inaccessible to an average reader since practically all methods uses frequency domain as well as complex analysis concepts to arrive at the filter. In this section we will find methods to arrive at the filter coefficients derived by Daubechies. As mentioned before Daubechies used ideas from complex analysis and Fourier transform to derive the coefficients. We, instead, go for a time domain approach for deriving the coefficients.

5.1 REFINEMENT RELATION FOR ORTHOGONAL WAVELET SYSTEMS

We have the relation

$$\phi(t) = \sum_{k=0}^{N-1} h(k) \sqrt{2} \phi(2t - k) = \sum_{k=0}^{N-1} c_k \phi(2t - k) \quad (5.1)$$

where $c_k = h(k) \cdot \sqrt{2}$.

We call c_k as un-normalized coefficients and $h(k)$ as normalized coefficients.

And

$$\psi(t) = \sum_{k=0}^{N-1} g(k) \sqrt{2} \phi(2t - k) = \sum_{k=0}^{N-1} d_k \phi(2t + k) \quad (5.2)$$

where $c'_k = g(k) \cdot \sqrt{2}$.

Wavelets must satisfy certain necessary conditions like orthogonality and certain desirable properties for specific kind of applications. These conditions, in turn, put restrictions on scaling and wavelet function coefficients. We will see one by one.

5.2 RESTRICTIONS ON FILTER COEFFICIENTS

5.2.1 Condition 1: Unit Area Under Scaling Function

It is clear that the scaling relation given by Eq. (5.1) determines ϕ only up to a multiplicative constant. It is necessary to impose certain conditions on ϕ in order to uniquely determine filter coefficients. Firstly, we require ϕ to be normalized. That is we set $\int \phi(t) dt = 1$. (In Haar scaling function, you might notice that it is true.) The scaling relation then impose a condition on the filter coefficients.

$$\begin{aligned} \int_{-\infty}^{\infty} \phi(t) dt &= \int_{-\infty}^{\infty} \sum_{k=0}^{N-1} c_k \phi(2t - k) dt \\ &= \sum_{k=0}^{N-1} c_k \int \phi(2t - k) dt \\ &= \frac{1}{2} \sum_{k=0}^{N-1} c_k \int \phi(y) dy \end{aligned}$$

Since

$$\int \phi(t) dt = \int \phi(y) dy$$

We obtain

$$\sum_{k=0}^{N-1} c_k = 2 \quad (5.3)$$

Utilizing the relation $c_k = h(k) \cdot \sqrt{2}$, we get the relation in terms of normalized coefficient as:

$$\sum_{k=0}^{N-1} h(k) = \frac{2}{\sqrt{2}} = \sqrt{2} \quad (5.4)$$

For Daubechies' 4-tap scaling function, this means

$$h(0) + h(1) + h(2) + h(3) = \sqrt{2}$$

Let us have a check for Haar.

We have

$$h(0) = h(1) = \frac{1}{\sqrt{2}}$$

Therefore,

$$h(0) + h(1) = \sqrt{2}$$

5.2.2 Condition 2: Orthonormality of Translates of Scaling Functions

The integer translates of scaling function must be orthonormal. This requires

$$\int \phi(t) \phi(t - k) dt = \delta_{0,k}$$

i.e.

$$\int \phi(t) \phi(t - k) dt = 1 \quad \text{if } k = 0$$

and

$$\int \phi(t) \phi(t - k) dt = 0 \quad \text{if } k \neq 0$$

Now,

$$\begin{aligned} \int \phi(t) \phi(t - k) dt &= \sum_{l=0}^{N-1} c_l \sum_{m=0}^{N-1} c_m \int \phi(2t - l) \cdot \phi(2t - 2k - m) dt \\ &= \frac{1}{2} \sum_{l=0}^{N-1} c_l \sum_{m=0}^{N-1} c_m \int \phi(y) \cdot \phi(y + l - 2k - m) dy \\ &= \frac{1}{2} \sum_{l=0}^{N-1} c_l \sum_{m=0}^{N-1} c_m \delta_{l,2k+m} \\ &= \frac{1}{2} \sum_{l=0}^{N-1} c_l c_{l-2k} \end{aligned} \tag{5.5}$$

If $k = 0$, we have the square normalization condition:

$$\frac{1}{2} \sum_{l=0}^{N-1} c_l^2 = 1 \quad \text{or} \quad \sum_{l=0}^{N-1} c_l^2 = 2 \tag{5.6}$$

This is equivalent to

$$\sum_{l=0}^{N-1} h^2(l) = \sqrt{2} \quad (5.7)$$

For $k \neq 0$, we have the orthonormality condition:

$$\sum_{l=0}^{N-1} c_l c_{l-2k} = 0 \quad \text{for all } k \neq 0 \quad (5.8)$$

We call condition given in Eq. (5.8) as double shift orthogonality conditions.

Let us see how this equation translates to a wavelet system with four coefficients, i.e., $N = 4$.

Let $k = 1$ (k is basically the integer representing translation in the relation $\int \phi(t)\phi(t-k) dt = \delta_{0,k}$)

Then the equation $\sum_{l=0}^{N-1} c_l c_{l-2k} = 0$ reduces to $\sum_{l=0}^3 c_l c_{l-2} = 0$ giving the following relation:
(remember that for $N = 4$, only c_0, c_1, c_2 and c_3 coefficients exist)

$$c_2 c_0 + c_3 c_1 = 0$$

Verify that for $N = 4$ and $k \geq 2$, no further relation between coefficients can be established.
Let us now try for $N = 6$. Start with fixing the value of k to 1.

$$\sum_{l=0}^5 c_l c_{l-2} = 0 \Rightarrow c_2 c_0 + c_3 c_1 + c_4 c_2 + c_5 c_3 = 0$$

For $k = 2$,

$$\sum_{l=0}^5 c_l c_{l-4} = 0 \Rightarrow c_4 c_0 + c_5 c_1 = 0$$

5.2.3 Condition 3: Orthonormality of Scaling and Wavelet Functions

We have a refinement relation that connects $\psi(t)$ and $\phi(2t - k)$ s. This relation defines coefficients $\{g(k)\}$. Now let $c'_k = \sqrt{2}g(k)$. Thus c'_k are basically unnormalized coefficients that define the relation between $\psi(t)$ and $\phi(2t - k)$, that is,

$$\psi(t) = \sum_{k=0}^{N-1} g(k) \sqrt{2} \phi(2t - k) = \sum_{k=0}^{N-1} c'_k \phi(2t - k)$$

We now establish the relation between c_k and c'_k . We have seen earlier that $\phi(t)$ and $\psi(t)$ are orthogonal. This implies $\int \phi(t) \psi(t) dt = 0$. It can be shown that, for this to be true $\psi(t)$ must take the following form: (The derivation of this is deferred to a later chapter.)

$$\psi(t) = \sum_{k=0}^{N-1} (-1)^k c_k \phi(2t + k - N + 1) \equiv \sum_k (-1)^{N-k-1} c_{N-k-1} \phi(2t - k) \quad (5.9)$$

This relation implies

$$c'_k = (-1)^{N-k-1} c_{N-k-1} \quad (5.10)$$

It will also be proved that

$$c'_k = (-1)^k c_{N-k-1} \quad (5.11)$$

is also a valid solution.

For $N = 4$, this means

$$c'_0 = (-1)^{4-0-1} c_{4-0-1} = -c_3$$

$$c'_1 = c_2$$

$$c'_2 = -c_1$$

and

$$c'_3 = -c_0$$

i.e., to get c'_k coefficients, simply reverse the coefficients c_k and change the sign of the coefficients in alternate positions.

5.2.4 Condition 4: Approximation Conditions (Smoothness Conditions)

In many applications, we need to approximate a signal using scaling function. So we ask the question: To what degree p , can monomials $1, t, t^2, \dots, t^p$ be reproduced exactly using a basis of scaling function? Or to what degree p the following is possible?

$$t^p = \sum_k d_k^p \phi(t - k) \quad (5.12)$$

Orthonormality of basis functions imply that

$$d_k^p = \int t^p \phi(t - k) dt \quad \{\text{project } t^p \text{ on to the basis } \phi(t - k)\} \quad (5.13)$$

To achieve this, scaling function should possess certain properties (suitable shape). But its shape depends on the coefficients c_k (or $h(k)$).

Now, we assume that it is possible to represent exactly monomials of order up to p using a given scaling function. This will impose a certain conditions on c_k . What are these conditions?

In this section, we show that it is quite easy to translate the condition on scaling functions on to conditions on wavelet function. This in turn allow us to find conditions on coefficients c_k .

We have

$$t^p = \sum_k d_k^p \phi(t - k)$$

Let us now try to project t^p on to $\psi(t)$.

$$\int t^p \psi(t) dt = \int \sum_k d_k^p \phi(t - k) \psi(t) dt = \sum_k d_k^p \int \phi(t - k) \psi(t) dt = 0$$

Since $\phi(t)$ and $\psi(t)$ are orthogonal, this means, if $\phi(t)$ is capable of expressing monomial of order up to p then corresponding wavelet function must have its moments of order up to p as zero. Note here that, in literature, the integral $\int_{-\infty}^{+\infty} t^p f(t) dt$ is called **p th moment of function $f(t)$** .

Let $p = 0$. This condition implies that

$$\int \psi(t) dt = 0$$

Using Eqs. (5.2) and (5.11), we obtain

$$\begin{aligned} & \int \left[\sum_k (-1)^k c_k \phi(2t + k - N + 1) \right] dt = 0 \\ & 0 = \sum_k (-1)^k c_k \int \phi(2t + k - N + 1) dt \\ & 0 = \frac{1}{2} \sum_k (-1)^k c_k \int \phi(y) dy \end{aligned}$$

Since $\int \phi(t) dt = 1$

$$\therefore \frac{1}{2} \sum_k (-1)^k c_k \int \phi(y) dy = \sum_k (-1)^k c_k = 0 \quad (5.14)$$

For a 4-tap filter (corresponding to a wavelet system of support 4), this reduces to

$$c_0 - c_1 + c_2 - c_3 = 0$$

For $p = 1$, we have

$$\int t \psi(t) dt = 0 \Rightarrow \sum_k (-1)^k k c_k = 0 \quad (5.15)$$

For a general $p > 0$, we have

$$\int t^p \psi(t) dt = 0 \Rightarrow \sum_k (-1)^k k^p c_k = 0 \quad (5.16)$$

In this manner, we can put conditions on scaling function coefficients, and then solve for the value of c_k s. The number of conditions that we can put depends on the number of coefficients, N , in the refinement relation. If $N = 6$, we say we have 6 degrees of freedom. Out of these 6, three (50%) degree of freedom will go for constraints on orthogonality. Remaining degree of freedom can be used at our will. What Daubechies did was that, the remaining degree of freedom was fully utilized on constraints on smoothness. We are now ready to obtain Daubechies wavelet coefficients.

5.3 DESIGNING DAUBECHIES ORTHOGONAL WAVELET SYSTEM COEFFICIENTS

Let us assume that $N = 4$.

<i>Constraint type</i>	<i>Constraints on coefficients</i>
Normalization	$c_0 + c_1 + c_2 + c_3 = 2$
Square normalization	$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 2$
Double-shift orthogonality	$c_0c_2 + c_1c_3 = 0$
Moment condition, $p = 0$	$c_0 - c_1 + c_2 - c_3 = 0$
Moment condition, $p = 1$	$0c_0 - 1c_1 + 2c_2 - 3c_3 = 0$

We have written five constraints. However it can be easily shown that the square normalization condition can be obtained from the normalization, orthonormality and $p = 0$ condition. So effectively there are only four constraints. This can be easily solved using MATLAB, Mathematica or even Excel spreadsheet solver. There are two solutions, the same up to a permutation but it is conventional to take the second of these:

$$c_0 = \frac{1}{4}(1 + \sqrt{3})$$

$$c_1 = \frac{1}{4}(3 + \sqrt{3})$$

$$c_2 = \frac{1}{4}(3 - \sqrt{3})$$

$$c_3 = \frac{1}{4}(1 - \sqrt{3})$$

Recall that $h(k) = c_k/\sqrt{2}$. Then h coefficients are given by

$$h(0) = \frac{1}{4\sqrt{2}}(1 + \sqrt{3})$$

$$h(1) = \frac{1}{4\sqrt{2}}(3 + \sqrt{3})$$

$$h(2) = \frac{1}{4\sqrt{2}}(1 - \sqrt{3})$$

$$h(3) = \frac{1}{4\sqrt{2}}(3 - \sqrt{3})$$

Once we obtain coefficients $\{h(k)\}$, we can find $\{g(k)\}$. Just reverse the coefficients and change the sign at the alternate positions. Therefore,

$$g(0) = \frac{1}{4\sqrt{2}}(3 - \sqrt{3})$$

$$g(1) = -\frac{1}{4\sqrt{2}}(1 - \sqrt{3})$$

$$g(2) = \frac{1}{4\sqrt{2}}(3 + \sqrt{3})$$

$$g(3) = -\frac{1}{4\sqrt{2}}(1 + \sqrt{3})$$

5.3.1 Constraints for Daubechies' 6 Tap Scaling Function

Let us suppose that $N = 6$.

<i>Constraint type</i>	<i>Constraints on coefficients</i>
Normalization	$c_0 + c_1 + c_2 + c_3 + c_4 + c_5 = 2$
Square normalization	$c_0^2 + c_1^2 + c_2^2 + c_3^2 + c_4^2 + c_5^2 = 2$
Double-shift orthogonality	$c_0c_2 + c_1c_3 + c_2c_4 + c_3c_5 = 0$ $c_0c_4 + c_1c_5 = 0$
Moment condition, $p = 0$	$c_0 - c_1 + c_2 - c_3 + c_4 - c_5 = 0$
Moment condition, $p = 1$	$0c_0 - 1c_1 + 2c_2 - 3c_3 + 4c_4 - 5c_5 = 0$
Moment condition, $p = 2$	$0c_0 - 1c_1 + 4c_2 - 9c_3 + 16c_4 - 25c_5 = 0$

After solving, we obtain $\{h(k)\}$ as follows:

$$h(0) = 0.3326705529500825$$

$$h(1) = 0.8068915093110924$$

$$h(2) = 0.4598775021184914$$

$$h(3) = -0.1350110200102546$$

$$h(4) = -0.0854412738820267$$

$$h(5) = 0.0352262918857095$$

Figure 5.1 shows corresponding scaling and wavelet function.

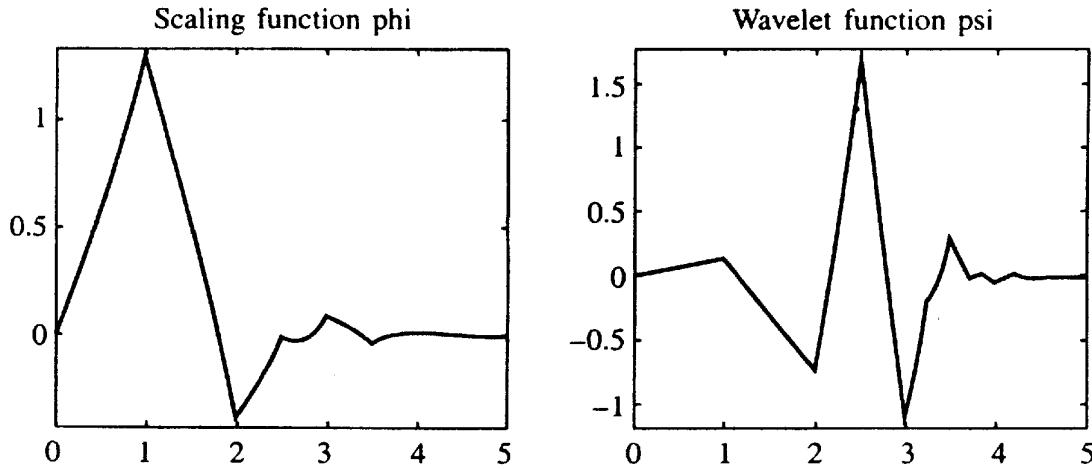


FIGURE 5.1 Daubechies scaling and wavelet function with 6 taps.

Filter coefficients with $N = 8$ are:

$$[0.230377813309, 0.714846570553, 0.630880767930, -0.027983769417, \\ -0.187034811719, 0.030841381836, 0.032883011667, -0.010597401785]$$

5.4 DESIGN OF COIFLET WAVELETS

Coiflet wavelets are obtained by imposing vanishing moment conditions on both scaling and wavelet functions. More conditions on vanishing moments imply more filter coefficients. The minimum number of taps is 6. If the number of taps $N = 6p$ then $2p$ number of vanishing moment conditions are imposed on wavelet function, $2p - 1$ on scaling function and the remaining on normality and orthogonality conditions. Thus, the conditions imposed are:

$$\int \phi(t) dt = 1$$

$$\int \phi(t) \phi(t - k) dt = \delta_{0,k}$$

$$\int t^n \psi(t) dt = 0 \quad \text{for } n = 0, 1, 2, \dots, 2p - 1$$

$$\int t^n \phi(t) dt = 0 \quad \text{for } n = 1, 2, \dots, 2p - 2$$

Note that $\int \phi(t) dt \neq 0$ and $\int \phi(t) \psi(t - k) dt = 0$ for all k , relate $h(n)$ and $g(n)$.

For $p = 1$, filter coefficients are given by

$$h(n) \equiv [0.038580777748, -0.126969125396, -0.077161555496, 0.607491641386, 0.745687558934, 0.226584265197]$$

Figure 5.2 shows corresponding scaling and wavelet function.

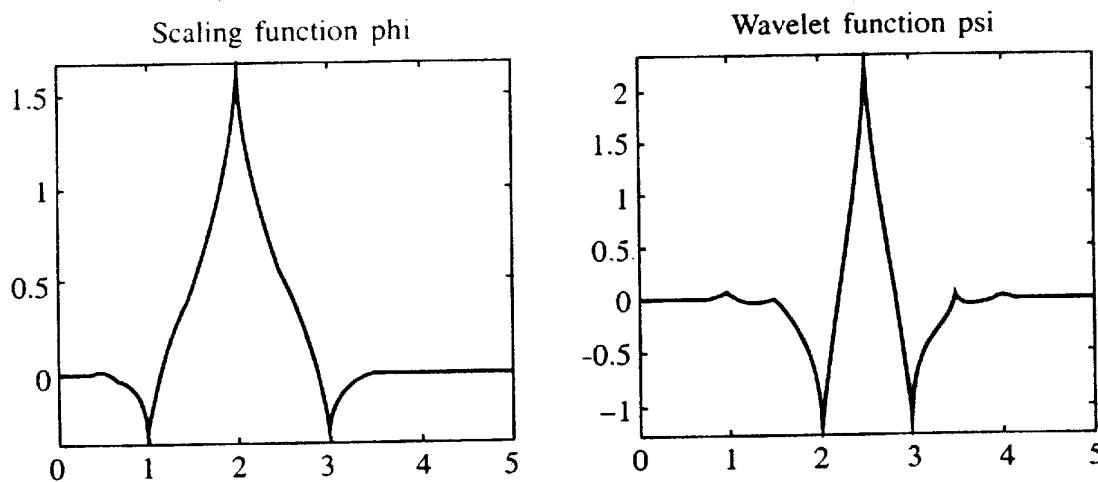


FIGURE 5.2 Coiflet scaling and wavelet function with 6 taps.

For $p = 2$

$$h(n) \equiv [0.016387336463, -0.041464936782, -0.067372554722, 0.386110066823, 0.812723635450, 0.417005184424, -0.076488599078, -0.059434418646, 0.023680171947, 0.005611434819, -0.001823208871, -0.000720549445]$$

5.5 SYMLETS

The solution for the wavelets which was given by Daubechies is not always unique. She gave solutions with minimal phase (maximum smoothness). Other choices can lead to more symmetric solutions. They are never completely symmetric though. Symlets have the following properties:

1. Orthogonal
2. Compact support
3. Filter length $N = 2p$
4. It has p vanishing moments
5. It is nearly linear phase

Design equation of Symlet wavelet is beyond the scope of this elementary book.

For $N = 8$, filter coefficients are:

$$h(n) \equiv [-0.107148901418, -0.041910965125, 0.703739068656, 1.136658243408, 0.421234534204, -0.140317624179, -0.017824701442, 0.045570345896]$$

Corresponding scaling and wavelet functions are shown in Figure 5.3.

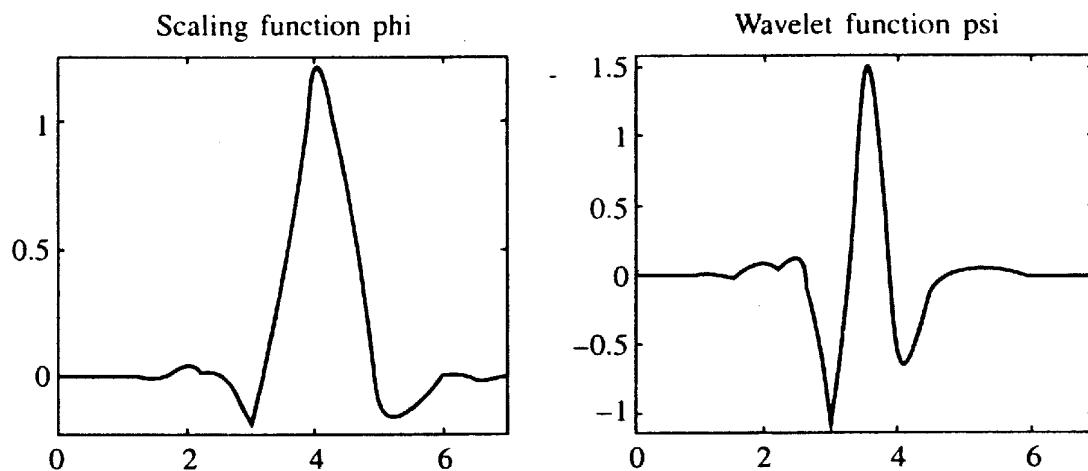


FIGURE 5.3 Symlets scaling and wavelet function with 8 taps.

For $N = 10$, filter coefficients are:

$$h(n) = [0.038654795955, 0.041746864422, -0.055344186117, 0.281990696854, 1.023052966894, 0.896581648380, 0.023478923136, -0.247951362613, -0.029842499869, 0.027632152958]$$

SUMMARY

In this chapter, we derived Daubechies orthogonal wavelet system coefficients $\{h(k)\}$ and $\{g(k)\}$ based on orthogonality and smoothness conditions that must be satisfied by scaling and wavelet functions. These conditions, in turn, impose restrictions on the value of filter coefficients through dilation (refinement) equations. A set of simultaneous non-linear equations on coefficients are formulated and solved to obtain numerical values for the coefficients. This straightforward approach obviate the need for understanding advanced Fourier transform methods that is conventionally used for the purpose.

EXERCISES

- 5.1 For 8-tap Daubechies wavelet system, derive the equations that must be satisfied by scaling function coefficients.
- 5.2 Write a general MATLAB or maple code that generate the equations mentioned in Question 5.1. Your input should only be the number of taps.
- 5.3 Derive the necessary condions for p vanishing moments for scaling function, in terms of scaling function coefficients.

5.4 Prove that any scaling function in an orthogonal wavelet system satisfy the relation:

$$\sum_n \phi(t-n) = 1$$

5.5 Let $D_j = \{k2^{-j}, k \in Z\}$ and $D = \bigcup_{j=0}^{\infty} D_j$ for Daubechies 4-tap wavelet system, prove the following:

$$(a) \quad \phi(t/2) = \frac{1+\sqrt{3}}{4} \phi(t)$$

$$(b) \quad \phi\left(\frac{t+1}{2}\right) = \frac{1-\sqrt{3}}{4} \phi(t) + \frac{1-\sqrt{3}}{4} t + \frac{2+\sqrt{3}}{4}$$

$$(c) \quad \phi\left(\frac{t+1}{2}\right) = \frac{1-\sqrt{3}}{4} \phi(t) + \frac{1-\sqrt{3}}{4} t + \frac{2+\sqrt{3}}{4}$$

$$(d) \quad \phi\left(\frac{t+2}{2}\right) = \frac{1+\sqrt{3}}{4} \phi(t+1) + \frac{1-\sqrt{3}}{4} t + \frac{\sqrt{3}}{4}$$

$$(e) \quad \phi\left(\frac{t+3}{2}\right) = \frac{1-\sqrt{3}}{4} \phi(t+1) - \frac{1+\sqrt{3}}{4} t + \frac{1}{4}$$

$$(f) \quad \phi\left(\frac{t+4}{2}\right) = \frac{1+\sqrt{3}}{4} \phi(t+2) - \frac{1-\sqrt{3}}{4} t + \frac{3-2\sqrt{3}}{4}$$

$$(g) \quad \phi\left(\frac{t+5}{2}\right) = \frac{1-\sqrt{3}}{4} \phi(t+2)$$

$$(h) \quad 2\phi(t) + \phi(t+1) = t + \frac{1+\sqrt{3}}{2}$$

$$(i) \quad 2\phi(t+2) + \phi(t+1) = -t + \frac{3-\sqrt{3}}{2}$$

$$(j) \quad \phi(t) + \phi(t+2) = t + \frac{-1+\sqrt{3}}{2}$$

5.6 Prove that the filter with non-zero taps $h(0) = \frac{\sqrt{2}}{2}$, $h(3) = \frac{-\sqrt{2}}{10}$ and $h(3-2k)$,

$= 12 \frac{\sqrt{2}}{5^{k+1}}$, $k = 1, 2, \dots$, is a wavelet low pass filter.

REFERENCES

- [1] Daubechies, I., *Ten Lectures on Wavelets*, CBMS-NSF, SIAM, 1992.
- [2] Vetterli, M. and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.
- [3] Burrus, C.S., Ramesh A. Gopinath, and Haitao Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, 1997.
- [4] Akansu, A.N. and M.J.T. Smith (Eds.), *Subband and Wavelet Transforms: Design and Applications*, Kluwer Academic, 1996.
- [5] Strang, G. and T.Q. Nguyen, *Wavelets and Filter Banks*, Revised Edition, Wellesley-Cambridge Press, Wellesley, MA, 1998.

CHAPTER

6

Discrete Wavelet Transform and Relation to Filter Banks

INTRODUCTION

We never have to deal most of the applications directly with the scaling functions or associated wavelets. We need only $h(n)$ and $g(n)$ in the refinement relation (dilation equation) and the discretized signal. In this context $h(n)$ and $g(n)$ are viewed as filters and the sequence of data as digital signal. This is from the viewpoint of filter banks developed in 1980s. Most of the results in wavelet theory can be explained either in terms of signal expansion or in terms of filter banks. Students in signal processing must have both viewpoints to get full understanding of this rich and fast growing field.

6.1 SIGNAL DECOMPOSITION (ANALYSIS)

As said earlier, we never see scaling function or wavelet function playing any role directly in the computation of signal expansion coefficients. This is because of the relationship between expansion coefficients at a lower scale and expansion coefficients at a higher scale. Our aim is to establish this relationship. Further, we use Haar scaling and wavelet function to geometrically interpret the result.

We start with refinement relation, the relation responsible for signal decomposition.

We have

$$\phi(t) = \sum_{n=0}^{N-1} h(n) \sqrt{2} \phi(2t - n) = \sum_{n=0}^{N-1} h(n) \phi_{1,n}(t)$$

Here, N is the number of coefficients in the refinement relation.

For Haar, $N = 2$, and the relation is:

$$\begin{aligned}\phi(t) &= h(0)\sqrt{2}\phi(2t) + h(1)\sqrt{2}\phi(2t-1) \\ &= h(0)\phi_{1,0}(t) + h(1)\phi_{1,1}(t)\end{aligned}$$

Note that $h(0)$ and $h(1)$ are normalized coefficients. For Haar, $h(0) = \frac{1}{\sqrt{2}}$ and $h(1) = \frac{1}{\sqrt{2}}$. We aim at relating j th level ϕ with the next higher level scale level $j+1$, so we write

$$\phi(2^j t - k) = \sum_{n=0}^{N-1} h(n)\sqrt{2}\phi[2(2^j t - k) - n] = \sum_{n=0}^{N-1} h(n)\sqrt{2}\phi(2^{j+1}t - 2k - n)$$

By putting $m = 2k + n$, we have

$$\phi(2^j t - k) = \sum_{m=2k}^{2k+N-1} h(m-2k)\sqrt{2}\phi(2^{j+1}t - m) \quad (6.1)$$

Similarly,

$$\psi(2^j t - k) = \sum_{m=2k}^{2k+N-1} g(m-2k)\sqrt{2}\phi(2^{j+1}t - m) \quad (6.2)$$

This relation plays very important role in the derivation. Therefore, we try to visualize this using Haar.

In Eq. (6.1), j and k are fixed by LHS and so are not a variable in RHS. On RHS, m is the only variable. Let us for simplicity take $j = 0$ and $k = 3$. Then what does the resulting RHS mean? See Figure 6.1 for interpretation with respect to Haar scaling function.

Substituting $j = 0$, $k = 3$ and $N = 2$ (for Haar) in Eq. (6.1), we get

$$\begin{aligned}\phi(2^0 t - 3) &= \sum_{m=6}^7 h(m-2k)\sqrt{2}\phi(2^1 t - m) \\ \phi(t-3) &= h(0)\cdot\sqrt{2}\phi(2t-6) + h(1)\sqrt{2}\phi(2t-7)\end{aligned}$$

Since for Haar $h(0) = \frac{1}{\sqrt{2}}$ and $h(1) = \frac{1}{\sqrt{2}}$. Therefore,

$$\phi(t-3) = \frac{1}{\sqrt{2}}\cdot\sqrt{2}\phi(2t-6) + \frac{1}{\sqrt{2}}\sqrt{2}\phi(2t-7) = \phi(2t-6) + \phi(2t-7)$$

This is obvious from Figure 6.1.

[Note that in the Figure 6.1, the value (height) of $\phi(2t-6)$ is shown slightly less than $\phi(t-3)$. Actually both heights are suppose to be same in the first half of $\phi(t-3)$. This is done for clarity. Similarly, $\phi(2t-7)$ and $\phi(t-3)$ must be same in the second half portion of $\phi(t-3)$. The

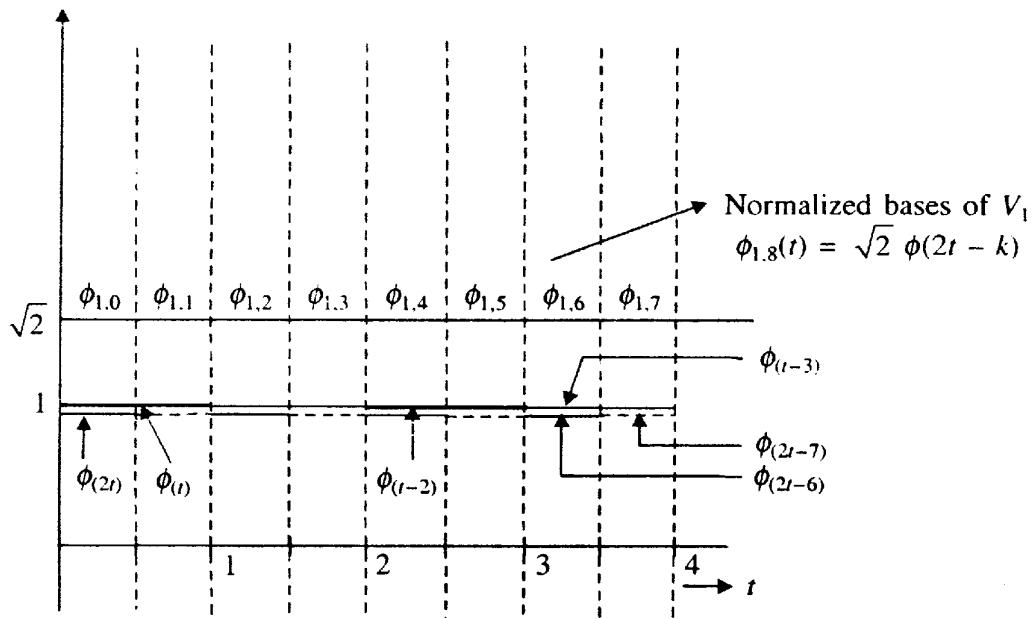


FIGURE 6.1 Haar scaling functions at level 0 and 1.

visualization is quite easy through Haar wavelet. If we had taken Daubechies wavelets, we would have found it very difficult to show it graphically.

If we denote V_j as:

$$V_j = \text{Span}_k \left\{ 2^{j/2} \phi(2^j t - k) \right\}$$

Then

$$f(t) \in V_{j+1} \Rightarrow f(t) = \sum_k s_{j+1}(k) 2^{(j+1)/2} \phi(2^{j+1}(t - k)) \quad (6.3)$$

That is, $f(t)$ is expressed as a linear combination of bases in V_{j+1} . At this scale, wavelets are not coming to picture. Since $V_{j+1} = V_j \oplus W_j$, to represent same signal, in the next lower scale, we require the help of wavelets. So, we have

$$f(t) = \sum_k s_j(k) 2^{j/2} \phi(2^j t - k) + \sum_k d_j(k) 2^{j/2} \psi(2^j t - k) \quad (6.4)$$

The $2^{j/2}$ in each term maintain the unity norm of the basis functions at various scales. How shall we find $s_j(k)$? Simple. Project $f(t)$ on to the corresponding normalized base $2^{j/2} \phi(2^j t - k)$, that is,

$$s_j(k) = \langle f(t), \phi_{j,k}(t) \rangle = \int f(t) 2^{j/2} \phi(2^j t - k) dt \quad (6.5)$$

Substituting Eq. (6.1) in Eq. (6.5) and interchanging summation and integral, we obtain

$$s_j(k) = \sum_{m=2k}^{2k+N-1} h(m-2k) \int f(t) 2^{(j+1)/2} \phi(2^{j+1} t - m) dt \quad (6.6)$$

The integral in Eq. (6.6) is basically projection of $f(t)$ on to the normalized base $2^{(j+1)/2}\phi(2^{j+1}t - m)$ and according to Eq. (6.5), it is $s_{j+1}(m)$.

Therefore,

$$s_j(k) = \sum_{m=2k}^{2k+N-1} h(m-2k) s_{j+1}(m) \quad (6.7)$$

The corresponding relationship for the wavelet coefficients is:

$$d_j(k) = \sum_{m=2k}^{2k+N-1} g(m-2k) s_{j+1}(m) \quad (6.8)$$

6.2 RELATION WITH FILTER BANKS

In signal processing, one of the most frequently used operation is the convolution. Convolution sum, $c(m)$, of two series $h(m)$ and $s(m)$ is defined as:

$$\begin{aligned} c(k) &= \sum_{m=-\infty}^{\infty} s(m) h(k-m) \\ &= \sum_{k=-\infty}^{\infty} s(m) h\{- (m-k)\} \end{aligned} \quad (6.9)$$

Note that the running parameter in the summation is m . k in the summation is fixed by the left side of the equation and hence is a constant. Also, the series $\{h(m-k)\}$, is a k -times translated (or delayed) version of the series $\{h(m)\}$. The series $\{h[-(m-k)]\}$ is a time reversed form of $\{h(m-k)\}$. Therefore, Eq. (6.9) can be interpreted as translation and folding (time reversal) of one series followed by multiplying and summing with the other series. The process is then repeated for each value of k . Since process involves folding of a translated series and then multiplication and summation with other series, the operation is aptly called **convolution**.

Let us try to visualize the operation by taking two small finite series. Let $\{h(k), k = 0, 1, 2, 3\}$ and $\{s(k), k = 0, 1, 2, 3, 4\}$. The two series can be visualized as follows in Figure 6.2:

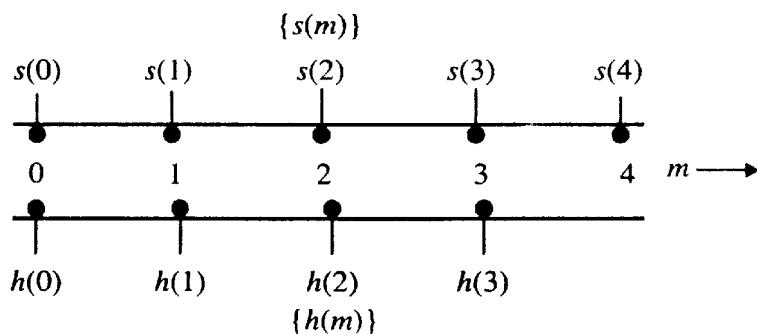


FIGURE 6.2 Filters for Convolutions.

Let us find out $c(0)$. By Eq. (6.9), we get

$$c(0) = \sum_{m=-\infty}^{\infty} s(m) h(-m) = s(0) h(0)$$

Figure 6.3 illustrates the process.

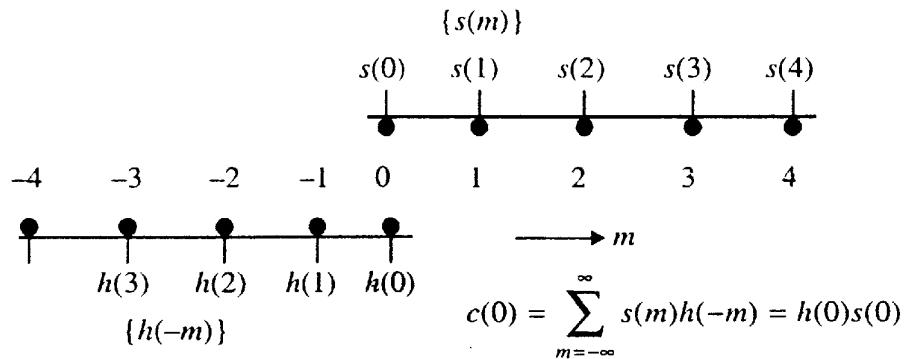


FIGURE 6.3 Finding convolution sum for $k = 0$.

By Eq. (6.9),

$$c(1) = \sum_{k=-\infty}^{\infty} s(m) h(1-m) = s(0) h(1) + s(1) h(0)$$

Figure 6.4 illustrates the process.

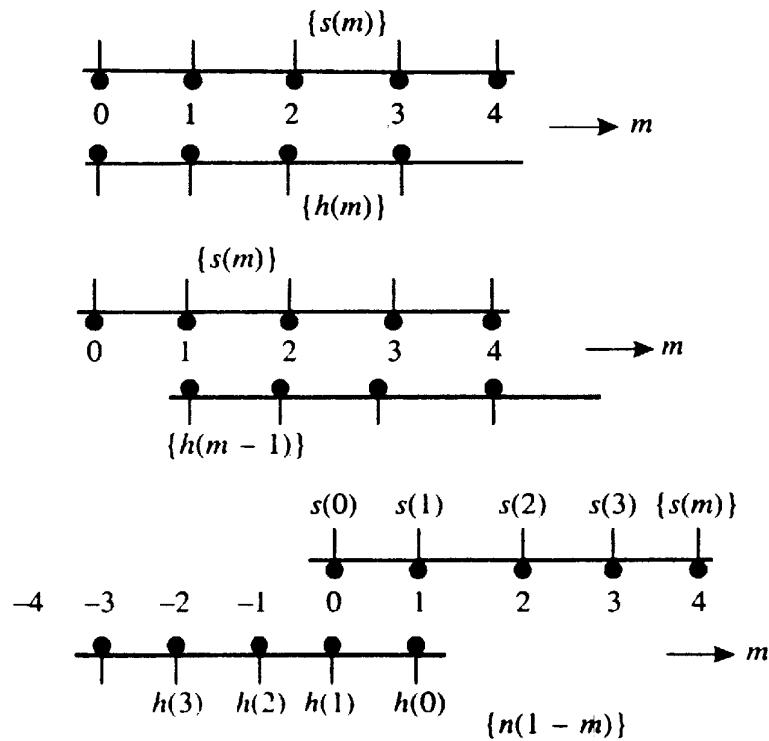


FIGURE 6.4 Finding convolution sum for $k = 1$.

We get the convolved sequence $c(k)$ by translating one of the sequences by k units (or k sample positions), folding it backwards, multiplying the corresponding (overlapping) terms in the two sequences and then adding it.

Let us now try to visualize Eqs. (6.7) and (6.8). We rewrite Eq. (6.7) in convolution format wherein, in the summation of product of two terms, the running parameter is '+' in one term and '-' in other term. In Eq. (6.7), the running parameter in the summation is m . Therefore, we write Eq. (6.7) as:

$$s_j(k) = \sum_{m=2k}^{2k+N-1} h\{-(2k-m)\} s_{j+1}(m) \quad (6.10)$$

$$= \sum_{m=2k}^{2k+N-1} h'(2k-m) s_{j+1}(m) \quad (6.11)$$

where $h'(n) = h(-n)$ represent a time reversed series of $h(n)$.

The parameter k is fixed by left hand side of the equation. Comparing Eqs. (6.9) and (6.11), we find that Eq. (6.11) is a sort of convolution between the series $\{h(-m)\}$ and $\{s_{j+1}(m)\}$. However there is a difficulty in interpreting the formula as a convolution. Translation quantity in Eq. (6.10) is $2k$. To find $s_j(k)$, first we translate $\{h(-m)\}$ by $2k$, fold it backwards, multiply the corresponding term in the series $\{s_{j+1}(m)\}$ and add. Since translation is by $2k$, we can visualize the process as an ordinary convolution followed by decimation of alternative terms. This will become obvious if we take a small example and substitute in Eq. (6.7).

Let us assume that our filter is given by $h(0)$ and $h(1)$, and the signal coefficients when expressed using the bases of V_{j+1} is $s_{j+1}(0), s_{j+1}(1), s_{j+1}(2), s_{j+1}(3), s_{j+1}(4)$ and $s_{j+1}(5)$. For all other indexes assume that $h(k)$ and $s_{j+1}(k)$ are zero. By Eq. (6.7),

$$s_j(k) = \sum_{m=2k}^{2k+N-1} h(m-2k) s_{j+1}(m)$$

where $N = 2$ (because we have only $h(0)$ and $h(1)$).

Therefore, for $k = 0$, we have

$$s_j(0) = \sum_{m=0}^1 h(m) s_{j+1}(m) = h(0)s_{j+1}(0) + h(1)s_{j+1}(1) \quad (6.12)$$

For $k = 1$, we have

$$s_j(1) = \sum_{m=2}^3 h(m-2) s_{j+1}(m) = h(0)s_{j+1}(2) + h(1)s_{j+1}(3) \quad (6.13)$$

For $k = 2$, we have

$$s_j(2) = \sum_{m=4}^5 h(m-4) s_{j+1}(m) = h(0)s_{j+1}(4) + h(1)s_{j+1}(5) \quad (6.14)$$

To give interpretation in terms of convolution, let us convolve $\{h(-m)\}$ series with the series $\{s_{j+1}(m)\}$. In our present case, $\{h(-m)\}$ series becomes $h(1)$ and $h(0)$, the time reversed form of series $\{h(0), h(1)\}$.

The situation is depicted in Figure 6.5.

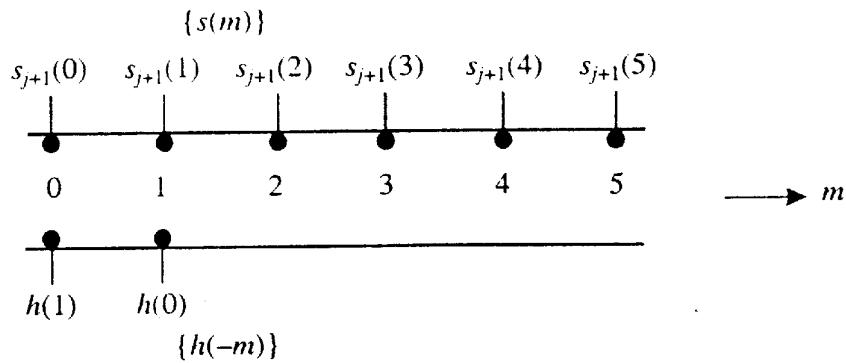


FIGURE 6.5 Filters for convolution.

When we convolve these two series, let the output sequence be represented by $c(k)$ (see Figure 6.6).

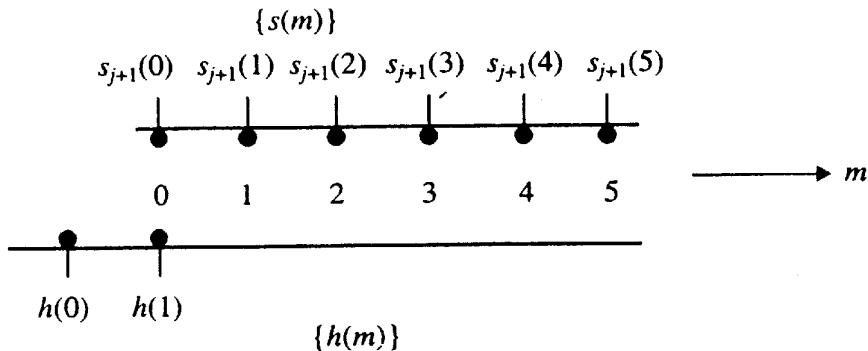


FIGURE 6.6 Computation of $c(0)$.

Now,

$$c(0) = h(1)s_{j+1}(0)$$

$$c(1) = h(0)s_{j+1}(0) + h(1)s_{j+1}(1)$$

$$c(2) = h(0)s_{j+1}(1) + h(1)s_{j+1}(2)$$

$$c(3) = h(0)s_{j+1}(2) + h(1)s_{j+1}(3)$$

$$c(4) = h(0)s_{j+1}(3) + h(1)s_{j+1}(4)$$

$$c(5) = h(0)s_{j+1}(4) + h(1)s_{j+1}(5)$$

It can be easily seen that $s_j(0)$, $s_j(1)$, $s_j(2)$ given by Eqs. (6.12)–(6.14) are $c(1)$, $c(3)$ and $c(5)$. Thus, Eq. (6.7) is equivalent to convolution followed by downsampling of even terms in the output series. So Eq. (6.7) is visualized as illustrated in Figure 6.7.

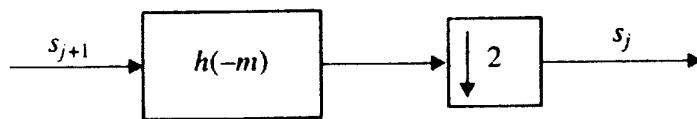


FIGURE 6.7 Relation between s_{j+1} and s_j .

Similarly Eq. (6.8) can be visualized as shown in Figure 6.8.

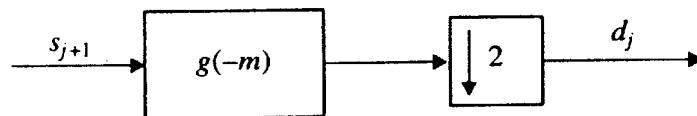


FIGURE 6.8 Relation between s_{j+1} and d_j .

Finally, the analysis, the decomposition of signal in V_{j+1} into sum of two signals, one in space V_j and the other in W_j is visualized as exhibited in Figure 6.9.

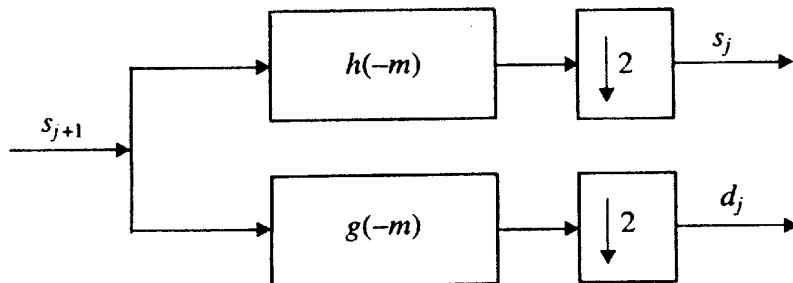


FIGURE 6.9 Relation between s_{j+1} , s_j and d_j .

This is one stage decomposition. In ‘filter bank theory’, it is called **two-band analysis**. The sequence of coefficients $\{s_j(k)\}$ can be further decomposed into sequences $\{s_{j-1}(k)\}$ and $\{d_{j-1}(k)\}$. At each stage, the length of the sequences approximately halves due to decimation by 2 after the convolution. Figure 6.10 shows, two-stage, two-band analysis tree.

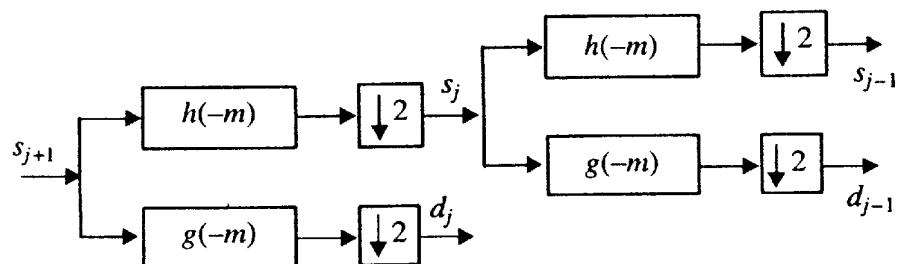


FIGURE 6.10 Two-band analysis of signals.

It will be shown later that the FIR filter represented by $h(-m)$ is a low pass filter and the one implemented by $g(-m)$ is a high pass filter. The sum of number of coefficients in sequence $\{s_j(k)\}$ and $\{d_j(k)\}$ are almost equal to the number of coefficients in the sequence $\{s_{j+1}(k)\}$. Hence there is possibility that no information is lost in the splitting of frequency bands and that the original sequence $\{s_{j+1}(k)\}$ can be recovered from the sequence $\{s_j(k)\}$ and $\{d_j(k)\}$. As we shall see, that is indeed the case. Downsampling will surely result in aliasing but the aliasing occurring in the one channel can be undone by using the downsampled signal (sequence) from the other channel.

6.3 FREQUENCY RESPONSE

The frequency response of a digital filter is the discrete-time Fourier transform of its filter coefficients $h(n)$. This is given by

$$H(\omega) = \sum_{k=-\infty}^{\infty} h(k)e^{j\omega n} \quad (6.15)$$

The magnitude of this complex valued function gives the ratio of the output to the input of the filter for a sampled sinusoid at a frequency of ω in radians per second. The angle of $H(\omega)$ is the phase shift between input and output.

The first stage of the two banks (channel) divides the spectrum of $\{s_{j+1}(k)\}$ into a low pass and high pass band. It results in scaling coefficients and wavelet coefficients $\{s_j(k)\}$ and $\{d_j(k)\}$. The second stage then divides that low pass band into another lower low pass band and high pass band. The first stage divides the spectrum into two equal parts. The second stage divides the lower half into quarters and so on. This results in a logarithmic set of bandwidths as given in Figure 6.12. Figures 6.13 and 6.14 correspond to frequency bands of stage 2 and stage 3 of wavelet decomposition tree shown in Figure 6.11.

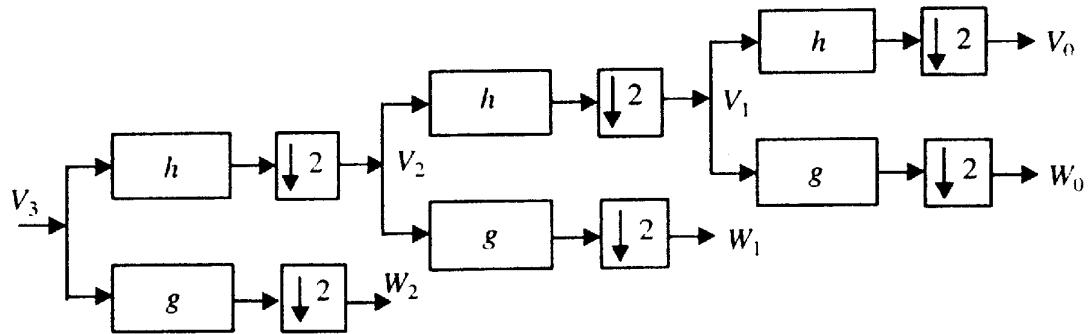


FIGURE 6.11 Three-stage analysis tree.

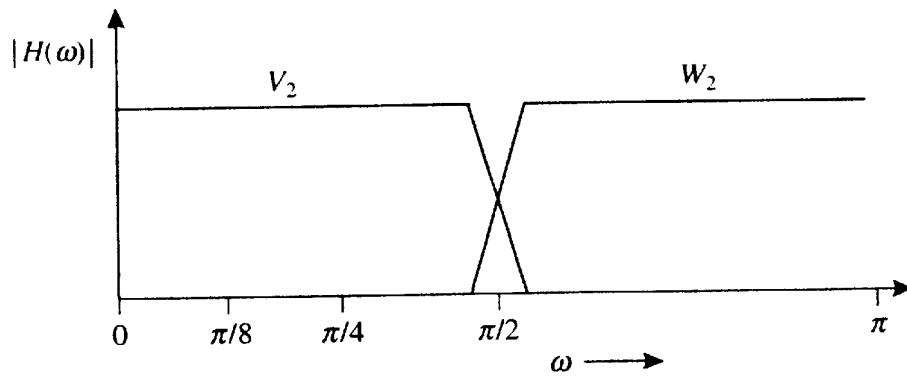


FIGURE 6.12 Splitting of frequency bands (after first stage).

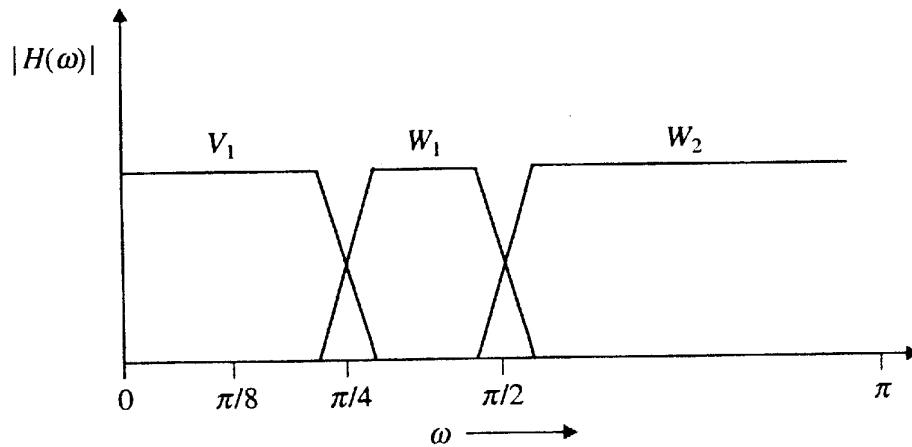


FIGURE 6.13 Splitting of frequency band (after second stage).

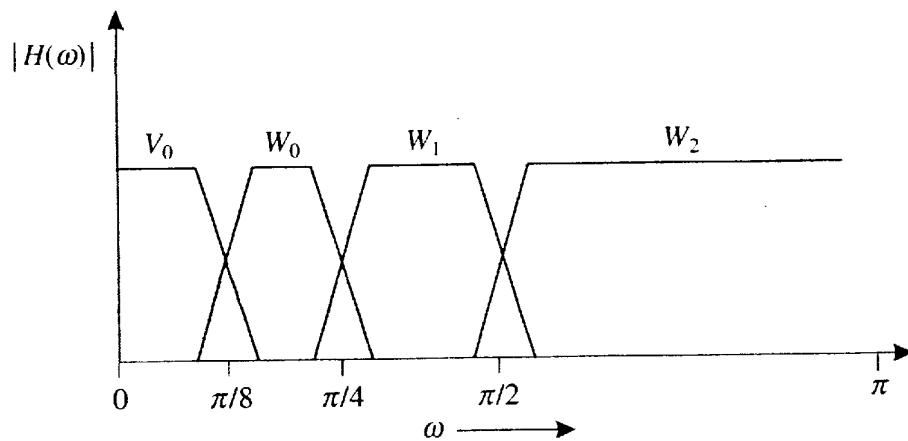


FIGURE 6.14 Splitting of frequency band (after third stage).

6.4 SIGNAL RECONSTRUCTION: SYNTHESIS FROM COARSE SCALE TO FINE SCALE

Here we address the problem of finding the original fine scale coefficients of the signal from the combination of scaling function and wavelet function coefficients at a coarse resolution. In other words we will try to arrive at a formula which finds series $\{s_{j+1}(k)\}$ from the series $\{s_j(k)\}$ and $\{d_j(k)\}$. To derive this we assume that signal $f(t) \in V_{j+1}$. $f(t)$ can be written in terms of basis functions of V_{j+1} , i.e.,

$$f(t) = \sum_k s_{j+1}(k) \cdot 2^{(j+1)/2} \phi(2^{j+1}t - k) \quad (6.16)$$

Since $V_{j+1} = V_j \oplus W_j$, $f(t)$ can be written using the bases of V_j and W_j ,

$$f(t) = \sum_m s_j(m) \cdot 2^{j/2} \phi(2^j t - m) + \sum_m d_j(m) \cdot 2^{j/2} \psi(2^j t - m) \quad (6.17)$$

We know from the refinement relation that

$$\phi(t) = \sum_n h(n) \sqrt{2} \phi(2t - n)$$

and

$$\psi(t) = \sum_n g(n) \sqrt{2} \phi(2t - n)$$

Substituting the above two relations in Eq. (6.17), we obtain

$$\begin{aligned} f(t) &= \sum_m s_j(m) \cdot 2^{j/2} \sum_n h(n) \sqrt{2} \phi\{2(2^j t - m) - n\} \\ &\quad + \sum_m d_j(m) \cdot 2^{j/2} \sum_n g(n) \sqrt{2} \phi\{2(2^j t - m) - n\} \end{aligned} \quad (6.18)$$

$$\begin{aligned} f(t) &= \sum_m s_j(m) \sum_n h(n) 2^{(j+1)/2} \phi(2^{j+1}t - 2m - n) \\ &\quad + \sum_m d_j(m) \sum_n g(n) 2^{(j+1)/2} \phi(2^{j+1}t - 2m - n) \end{aligned} \quad (6.19)$$

Multiplying both sides of Eq. (6.19) by $2^{(j+1)/2} \phi(2^{j+1}t - k)$ and integrating, we obtain

$$\begin{aligned} &\int f(t) 2^{(j+1)/2} \phi(2^{j+1}t - k) dt \\ &= \int \left[\sum_m s_j(m) \sum_n h(n) 2^{(j+1)/2} \phi(2^{j+1}t - 2m - n) \cdot 2^{(j+1)/2} \phi(2^{j+1}t - k) \right] dt \end{aligned}$$

$$+ \int \left[\sum_m d_j(m) \sum_n g(n) 2^{(j+1)/2} \phi(2^{j+1}t - 2m - n) \cdot 2^{(j+1)/2} \phi(2^{j+1}t - k) \right] dt \quad (6.20)$$

On left hand side of Eq. (6.20), we are projecting $f(t)$ on to the base $2^{(j+1)/2} \phi(2^{j+1}t - k)$ which is equal to $s_{j+1}(k)$. On the right hand side, since the bases are orthogonal, only one integral of the product of $\phi(\cdot)$'s remains and all other vanishes. Non-zero (unity) value for the product occurs when $(2m + n) = k$ or when $n = k - 2m$. Therefore,

$$s_{j+1}(k) = \sum_m s_j(m) h(k - 2m) + \sum_m d_j(m) g(k - 2m) \quad (6.21)$$

6.4.1 Upsampling and Filtering

Let us try to visualize RHS of Eq. (6.21) in terms of convolution. If the relation was of the form $\sum_m s_j(m) h(k - m) + \sum_k d_j(m) g(k - m)$ then both the summations are perfect convolution sums. Since the translation quantity is $2m$, each of $h(k)$ and $g(k)$ is not getting multiplied with s and d terms. For a given k , either odd indexed term of $h(k)$ and $g(k)$ or even indexed term of $h(k)$ and $g(k)$ are participating in the summation. So, to make it a perfect convolution sum, without affecting the final result, simply upsample (add 'zero' between each term in s and d) and do the ordinary convolution. This can be visualized as depicted in Figures 6.15 and 6.16.

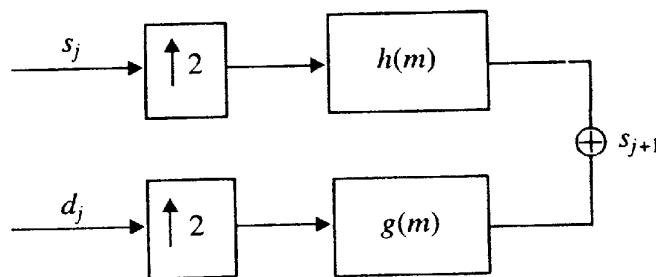


FIGURE 6.15 Synthesis of s_{j+1} from s_j and d_j .

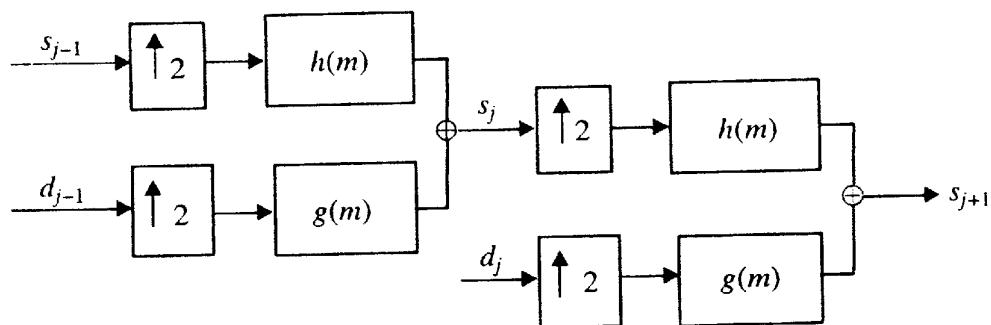


FIGURE 6.16 Two-stage synthesis process.

Let us further visualize Eq. (6.21) through an example. Assuming that $s_j(0)$, $s_j(1)$, $s_j(2)$, $s_j(3)$, and $s_j(4)$ be the s coefficients and $d_j(0)$, $d_j(1)$, $d_j(2)$, $d_j(3)$, and $d_j(4)$ be the d coefficients. Let $h(0)$ and $h(1)$ be the scaling function filter coefficients and $g(0)$ and $g(1)$ be the wavelet function filter coefficients. By Eq. (6.21),

$$s_{j+1}(0) = \sum_m s_j(m) h(0 - 2m) + \sum_m d_j(m) g(0 - 2m)$$

Here m can take only value 0 for getting non-zero product under summation. Therefore,

$$s_{j+1}(0) = s_j(0)h(0) + d_j(0)g(0)$$

$$s_{j+1}(1) = \sum_m s_j(m)h(1 - 2m) + \sum_m d_j(m)g(1 - 2m) = s_j(0)h(1) + d_j(0)g(1)$$

$$s_{j+1}(2) = \sum_m s_j(m)h(2 - 2m) + \sum_m d_j(m)g(2 - 2m) = s_j(1)h(0) + d_j(1)g(0)$$

$$s_{j+1}(3) = \sum_m s_j(m)h(3 - 2m) + \sum_m d_j(m)g(3 - 2m) = s_j(1)h(1) + d_j(1)g(1)$$

$$s_{j+1}(4) = \sum_m s_j(m)h(4 - 2m) + \sum_m d_j(m)g(4 - 2m) = s_j(2)h(0) + d_j(2)g(0)$$

$$s_{j+1}(5) = \sum_m s_j(m)h(5 - 2m) + \sum_m d_j(m)g(5 - 2m) = s_j(2)h(1) + d_j(2)g(1)$$

$$s_{j+1}(6) = \sum_m s_j(m)h(6 - 2m) + \sum_m d_j(m)g(6 - 2m) = s_j(3)h(0) + d_j(3)g(0)$$

$$s_{j+1}(7) = \sum_m s_j(m)h(7 - 2m) + \sum_m d_j(m)g(7 - 2m) = s_j(3)h(1) + d_j(3)g(1)$$

$$s_{j+1}(8) = \sum_m s_j(m)h(8 - 2m) + \sum_m d_j(m)g(8 - 2m) = s_j(4)h(0) + d_j(4)g(0)$$

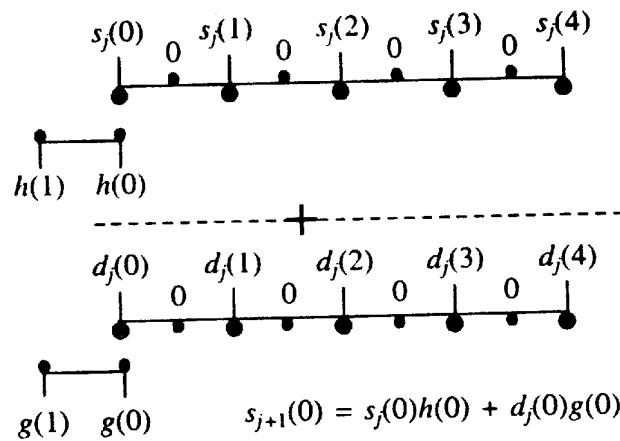
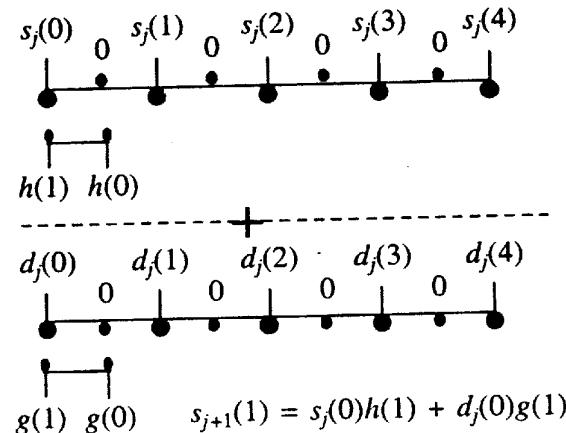
$$s_{j+1}(9) = \sum_m s_j(m)h(9 - 2m) + \sum_m d_j(m)g(9 - 2m) = s_j(4)h(1) + d_j(4)g(1)$$

The same result would have been obtained by upsampling $\{s_j(m)\}$ and $\{d_j(m)\}$ series and then convolving with $\{h(0), h(1)\}$ and $\{g(0), g(1)\}$, respectively.

Figures 6.17 and 6.18 show the pictorial representation of the computation of $s_{j+1}(0)$ and $s_{j+1}(1)$. Note that the result is same.

6.5 PERFECT MATCHING FILTERS

Wavelet-system analysis filters $\{h(-m), g(-m)\}$ decompose the signal into frequency bands and wavelet-system synthesis filters $\{h(m), g(m)\}$ reconstruct the decomposed signal back into the

FIGURE 6.17 Computation of $s_{j+1}(0)$.FIGURE 6.18 Computation of $s_{j+1}(1)$.

original signal (see Figure 6.19). Set of such system of filters are called **perfectly matching filters**. Since $h(m)$ and $g(m)$ are orthogonal (in the vectorial sense) and analysis filters are mirror filters. Since $h(m)$ and $g(m)$ are orthogonal (in the vectorial sense) and analysis filters are mirror filters, these filters are also called **quadrature mirror filters or QMF**.

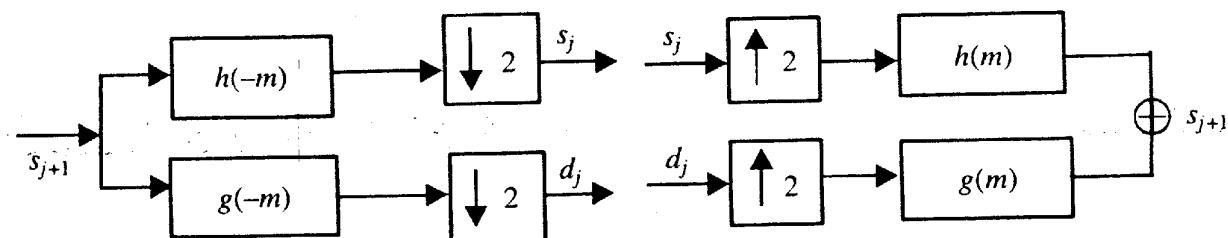


Figure 6.19 Perfectly matching filters.

6.6 COMPUTING INITIAL s_{j+1} COEFFICIENTS

In all the previous discussions we have conveniently assumed that our signal $f(t)$ is an element of space V_{j+1} and $\{s_{j+1}(k), k \in \mathbb{N}\}$ are the corresponding scaling function coefficients. There are two issues here:

1. How do we know that the signal actually belong to space spanned by scaling function of our chosen wavelet system?
2. Secondly how do we compute $\{s_{j+1}(k), k \in \mathbb{N}\}$?

The answer to the first question from practical signal processing point of view is that the signal may not be expressible as linear combination of our chosen scaling function exactly but we can very closely approximate it. We can think of this as the projection of our signal on to the space spanned by the scaling function and its translates.

Given the closely sampled continuous signal, though we can compute the s_{j+1} coefficients using Eq. (6.5) for a particular $j + 1$, usually we assume that sampled signal value itself as s_{j+1} coefficients for some high enough value of level $j + 1$. As the scale level j becomes high, scaling function gets compressed and approaches a pulse signal of unit strength and, therefore, sampled signal itself become scaling function coefficients. This assumption is not 100% right but for most of the practical application this does not matter much. Once we have s_{j+1} coefficients, we proceed to find out s_j and d_j and so on. Also note that, through the synthesis filters we can get back s_{j+1} coefficients which is our original signal.

SUMMARY

The recognition and the study of several phenomena strongly depends on the resolution at which the analysis is carried out. The availability of an efficient way to represent the same phenomena at different resolutions is, therefore, very important (from both a mathematical and a computational point of view). Wavelets can be used in a Multiresolution Analysis (MRA), where the signal and the difference from the previous resolution level are orthogonal, to provide an efficient tool for investigating resolution dependent phenomena. In this chapter we developed formulae for decomposing a signal at high resolution into two orthogonal signals at coarser resolution and vice versa.

EXERCISES

- 6.1** (a) Prove the following statement: The family $\{\psi(t - n)\}$ is an orthonormal basis for W_0 if and only if $|G(\omega)|^2 + |G(\omega + \pi)|^2 = 2$, and $H(\omega) \bar{H}(\omega) + G(\omega) \bar{G}(\omega) = 0$
- (b) Derive an expression for $G(\omega)$ in terms of $H(\omega)$.

- 6.2** If $\phi_1(t)$ and $\phi_2(t)$ satisfy dilation (recursion) equation,
- Does their product $P(t) = \phi_1(t) \phi_2(t)$ satisfy a dilation equation? Justify your answer.
 - Does their convolution $P(t) = \phi_1(t) * \phi_2(t)$ satisfy a dilation equation? Justify your answer.

- 6.3** (a) Verify that $h(n) = \left\{ \frac{2}{5\sqrt{2}}, \frac{6}{5\sqrt{2}}, \frac{3}{5\sqrt{2}}, \frac{-1}{5\sqrt{2}} \right\}$ is a quadrature mirror filter.

[Hint: Show that $\sum_n h(n) h(n-2k) = \delta(k)$ for $k = 0$ and $k = 1$]

- (b) Find the corresponding wavelet filter $g(n)$ for an orthogonal wavelet system.

- 6.4** (a) Show that Fourier transform of the recursive relation $\phi(t) = \sqrt{2} \sum_k h(k) \phi(2t - k)$ is:

$$\phi(\omega) = \frac{1}{\sqrt{2}} \phi(\omega/2) H(\omega/2) = \prod_{k=1}^{\infty} \left(\frac{1}{2} \right) \left(1 + e^{\frac{-j\omega}{2^k}} \right)$$

- (b) For Haar wavelet, show that $\phi(\omega) = e^{\frac{-j\omega}{2}} \left(\frac{\sin(\omega/2)}{\omega/2} \right)$

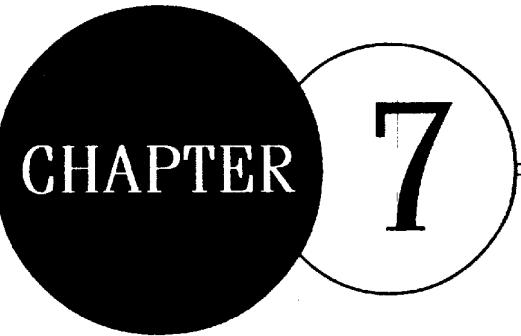
- (c) Show that Fourier transform of Haar wavelet function $\psi(t)$ is:

$$\Psi(\omega) = j e^{\frac{-j\omega}{2}} \left(\frac{\sin^2(\omega/4)}{\omega/4} \right)$$

- 6.5** Prove that $\psi(t)$ has p vanishing moments if and only if $g(n)$ has p discrete vanishing moments.
- 6.6** Let h be the filter associated with the scaling function $\phi(t)$. Prove that if $H(\omega)$ is p -regular [or $H(\omega)$ has p zeroes at $\omega = \pi$] then $\phi'(2\pi k) = 0$ for any integer $k \neq 0$ and $l < p$. Note that $\phi'(2\pi k)$ refers to the l th derivative of the scaling function in the frequency domain.

REFERENCES

- [1] Burrus, C.S. and A. Ramesh Gopinath, and Haitao Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, NJ, 1997.
- [2] Strang, G. and T. Nguyen, *Wavelets and Filter Banks*, Wellesley, Cambridge Press, 1996.
- [3] Vaidyanathan, P.P., *Multirate Systems and Filter Banks*, Prentice Hall, NJ, 1993.
- [4] Vetterli, M. and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, NJ, 1995.
- [5] Mallat, S., *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA, 1999.



Generating and Plotting of Parametric Wavelets

INTRODUCTION

So far for designing wavelet system we were trying to solve a set of simultaneous equations corresponding to various constraints on coefficients. Here we address another problem. Can we generate scaling function coefficients corresponding to a orthogonal wavelet system, randomly, by varying certain parameters? The answer is yes. If we want to generate 8-tap scaling function coefficients $h(k)$, vary 4 angular parameters such that their sum is $\pi/4$. That means basically we can vary 3 angular parameters independently in the range 0 to 2π . This section is devoted to explore the theory behind such wavelets. The importance of the concept is that, we can easily optimize the wavelet system for specific application. In Sections 7.1 and 7.2 we describe a very general parameterization method based on factorization of polyphase matrix. Section 7.3 gives formula (no derivation is provided) for Pollen-type parameterizations of wavelet bases. Section 7.4 describes various methods of plotting scaling and wavelet functions when the filter coefficients are given.

7.1 ORTHOGONALITY CONDITIONS AND PARAMETERIZATION

We will explain the concept by taking the example of a 4-tap wavelet system. Let $h(0)$, $h(1)$, $h(2)$, and $h(3)$ be the scaling function coefficients. For an orthogonal wavelet system, corresponding scaling function and wavelet system must satisfy the following relation implying certain constraints on value of coefficients:

Normality:

$$\int \phi(t) dt = 1 \Rightarrow \sum_k h(k) = \sqrt{2} \quad (7.1)$$

Orthogonality of integer translates of $\phi(t)$:

$$\int \phi(t) \phi(t-k) dt = \delta_{k,0} \Rightarrow \sum_k h(k) h(k-2n) = \delta_{n,0}, n = 0, 1, 2, \dots, \frac{N}{2}-1 \quad (7.2)$$

For $N = 4$ and $n = 0$, we have

$$\sum_k h^2(k) = 1 \Rightarrow h^2(0) + h^2(1) + h^2(2) + h^2(3) = 1 \quad (7.2a)$$

For $N = 4$ and $n = 1$, we have

$$h(0) h(2) + h(1) h(3) = 0 \quad (7.2b)$$

Orthogonality of $\phi(t)$ and $\psi(t)$:

$$\int \phi(t) \psi(t-k) dt = 0 \Rightarrow g(n) = (-1)^n h(N-1-n) \quad (7.3)$$

First vanishing moment of $\psi(t)$:

$$\int \psi(t) dt = 0 \Rightarrow \sum_n g(n) = 0 \Rightarrow \sum_n (-1)^n h(N-1-n) = 0 \Rightarrow \sum_{n \text{ even}} h(n) = \sum_{n \text{ odd}} h(n) \quad (7.4)$$

For $N = 4$, we have

$$g(n) \equiv \{h(3), -h(2), h(1), -h(0)\}$$

Therefore,

$$\sum_n g(n) = 0 \Rightarrow h(0) + h(2) = h(1) + h(3) \quad (7.4a)$$

From Eqs. (7.1) and (7.4a), we have, for a 4-tap wavelet system,

$$h(0) + h(2) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} = h(1) + h(3)$$

or, in general

$$\sum_{n \text{ even}} h(n) = \sum_{n \text{ odd}} h(n) = \frac{1}{\sqrt{2}} \quad (7.5)$$

From Eq. (7.5), we have

$$\left[\sum_{n \text{ even}} h(n) \right]^2 + \left[\sum_{n \text{ odd}} h(n) \right]^2 = \frac{1}{2} + \frac{1}{2} = 1 \quad (7.6)$$

Equations (7.5) and (7.6) play the key role in parametric wavelets. These are basically derived from Eqs. (7.1) and (7.4). You may wonder what is the role of Eqs. (7.2) and (7.3). It can be shown that, if Eqs. (7.5) and (7.6) are true then automatically Eqs. (7.1) to (7.4) are true.

For a 4-tap wavelet system if we set

$$h(0) + h(2) = \cos(\theta_1 + \theta_2)$$

and

$$h(1) + h(3) = \sin(\theta_1 + \theta_2)$$

where $\theta_1 + \theta_2 = \pi/4$.

we see that both Eqs. (7.5) and (7.6) are satisfied since $\cos(\pi/4) = \sin(\pi/4) = 1/\sqrt{2}$ and $\cos^2(\pi/4) + \sin^2(\pi/4) = 1$. We now expand $\cos(\theta_1 + \theta_2)$ and $\sin(\theta_1 + \theta_2)$ and then assign each term to $h(k)$ s. That is,

$$\cos(\theta_1 + \theta_2) = \cos \theta_1 \cos \theta_2 - \sin \theta_1 \sin \theta_2 = h(0) + h(2)$$

$$\sin(\theta_1 + \theta_2) = \sin \theta_1 \cos \theta_2 + \cos \theta_1 \sin \theta_2 = h(1) + h(3)$$

Therefore, we obtain

$$h(0) = \cos \theta_1 \cos \theta_2$$

$$h(2) = -\sin \theta_1 \sin \theta_2$$

$$h(1) = \sin \theta_1 \cos \theta_2$$

$$h(3) = \cos \theta_1 \sin \theta_2$$

Now, by choosing a value of θ_1 randomly and taking $\theta_2 = \pi/4 - \theta_1$ we can generate scaling function coefficients. We can easily verify that by choosing $\theta_1 = \pi/3$ and $\theta_2 = \pi/4 - \theta_1 = -\pi/12$ we obtain Daubechies Daub-4 wavelet system of coefficients.

Generalizing this idea, we have the relation:

$$\sum_{n..even} h(n) = \cos(\theta_1 + \theta_2 + \dots + \theta_{N/2}) = \frac{1}{\sqrt{2}} \quad (7.7)$$

$$\sum_{n..odd} h(n) = \sin(\theta_1 + \theta_2 + \dots + \theta_{N/2}) = \frac{1}{\sqrt{2}} \quad (7.8)$$

Equation (7.7) implies that

$$\sum_{i=1}^{N/2} \theta_i = \pi/4 \quad (7.9)$$

Expanding $\cos(\cdot)$ and $\sin(\cdot)$ functions and distributing the trigonometric monomial to various filter coefficients is a tedious task for scaling function filters with higher order taps. However this problem can be solved in a systematic and elegant way using the lattice factorization from the theory of filter banks. For more details, readers can refer the book by Gilbert Strang and Neugen. Polyphase matrix formulation and its factorization is the key to this method.

7.2 POLYPHASE MATRIX AND RECURRENCE RELATION

It is a matrix with elements as the Z-transform of the even and odd indexed coefficients of scaling function and wavelet function filter coefficients, i.e.,

$$P(z) = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix}$$

where

$H_e(z)$ = Z-transform of the sequence $\{h(2n), n = 0, 1, \dots\}$

$H_o(z)$ = Z-transform of the sequence $\{h(2n + 1), n = 0, 1, \dots\}$

$G_e(z)$ = Z-transform of the sequence $\{g(2n), n = 0, 1, \dots\}$

$G_o(z)$ = Z-transform of the sequence $\{g(2n + 1), n = 0, 1, \dots\}$

For a 2-tap orthogonal wavelet system, let us denote polyphase matrix as:

$$P_1(z) = \begin{bmatrix} h(0) & h(1) \\ g(0) & g(1) \end{bmatrix}$$

For a 4-tap orthogonal wavelet system,

$$P_2(z) = \begin{bmatrix} h(0) + z^{-1}h(2) & h(1) + z^{-1}h(3) \\ g(0) + z^{-1}g(2) & g(1) + z^{-1}g(3) \end{bmatrix}$$

For a 2-tap wavelet system, we equate

$$P_1(z) = \begin{bmatrix} h(0) & h(1) \\ g(0) & g(1) \end{bmatrix} = \begin{bmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{bmatrix}$$

We have only one argument θ_1 here. Therefore, by Eq. (7.9), $\theta_1 = \pi/4$ implying that

$$\begin{bmatrix} h(0) & h(1) \\ g(0) & g(1) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

We can easily see that it is the coefficients for Haar wavelet system.

For a 4-tap orthogonal wavelet system, we equate

$$\begin{aligned} P_2(z) &= \begin{bmatrix} h(0) + z^{-1}h(2) & h(1) + z^{-1}h(3) \\ g(0) + z^{-1}g(2) & g(1) + z^{-1}g(3) \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} \cos \theta_2 & \sin \theta_2 \\ -\sin \theta_2 & \cos \theta_2 \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta_1 & \sin \theta_1 \\ -\sin \theta_1 & \cos \theta_1 \end{bmatrix} \begin{bmatrix} \cos \theta_2 & \sin \theta_2 \\ -z^{-1} \sin \theta_2 & z^{-1} \cos \theta_2 \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} \cos \theta_1 \cos \theta_2 - z^{-1} \sin \theta_1 \sin \theta_2 & \cos \theta_1 \sin \theta_2 + z^{-1} \sin \theta_1 \cos \theta_2 \\ -\sin \theta_1 \cos \theta_2 - z^{-1} \cos \theta_1 \sin \theta_2 & -\sin \theta_1 \sin \theta_2 - z^{-1} \cos \theta_1 \cos \theta_2 \end{bmatrix}$$

Equating the coefficients of equal powers of z on both sides, we obtain

$$h(0) = \cos \theta_1 \cos \theta_2$$

$$h(2) = -\sin \theta_1 \sin \theta_2$$

$$h(1) = \sin \theta_1 \cos \theta_2$$

$$h(3) = \cos \theta_1 \sin \theta_2$$

Similarly, we obtain the equations for $g(k)$ s. Note that, this construction gives the same result as we obtained earlier.

Recursive relation for $P_n(z)$ for N -tap wavelet system, where $N = 2n$, can be expressed as:

$$P_n(z) = P_{n-1}(z)\Lambda(z) \begin{bmatrix} \cos(\theta_{n-1}) & \sin(\theta_{n-1}) \\ -\sin(\theta_{n-1}) & \cos(\theta_{n-1}) \end{bmatrix} \quad (7.10)$$

where

$$\Lambda(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}$$

This relation is the polyphase factorization theorem in filter banks.

Section 7.2.1 gives a maple code generating the coefficients symbolically in terms of angular parameters. MATLAB users can easily run this code since maple is the symbolic computation engine of MATLAB.

7.2.1 Maple Code

```
# Maple program for the parameterization of a
# length N scaling filter in terms of angles.

with(linalg):      # load linear algebra package

N := 8;            # filter length
K := (N-2)/2;     # number of angles (N = 2K+2)
R := t -> matrix(2,2, [cos(t), sin(t), -sin(t), cos(t)]);
Lambda := matrix(2,2,[1, 0, 0, 1/z]);

Hp := R(t0);      # construct polyphase matrix
i := 'i':
for i from 1 to K do
  Hp := evalm(R(t.i) &* Lambda &* Hp):
```

```

od:
H00 := expand(Hp[1,1]): # extract H00 and H01
H01 := expand(Hp[1,2]): # expand polynomials

# display coefficients as functions of angles
lprint(t.K = Pi/4 - sum('t.i','i'=0..(K-1)));
i := 'i':
n := 1:
for i from 0 to K do
    lprint(h[n] = coeff(H00,z,-i));
    n := n + 1:
    lprint(h[n] = coeff(H01,z,-i));
    n := n + 1:
od:

```

7.2.2 Parameterization of Length-6 Scaling Filter

```

function h = h6(t0,t1)
% h = h6(t0,t1)
% length 6 orthogonal scaling filter as a function of 2 angles.
t2 = 1/4*pi-t0-t1;
h(1) = cos(t2)*cos(t1)*cos(t0);
h(2) = cos(t2)*cos(t1)*sin(t0);
h(3) = -cos(t2)*sin(t1)*sin(t0)-sin(t2)*sin(t1)*cos(t0);
h(4) = cos(t2)*sin(t1)*cos(t0)-sin(t2)*sin(t1)*sin(t0);
h(5) = -sin(t2)*cos(t1)*sin(t0);
h(6) = sin(t2)*cos(t1)*cos(t0);

```

7.2.3 Parameterization of Length-8 Scaling Filter

```

function h = h8(t0,t1,t2)
% h = h8(t0,t1,t2)
% length 8 orthogonal scaling filter as a function of 3 angles.
t3 = 1/4*pi-t0-t1-t2;
h(1) = cos(t3)*cos(t2)*cos(t1)*cos(t0);
h(2) = cos(t3)*cos(t2)*cos(t1)*sin(t0);
h(3) = -cos(t3)*cos(t2)*sin(t1)*sin(t0)-
cos(t3)*sin(t2)*sin(t1)*cos(t0)-
sin(t3)*sin(t2)*cos(t1)*cos(t0);

```

```

h(4) = cos(t3)*cos(t2)*sin(t1)*cos(t0)-
       cos(t3)*sin(t2)*sin(t1)*sin(t0)-
       sin(t3)*sin(t2)*cos(t1)*sin(t0);

h(5) = -cos(t3)*sin(t2)*cos(t1)*sin(t0)-
       sin(t3)*sin(t2)*sin(t1)*sin(t0)-
       sin(t3)*cos(t2)*sin(t1)*cos(t0);

h(6) = cos(t3)*sin(t2)*cos(t1)*cos(t0)-
       sin(t3)*sin(t2)*sin(t1)*cos(t0)-
       sin(t3)*cos(t2)*sin(t1)*sin(t0);

h(7) = -sin(t3)*cos(t2)*cos(t1)*sin(t0);

h(8) = sin(t3)*cos(t2)*cos(t1)*cos(t0);

```

7.2.4 Parameterization of Length-10 Scaling Filter

```

function h = h10(t0,t1,t2,t3)
% h = h10(t0,t1,t2,t3)
% length 10 orthogonal scaling filter as a function of 4 angles.

t4 = 1/4*pi-t0-t1-t2-t3;

h(1) = cos(t4)*cos(t3)*cos(t2)*cos(t1)*cos(t0);

h(2) = cos(t4)*cos(t3)*cos(t2)*cos(t1)*sin(t0);

h(3) = -cos(t4)*cos(t3)*cos(t2)*sin(t1)*sin(t0)-
       cos(t4)*cos(t3)*sin(t2)*sin(t1)*cos(t0)-
       cos(t4)*sin(t3)*sin(t2)*cos(t1)*cos(t0)-
       sin(t4)*sin(t3)*cos(t2)*cos(t1)*cos(t0);

h(4) = cos(t4)*cos(t3)*cos(t2)*sin(t1)*cos(t0)-
       cos(t4)*cos(t3)*sin(t2)*sin(t1)*sin(t0)-
       cos(t4)*sin(t3)*sin(t2)*cos(t1)*sin(t0)-
       sin(t4)*sin(t3)*cos(t2)*cos(t1)*sin(t0);

h(5) = -cos(t4)*cos(t3)*sin(t2)*cos(t1)*sin(t0)-
       cos(t4)*sin(t3)*sin(t2)*sin(t1)*sin(t0)-
       cos(t4)*sin(t3)*cos(t2)*sin(t1)*cos(t0)-
       sin(t4)*sin(t3)*cos(t2)*sin(t1)*sin(t0)-
       sin(t4)*cos(t3)*sin(t2)*cos(t1)*cos(t0);

```

```

h(6) = cos(t4)*cos(t3)*sin(t2)*cos(t1)*cos(t0)-
       cos(t4)*sin(t3)*sin(t2)*sin(t1)*cos(t0)-
       cos(t4)*sin(t3)*cos(t2)*sin(t1)*sin(t0)-
       sin(t4)*sin(t3)*cos(t2)*sin(t1)*cos(t0)+
       sin(t4)*sin(t3)*sin(t2)*sin(t1)*sin(t0)-
       sin(t4)*cos(t3)*sin(t2)*cos(t1)*sin(t0);

h(7) = -cos(t4)*sin(t3)*cos(t2)*cos(t1)*sin(t0)-
       sin(t4)*sin(t3)*sin(t2)*cos(t1)*sin(t0)-
       sin(t4)*cos(t3)*sin(t2)*sin(t1)*sin(t0)-
       sin(t4)*cos(t3)*cos(t2)*sin(t1)*cos(t0);

h(8) = cos(t4)*sin(t3)*cos(t2)*cos(t1)*cos(t0)-
       sin(t4)*sin(t3)*sin(t2)*cos(t1)*cos(t0)-
       sin(t4)*cos(t3)*sin(t2)*sin(t1)*cos(t0)-
       vsin(t4)*cos(t3)*cos(t2)*sin(t1)*sin(t0);

```

7.3 POLLEN-TYPE PARAMETERIZATIONS OF WAVELET BASES

Brani Vidakovic [4] in his book has given this formula for Pollen-type parameterization. We give it here for the ready reference. For a four-tap wavelet system, scaling function coefficients are given by

$$h(0) = (1 + \cos \theta - \sin \theta)/s$$

$$h(1) = (1 + \cos \theta + \sin \theta)/s$$

$$h(2) = (1 - \cos \theta + \sin \theta)/s$$

$$h(3) = (1 + \cos \theta - \sin \theta)/s$$

Here $s = 2\sqrt{2}$.

For a six-tap wavelet system, scaling function coefficients are given by

$$h(0) = (1 + \cos \theta_1 - \cos \theta_2 - \cos \theta_1 \cos \theta_2 + \sin \theta_1 - \cos \theta_2 \sin \theta_1 - \sin \theta_2 + \cos \theta_1 \sin \theta_2 - \sin \theta_1 \sin \theta_2)/(2s)$$

$$h(1) = (1 - \cos \theta_1 + \cos \theta_2 - \cos \theta_1 \cos \theta_2 + \sin \theta_1 + \cos \theta_2 \sin \theta_1 - \sin \theta_2 - \cos \theta_1 \sin \theta_2 - \sin \theta_1 \sin \theta_2)/(2s)$$

$$h(2) = (1 + \cos \theta_1 \cos \theta_2 + \cos \theta_2 \sin \theta_1 - \cos \theta_1 \sin \theta_2 + \sin \theta_1 \sin \theta_2)/s$$

$$h(3) = (1 + \cos \theta_1 \cos \theta_2 - \cos \theta_2 \sin \theta_1 + \cos \theta_1 \sin \theta_2 + \sin \theta_1 \sin \theta_2)/s$$

$$h(4) = (1 - \cos \theta_1 + \cos \theta_2 - \cos \theta_1 \cos \theta_2 - \sin \theta_1 - \cos \theta_2 \sin \theta_1 + \sin \theta_2 + \cos \theta_1 \sin \theta_2 - \sin \theta_1 \sin \theta_2)/(2s)$$

$$h(5) = (1 + \cos \theta_1 - \cos \theta_2 - \cos \theta_1 \cos \theta_2 - \sin \theta_1 + \cos \theta_2 \sin \theta_1 + \sin \theta_2 - \cos \theta_1 \sin \theta_2 - \sin \theta_1 \sin \theta_2)/(2s)$$

7.4 PRECISE NUMERICAL EVALUATION OF ϕ AND ψ

Generally, there are no explicit formulae for the basic functions ϕ and ψ . Hence most algorithms concerning scaling functions and wavelets are formulated in term s of the filter coefficients. A good example is the task of computing the function values of ϕ and ψ . Such an algorithm is useful when one wants to make plots of scaling functions and wavelets or of linear combinations of such functions. In Section 4.10 of Chapter 4, we have given a brute force method of plotting scaling and wavelet function. In this section we explain accurate methods of finding values of ϕ and ψ .

7.4.1 Method 1. Cascade Algorithm: Direct Evaluation at Dyadic Rational Points

Computing ϕ at integers

The scaling function ϕ has support on the interval $[0, N - 1]$, with $\phi(0) = 0$ and $\phi(N - 1) = 0$ because it is continuous for $N \geq 4$. We will discard $\phi(N - 1)$ in our computations, but, for reasons explained in the next section on computing ϕ at dyadic rationals, we keep $\phi(0)$.

Putting $t = 0, 1, \dots, N - 2$ in the dilation equation $\phi(t) = \sum_{k=0}^{N-1} h(k) \sqrt{2} \phi(2t - k)$ yields a homogeneous linear system of equations, (shown here for $N = 6$).

$$\begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h(0) & & & & & \phi(0) \\ h(2) & h(1) & h(0) & & & \phi(1) \\ h(4) & h(3) & h(2) & h(1) & h(0) & \phi(2) \\ & h(5) & h(4) & h(3) & h(2) & \phi(3) \\ & & & h(5) & h(4) & \phi(4) \end{bmatrix} \begin{bmatrix} \Phi(0) \\ \Phi(1) \\ \Phi(2) \\ \Phi(3) \\ \Phi(4) \end{bmatrix} = A_0 \Phi(0) \quad (7.11)$$

where we have defined the vector valued function,

$$\Phi(t) = [\phi(t), \phi(t+1), \dots, \phi(t+N-2)]^T$$

Consider the eigenvalue problem for A_0 .

$$A_0 \Phi(0) = \lambda \Phi(0) \quad (7.12)$$

Equation (7.11) has a solution if $\lambda = 1$ is among the eigenvalues of A_0 . Hence the computational problem amounts to finding the eigensolutions of Eq. (7.12). It is shown by G. Strang and T. Nguyen [5] that Eq. (7.11) indeed has an eigenvalue of 1.

Computing ϕ at dyadic rationals

From Eq. (7.11) we can use dilation equation again to obtain ϕ at all midpoints between integers in the interval, namely the vector $\Phi(1/2)$. Substituting into the dilation equation yields another matrix equation of the form (still shown for $N = 6$):

$$\Phi(1/2) = \begin{bmatrix} \phi(1/2) \\ \phi(3/2) \\ \phi(5/2) \\ \phi(7/2) \\ \phi(9/2) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h(1) & h(0) & & & \\ h(3) & h(2) & h(1) & h(0) & \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ & & h(5) & h(4) & h(3) \\ & & & & h(5) \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = A_1 \Phi(0) \quad (7.13)$$

Equation (7.13) is an explicit formula, hence

$$\Phi(0) = A_0 \Phi(0)$$

$$\Phi(1/2) = A_1 \Phi(0)$$

This pattern continues to fractions of the form $k/4$, where k is odd and we get the following system:

$$\begin{bmatrix} * & \phi(1/4) \\ \circ & \phi(3/4) \\ * & \phi(5/4) \\ \circ & \phi(7/4) \\ * & \phi(9/4) \\ \circ & \phi(11/4) \\ * & \phi(13/4) \\ \circ & \phi(15/4) \\ * & \phi(17/4) \\ \circ & \phi(19/4) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h(0) \\ h(1) & h(0) \\ h(2) & h(1) & h(0) \\ h(3) & h(2) & h(1) & h(0) \\ h(4) & h(3) & h(2) & h(1) & h(0) \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ h(5) & h(4) & h(3) & h(2) & h(1) \end{bmatrix} \begin{bmatrix} \phi(1/2) \\ \phi(3/2) \\ \phi(5/2) \\ \phi(7/2) \\ \phi(9/2) \end{bmatrix} \quad (7.14)$$

Equation (7.14) could be used as it is but we observe that if we split it in two systems, one with the equations marked with * and the other with the equations marked with \circ , we can reuse the matrices A_0 and A_1 . This pattern repeats itself as follows:

$$\Phi(1/4) = A_0 \Phi(1/2)$$

$$\Phi(3/4) = A_1 \Phi(1/2)$$

$$\Phi(1/8) = A_0 \Phi(1/4)$$

$$\Phi(5/8) = A_1 \Phi(1/4)$$

$$\Phi(3/8) = A_0 \Phi(3/4)$$

$$\Phi(7/8) = A_1 \Phi(3/4)$$

$$\begin{array}{ll}
 \Phi(1/16) = A_0\Phi(1/8) & \Phi(9/16) = A_1\Phi(1/8) \\
 \Phi(3/16) = A_0\Phi(3/8) & \Phi(11/16) = A_1\Phi(3/8) \\
 \Phi(5/16) = A_0\Phi(5/8) & \Phi(13/16) = A_1\Phi(5/8) \\
 \Phi(7/16) = A_0\Phi(7/8) & \Phi(15/16) = A_1\Phi(7/8)
 \end{array}$$

This is the reason why we keep $\phi(0)$ in the initial eigenvalue Eq. (7.11). We can use the same two matrices for all steps in the algorithm and we can continue as follows until a desired resolution 2^q is obtained:

for $j = 2, 3, \dots, q$

for $k = 1, 3, \dots, 2^{j-1} - 1$

$$\Phi\left(\frac{k}{2^j}\right) = A_0\Phi\left(\frac{k}{2^{j-1}}\right)$$

$$\Phi\left(\frac{k}{2^j} + \frac{1}{2}\right) = A_1\Phi\left(\frac{k}{2^{j-1}}\right)$$

end

end

Function values of the basic wavelet

Function values of ψ follows immediately from the computed values of ϕ by the wavelet equation

$$\psi(t) = \sum_{k=0}^{N-1} g(k)\sqrt{2}\phi(2t - k)$$

However, function values of ϕ are only needed at even numerators:

$$\psi\left(\frac{m}{2^q}\right) = \sqrt{2} \sum_{k=0}^{N-1} g(k) \phi\left(\frac{2m}{2^q} - k\right) \quad (7.15)$$

The MATLAB function `cascade_daub(N, q)` computes $\phi(t)$ and $\psi(t)$ at $t = k/2^q$

where $k = 0, \frac{1}{2^q}, \dots, \frac{N-1}{2^q}$.

Figures 7.1 to 7.3 show $\phi(t)$ and $\psi(t)$ for different values of N drawn using the method described above.

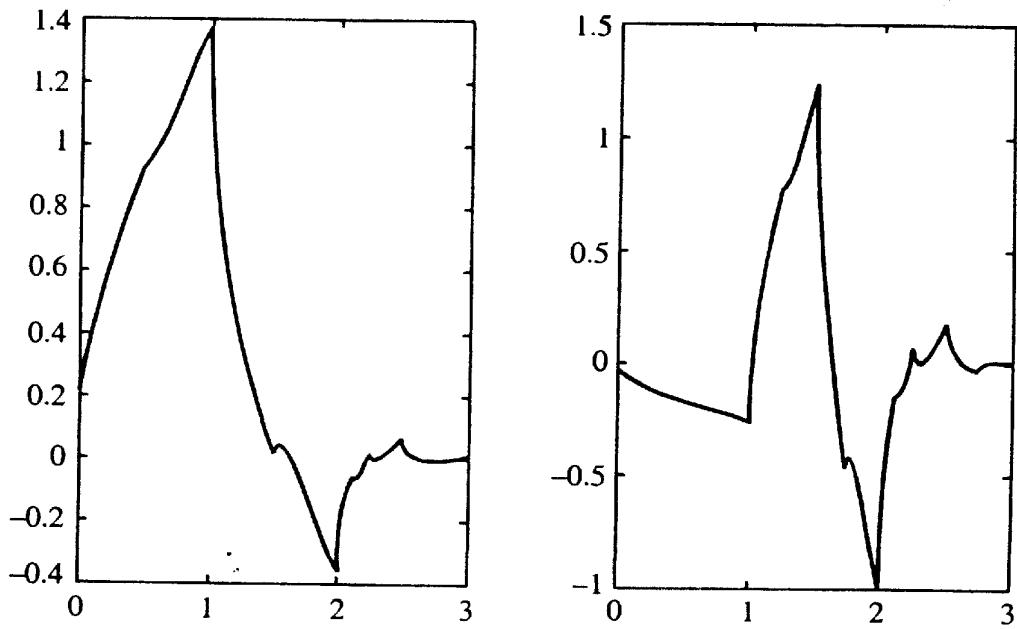


FIGURE 7.1 Daubechies 4-tap scaling and wavelet function.

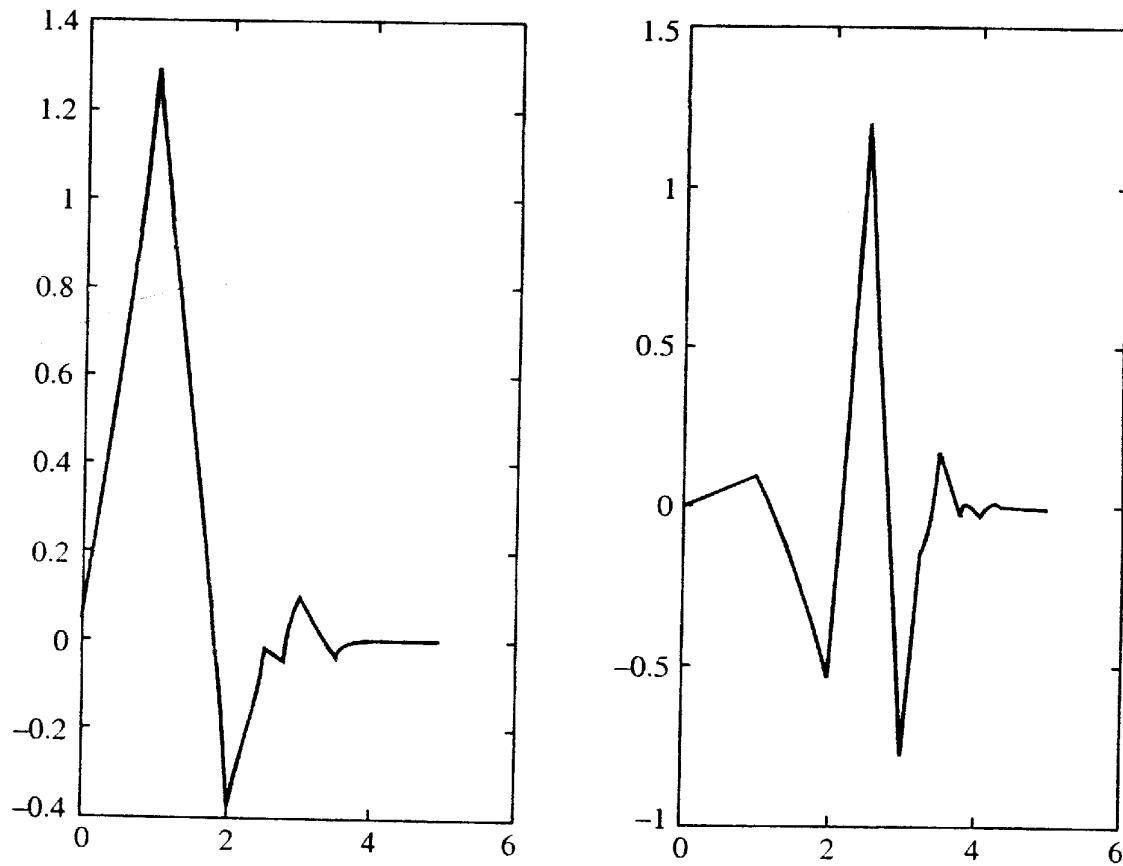


FIGURE 7.2 Daubechies 6-tap scaling and wavelet function.

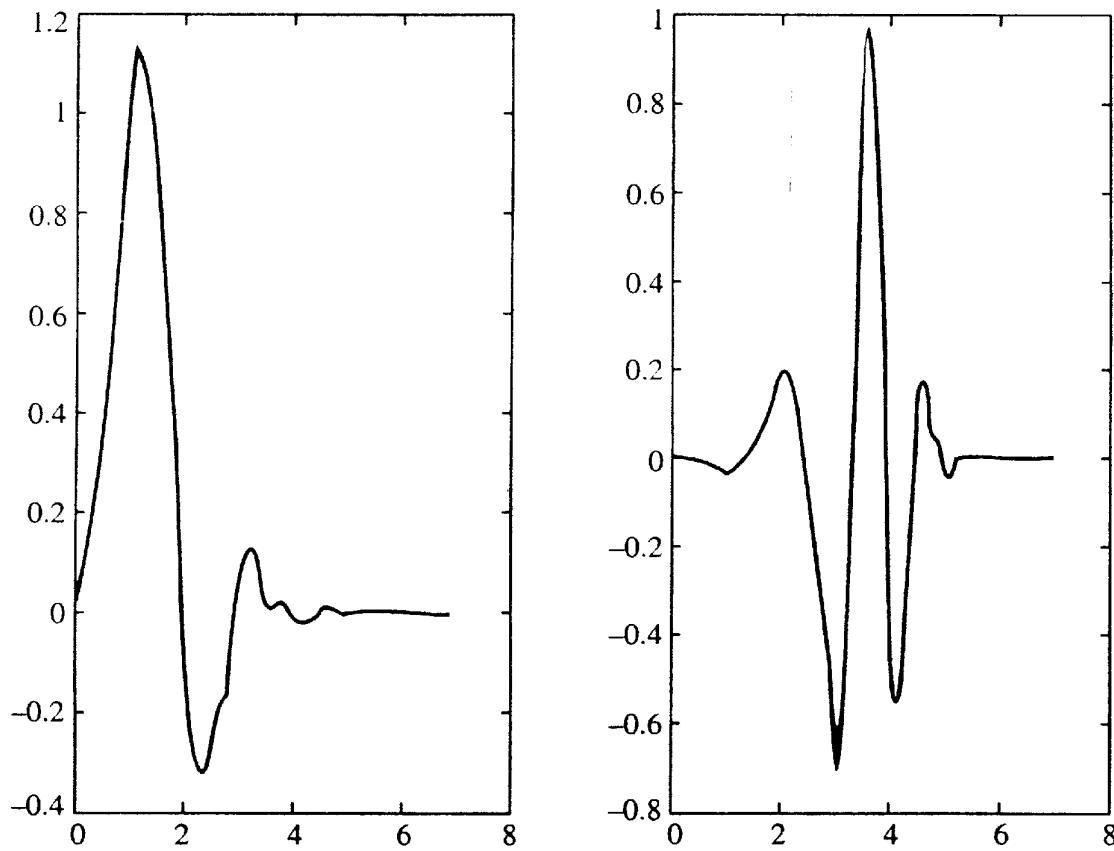


FIGURE 7.3 Daubechies 8-tap scaling and wavelet function.

7.4.2 Method 2. Successive Approximation

We will now discuss a method for the construction of scaling functions using successive approximation. Using scaling functions obtained in this way, associated wavelet function can be constructed.

The approximation method is based on the recursive formula

$$\phi(t) = \sum_{k=0}^{N-1} h(k) \sqrt{2} \phi(2t - k)$$

which becomes analogously

$$\Phi(j\omega) = \frac{1}{\sqrt{2}} \Phi\left(\frac{j\omega}{2}\right) H(e^{j\omega/2}) \quad (7.16)$$

in the frequency domain. If we carry out this recursion i times, we obtain

$$\Phi(j\omega) = \Phi\left(\frac{j\omega}{2^{i+1}}\right) \prod_{k=0}^i \frac{1}{\sqrt{2}} H(e^{j\omega/2^{k+1}}) \quad (7.17)$$

As $i \rightarrow \infty$, the factor on the left $\Phi(j\omega/2^{i+1})$ converges to the value $\Phi(0) = 1$ (since the value of $\int \phi(t) dt = 1$). Hence, we have

$$\Phi(j\omega) = \prod_{k=0}^{\infty} \frac{1}{\sqrt{2}} H(e^{j\omega/2^{k+1}}) \quad (7.18)$$

The limit of RHS exists and is the Fourier transform of a continuous scaling function. Rewriting Eq. (7.18) we have an infinite product:

$$\Phi(j\omega) = \frac{1}{\sqrt{2}} H(e^{j\omega/2^0}) \frac{1}{\sqrt{2}} H(e^{j\omega/2^1}) \frac{1}{\sqrt{2}} H(e^{j\omega/2^2}) \dots \quad (7.19)$$

Here $H(e^{j\omega/2^0})$ is just the Fourier transform of the filter coefficients $\{h(k)\}$, $H(e^{j\omega/2^1})$ is the Fourier transform of the upsampled sequence $\{h(k)\}$, that is,

$$H(e^{j\omega/2^1}) = \text{Fourier transform of } \{h(0), 0, h(1), 0, h(2), 0, \dots, h(N-1)\} \quad (7.20)$$

Also note that

$$\frac{1}{\sqrt{2}} H(e^{j\omega/2^1}) = \text{Fourier transform of } \sqrt{2} \{h(0), 0, h(1), 0, h(2), 0, \dots, h(N-1)\} \quad (7.21)$$

Therefore, in the time domain, the discrete-time values

$$h^{(i)}(n) = \prod_{k=0}^i *h_{0k}(n)$$

can be obtained in the i -th iteration step by i -fold convolution of the dyadic upsampled impulse response,

$$h_{0k}(n) = \begin{cases} 2^{1/2} h(m) & \text{if } n = 2^k m \\ 0 & \text{otherwise} \end{cases} \quad (7.22)$$

Hence, we can derive the “step function”,

$$f^{(i)}(t) = h^{(i)}(n) \text{ with } n/2^i \leq t \leq (n+1)/2^i \quad (7.23)$$

which approximates the scaling function $\phi(t)$ as $i \rightarrow \infty$. In practical applications a few iteration steps are enough to obtain the scaling function in the given graphic resolution.

Example: In this example, we will deal with the construction of the scaling function according to Daubechies with $N = 4$. The coefficients $h(k)$ are as follows:

$$h(0) = \frac{1 + \sqrt{3}}{4\sqrt{2}}$$

$$h(1) = \frac{3 + \sqrt{3}}{4\sqrt{2}}$$

$$h(2) = \frac{3 - \sqrt{3}}{4\sqrt{2}}$$

$$h(3) = \frac{1 - \sqrt{3}}{4\sqrt{2}}$$

By considering the factor $2^{1/2}$, we obtain the impulse response from Eq. (7.22) as:

$$h_{00}(n) = [0.683 \quad 1.183 \quad 0.317 \quad -0.183] \quad (7.24)$$

and

$$h_{01}(n) = [0.683 \quad 0 \quad 1.183 \quad 0 \quad 0.317 \quad 0 \quad -0.183] \quad (7.25)$$

A convolution of these two sequences results in the discrete-time values of $\phi(t)$ as $h^{(1)}(n)$ which is shown in Figure 7.4.

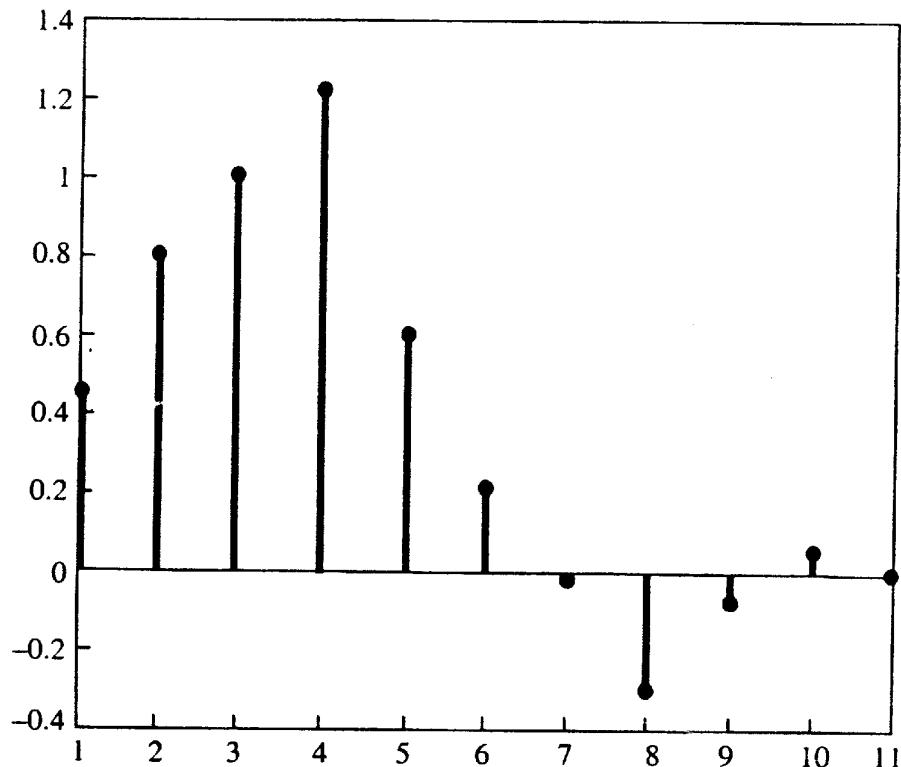


FIGURE 7.4 The stem plot of $\phi(t)$ after one iteration (Daubechies 4-tap scaling function).

The sequence $h_{02}(n)$ has the same numerical values as $h_{01}(n)$ but three zeroes are inserted between each of the non-zero values. The result $h^{(2)}(n)$ is obtained by convolving $h^{(1)}(n)$ with $h_{02}(n)$. Figure 7.5 shows the $h^{(2)}(n)$ values.

Although Eq. (7.23) suggests the derivation of a step function $f^{(i)}(t)$ from the impulse response, it is seen that the iterative approximation of the scaling function can also be portrayed by linear interpolation between the values of the impulse response $h^{(i)}(n)$.

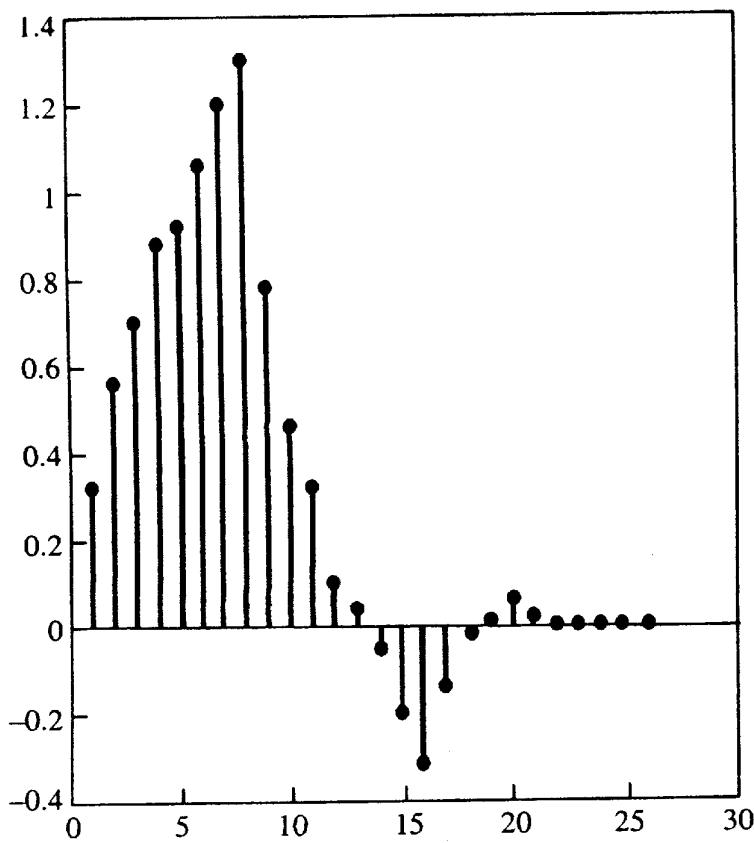


FIGURE 7.5 The stem plot of $\phi(t)$ after two iterations.

The impulse response $h^{(i)}(n)$ has $(n - 1)(2^{i+1} - 1) + 1$ coefficients, where N is the number of coefficients of $h(n)$. If we compare running index with the discrete-time steps

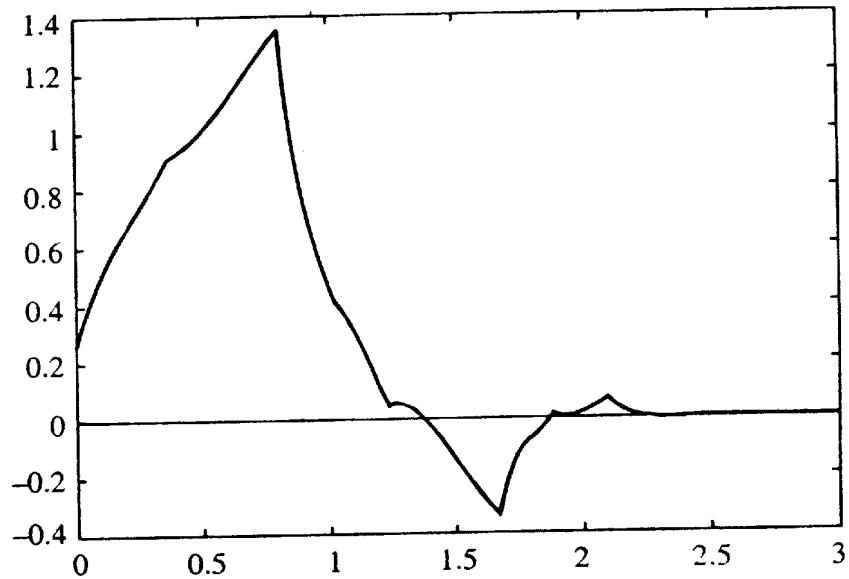
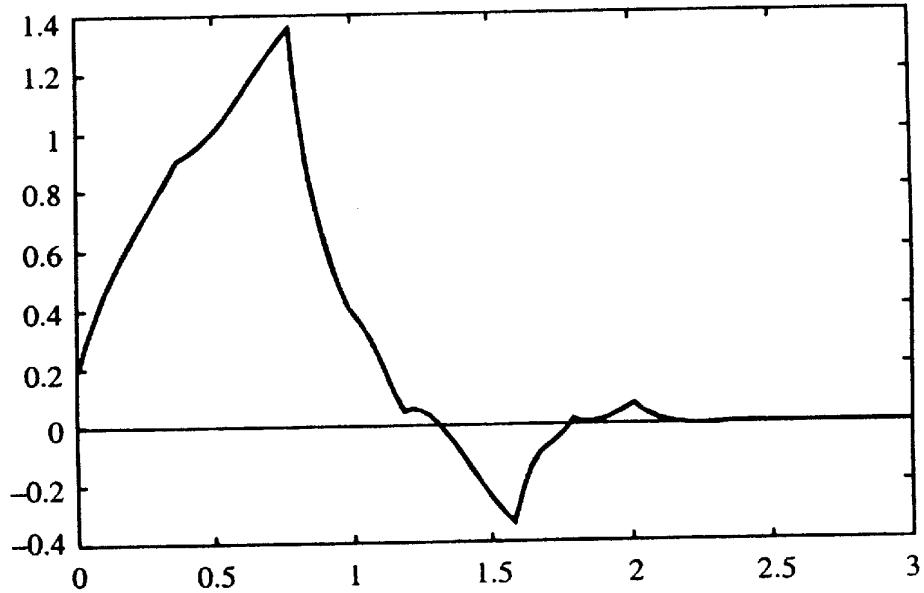
$$t = n/2^{i+1} \quad (7.26)$$

we get the right time-scaling for the linear interpolation mentioned above. It then falls in the interval $0 \leq t \leq N - 1$ as does the scaling function to be approximated. Figures 7.6 and 7.7 show the iterative approximation for $i = 3$ and $i = 4$. The plot has almost converged with four iterations.

7.4.3 Method 3. Daubechies-Lagarias Algorithm

Daubechies-Lagarias algorithm provides a fast and economic calculation of the value of scaling and wavelets functions at the point to a desired accuracy. Such evaluations are essential for wavelet density estimators, wavelet based classification, etc.

We give here a description of the algorithm without any proof of the method. Let ϕ be the scaling function of a compactly supported wavelets generating an orthogonal multiresolution

FIGURE 7.6 The continuous plot of $\phi(t)$ after third iteration.FIGURE 7.7 The continuous plot of $\phi(t)$ after four iteration.

analysis. Suppose the support of ϕ is $[0, N - 1]$. Let $x \in (0, 1)$ and let $dyad(x) = \{d_1, d_2, d_3, \dots, d_n, \dots\}$ be the set of numbers between the digits 0 and 1 in dyadic representation of x :

$$x = \sum_{j=1}^{\infty} d_j 2^{-j}$$

By $dyad(x, n)$ we denote the subset of the first n digits from $dyad(x)$, i.e.,

$$dyad(x, n) = \{d_1, d_2, d_3, \dots, d_n\}$$

Let $h = \{h(0), h(1), \dots, h(N-1)\}$ be the vector of scaling function coefficients. Define two $(N-1) \times (N-1)$ matrices as:

$$T_0 = \sqrt{2}h(2i-j-1) \text{ for } 1 \leq i, j \leq N-1 \quad (7.27)$$

and

$$T_1 = \sqrt{2}(h(2i-j)) \text{ for } 1 \leq i, j \leq N-1 \quad (7.28)$$

Theorem 1 (Daubechies and Lagarias, 1992)

$$\lim_{n \rightarrow \infty} T_{d1} \cdot T_{d2} \cdots \cdot T_{dn} = \begin{bmatrix} \phi(x) & \phi(x) & \dots & \phi(x) \\ \phi(x+1) & \phi(x+1) & \dots & \phi(x+1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi(x+N-2) & \phi(x+N-2) & \dots & \phi(x+N-2) \end{bmatrix} \quad (7.29)$$

Example: Consider the Daubechies-4 case. The corresponding filter is:

$$\left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}} \right)$$

According to Eqs. (7.27) and (7.28), the matrices T_0 and T_1 are given as:

$$T_0 = \begin{bmatrix} \frac{1+\sqrt{3}}{4} & 0 & 0 \\ \frac{3-\sqrt{3}}{4} & \frac{3+\sqrt{3}}{4} & \frac{1+\sqrt{3}}{4} \\ 0 & \frac{1-\sqrt{3}}{4} & \frac{3-\sqrt{3}}{4} \end{bmatrix}$$

and

$$T_1 = \begin{bmatrix} \frac{3+\sqrt{3}}{4} & \frac{1+\sqrt{3}}{4} & 0 \\ \frac{1-\sqrt{3}}{4} & \frac{3-\sqrt{3}}{4} & \frac{3+\sqrt{3}}{4} \\ 0 & 0 & \frac{1-\sqrt{3}}{4} \end{bmatrix}$$

If, for instance, $x = 0.45$, then $\text{dyad}(0.45, 20) = \{0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1\}$. The values $\phi(0.45)$, $\phi(1.45)$ and $\phi(2.45)$ are calculated as:

$$\begin{aligned} \prod_{i \in \text{dyad}(0.45, 20)} T_i &= T_0 T_1 T_1 T_1 T_0 T_0 T_1 T_1 T_0 T_0 T_1 T_1 T_0 T_0 T_1 T_1 T_0 T_0 T_1 T_1 \\ &= \begin{bmatrix} 0.86480542 & 0.86480459 & 0.86480336 \\ 0.08641418 & 0.08641568 & 0.08641719 \\ 0.04878000 & 0.04877973 & 0.04877945 \end{bmatrix} \end{aligned}$$

We may take average of first row as $\phi(0.45)$, average of second and third row as $\phi(1.45)$ and $\phi(2.45)$, respectively.

The Daubechies and Lagarias algorithm gives only the values of the scaling function. However, most of the evaluation needed involves the mother wavelet. It turns out that another algorithm is necessary, due to theorem 2.

Theorem 2 Let x be the arbitrary real number. Let the scaling function coefficients be given by $h = \{h(0), h(1), \dots, h(N - 1)\}$ and let $n = N/2$. Define a vector u with $N - 1$ components as

$$u(x) = \left\{ (-1)^{1-[2x]} h(i + 1 - 2[x]), i = 0, 1, 2, \dots, N - 2 \right\} \quad (7.30)$$

where $[x]$ represents highest integer less than x . If for some i the index is $i + 1 - 2[x]$ is negative or larger than N then the corresponding component of u is zero.

Let e be a vector of $N - 1$ ones and v be defined as:

$$v(x, n) = \frac{1}{N-1} e' \prod_{i \in \text{dyad}(2x, n)} T_i \quad (7.31)$$

Then

$$\psi(x) = \lim_{n \rightarrow \infty} u(x)' v(x, n) \quad (7.32)$$

7.4.4 Method 4. Subdivision Scheme

Subdivision scheme is a method used in computer graphics for interpolation. This idea is used here in plotting $\phi(t)$ and $\psi(t)$. More details about the theoretical foundations of this method can be found in [6]. Here we will describe the process.

Generating $\phi(t)$

The system for generating $\phi(t)$ is basically one channel that we use to reconstruct the signal from wavelet coefficients s_j and d_j . We assume a pulse of unit strength is given to the system as input. After upsampling, it is convolved with filter $h(n)$ and the output is fed back as input. Repeat the upsampling-convolution process to the desired number of times (see Figure 7.8). The final output is equally spaced samples of $\phi(t)$ in the range $[0, N - 1]$ for a scaling function with

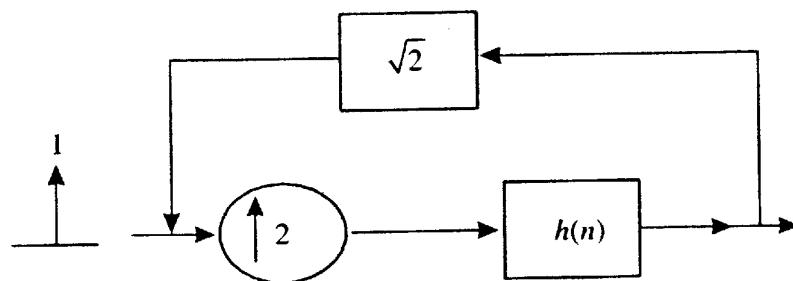


FIGURE 7.8 System for generating $\phi(t)$ through subdivision scheme.

N taps. Multiplication factor $\sqrt{2}$ can be absorbed in the filter itself. Following is MATLAB code for generating Daubechies' 4-tap wavelet scaling function. In this code $\sqrt{2}$ is absorbed in filter coefficients. Figure 7.9 is the output of the MATLAB code.

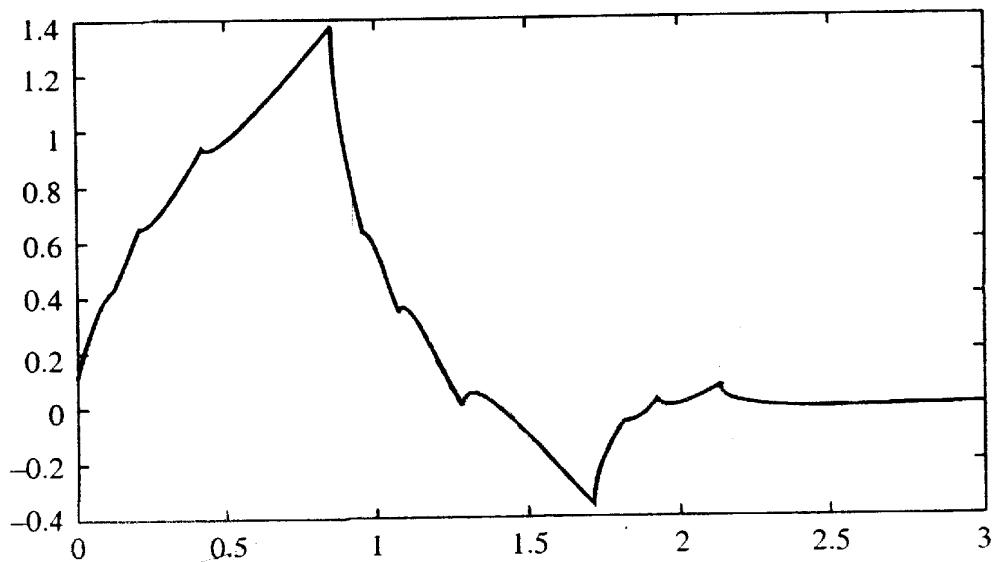


FIGURE 7.9 Output of the MATLAB code for generating $\phi(t)$.

```

h=[0.683 1.183 0.317 -0.183]; %sqrt(2) h(n)
phi=1; %initial pulse
n=10; % no of iteration
for i=1:n
    phi=conv(h,phi);
    if i<n % last convolved need not go for upsampling
        d=upsample(d,2);
    end
end
x=linspace(0,3,length(d));
plot(x,d)

```

Generating $\psi(t)$

Here we give a pulse of unit strength to an upsampler (this upsampler is not really needed. It is put there to show that we are basically using the same signal reconstruction or synthesis steps that we used in multiresolution analysis and synthesis, see Figure 7.10). The output we obtain

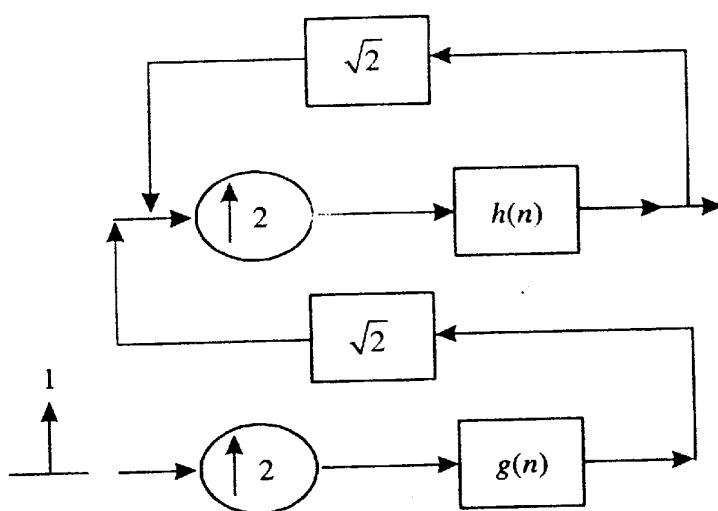


FIGURE 7.10 System for generating $\psi(t)$.

after convolution with $g(n)$, goes to upsampler-convolution loop with $h(n)$ as filter. Number of looping depends on number of equally spaced sampled points of $\psi(t)$ we require in the range $[0, N - 1]$. The following MATLAB code generate $\psi(t)$ of Daubechies 4-tap wavelet function and Figure 7.11 shows the corresponding output:

```

h=[0.683 1.183 0.317 -0.183];
g=[-0.183 -0.317 1.183 -0.683];
psi=1;
psi=conv(psi,g);
n=10;
psi=upsample(psi,2);
for i=1:n
    psi=conv(psi,h);
    if i<n
        psi=upsample(psi,2);
    end
end
x=linspace(0,3,length(psi));
plot(x,psi)
  
```

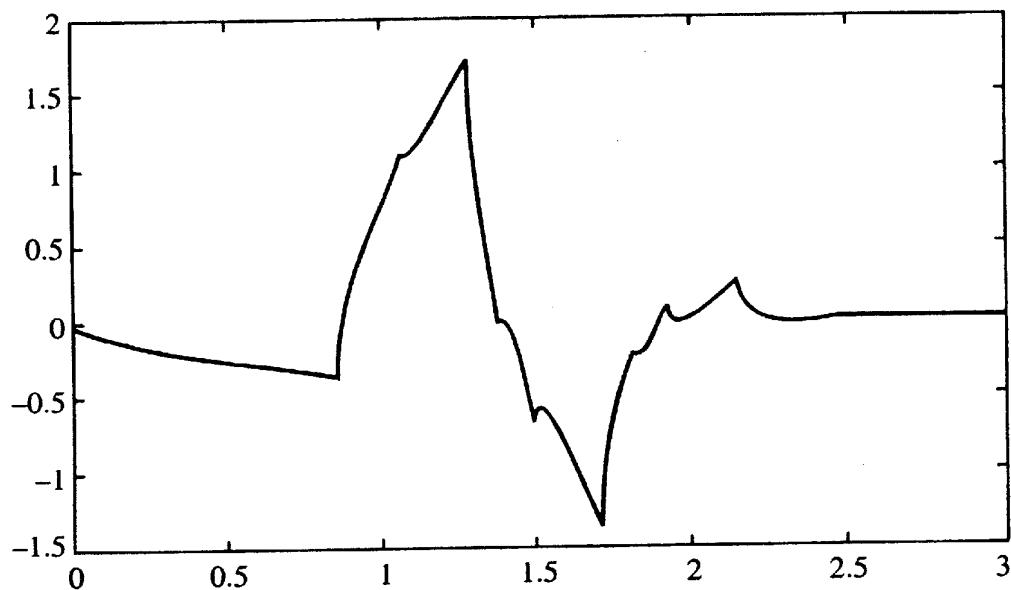


FIGURE 7.11 Output of the MATLAB code for generating $\psi(t)$.

SUMMARY

Parameterization of wavelets allow generation of infinite number of orthogonal wavelet systems. We have discussed an elegant method of generating such orthogonal wavelet system coefficients. Further, most of the wavelets are fractal in nature and hence iterative procedures are required for plotting the scaling and wavelet function. Several simple methods are discussed to plot such functions. MATLAB codes are given for generation as well as plotting.

EXERCISES

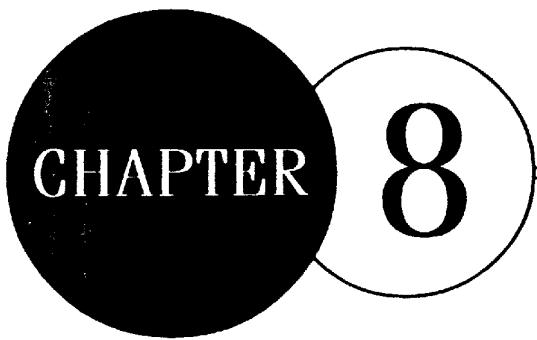
- 7.1 $h(n)$ for an orthogonal wavelet system is given as $\left\{ \frac{2}{5\sqrt{2}}, \frac{6}{5\sqrt{2}}, \frac{3}{5\sqrt{2}}, \frac{-1}{5\sqrt{2}} \right\}$.

Generate the corresponding scaling and wavelet functions using the successive approximation algorithm.

- 7.2 Using Daubechies-Lagarias algorithm find the value of $\psi(1/2)$ and $\psi(1)$ and $\phi(1/4)$ for 4-tap Daubechies' wavelet system.
- 7.3 Write a MATLAB code for cascade algorithm for finding values of $\phi(t)$ at dyadic points.
- 7.4 Write a MATLAB code which takes the angle values of parametric wavelets and then draw the corresponding scaling and wavelet function.

REFERENCES

- [1] Daubechies, I. and J. Lagarias, Two-scale difference equations I. Existence and global regularity of solutions, *SIAM J. Math. Anal.*, 22 (5), 1388–1410, 1991.
- [2] Daubechies, I. and J. Lagarias, Two-scale difference equations II. Local regularity, infinite products of matrices and fractals, *SIAM J. Math. Anal.*, 23 (4), 1031–1079, 1992.
- [3] Pollen, D., SU_12 , ($F(z, 1/z)$) for F a subfield of C , *J. Amer. Math. Soc.*, 3, 611–624, 1990.
- [4] Vidakovic, B., *Statistical Modeling by Wavelets*, Wiley, NY, 1999.
- [5] Strang, G. and T. Nguyen, *Wavelets and Filter Banks*, Wellesley Cambridge Press, 1996.
- [6] Sweldons, W. and Schroder Peter, Building your own wavelets at home, *SIGGRAPH '96, ACM Conference on Graphics and Interactive Techniques*.



Biorthogonal Wavelets

INTRODUCTION

Orthogonality of vectors and functions allow us to represent general vectors and functions in terms of a set of complete orthogonal vectors and functions, respectively. But biorthogonality is a concept which, engineers are not much familiar with. In signal processing, especially in wavelets, however, it is a concept which nobody can ignore. Biorthogonal wavelets are the working horse behind many commercial applications like finger print image compression.

8.1 BIORTHOGONALITY IN VECTOR SPACE

Consider vectors $p_1 = (1, 0)$ and $p_2 = (1/2, \sqrt{3}/2)$. They form a basis for \mathbb{R}^2 . We may note that, p_1 and p_2 are not orthogonal. Since they are not collinear, they span the space \mathbb{R}^2 . Let T be a matrix in which p_1 and p_2 are rows. Thus, T is given by

$$T = \begin{bmatrix} 1 & 0 \\ 1/2 & \sqrt{3}/2 \end{bmatrix}$$

T^{-1} is given by

$$\begin{bmatrix} 1 & 0 \\ -1/\sqrt{3} & 2/\sqrt{3} \end{bmatrix}$$

By taking columns of T^{-1} , let us form vectors d_1 and d_2 so that $d_1 = (1, -1/\sqrt{3})$ and $d_2 = (0, 2/\sqrt{3})$. d_1 and d_2 are not orthogonal but as elements of a basis set they can span \mathbb{R}^2 . We call the basis set $\{d_1, d_2\}$ as dual basis set corresponding to the primal basis set $\{p_1, p_2\}$. Let us now denote the inner product (or dot product) of two vectors p_1 and p_2 as $\langle p_1, p_2 \rangle$.

Since $T T^{-1} = I$, (Identity matrix) a little consideration will show that

$$\begin{aligned}\langle p_1, d_1 \rangle &= 1 \quad [\text{Check: } (1, 0)^T (1, 1/\sqrt{3}) = 1] \\ \langle p_1, d_2 \rangle &= 0, \langle p_2, d_1 \rangle = 0, \langle p_2, d_2 \rangle = 1\end{aligned}$$

The elements of basis set $\{p_1, p_2\}$ are themselves not orthogonal but p_1 is orthogonal to d_2 and p_2 is orthogonal to d_1 . Moreover $\langle p_1, d_1 \rangle = 1$ and $\langle p_2, d_2 \rangle = 1$. Let us see its extension to n dimensional space.

Let $\{p_1, p_2, \dots, p_n\}$ be a set of n independent vectors (each an n -tuple) that form the primal basis set. These vectors are not orthogonal among themselves but they are independent. By taking these vectors as rows of a matrix, we find the inverse of the matrix and take its column vectors as elements of dual basis set. Let it be $\{d_1, d_2, \dots, d_n\}$. These vectors follow the following two relationships:

$$p_i \perp d_j \text{ for every } i \neq j$$

or

$$\langle p_i, d_j \rangle = 0 \text{ for every } i \neq j$$

and

$$\langle p_i, d_i \rangle = 1 \text{ for every } i$$

Consider the two-dimensional example we have just seen. Let us try to express vector $(1, 1)$ using primal basis set $\{p_1, p_2\}$.

$$(1, 1) = ap_1 + bp_2$$

where a and b are two scalars.

$$(1, 1) = a(1, 0) + b\left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)$$

$$\Rightarrow a + b/2 = 1 \quad \text{and} \quad b\left(\frac{\sqrt{3}}{2}\right) = 1$$

$$\Rightarrow b = \frac{2}{\sqrt{3}} \quad \text{and} \quad a = 1 - \frac{1}{\sqrt{3}}$$

That is $(1, 1) = (1 - 1/\sqrt{3})p_1 + 2/\sqrt{3}p_2$.

Can we get these coefficients by projecting the vector $(1, 1)$ on to p_1 and p_2 ? A little thinking will show that we will not get it because p_1 and p_2 are not orthogonal. Now,

Projecting $(1, 1)$ on to d_1 , we get

$$(1, 1)^T (1, -1/\sqrt{3}) = 1 - 1/\sqrt{3} = a$$

Projecting $(1, 1)$ on to d_2 , we get

$$(1, 1)^T (0, 2/\sqrt{3}) = 2/\sqrt{3} = b$$

This is the crux of biorthogonality. Now, we are ready for extending this concept to function spaces.

8.2 BIORTHOGONAL WAVELET SYSTEMS

In many filtering applications we need filters with symmetrical coefficients to achieve linear phase. None of the orthogonal wavelet systems except Haar are having symmetrical coefficients. But Haar is too inadequate for many practical applications. Biorthogonal wavelet system can be designed to have this property. That is our motivation for designing such wavelet system. Non-zero coefficients in analysis filters and synthesis filters are not same.

Let $\phi(t) = \sum_{k=0}^{N-1} h(k) \sqrt{2} \phi(2t - k)$ (8.1)

and its translates form the primal scaling function basis and corresponding space be V_0 .

Let $\tilde{\phi}(t) = \sum_{k=0}^{N-1} \tilde{h}(k) \sqrt{2} \tilde{\phi}(2t - k)$ (8.2)

and translates form the dual scaling function basis and corresponding space be \tilde{V}_0 . This means that $\phi(t)$ and its translates are not orthogonal among themselves but orthogonal to translates of $\tilde{\phi}(t)$ (except one basis function as in vector space) such that

$$\phi(t - k) \perp \tilde{\phi}(t - m) \quad \text{for every } k \neq m$$

which in turn means

$$\int \phi(t - k) \tilde{\phi}(t - m) dt = 0 \quad \text{for } k \neq m$$

Also we need that, like in vector space, projecting $\phi(t - k)$ onto $\tilde{\phi}(t - k)$ should result in unity. That is

$$\int \phi(t - k) \tilde{\phi}(t - k) dt = 1 \quad \text{for every } k$$

These two conditions together we put as:

$$\int \phi(t) \tilde{\phi}(t - k) dt = \delta_{k,0} \quad (8.3)$$

Like scaling functions, wavelet function also follow the scaling relation given by

$$\psi(t) = \sum_k g(k) \sqrt{2} \phi(2t - k)$$

and

$$\tilde{\psi}(t) = \sum_k g(k) \sqrt{2} \tilde{\phi}(2t - k) \quad (8.4)$$

In orthogonal wavelet system, $\phi(t)$ is orthogonal to $\psi(t)$ and its translates. In biorthogonal system our requirement is that $\phi(t)$ be orthogonal to $\tilde{\psi}(t)$ and its translates.

Similarly $\tilde{\phi}(t)$ must be orthogonal to $\psi(t)$ and its translates. Since $\phi(t) \perp \tilde{\psi}(t)$, $\tilde{\psi}(t)$ can be written as:

$$\tilde{\psi}(t) = \sum_{k=0}^{N-1} (-1)^n h(N-k-1) \tilde{\phi}(2t-k) \quad (8.5)$$

(The condition $\phi(t) \perp \tilde{\psi}(t)$ relates $g(k)$ with $h(k)$. $g(k)$ becomes alternate flip of $h(k)$) The derivation of the result follow in the same line as in orthogonal wavelets.

Similarly, since $\tilde{\phi}(t) \perp \psi(t)$,

$$\psi(t) = \sum_{k=0}^{N-1} (-1)^n \tilde{h}(N-k-1) \phi(2t-k) \quad (8.6)$$

Let $\psi(t)$ and its translates span the space W_0 and $\tilde{\psi}(t)$ and its translates span the space \tilde{W}_0 . Like primal and dual scaling functions, $\psi(t)$ is not orthogonal to its own translates but are orthogonal to translates of $\tilde{\psi}(t)$, that is,

$$\int \psi(t-k) \tilde{\psi}(t-m) dt = \delta_{k-m} \quad (8.7)$$

In orthogonal wavelet system $\psi(t)$ is orthogonal to any scaled version of itself and its translates, that is,

$$\int \psi(t-k) 2^{j/2} \tilde{\psi}(2^j t - m) dt = \delta_{j,k-m} \quad (8.8)$$

The integral is zero except when $j = 0$ and $k = m$.

When $j = 0$ and $k = m$, the integral becomes

$$\int \psi(t) \psi(t) dt = 1$$

So in biorthogonal system, the most general equation connecting $\psi(t)$ and $\tilde{\psi}(t)$ is:

$$\int \psi_{j,k}(t) \tilde{\psi}_{j,k}(t) dt = \delta_{j-j} \delta_{k-k} \quad (8.9)$$

Now, we conclude the discussion by putting all the relations in biorthogonal wavelet systems in one place. We have the following relationships:

$$1. V_{-1} \subset V_0 \subset V_1 \subset \dots \subset V_\infty$$

$$2. \tilde{V}_{-1} \subset \tilde{V}_0 \subset \tilde{V}_1 \subset \tilde{V}_2 \subset \dots \subset \tilde{V}_\infty$$

where $V_j = \text{span}_k \{2^{j/2} \phi(2^j t - k)\}$ and $\tilde{V}_j = \text{span}_k \{2^{j/2} \tilde{\phi}(2^j t - k)\}$

In orthogonal wavelet system we have V_j orthogonal to W_j .

3. But in biorthogonal system V_j is orthogonal only to \tilde{W}_j . Similarly, \tilde{V}_j is orthogonal only to W_j .
4. $W_k \perp \tilde{W}_j$.

8.3 SIGNAL REPRESENTATION USING BIORTHOGONAL WAVELET SYSTEM

Let $f(t)$ be a signal (function) from V_j and is given by

$$f(t) = \sum_k s_j(k) 2^{j/2} \phi(2^j t - k)$$

Since $\phi_{j,k}(t)$ s are not orthogonal, to get $s_j(k)$ we project $f(t)$ on to the dual basis $\tilde{\phi}_{j,k}(t)$, i.e.,

$$s_j(k) = \int f(t) 2^{j/2} \tilde{\phi}(2^j t - k) dt = \int f(t) \tilde{\phi}_{j,k}(t) dt$$

As $V_j = V_{j-1} \oplus W_{j-1}$, $f(t)$ can be expressed using the basis of V_{j-1} and W_{j-1} . So projection on to V_j is equivalent to projection on to V_{j-1} and W_{j-1} giving two sets of coefficients. Since bases of V_{j-1} and W_{j-1} are not orthogonal among themselves, we have to project $f(t)$ on to its duals. The filters taking part in analysis are $\tilde{h}(k)$ and $\tilde{g}(k)$ while during the synthesis phase we use filters $h(k)$ and $g(k)$.

8.4 BIORTHOGONAL ANALYSIS

We have

$$\phi(t) = \sum_k h(k) \sqrt{2} \phi(2t - k)$$

and

$$\tilde{\phi}(t) = \sum_k \tilde{h}(k) \sqrt{2} \tilde{\phi}(2t - k)$$

Now,

$$\tilde{\phi}(2^j t - k) = \sum_n \tilde{h}(n) \sqrt{2} \tilde{\phi}\{2(2^j t - k) - n\} = \sum_n \tilde{h}(n) \sqrt{2} \tilde{\phi}(2^{j+1} t - 2k - n) \quad (8.10)$$

Similarly, as in orthogonal wavelet system

$$\tilde{\psi}(2^j t - k) = \sum_n \tilde{g}(n) \sqrt{2} \tilde{\phi}\{2(2^j t - k) - n\} = \sum_n \tilde{g}(n) \sqrt{2} \tilde{\phi}(2^{j+1} t - 2k - n) \quad (8.11)$$

By substituting variable $m = 2k + n$ in Eqs. (8.10) and (8.11), we get

$$\tilde{\phi}(2^j t - k) = \sum_m \tilde{h}(m - 2k) \sqrt{2} \tilde{\phi}(2^{j+1} t - m) \quad (8.12)$$

$$\tilde{\psi}(2^j t - k) = \sum_m \tilde{g}(m - 2k) \sqrt{2} \tilde{\phi}(2^{j+1} t - m) \quad (8.13)$$

Let $f(t)$ be the element of V_{j+1} . Then

$$f(t) = \sum_k s_{j+1}(k) 2^{(j+1)/2} \phi(2^{j+1} t - k) = \sum_k s_j(k) 2^{j/2} \phi(2^j t - k) + \sum_k d_j(k) 2^{j/2} \psi(2^j t - k)$$

where

$$s_{j+1}(k) = \int f(t) 2^{(j+1)/2} \tilde{\phi}(2^{j+1} t - k) dt$$

$$s_j(k) = \int f(t) 2^{j/2} \tilde{\phi}(2^j t - k) dt$$

and

$$d_j(k) = \int f(t) 2^{j/2} \tilde{\psi}(2^j t - k) dt$$

Using the relation

$$\tilde{\phi}(2^j t - k) = \sum_m \tilde{h}(m - 2k) \sqrt{2} \tilde{\phi}(2^{j+1} t - m), s_j(k) \text{ can be written as:}$$

$$\begin{aligned} s_j(k) &= \int f(t) 2^{j/2} \tilde{\phi}(2^j t - k) dt = \int f(t) 2^{j/2} \sum_m \tilde{h}(m - 2k) \sqrt{2} \tilde{\phi}(2^{j+1} t - m) dt \\ &= \sum_m \tilde{h}(m - 2k) \int f(t) 2^{(j+1)/2} \tilde{\phi}(2^{j+1} t - m) dt \\ &= \sum_{m=2k}^{2k+N-1} \tilde{h}(m - 2k) s_{j+1}(m) \end{aligned} \quad (8.14)$$

(The relation assumes that $\tilde{h}(k)$ are defined for $k = 0$ to $N - 1$. The limit for m depends on to what indexes k , $\tilde{h}(k)$ are defined.)

Similarly,

$$\begin{aligned} d_j(k) &= \int f(t) 2^{j/2} \tilde{\psi}(2^j t - k) dt = \int f(t) 2^{j/2} \sum_m \tilde{g}(m - 2k) \sqrt{2} \tilde{\phi}(2^{j+1} t - m) dt \\ &= \sum_m \tilde{g}(m - 2k) \int f(t) 2^{(j+1)/2} \tilde{\phi}(2^{j+1} t - m) dt \\ &= \sum_{m=2k}^{2k+N-1} \tilde{g}(m - 2k) s_{j+1}(m) \end{aligned} \quad (8.15)$$

(The relation also assumes that $\tilde{h}(k)$ are defined for $k = 0$ to $N - 1$.)

For interpretation of the result reader can refer to chapter on orthogonal wavelets. Now, we move on to synthesis.

8.5 BIORTHOGONAL SYNTHESIS—FROM COARSE SCALE TO FINE SCALE

In this section we will try to get formula for getting s_{j+1} coefficients from s_j and d_j coefficients. Let $f(t)$ be the element of V_{j+1} . Then

$$\begin{aligned} f(t) &= \sum_k s_{j+1}(k) 2^{(j+1)/2} \phi(2^{j+1}t - k) \\ &= \sum_k s_j(k) 2^{j/2} \phi(2^j t - k) + \sum_k d_j(k) 2^{j/2} \psi(2^j t - k) \end{aligned}$$

we have

$$\phi(2^j t - k) = \sum_m h(m - 2k) \sqrt{2} \phi(2^{j+1} t - m)$$

and

$$\psi(2^j t - k) = \sum_m g(m - 2k) \sqrt{2} \phi(2^{j+1} t - m)$$

Substituting in the above equation, we get

$$\begin{aligned} f(t) &= \sum_k s_j(k) \sum_m h(m - 2k) 2^{(j+1)/2} \phi(2^{j+1} t - m) \\ &\quad + \sum_k d_j(k) \sum_m g(m - 2k) 2^{(j+1)/2} \phi(2^{j+1} t - m) \\ f(t) &= \sum_k s_j(k) \sum_m h(m - 2k) \phi_{j+1,m}(t) + \sum_k d_j(k) \sum_m g(m - 2k) \phi_{j+1,m}(t) \end{aligned}$$

Let us now project $f(t)$ on to normalized base $\tilde{\phi}_{j+1,n}(t) = 2^{(j+1)/2} \tilde{\phi}(2^{j+1}t - n)$ to get $s_{j+1}(n)$. This gives

$$\begin{aligned} s_{j+1}(n) &= \int \left[\sum_k s_j(k) \sum_m h(m - 2k) \phi_{j+1,m}(t) \right] \tilde{\phi}_{j+1,n}(t) dt \\ &\quad + \int \left[\sum_k d_j(k) \sum_m g(m - 2k) \phi_{j+1,m}(t) \right] \tilde{\phi}_{j+1,n}(t) dt \end{aligned}$$

Since $\int \phi_{j+1,m}(t) \tilde{\phi}_{j+1,n}(t) dt = \delta_{m-n}$ (That is only when $m = n$, the integral exists.)

$$s_{j+1}(n) = \sum_k s_j(k) h(n - 2k) + \sum_k d_j(k) g(n - 2k) \quad (8.16)$$

Note that the synthesis involves filters h and g .

8.6 CONSTRUCTION OF BIORTHOGONAL WAVELET SYSTEMS

To understand the whole theory better let us first consider some biorthogonal filters and construct corresponding scaling functions and wavelet functions. Spline based biorthogonal wavelet systems are most easy to construct. At first we will define what splines are and then we make a dual wavelet system. More details on splines and multiresolution properties of splines are given in Chapter 13.

8.6.1 B-Splines

B-splines form a set of scaling functions satisfying the dilation equation with binomial filter coefficients. However, B-splines other than the zeroth order B-spline (the Haar function) are not orthogonal to its own shifts. Hence to form wavelet system, biorthogonal scaling functions are used.

B-splines are produced by convolving Haar scaling function with itself. To avoid confusion in notation (splines are indexed by the order of the polynomial), let us time being write Haar function $\phi(t)$ as:

$$\begin{aligned} N_1(t) &= 1 & 0 \leq t \leq 1 \\ &= 0 & \text{elsewhere} \end{aligned} \quad (8.17)$$

Let $N_{1,k}(t) = \phi(t - k)$ represents translates of $N_1(t)$, which we know are orthogonal. Also we know

$$N_1(t) = \phi(2t) + \phi(2t - 1)$$

Convolving $\phi(t)$ with itself we get a new function having a tent shape. Let us denote this function by $N_2(t)$.

$$\begin{aligned} N_2(t) &= \int \phi(\tau) \phi(t - \tau) d\tau \\ N_2(t) &= \begin{cases} t & 0 \leq t \leq 1 \\ 2 - t & 1 \leq t \leq 2 \end{cases} \end{aligned} \quad (8.18)$$

The function is shown in Figure 8.1. We call this function a spline of order 2. The beauty of this spline function is that it can be expressed as a linear combination of scaled version of itselfs and its translates. Figure 8.2 illustrates the concept.

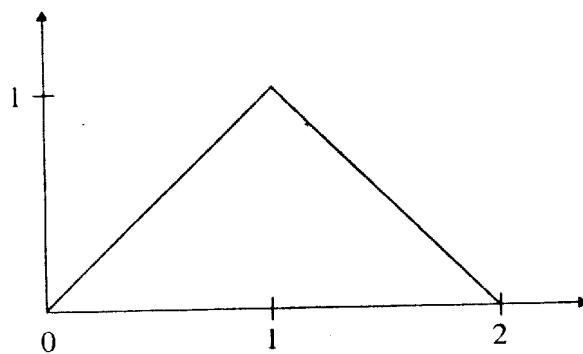


FIGURE 8.1 B-spline of order 2.

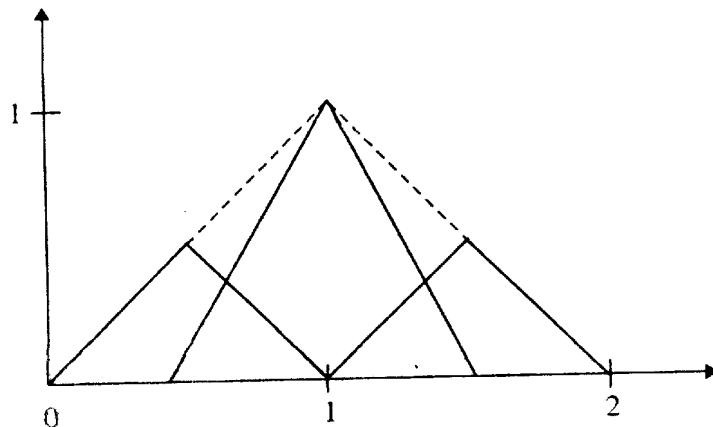


FIGURE 8.2 Construction of B-spline of order 2 from its scaled version.

Now,

$$N_2(t) = \frac{1}{2} N_2(2t) + \frac{2}{2} N_2(2t - 1) + \frac{1}{2} N_2(2t - 2) \quad (8.19)$$

The scaling function coefficients are $1/2(1, 2, 1)$. Note that $(1, 2, 1)$ are binomial coefficients which in turn are obtained by convolving $(1, 1)$ with $(1, 1)$. Similarly, $N_3(t)$ is obtained by convolving box function $\phi(t)$ (Haar scaling function) by itself 3 times. It is equivalent to convolving $\phi(t)$ with $N_2(t)$. Therefore, $N_3(t)$ is given by

$$N_3(t) = \int N_2(\tau) \phi(t - \tau) d\tau$$

$$N_3(t) = \begin{cases} \frac{1}{2}t^2 & 0 \leq t \leq 1 \\ \frac{1}{2}(-2t^2 + 6t - 3) & 1 \leq t \leq 2 \\ \frac{1}{2}(t^2 - 6t + 9) & 2 \leq t \leq 3 \end{cases} \quad (8.20)$$

The two scale relation (dilation equation) for this function is given by

$$N_3(t) = \frac{1}{4} N_3(2t) + \frac{3}{4} N_3(2t - 1) + \frac{3}{4} N_3(2t - 2) + \frac{1}{4} N_3(2t - 3) \quad (8.21)$$

The five components of this equation are shown in Figure 8.3 where the original function is shown with dashed lines and the four scaled and translated functions are shown using solid lines.

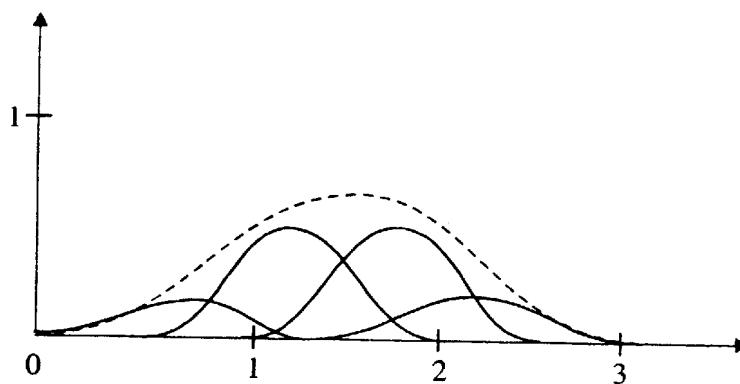


FIGURE 8.3 Construction of B-spline of order 3 from its scaled version.

The scaling coefficients are $1/4(1, 3, 3, 1)$. Note, again that $(1, 3, 3, 1)$ are binomial coefficients [convolve $(1, 2, 1)$ with $(1, 1)$ to get $(1, 3, 3, 1)$]. Now, we can generalize the scaling relation for the spline $N_k(t)$,

$$N_k(t) = \sum_{i=0}^k p_i N_k(2t - i)$$

where

$$p_i = \frac{1}{2^{k-1}} \binom{k}{i} \quad (8.22)$$

We are now ready for designing spline based biorthogonal wavelet system.

8.6.2 B-spline Biorthogonal Wavelet System or Cohen-Daubechies-Feauveau Wavelets (CDF)

We have understood that the splines of any order follow scaling relation. But their integer translates (Except Haar) are not orthogonal. Therefore, we try to find a dual scaling function. Let

$$\phi(t) = N_2(t) \quad \text{and} \quad V_0 \equiv \text{span} \left\{ \sum_k N_2(t - k) \right\}$$

Let us try to find a matching $\tilde{\phi}(t)$ whose scaling relation have five continuous non-zero coefficients. We say support of $\tilde{\phi}(t)$ is 5. We could have chosen any positive odd number (other

than 3, the support of its dual, that is $\phi(t)$. Why this restriction? This restriction follows from the requirement that $\int \phi(t) \tilde{\phi}(t - k) dt = \delta_{k,0}$. We will explore it further.

For the spline scaling function, $N_2(t)$, scaling function coefficients are $1/2, 1, 1/2$. However, these are not normalized. How do we find normalized scaling function coefficients? We know that the sum of normalized coefficients is $\sqrt{2}$.

$$\therefore a \left(\frac{1}{2} + 1 + \frac{1}{2} \right) = \sqrt{2} \Rightarrow a = \frac{1}{\sqrt{2}}$$

Therefore, normalized coefficients of $\phi(t) = N_2(t)$ are: $\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}$.

Let $\tilde{h}(0), \tilde{h}(1), \tilde{h}(2), \tilde{h}(3), \tilde{h}(4)$ be the normalized scaling function coefficients of $\tilde{\phi}(t)$. Since $\int \phi(t) \tilde{\phi}(t - k) dt = \delta_{k,0}$, we have

$$\sum_n h(n) \tilde{h}(n - 2k) = \delta_{k,0} \quad (8.23)$$

This is most important relation that decides $\tilde{\phi}(t)$.

In the orthogonal case, we have

$$\sum_n h(n) h(n - 2k) = \delta_{k,0}$$

i.e., $h(n)$ is orthogonal to even translates of itself. Here \tilde{h} is orthogonal to h , thus the name biorthogonal. Equation (8.23) is the key to the understanding of the biorthogonal wavelets. Let us assume $\tilde{h}(n)$ is non-zero when $\tilde{N}_1 \leq n \leq \tilde{N}_2$ and $h(n)$ is non-zero when $N_1 \leq n \leq N_2$. Equation (8.23) implies that [1]

$$N_2 - \tilde{N}_1 = 2k + 1 \text{ and } \tilde{N}_2 - N_1 = 2\tilde{k} + 1, k, \tilde{k} \in \mathbb{Z} \quad (8.24)$$

In the orthogonal case, this reduces to the well known fact that the length of h has to be even. Equation (8.23) also implies that the difference between the lengths of \tilde{h} and h must be even. Thus, their length must be both even or both odd. Now, try to visualize Eqs. (8.23) and (8.24).

The positioning of scaling function coefficients is very important. This decides the constraints that we are going to put on coefficients. If Eq. (8.24) is not satisfied, we won't get a feasible solution for our $\tilde{h}(k)$ coefficients. Many applications require symmetric scaling function coefficients. So let us put this constraint first. For symmetry, we need $\tilde{h}(0) = \tilde{h}(4)$ and $\tilde{h}(1) = \tilde{h}(3)$. So the Figure 8.4 can be redrawn as depicted in Figure 8.5.

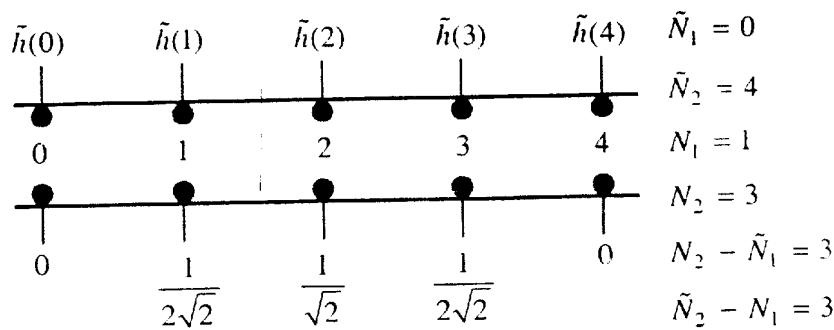


FIGURE 8.4 Relative positioning of scaling filter coefficients.

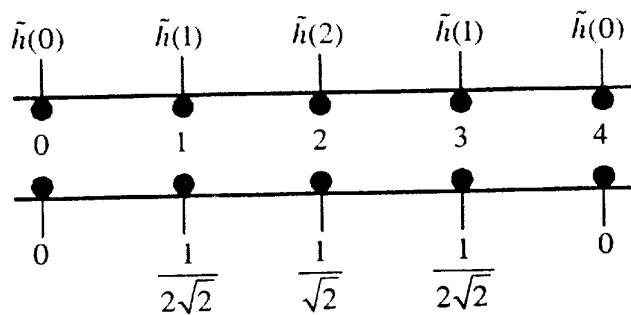


FIGURE 8.5 Positioning of scaling filter coefficients.

Now, apply conditions on coefficients. For $\int \tilde{\phi}(t) dt = 1$, we have the condition

$$\sum_k \tilde{h}(k) = \sqrt{2} \quad (8.25)$$

This implies $2\tilde{h}(0) + 2\tilde{h}(1) + \tilde{h}(2) = \sqrt{2}$

For $k = 0$, Eq. (8.23) gives

$$\frac{1}{2\sqrt{2}} \tilde{h}(1) + \frac{1}{\sqrt{2}} \tilde{h}(2) + \frac{1}{2\sqrt{2}} \tilde{h}(1) = 1 \Rightarrow \frac{1}{\sqrt{2}} \tilde{h}(1) + \tilde{h}(2) = 1 \quad (8.26)$$

(That is multiply the corresponding terms in the two rows of Figure 8.5.)

For $k = 1$, h coefficients shift by 2. This can be visualized as illustrated in Figure 8.6.

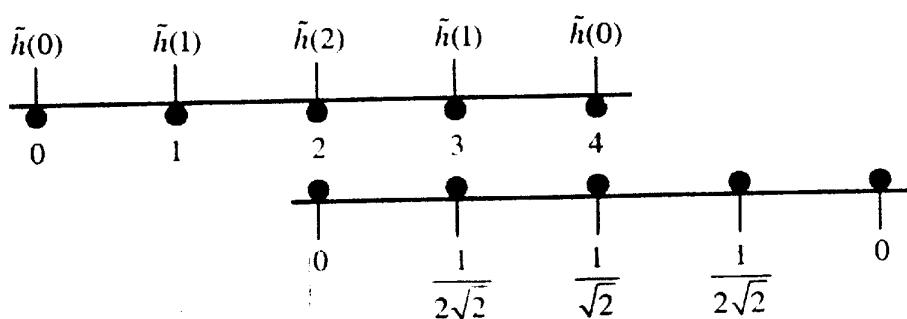


FIGURE 8.6 Positioning of scaling filter coefficients.

For $k = 1$, Eq. (8.23) gives us

$$\frac{1}{2\sqrt{2}} \tilde{h}(1) + \frac{1}{\sqrt{2}} \tilde{h}(0) = 0 \Rightarrow \frac{1}{2} \tilde{h}(1) + \tilde{h}(0) = 0 \quad (8.27)$$

$k = -1$ will also give the same condition as given in Eq. (8.27), since we assumed $\tilde{h}(k)$ are symmetric. No more condition can be derived from Eq. (8.23).

Note that we have only three unknowns and we already have three equations. But it can be easily verified that only two are independent. We need one more equation to get unique solution.

Since $\tilde{\phi}(t) \perp \psi(t)$, we have

$$\psi(t) = \sum_{k=0}^{N-1} (-1)^k \tilde{h}(N-k-1) \phi(2t-k)$$

$\psi(t)$ function depends on $\tilde{h}(k)$ and we require $\psi(t)$ such that $\int \psi(t) dt = 0$. This requires that

$\sum_{k=0}^{N-1} (-1)^k \tilde{h}(N-k-1) = 0$ which is our vanishing moment condition. Thus, we have

$$\tilde{h}(4) - \tilde{h}(3) + \tilde{h}(2) - \tilde{h}(1) + \tilde{h}(0) = 0 \Rightarrow 2\tilde{h}(0) - 2\tilde{h}(1) + \tilde{h}(2) = 0 \quad (8.28)$$

Let us put together all conditions.

1. **Normality:** $2\tilde{h}(0) + 2\tilde{h}(1) + \tilde{h}(2) = \sqrt{2}$

2. **Biorthogonality:** For $k = 0$, $\frac{1}{\sqrt{2}} \tilde{h}(1) + \tilde{h}(2) = 1$

For $k = 1$, $\frac{1}{2} \tilde{h}(1) + \tilde{h}(0) = 0$

3. **Vanishing moment:** $2\tilde{h}(0) - 2\tilde{h}(1) + \tilde{h}(2) = 0$

Solving the above four equations, we obtain

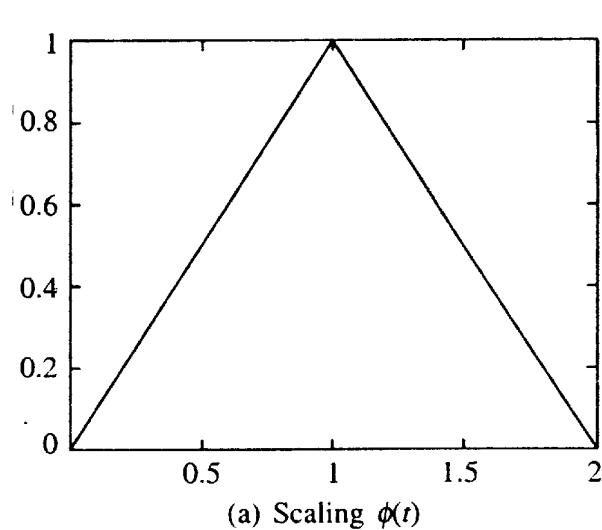
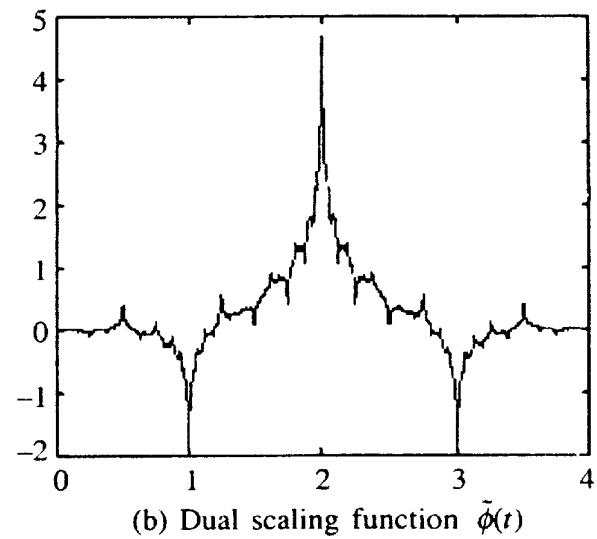
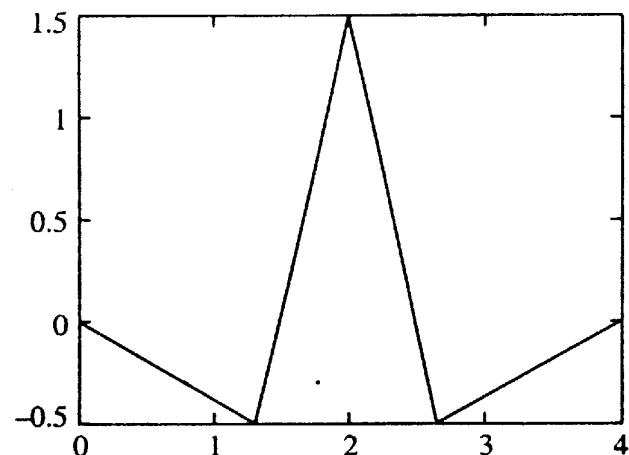
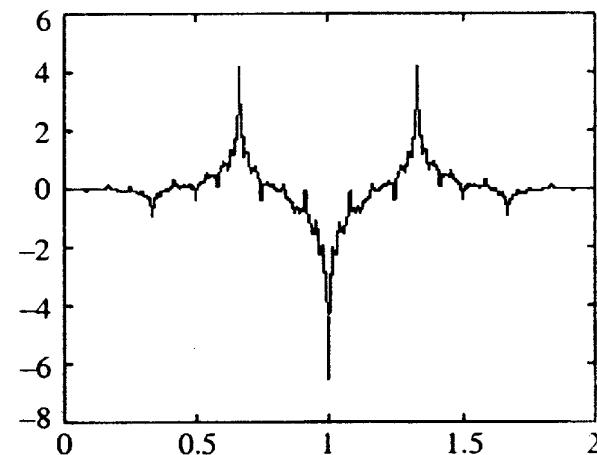
$$\tilde{h}(0) = \frac{-\sqrt{2}}{8}, \quad \tilde{h}(1) = \frac{\sqrt{2}}{4}, \quad \tilde{h}(2) = \frac{3\sqrt{2}}{4}$$

By symmetry,

$$\tilde{h}(3) = \frac{\sqrt{2}}{4}, \quad \tilde{h}(4) = \frac{-\sqrt{2}}{8}$$

These equations can be easily solved using Excel spreadsheet solver.

Figure 8.7 shows the corresponding scaling function, wavelet function and their duals. Table 8.1 shows the complete set of filters. Note the relative positioning of filter coefficients. From the table it can be easily seen that, filter h and \tilde{g} are double-shift orthogonal. Similarly \tilde{h} and g are double-shift orthogonal.

(a) Scaling $\phi(t)$ (b) Dual scaling function $\tilde{\phi}(t)$ (c) Wavelet function $\psi(t)$ (d) Dual wavelet function $\tilde{\psi}(t)$ **FIGURE 8.7** Biorthogonal scaling and wavelet functions.**TABLE 8.1** Biorthogonal Spline Wavelet Filter Coefficients

k	z^2	z^1	z^0	z^{-1}	z^{-2}	z^{-3}
\tilde{h}	$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$	
\tilde{g}			$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{-1}{4}$	
h		$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{4}$		
g		$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$

SUMMARY

As we know, a basis that spans a space does not have to be orthogonal. In order to gain greater flexibility in the construction of wavelet bases, we resorted to relaxing the orthogonality condition and allowed non-orthogonal wavelet bases. For example, it is well-known that the Haar wavelet is the only known wavelet that is compactly supported, orthogonal and symmetric. In many applications, the symmetry of the filter coefficients is often desirable since it results in linear phase of the transfer function. So in order to construct more families of compactly supported, symmetric wavelets in this chapter, we did forego the requirement of orthogonality and, in particular, we introduced the so-called **biorthogonal wavelets**. Then we derived expression for biorthogonal analysis and synthesis. In the last section we showed how biorthogonal wavelets can be constructed from B-spline wavelet bases. More about the construction of biorthogonal wavelet will be explained in Chapter 10.

EXERCISES

- 8.1** Suppose that h, \tilde{h} define two low pass filters in a biorthogonal system.
- Show that $\bar{H}(\omega)\tilde{H}(w) + \bar{H}(\omega + \pi)\tilde{H}(w + \pi) = 2$
 - Prove that $h_{\text{new}}(n) = \frac{1}{2}[h(n) + h(n - 1)]$, $\tilde{h}_{\text{new}}(n) = \frac{1}{2}[\tilde{h}(n) + \tilde{h}(n - 1)]$ defines a new pair of biorthogonal system.
- 8.2** The Deslauries-Dubec filters are $H(\omega)$ and $\tilde{H}(\omega) = \frac{1}{16}(-e^{-3j\omega} + 9e^{-j\omega} + 16 + 9e^{j\omega} - e^{3j\omega})$. Compute $h_{\text{new}}(n)$ and $\tilde{h}_{\text{new}}(n)$ as well as corresponding wavelets ψ_{new} and $\tilde{\psi}_{\text{new}}$.
- 8.3** Prove that the hat function is not orthogonal to its integer translates.
- 8.4** Prove that in a biorthogonal system with h, \tilde{h} as low pass filters,

$$\sum_k h(k) = \sum_k \tilde{h}(k) = \sqrt{2}$$

and

$$\sum_k \tilde{h}(k)h(k - 2l) = \delta_l$$

- 8.5** How many vanishing moments have the primary and scaling functions in the biorthogonal coiflet family (In this family scaling functions also have vanishing moments) defined by

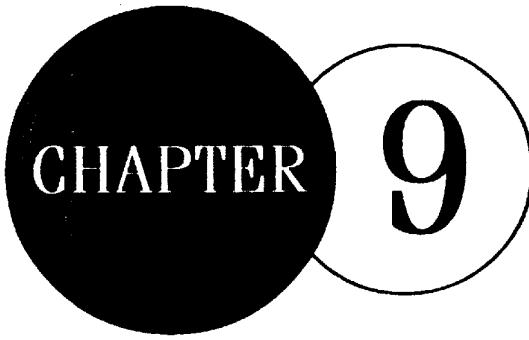
$$h \equiv \left\{ \frac{1}{16\sqrt{2}} (-1, 0, 9, 16, 9, 0, -1) \right\}$$

and

$$\tilde{h} \equiv \left\{ \frac{1}{256\sqrt{2}} (-1, 0, 18, -16, -63, 144, 348, 144, -63, -16, 18, 0, -1) \right\}$$

REFERENCES

- [1] Cohen, A., I. Daubechies, and J. Feauveau, Biorthogonal bases of compactly supported wavelets, *Comm. Pure Appl. Math.*, 45, 485–560, 1992.
- [2] Burrus, C.S., Ramesh A. Gopinath, and Haitao Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, Englewood Cliffs, 1997.



Designing Wavelets—Frequency Domain Approach

INTRODUCTION

In the previous chapter we determined some basic properties of the recursion coefficients $\{h(k)\}$ that must have to make the decomposition/reconstruction work. We will repeat those conditions here and also rephrase them in terms of $H(\omega)$ and $h(z)$. Along the way, we will answer the question of how to find the $\{g(k)\}$ from $\{h(k)\}$. One such possibility was given in Chapter 4 but now we can give the complete answer. Then we will discuss what other desirable properties we may want our wavelets to have and how they are expressed in terms of $\{h(k)\}$, $H(\omega)$ and $h(z)$. Together, this will allow us to write down the set of equations we need to solve to find wavelets with particular properties. We will actually solve these equations for one important special case.

9.1 BASIC PROPERTIES OF FILTER COEFFICIENTS

In terms of $\{h(k)\}$

Let us review what we have learnt in Chapter 4. We determined that the following conditions are necessary for $\{h(k)\}$ and $\{g(k)\}$ to produce a scaling function and wavelet:

$$\sum_k h(k) = \sqrt{2} \quad (9.1a)$$

$$\sum_k h(k) \bar{h}(k + 2j) = \delta_{0j} \quad \text{for all } j \in \mathbb{Z} \quad (9.1b)$$

$$\sum_k h(2k) = \sum_k h(2k+1) = 1/\sqrt{2} \quad (9.1c)$$

$$\sum_k g(k) = 0 \quad (9.1d)$$

$$\sum_k g(k) \bar{g}(k+2j) = \delta_{0j} \quad \text{for all } j \in \mathbb{Z} \quad (9.1e)$$

$$\sum_k h(k) \bar{g}(k+2j) = 0 \quad \text{for all } j \in \mathbb{Z} \quad (9.1f)$$

Relations (9.1a) and (9.1b) imply (9.1c). Condition (9.1a) must be satisfied to make the recursion relation work. Condition (9.1b) says that the integer translates of ϕ should be orthonormal (alternately filter coefficients must be double-shift orthogonal). Equation (9.1c) says that the first moment of $\psi(t)$ [or area under $\psi(t)$] must be zero. Equation (9.1f) must be satisfied for orthogonality of $\phi(t)$ and $\psi(t)$ and their translates. We have assumed that the coefficients can be complex. That is why conditions (9.1b), (9.1e) and (9.1f) contain complex conjugates.

In terms of $H(\omega)$

Recall that $H(\omega)$ is the Fourier transform of $\{h(k)\}$. Let $G(\omega)$ be the Fourier transform of $\{g(k)\}$. Thus, we have

$$H(\omega) = \frac{1}{\sqrt{2}} \sum_k h(k) e^{-ik\omega}$$

$$G(\omega) = \frac{1}{\sqrt{2}} \sum_k g(k) e^{-ik\omega}$$

The conditions in terms of H and G are:

$$H(0) = 1 \quad (9.2a)$$

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1 \quad (9.2b)$$

$$H(\pi) = 0 \quad (9.2c)$$

$$G(0) = 0 \quad (9.2d)$$

$$|G(\omega)|^2 + |G(\omega + \pi)|^2 = 1 \quad (9.2e)$$

$$H(\omega) \overline{G(\omega)} + H(\omega + \pi) \overline{G(\omega + \pi)} = 0 \quad (9.2f)$$

In terms of $H_z(z)$

Let $H_z(z)$ and $G_z(z)$ represent, Z-transform of $\{h(k)\}$ and $\{g(k)\}$, respectively. The subscript z is to distinguish between Fourier and Z-transform. Instead of deriving everything from scratch, we just translate it from $H(\omega)$ using the Table 9.1.

TABLE 9.1 Relationship Between $H_z(z)$ and $H(\omega)$

$H(\omega)$	$H_z(z)$
$e^{-i\omega}$	z
$\omega = 0$	$z = 1$
$\omega = \pi$	$z = -1$
$e^{i\omega}$	$1/z$
$e^{-i(\omega+\pi)}$	$-z$
$e^{i(\omega+\pi)}$	$-1/z$
$H(\omega) = \frac{1}{\sqrt{2}} \sum_k h(k) e^{-ik\omega}$	$H_z(z) = \frac{1}{\sqrt{2}} \sum_k h(k) z^k$
$\overline{H(\omega)} = \frac{1}{\sqrt{2}} \sum_k \bar{h}(k) e^{ik\omega}$	$\overline{H_z(z)} = \frac{1}{\sqrt{2}} \sum_k \bar{h}(k) z^{-k}$
$H(\omega + \pi) = \frac{1}{\sqrt{2}} \sum_k (-1)^k h(k) e^{-ik\omega}$	$H_z(-z) = \frac{1}{\sqrt{2}} \sum_k (-1)^k h(k) z^k$
$\overline{H(\omega + \pi)} = \frac{1}{\sqrt{2}} \sum_k (-1)^k \bar{h}(k) e^{ik\omega}$	$\overline{H_z(-z)} = \frac{1}{\sqrt{2}} \sum_k (-1)^k \bar{h}(k) z^{-k}$

The function \bar{H}_z is in general not the complex conjugate of H_z . It is as defined in the above table or equivalently

$$\overline{H_z(z)} = \overline{H_z(1/\bar{z})}$$

Note however, that many authors use only z with $|z| = 1$, in which case \bar{H}_z is complex conjugate of H_z .

The conditions in terms of H_z and G_z are:

$$H_z(1) = 1 \quad (9.3a)$$

$$H_z(z) \overline{H_z(z)} + H_z(-z) \overline{H_z(-z)} = 1 \quad (9.3b)$$

$$H_z(-1) = 0 \quad (9.3c)$$

$$G_z(1) = 0 \quad (9.3d)$$

$$G_z(z) \overline{G_z(z)} + G_z(-z) \overline{H_z(-z)} = 1 \quad (9.3e)$$

$$H_z(z) \overline{G_z(z)} + H_z(-z) \overline{G_z(-z)} = 0 \quad (9.3f)$$

9.2 CHOICE OF WAVELET FUNCTION COEFFICIENTS $\{g(k)\}$

We can now answer the following questions: Assuming that the $\{g(k)\}$ satisfy (a), (b) [and, therefore, automatically (c)], what choice of $\{g(k)\}$ will make the remaining conditions valid?

In terms of $H(\omega)$

Define the matrix

$$M(\omega) = \begin{bmatrix} H(\omega) & H(\omega + \pi) \\ G(\omega) & G(\omega + \pi) \end{bmatrix} \quad (9.4)$$

and its complex conjugate transpose

$$M^*(\omega) = \begin{bmatrix} \overline{H(\omega)} & \overline{G(\omega)} \\ \overline{H(\omega + \pi)} & \overline{G(\omega + \pi)} \end{bmatrix} \quad (9.5)$$

Conditions (9.2b), (9.2e) and (9.2f) are equivalent to

$$M(\omega) M^*(\omega) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (9.6)$$

so $M^* = (\overline{M})^T$ is the inverse of M . (A matrix which satisfy $M^* = (\overline{M})^T = M^{-1}$ is called **unitary**.) One consequence of this is:

$$M^*(\omega) M(\omega) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

which leads to a few more relations, for example,

$$|H(\omega)|^2 + |G(\omega)|^2 = 1 \quad (9.7)$$

A more important consequence is that we can now get direct relations between H and G , using the inversion formula for 2×2 matrices.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{(ad - bc)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad (9.8)$$

Define the determinant of $M(\omega)$ as:

$$\Delta(\omega) = H(\omega) G(\omega + \pi) - G(\omega) H(\omega + \pi) \quad (9.9)$$

Then

$$M^{-1}(\omega) = \frac{1}{\Delta(\omega)} \begin{bmatrix} G(\omega + \pi) & -H(\omega + \pi) \\ -G(\omega) & H(\omega) \end{bmatrix} = \begin{bmatrix} \overline{H(\omega)} & \overline{G(\omega)} \\ \overline{H(\omega + \pi)} & \overline{G(\omega + \pi)} \end{bmatrix} = M^*(\omega) \quad (9.10)$$

Comparing the lower left corner, we have

$$G(\omega) = -\Delta(\omega) \overline{H(\omega + \pi)} \quad (9.11)$$

Comparing the top left corner, we have

$$G(\omega + \pi) = \Delta(\omega) \overline{H(\omega)} \Rightarrow G(\omega) = \Delta(\omega + \pi) \overline{H(\omega + \pi)} \quad (9.12)$$

Therefore, we must have

$$\Delta(\omega + \pi) = -\Delta(\omega) \quad (9.13)$$

Also, the determinant of any unitary matrix must have absolute value 1. One can check directly that these conditions on $\Delta(\omega)$ are also sufficient. If $H(\omega)$ has been found then suitable $G(\omega)$ are of the form:

$$G(\omega) = -\Delta(\omega) \overline{H(\omega + \pi)} \quad (9.14)$$

where $\Delta(\omega)$ is any 2π periodic function with

$$\Delta(\omega + \pi) = -\Delta(\omega) \quad (9.15)$$

$$|\Delta(\omega)| = 1 \text{ (for all } \omega) \quad (9.16)$$

Conversely, every such $\Delta(\omega)$ will produce a suitable $G(\omega)$.

As a special case, if $\{h(k)\}$, $\{g(k)\}$ are finite sequences, $\Delta(\omega)$ must be of the form:

$$\Delta(\omega) = \alpha e^{-j(N-1)\omega} \quad (9.17)$$

for some even integer N , $j = \sqrt{-1}$ and $|\alpha| = 1$

This leads to

$$g(k) = \alpha(-1)^k \bar{h}(N-1-k) \quad (9.18)$$

If $\{h(k)\}$, $\{g(k)\}$ are real then $\alpha = \pm 1$, which leads to

$$g(k) = \alpha(-1)^k h(N-1-k) \quad (9.19)$$

The choice of $\Delta(\omega) = e^{-j\omega}$ leads to $g(k) = (-1)^k h(1-k)$, which is what some authors prefer. If $\{h(k)\} = \{h(0), h(1), \dots, h(L)\}$, where $L = N - 1$, the more usual choice is $\Delta(\omega) = e^{-jL\omega}$ which leads to

$$g(k) = (-1)^k h(L-k) = (-1)^k h(N-1-k) \quad (9.20)$$

Derivation in terms of $H_z(z)$

We copy the approach from $H(\omega)$ and define

$$M_z(z) = \begin{bmatrix} H_z(z) & H_z(-z) \\ G_z(z) & G_z(-z) \end{bmatrix} \quad (9.21)$$

Then

$$M_z^{-1}(z) = \begin{bmatrix} \overline{H_z(z)} & \overline{G_z(z)} \\ \overline{H_z(-z)} & \overline{G_z(-z)} \end{bmatrix} \quad (9.22)$$

The determinant of $M_z(z)$ is:

$$\Delta(z) = H_z(z) G_z(-z) - H_z(-z) G_z(z) \quad (9.23)$$

For comparing the inversion formula for $M_z(z)$ with $M_z^{-1}(z)$, we define as before

$$G_z(z) = -\Delta(z) \overline{H_z(-z)} \quad (9.24)$$

$$\Delta(z) \overline{\Delta(z)} = 1 \quad (9.25)$$

If $H_z(z)$ has been found then all suitable $G_z(z)$ are of the form:

$$G_z(z) = -\Delta(z) \overline{H_z(-z)} \quad (9.26)$$

where $\Delta(z)$ is any function with

$$\Delta(-z) = -\Delta(z) \quad (9.27)$$

$$\Delta(z) \overline{\Delta(z)} = 1 \quad (9.28)$$

Conversely, every such $\Delta(z)$ will produce a suitable $G_z(z)$.

As a special case, if $\{h(k)\}$, $\{g(k)\}$ are finite sequences, $H_z(z)$, $G_z(z)$ and $\Delta(z)$ are polynomials. $\Delta(z)$ is an odd polynomial and is never zero, except possibly at $z = 0$ (because $M_z^{-1}(z)$ is defined everywhere except possibly at $z = 0$), so $\Delta(z)$ must of the form $\Delta(z) = \alpha z^{N-1}$ where N is an even integer and $|\alpha| = 1$ (since $\Delta(z) \overline{\Delta(z)} = |\alpha|^2 = 1$).

If $\{h(k)\}$, $\{g(k)\}$ are real then $\alpha = \pm 1$, which again leads to

$$g(k) = \pm(-1)^k h(N - 1 - k)$$

The choice of $\Delta(z) = z$ leads to $g(k) = (-1)^k h(1 - k)$, the choice $\Delta(z) = z^L$ leads to

$$g(k) = (-1)^k h(L - k) \text{ where } L \text{ is an odd integer}$$

or

$$g(k) = (-1)^k h(N - 1 - k) \text{ where } N \text{ is an even integer}$$

9.3 VANISHING MOMENT CONDITIONS IN FOURIER DOMAIN

Let Fourier transform of $\phi(t)$ and $\psi(t)$ be respectively $\hat{\phi}(\omega)$ and $\hat{\psi}(\omega)$. If $\psi(t)$ has M vanishing moments then

$$\hat{\psi}(0) = \hat{\psi}'(0) = \dots = \hat{\psi}^{(M-1)}(0) = 0 \quad (9.29)$$

Next, we differentiate the relation $\hat{\psi}(\omega) = G(\omega/2) \hat{\phi}(\omega/2)$ repeatedly to get

$$\hat{\psi}'(\omega) = \frac{1}{2} [G'(\omega/2) \hat{\phi}(\omega/2) + G(\omega/2) \hat{\phi}'(\omega/2)] \quad (9.30)$$

$$\hat{\psi}''(\omega) = \frac{1}{4} [G''(\omega/2) \hat{\phi}(\omega/2) + 2G'(\omega/2) \hat{\phi}'(\omega/2) + G(\omega/2) \hat{\phi}''(\omega/2)] \quad (9.31)$$

$$\begin{aligned} \hat{\psi}'''(\omega) &= \frac{1}{8} \left[G''' \left(\frac{\omega}{2} \right) \hat{\phi} \left(\frac{\omega}{2} \right) + 3G'' \left(\frac{\omega}{2} \right) \hat{\phi}' \left(\frac{\omega}{2} \right) \right. \\ &\quad \left. + 3G' \left(\frac{\omega}{2} \right) \hat{\phi}'' \left(\frac{\omega}{2} \right) + G \left(\frac{\omega}{2} \right) \hat{\phi}''' \left(\frac{\omega}{2} \right) \right] \end{aligned} \quad (9.32)$$

Now, we evaluate for $\omega = 0$

$$\hat{\psi}(0) = G(0) \hat{\phi}(0)$$

$$\hat{\psi}'(0) = \frac{1}{2} [G'(0) \hat{\phi}(0) + G(0) \hat{\phi}'(0)]$$

$$\hat{\psi}''(0) = \frac{1}{4} [G''(0) \hat{\phi}(0) + 2G'(0) \hat{\phi}'(0) + G(0) \hat{\phi}''(0)]$$

$$\hat{\psi}'''(0) = \frac{1}{8} [G'''(0) \hat{\phi}(0) + 3G''(0) \hat{\phi}'(0) + 3G'(0) \hat{\phi}''(0) + G(0) \hat{\phi}'''(0)]$$

We know that $\hat{\phi}(0) \neq 0$. (Since average value of $\phi(t)$ is not zero). This means that, for $\hat{\psi}(0)$ to be equal to 0, $G(0)$ must be equal to zero.

Again $\hat{\phi}(0) \neq 0$, $G(0) = 0$ and $\hat{\psi}'(0) = 0$ imply that $G'(0) = 0$.

Similarly, $\hat{\phi}(0) \neq 0$, $G(0) = 0$, $G'(0) = 0$ and $\hat{\psi}''(0) = 0$ imply that $G''(0) = 0$ and so on. We thus deduce that G and its first $M - 1$ derivatives must vanish origin.

We want a condition on H , so we use the basic relationship $G(\omega) = -\Delta(\omega) \overline{H(\omega + \pi)}$ to deduct that M vanishing moments are equivalent to

$$H(\pi) = H'(\pi) = \dots = H^{(M-1)}(\pi) = 0 \quad (9.33)$$

The theory of complex functions states that this is equivalent to

$$H(\omega) = \left(\frac{1 + e^{-j\omega}}{2} \right)^M L(\omega) \quad (9.34)$$

where L is a trigonometric polynomial.

9.4 DERIVATION OF DAUBECHIES WAVELETS

Following steps are involved in derivation of daubechies wavelets:

Step 1 Fix M , the number of vanishing moments. We write down the equations we need to satisfy, using $H(\omega)$ approach:

$$H(\omega) = \left(\frac{1 + e^{-j\omega}}{2} \right)^M L(\omega) \quad (9.35)$$

$$H(0) = 1 \quad (9.36)$$

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1 \quad (9.37)$$

Step 2 Convert second (9.36) and third (9.37) conditions on to a condition on L :

$$\left[\cos^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega)|^2 + \left[\sin^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega + \pi)|^2 = 1 \quad (9.38)$$

Step 3 Show that

$$|L(\omega)|^2 = P \left[\sin^2 \left(\frac{\omega}{2} \right) \right] \quad (9.39)$$

$$|L(\omega + \pi)|^2 = P \left[\cos^2 \left(\frac{\omega}{2} \right) \right] \quad (9.40)$$

for some polynomial P .

Step 4 Convert the equation for L into an equation for P .

$$(1 - y)^M P(y) + y^M P(1 - y) = 1 \quad (9.41)$$

Step 5 Find the polynomial P of the lowest possible degree that solves the equation and convert it back to $|L(\omega)|^2$.

Step 6 Factor $|L(\omega)|^2$ to find $L(\omega)$, and put everything back together.

Example

Step 1

$$\begin{aligned} |H(\omega)|^2 &= H(\omega) \overline{H(\omega)} = \left(\frac{1 + e^{-j\omega}}{2} \right)^M L(\omega) \left(\frac{1 + e^{j\omega}}{2} \right)^M \overline{L(\omega)} \\ &= \left(\frac{1 + e^{-j\omega}}{2} \cdot \frac{1 + e^{j\omega}}{2} \right)^M |L(\omega)|^2 = \left[\cos^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega)|^2 \end{aligned}$$

$$\text{Likewise } |H(\omega + \pi)|^2 = \left[\cos^2 \left(\frac{\omega + \pi}{2} \right) \right]^M |L(\omega + \pi)|^2 = \left[\sin^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega + \pi)|^2$$

Putting these together, we obtain

$$\left[\cos^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega)|^2 + \left[\sin^2 \left(\frac{\omega}{2} \right) \right]^M |L(\omega + \pi)|^2 = 1$$

Step 2 Note that since $\cos(0) = 1$ and $\sin(0) = 0$, we automatically get that $|L(0)| = 1$. By introducing a scale factor of absolute value 1, if necessary, we can assume that $L(0) = 1$, which takes care of second condition in step 1.

Step 3 Here is the point where we need the fact that the coefficients are real. Since

$$L(\omega) = \sum_k l_k e^{-jk\omega}$$

$$\overline{L(\omega)} = \sum_k l_k e^{jk\omega}$$

We get

$$|L(\omega)|^2 = L(\omega) \overline{L(\omega)} = \sum_{k,i} l_k l_i e^{-j(k-i)\omega}$$

We take out the term with $i = k$ and group together the remaining l_k and l_i terms:

$$\begin{aligned} |L(\omega)|^2 &= \sum_k l_k^2 + \sum_{i>k} l_k l_i [e^{j(i-k)\omega} + e^{-j(i-k)\omega}] \\ &= \sum_k l_k^2 + 2 \sum_{i>k} l_k l_i \cos(i-k)\omega = \sum_m \alpha_m \cos m\omega, \text{ where } \alpha_0 = \sum_k l_k^2 \end{aligned}$$

Using the trigonometric identities,

$$\cos 2\omega = 2 \cos^2 \omega - 1$$

$$\cos 3\omega = 4 \cos^3 \omega - 3 \cos \omega$$

and so on. We convert this sum to powers of $\cos \omega$.

$$|L(\omega)|^2 = \sum_m \alpha_m \cos m\omega = \sum_k \beta_k \cos^k \omega$$

then we use the substitution $\cos \omega = 1 - 2 \sin^2(\omega/2)$ to get

$$|L(\omega)|^2 = \sum_k \gamma_k \left[\sin^2\left(\frac{\omega}{2}\right) \right]^k = P \left[\sin^2\left(\frac{\omega}{2}\right) \right]$$

Now,

$$|L(\omega + \pi)|^2 = P \left[\sin^2\left(\frac{\omega + \pi}{2}\right) \right] = P \left[\cos^2\left(\frac{\omega}{2}\right) \right]$$

Step 4 We make use of the substitution

$$y = \sin^2\left(\frac{\omega}{2}\right)$$

into the equation

$$\left[\cos^2\left(\frac{\omega}{2}\right) \right]^M |L(\omega)|^2 + \left[\sin^2\left(\frac{\omega}{2}\right) \right]^M |L(\omega + \pi)|^2 = 1$$

We get

$$(1 - y)^M P(y) + y^M P(1 - y) = 1 \quad (9.42)$$

Step 5 Using a theorem due to Bezout, it can be shown that the equation for P has a unique shortest solution of degree $M - 1$. The solution itself can be found by a trick. We write the equation in the form:

$$P(y) = (1 - y)^{-M} - y^M (1 - y)^{-M} P(1 - y) \quad (9.43)$$

This must be valid for all y . For $|y| < 1$, we can expand $(1 - y)^{-M}$ into a power series. Since P is of degree $M - 1$ and the second term on the right hand side starts with a power of y^M , the shortest $P(y)$ must be equal to first M terms in the power series for $(1 - y)^{-M}$.

or

$$P(y) = \sum_{k=0}^{M-1} \binom{M+k-1}{k} y^k \quad (9.44)$$

One thing we did not mention before is that we also must have $P(y) \geq 0$ for $y \in [0, 1]$, since P is the square of L and $y = \sin^2(\omega/2)$ takes on values in $[0, 1]$. By a lucky coincidence, this is satisfied here for all value of M .

Maybe this is a good point to talk about the number of coefficients in these trig polynomials and regular polynomials. Suppose $L(\omega)$ has degree K . Then $|L(\omega)|^2$, written as a sum of exponentials, has terms going from $-K$ to K . We collapse the exponentials into cosines by grouping positive and negative powers together then we are back to summing from 0 to K . Nothing changes during the next few steps, so P also has degree K , which we now know to be $M - 1$. This translates into an H of degree $(2M - 1)$ or $2M$ coefficients $\{h(k)\}$. Thus the **Daubechies wavelet with M vanishing moments has $2M$ coefficients**.

Step 6 Now we have P and we have to find our way back to L . This process is called **spectral factorization**. Daubechies used the following method, based on a theorem of Riesz:

Substituting all the substitutions backwards until we have $|L(\omega)|^2$ in terms of complex exponentials again. To make life easier, let $z = e^{-j\omega}$ so $|L(\omega)|^2$ turns into a z^{-M+1} times a polynomial in z of degree $2M - 2$. Find all roots of this polynomial. These roots turn out to come in groups of four, consisting of z_i , $1/z_i$, \bar{z}_i and $1/\bar{z}_i$. If z_i is real or on the unit circle, there are only two roots in a group instead of 4.

We select one pair z_i , $1/\bar{z}_i$ from each group of 4 or one root from each pair. If we multiply the terms $(z - z_k)$ for all k together, we get the original polynomial (except for a constant). If we multiply the terms $(z - z_i)$ only for the selected roots, we get the square root we want, except for a constant (this is the content of the Riesz theorem). We can then replace z by $e^{-j\omega}$ again.

This whole process is quite a *mathematical tour de force*. It can be illustrated by deriving the Daubechies wavelet with one and two vanishing moments. We can start with step 5, the other steps were only necessary for the proof. These two cases can be done by hand. For more vanishing moments, these equations must be solved numerically.

9.4.1 Daubechies Wavelets with One Vanishing Moment

We have $M = 1$

$$\therefore P(y) = 1$$

which translates into $|L(\omega)|^2 = 1$, and obviously $L(\omega) = 1$. This leads to

$$H(\omega) = \left(\frac{1 + e^{-j\omega}}{2} \right)^M \quad L(\omega) = \left(\frac{1 + e^{-j\omega}}{2} \right) 1$$

which are the Haar wavelets.

9.4.2 Daubechies Wavelets with Two Vanishing Moment

We have $M = 2$

$$\therefore P(y) = 1 = 2y$$

Recall that $y = \sin^2(\omega/2)$, plug that in and convert everything back into the complex exponentials:

$$\begin{aligned} |L(\omega)|^2 = P(y) &= 1 + 2y = 1 + 2 \sin^2(\omega/2) = -\frac{e^{-j\omega}}{2} + 2 + \frac{e^{j\omega}}{2} \\ &= z^{-1} \left(-\frac{z^2}{2} + 2z - \frac{1}{2} \right) \end{aligned}$$

The corresponding polynomial is $-\frac{z^2}{2} + 2z - \frac{1}{2}$, which has roots $2 \pm \sqrt{3}$. We choose one of them, like $2 + \sqrt{3}$. The Riesz theorem says that $L(\omega)$ should be

$$L(\omega) = \text{const.} [z - (2 + \sqrt{3})] = \text{const.} [e^{-j\omega} - (2 + \sqrt{3})]$$

The constant can be found by multiplying it back together or from the Riesz theorem. It turns out to be $(\sqrt{3} - 1)/2$.

So,

$$H(\omega) = \left(\frac{1 + e^{-j\omega}}{2} \right)^2 \cdot \frac{\sqrt{3} - 1}{2} \cdot [e^{-j\omega} - (2 + \sqrt{3})] = \sum_{k=0}^3 h(k) e^{-jk\omega}$$

where

$$h(0) = \frac{1 + \sqrt{3}}{4\sqrt{2}}$$

$$h(1) = \frac{3 + \sqrt{3}}{4\sqrt{2}}$$

$$h(2) = \frac{3 - \sqrt{3}}{4\sqrt{2}}$$

$$h(3) = \frac{1 - \sqrt{3}}{4\sqrt{2}}$$

SUMMARY

In this chapter, we derived Daubechies wavelet coefficients employing a frequency domain approach. This in fact is the method used by Daubechies herself. The method uses certain concepts from complex analysis like Riesz theorem to arrive at the solution.

EXERCISES

9.1 If $z = e^{-i\omega}$ then $\left| (z - z_j)(z - \bar{z}_j^{-1}) \right| = |z_j|^{-1} |z - z_j|^2$

(Hint: Start with $(e^{-i\omega} - z_j)(e^{-i\omega} - \bar{z}_j^{-1})$. Factor out $-\bar{z}_j^{-1}e^{-i\omega}$ from the second factor and then take the absolute value.)

- 9.2** Derive Daubechies wavelet system with three vanishing moment (a 6-tap wavelet system).
- 9.3** Write a MATLAB code for generating Daubechies' orthogonal wavelet system coefficients.

REFERENCES

- [1] Daubechies, I. and J. Lagarias, Two-scale difference equations I. Existence and global regularity of solutions, *SIAM J. Math. Anal.*, 22 (5), 1388–1410, 1991.
- [2] Daubechies, I. and J. Lagarias, Two-scale difference equations II. Local regularity, infinite products of matrices and fractals, *SIAM J. Math. Anal.*, 23 (4), 1031–1079, 1992.
- [3] Daubechies, I., *Ten lectures on wavelets*, SIAM, 1992.
- [4] Vidakovic, B., *Statistical Modeling by Wavelets*, Wiley, NY, 1999.
- [5] Strang, G. and T.Q. Nguyen, *Wavelets and Filter Banks*, Revised Edition, Wellesley-Cambridge Press, Wellesley, MA, 1998.



CHAPTER 10

Lifting Scheme

INTRODUCTION

The *lifting scheme* [1 to 6] is a new approach to construct the so-called second generation wavelets, i.e., wavelets which are not necessarily translations and dilations of one function. The latter we refer to as a first generation wavelets or classical wavelets. The lifting scheme has some additional advantages in comparison with the classical wavelets. This transform works for signals of an arbitrary size with correct treatment of the boundaries. Also, all computations can be done in-place. Another feature of the lifting scheme is that all constructions are derived in the spatial domain. This is in contrast to the traditional approach, which relies heavily on the frequency domain. Staying in the spatial domain leads to two major advantages. First, it does not require the machinery of Fourier analysis as a prerequisite. This leads to a more intuitively appealing treatment better suited to those interested in applications, rather than mathematical foundations. Secondly, lifting leads to algorithms that can easily be generalized to complex geometric situations that typically occur in computer graphics. This will lead to second generation wavelets. Moreover, the lifting scheme makes computational time optimal, sometimes increasing the speed of calculations by factor 2. Another important feature of the lifting scheme is that every filter bank based on lifting automatically satisfies perfect reconstruction properties. The lifting scheme starts with a set of well-known filters, thereafter lifting steps are used in an attempt to improve (lift) the properties of a corresponding wavelet decomposition. There are two types of lifting steps: the *primal lifting step* and the *dual lifting step*, which we will explore in detail in the forthcoming sections. A number of such lifting steps (dual and primal ones being interchanged) can be used in order to obtain desired properties of a wavelet transform.

Daubechies and Sweldson [1, 2] further showed that any orthogonal and biorthogonal wavelet-analysis-process can be made into lifting steps by employing factorization of polyphase matrix. We will use this result in the next section to get a feeling of the lifting scheme. The approach and philosophy used in lifting is quite different from conventional approach and the reader who has trodden the path laid out in this book so far may feel sudden change in the way

we look at wavelet transform. To make a smooth transition from the old approach to the new one, we convert the conventional wavelet transform matrix into lifting steps. Further, we move on to interpreting the result (elementary matrices representing lifting) so obtained in a geometrical way.

10.1 WAVELET TRANSFORM USING POLYPHASE MATRIX FACTORIZATION

Let us take a signal of length 8 and choose Daubechies 4-tap wavelet for analysis. Let the notation and their relative position (index) for synthesis filters $h(n)$ and $g(n)$ be as shown in Table 10.1. Analysis filters are $h(-n)$ and $g(-n)$. These filters are to be convolved with original signal and get downsampled.

TABLE 10.1 Four-tap Analysis Filters

k	z^2	z^1	z^0	z^{-1}	z^{-2}	z^{-3}
$h(k)$			h_0	h_1	h_2	h_3
$g(k)$	g_0	g_1	g_2	g_3		

Further, we split signal into two streams s stream and d stream. s corresponds to even indexed signal and d for odd indexed in the original signal. With this new notation, the wavelet transform in matrix form is as given in Eq. (10.1). We have assumed here that the signal is periodic. More about dealing with finite signal is given in a later section.

$$\begin{bmatrix} s_0^{\text{new}} \\ s_1^{\text{new}} \\ s_2^{\text{new}} \\ s_3^{\text{new}} \\ d_0^{\text{new}} \\ d_1^{\text{new}} \\ d_2^{\text{new}} \\ d_3^{\text{new}} \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \\ g_2 & g_3 & 0 & 0 & 0 & 0 & g_0 & g_1 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \end{bmatrix} \begin{bmatrix} s_0 \\ d_0 \\ s_1 \\ d_1 \\ s_2 \\ d_2 \\ s_3 \\ d_3 \end{bmatrix} \quad (10.1)$$

On RHS of Eq. (10.1), multiplication of the first four rows of coefficient matrix with signal give four smooth coefficients and the multiplication of remaining four, give four detail coefficients.

It is clear that we can split this computation into two such that even and odd indexed signal values are separated in the multiplication.

$$\begin{bmatrix} h_0 & h_2 & 0 & 0 \\ 0 & h_0 & h_2 & 0 \\ 0 & 0 & h_0 & h_2 \\ h_0 & 0 & 0 & h_2 \\ g_2 & 0 & 0 & g_0 \\ g_0 & g_2 & 0 & 0 \\ 0 & g_0 & g_2 & 0 \\ 0 & 0 & g_0 & g_2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} h_1 & h_3 & 0 & 0 \\ 0 & h_1 & h_3 & 0 \\ 0 & 0 & h_1 & h_3 \\ h_3 & 0 & 0 & h_1 \\ g_3 & 0 & 0 & g_1 \\ g_1 & g_3 & 0 & 0 \\ 0 & g_1 & g_3 & 0 \\ 0 & 0 & g_1 & g_3 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (10.2)$$

Equation (10.2) can further split into two different matrix multiplication processes such that the first one gives filtered smooth signal and the other one gives detail coefficients.

$$\begin{bmatrix} s_0^{\text{new}} \\ s_1^{\text{new}} \\ s_2^{\text{new}} \\ s_3^{\text{new}} \end{bmatrix} = \begin{bmatrix} h_0 & h_2 & 0 & 0 \\ 0 & h_0 & h_2 & 0 \\ 0 & 0 & h_0 & h_2 \\ h_2 & 0 & 0 & h_0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} h_1 & h_3 & 0 & 0 \\ 0 & h_1 & h_3 & 0 \\ 0 & 0 & h_1 & h_3 \\ h_3 & 0 & 0 & h_1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (10.3)$$

and

$$\begin{bmatrix} d_0^{\text{new}} \\ d_1^{\text{new}} \\ d_2^{\text{new}} \\ d_3^{\text{new}} \end{bmatrix} = \begin{bmatrix} g_2 & 0 & 0 & g_0 \\ g_0 & g_2 & 0 & 0 \\ 0 & g_0 & g_2 & 0 \\ 0 & 0 & g_0 & g_2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} g_3 & 0 & 0 & g_1 \\ g_1 & g_3 & 0 & 0 \\ 0 & g_1 & g_3 & 0 \\ 0 & 0 & g_1 & g_3 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (10.4)$$

Note that Eqs. (10.3) and (10.4) are two convolution sums. We know that convolution sum in time domain is equivalent to multiplication in the transform (viz., Z-transform and Fourier transform) domain. Therefore, Eqs. (10.3) and (10.4) can be put together in matrix form as:

$$\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} h_0 + \frac{h_2}{z} & h_1 + \frac{h_3}{z} \\ zg_0 + g_2 & zg_1 + g_3 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} = P(z) \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} \quad (10.5)$$

$P(z)$ is called **polyphase matrix**. Daubechies and Sweldone [1, 2] could factor polyphase matrix into a few simple matrices with certain special properties. The details of factorization is given in Section 10.4.3. For Daubechies' four-tap wavelet system, the factored matrix are as given in Eq. (10.6).

$$\begin{bmatrix} h_0 + \frac{h_2}{z} & h_1 + \frac{h_3}{z} \\ zg_0 + g_2 & zg_1 + g_3 \end{bmatrix} \equiv \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2-\sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \quad (10.6)$$

So Eq. (10.5) can be rewritten as:

$$\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2+\sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} \quad (10.7)$$

Let us now compute the rightmost two matrices of Eq. (10.2). Suppose the output be represented as:

$$\begin{bmatrix} S^{[1]}(z) \\ D^{[1]}(z) \end{bmatrix}$$

Here the superscript indicates the stage of multiplication.

$$\begin{bmatrix} S^{[1]}(z) \\ D^{[1]}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$$

This is equivalent to the following time-domain relationship:

$$s_l^{[1]} = s_l$$

$$d_l^{[1]} = d_l - \sqrt{3}s_l$$

For the eight data that we have considered, this means

$$s^{[1]} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad \text{and} \quad d^{[1]} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} - \sqrt{3} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Now,

$$\begin{bmatrix} S^{[2]}(z) \\ D^{[2]}(z) \end{bmatrix} = \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} = \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S^{[1]}(z) \\ D^{[1]}(z) \end{bmatrix}$$

This leads to

$$S^{[2]}(z) = S^{[1]}(z) + \frac{\sqrt{3}}{4} D^{[1]}(z) + \frac{\sqrt{3}-2}{4} z^{-1} D^{[1]}(z)$$

$$D^{[2]}(z) = D^{[1]}(z)$$

The reader must remember here that multiplying a sequence by z^{-1} is equivalent to shifting the sequence right. Hence the time domain relation is:

$$s_l^{[2]} = s_l^{[1]} + \frac{\sqrt{3}}{4} d_l^{[1]} + \left(-\frac{1}{2} + \frac{\sqrt{3}}{4} \right) d_{l+1}^{[1]}$$

$$d_l^{[2]} = d_l^{[1]}$$

We can visualize multiplying a sequence $\{a, b, c, d\}$ by z^{-1} (we imagine z^{-1} as an operator that operates on sequences) as:

$$(z^{-1}) \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix}$$

Here, the result is a circular shift. This is true if we assume a periodic signal. More details about dealing the end points of a finite signal are given in Section 10.5. Similarly, we can visualize multiplying a sequence $\{a, b, c, d\}$ by z as:

$$(z) \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix}$$

We move on to next stage. Let

$$\begin{bmatrix} S^{[3]}(z) \\ D^{[3]}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} S^{[2]}(z) \\ D^{[2]}(z) \end{bmatrix}$$

In time domain, the above relation is translated as:

$$s_l^{[3]} = s_l^{[2]}$$

$$d_l^{[3]} = d_l^{[1]} + s_{l-1}^{[2]}$$

Finally,

$$\begin{aligned} \begin{bmatrix} S^{[4]}(z) \\ D^{[4]}(z) \end{bmatrix} &= \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2-\sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2-\sqrt{3}} \end{bmatrix} \begin{bmatrix} S^{[3]}(z) \\ D^{[3]}(z) \end{bmatrix} \end{aligned}$$

$$s_l^{[4]} \leftarrow \sqrt{2 + \sqrt{3}} s_l^{[3]}$$

$$d_l^{[4]} \leftarrow \sqrt{2 - \sqrt{3}} d_l^{[3]}$$

Now, the transform is complete.

Thus, in four stages we obtain low pass and high pass filtered coefficients. In some stages, only one among the series $\{s_k\}$ and $\{d_k\}$ gets updated. Omitting those updates and the superscript symbol used for identifying stages of updating, the whole series of updates can be written as:

$$d_l \leftarrow -\sqrt{3}s_l + d_l$$

$$s_l \leftarrow s_l + \frac{\sqrt{3}}{4}d_l + \frac{\sqrt{3}-2}{4}d_{l+1}$$

$$d_l \leftarrow -s_{l-1} + d_l$$

$$s_l \leftarrow \sqrt{2 - \sqrt{3}}s_l$$

$$d_l \leftarrow \sqrt{2 + \sqrt{3}}d_l$$

The equation involves only multiplication and addition of $\{s_k\}$ and $\{d_k\}$ arrays and computation is found to be faster than conventional ‘convolution and downsampling’ operation.

The **steps** that effect changes in value of $\{d_k\}$ are called **prediction** steps and the one that effect change in value of $\{s_k\}$ are called **update** steps. The reason for calling these steps as ‘prediction’ and ‘update’ will be explained in Section 10.2.

10.1.1 Inverse Lifting

One great advantage of the lifting scheme is that the inverse discrete wavelet transform can be trivially implemented. This is possible because of the special nature of the matrices resulting from the factorization of the polyphase matrix. All constituent matrices of $P(z)$ are 2×2 and they are either diagonal or triangular (upper or lower). Wherever the matrix is triangular, the main diagonal elements are 1s. Since

$$\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} c & 0 \\ 0 & d \end{bmatrix}^{-1} = \begin{bmatrix} 1/c & 0 \\ 0 & 1/d \end{bmatrix}$$

original data $\begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$ can easily obtained from the equation for $\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix}$

Inverting the following forward wavelet transform equation:

$$\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} \sqrt{2 + \sqrt{3}} & 0 \\ 0 & \sqrt{2 - \sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$$

We obtain,

$$\begin{aligned}
 \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} &= \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2-\sqrt{3}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4z} \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix}^{-1} \begin{bmatrix} \sqrt{2+\sqrt{3}} & 0 \\ 0 & \sqrt{2-\sqrt{3}} \end{bmatrix}^{-1} \begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} \\
 &= \begin{bmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} - \frac{\sqrt{3}-2}{4z} & 1 \end{bmatrix} \begin{bmatrix} 1 & -z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2+\sqrt{3}}} & 0 \\ 0 & \frac{1}{\sqrt{2-\sqrt{3}}} \end{bmatrix} \begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix}
 \end{aligned}$$

This leads to the following lifting implementation of inverse wavelet transform:

$$d_l \leftarrow \frac{1}{\sqrt{2+\sqrt{3}}} d_l$$

$$s_l \leftarrow \frac{1}{\sqrt{2-\sqrt{3}}} s_l$$

$$d_l \leftarrow s_{l-1} + d_l$$

$$s_l \leftarrow s_l - \frac{\sqrt{3}}{4} d_l - \frac{\sqrt{3}-2}{4} d_{l+1}$$

$$d_l \leftarrow \sqrt{3}s_l + d_l$$

These equations can be easily written down by looking at the forward transform. Simply undo the forward transform starting from the last operation performed in the forward wavelet transform.

10.1.2 Example: Forward Wavelet Transform

Let our sample be $\{1, 2, 4, 7, 5, 3, 1, 3\}$. We apply wavelet transform using Daubechies' 4-tap wavelet, which we have considered in the previous sections. Following are steps involved and the output of each step. We will use 'predict' and 'update' steps given in Section 10.1 for Daubechies' 4-tap wavelet system.

Step 1 Split the data into two streams s and d based on even and odd index.

$$s = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 1 \end{bmatrix} \quad d = \begin{bmatrix} 2 \\ 7 \\ 3 \\ 3 \end{bmatrix}$$

Step 2 Apply $d_l \leftarrow -\sqrt{3}s_l + d_l$

$$d = -\sqrt{3} \begin{bmatrix} 1 \\ 4 \\ 5 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 7 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.267949192 \\ 0.07179677 \\ -5.660254038 \\ 1.267949192 \end{bmatrix}$$

Step 3 $s_l \leftarrow s_l + \frac{\sqrt{3}}{4} d_l + \frac{\sqrt{3}-2}{4} d_{l+1}$

$$s = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 1 \end{bmatrix} + \frac{\sqrt{3}}{4} \begin{bmatrix} 0.267949192 \\ 0.07179677 \\ -5.660254038 \\ 1.267949192 \end{bmatrix} + \frac{\sqrt{3}-2}{4} \begin{bmatrix} 1.267949192 \\ 0.267949192 \\ 0.07179677 \\ -5.660254038 \end{bmatrix} = \begin{bmatrix} 1.031089 \\ 4.01314 \\ 2.544229 \\ 1.928203 \end{bmatrix}$$

Step 4 $d_l \leftarrow -s_{l-1} + d_l$

$$d = - \begin{bmatrix} 4.0314 \\ 2.544229 \\ 1.928203 \\ 1.031089 \end{bmatrix} + \begin{bmatrix} 0.267949192 \\ 0.07179677 \\ -5.660254038 \\ 1.267949192 \end{bmatrix} = \begin{bmatrix} -3.74519 \\ -2.47243 \\ -7.58846 \\ 0.23686 \end{bmatrix}$$

Step 5 $s_l \leftarrow \sqrt{2-\sqrt{3}}s_l$

$$s = \begin{bmatrix} 0.533731 \\ 2.077354 \\ 1.31699 \\ 0.998111 \end{bmatrix}$$

Step 6 $d_l \leftarrow \sqrt{2 + \sqrt{3}} d_l$

$$\mathbf{d} = \begin{bmatrix} -7.23515 \\ -4.77637 \\ -14.6598 \\ 0.457579 \end{bmatrix}$$

Inverse wavelet transform can be implemented in a similar fashion.

10.2 GEOMETRICAL FOUNDATIONS OF LIFTING SCHEME

Consider two numbers a and b and think of them as two neighbouring samples of a sequence. So a and b have some correlation which we would like to take advantage of. We propose a well known, simple linear transform which replaces a and b by their average s and difference d , i.e.,

$$s = \frac{a+b}{2} \quad (10.8)$$

and

$$d = \frac{a-b}{2} \quad (10.9)$$

The idea is that if a and b are highly correlated, the expected absolute value of their difference d will be small and can be represented with fewer bits. In case that $a = b$, the difference is simply zero. We have not lost any information because we can always recover a and b from the given s and d as:

$$a = s - \frac{d}{2} \quad (10.10)$$

and

$$b = s + \frac{d}{2} \quad (10.11)$$

This simple observation is the key behind the so-called Haar wavelet transform. Consider a signal s_j (we assume signal in space V_j) of 2^j sample values $s_{j,k}$:

$$\{s_{j,k} \mid k = 0, 1, 2, \dots, 2^j\}$$

Apply the average and difference transform for each pair $a = s_{j,2k}$ and $s_{j,2k+1}$. There are 2^{j-1} such pairs ($k = 0, 1, 2, \dots, 2^{j-1}$). Denote the result by $s_{j-1,k}$ and $d_{j-1,k}$:

$$s_{j-1,k} = \frac{s_{j,2k} + s_{j,2k+1}}{2} \quad (10.12)$$

$$d_{j-1,k} = \frac{d_{j,2k} - d_{j,2k+1}}{2} \quad (10.13)$$

The input signal s_j which has 2^j samples is split into two signals: s_{j-1} and d_{j-1} , where

$$s_{j-1} = \{s_{j-1,k} \mid k = 0, 1, 2, \dots, 2^{j-1}\}$$

$$d_{j-1} = \{d_{j-1,k} \mid k = 0, 1, 2, \dots, 2^{j-1}\}$$

Given the averages s_{j-1} and differences d_{j-1} , one can recover the original signal s_j .

We can think of the averages s_{j-1} as a coarser resolution representation of the signal s_j and of the differences d_{j-1} as the information needed to go from the coarser representation back to the original signal. If the original signal has some local coherence, e.g., if the samples are values of a smoothly varying function then the coarse representation closely resembles the original signal and the detail is very small and thus can be represented efficiently.

We can apply the same transform to the coarser signal s_{j-1} itself. By taking the averages and differences, we can split it in a (yet) coarser signal s_{j-2} and another difference signal d_{j-2} where each of them contain 2^{j-2} samples. We can do this j times before we run out of samples (see Figure 10.1). This is the Haar transform. We end up with j detail signals d_n with $0 \leq n \leq j-1$ each with 2^n coefficients, and one signal s_0 on the very coarsest scale.

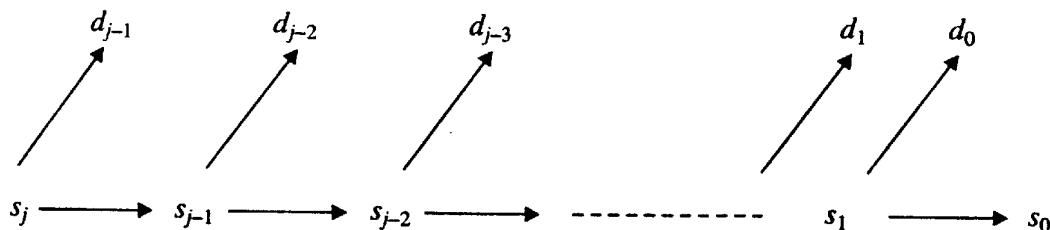


FIGURE 10.1 Signal decomposition.

The coarsest level signal s_0 contains only one sample $s_{0,0}$; which is the average of all the samples of the original signal, i.e., it is the DC component or zero frequency of the signal. By using the inverse transform we start from s_0 and d_0 $0 \leq n \leq j$ and obtain s_j again. This adds up to

$$1 + \sum_{n=0}^{j-1} 2^n = 2^j \quad (10.14)$$

which exactly is the number of samples of the original signal. The whole Haar transform can be thought of as applying a $N \times N$ matrix ($N = 2^j$) to the signal s_j . The cost of computing the transform is only proportional to N . This is remarkable as in general a linear transformation of an N vector requires $O(N^2)$ operations. Compare this to the fast Fourier transform, whose cost is $O(N \log N)$. It is the hierarchical structure of a wavelet transform which allows switching to and from the wavelet representation in $O(N)$ time.

10.2.1 Haar and Lifting

In this section we will show a new way of looking at the Haar transform. The novelty lies in the way we compute the difference and average of two numbers a and b . Assume we want to compute the whole transform in-place, i.e., without using auxiliary memory locations, by overwriting the locations that hold a and b with the values of s and d , respectively. This cannot immediately be done with the formulas given by Eqs. (10.8) and (10.9). Indeed, assume that we want to store s in the same location as a and d in the same location as b . Then Eq. (10.8) would lead to the wrong result. Computing s and overwriting a leads to a wrong d (assuming we compute the average after the difference). We, therefore, use another implementation, again in two steps. Firstly, we compute the difference:

$$d = b - a \quad (10.15)$$

and store it in the location for b . As we now lost the value of b , we next use a and the newly computed difference d to find the average as:

$$s = a + \frac{d}{2} \quad (10.16)$$

This gives the same result because $a + d/2 = a + (b - a) = 2 = (a + b)/2$. The advantage of the splitting into two steps is that we can overwrite b with d and a with s , requiring no auxiliary storage. This particular scheme of writing a transform is a first, simple instance of the lifting scheme.

10.2.2 The Lifting Scheme

In this section we describe the lifting scheme in more detail. Consider a signal s_j with 2^j samples which we want to transform into a coarser signal s_{j-1} and a detail signal d_{j-1} . A typical case of a wavelet transform built through lifting consists of three steps: split, predict and update (see Figure 10.2). Let us discuss each stage in more detail.

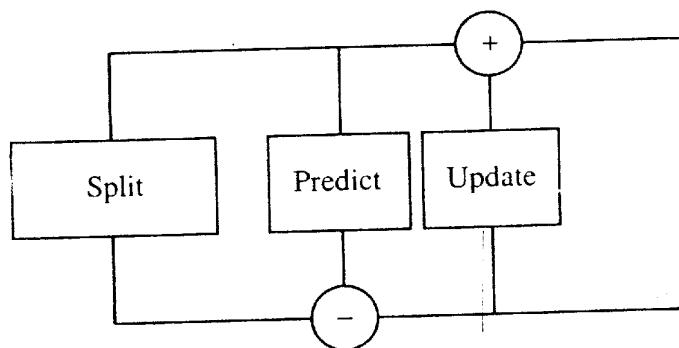


FIGURE 10.2 Steps in lifting scheme.

Split

This stage does not do much except for splitting the signal into two disjoint sets of samples. In our case one group consists of the even indexed samples s_{2l} and the other group consists of the odd indexed samples s_{2l+1} . Each group contains half as many samples as the original signal. The splitting into evens and odds is called the **lazy wavelet transform**. We, thus, built an operator so that

$$(\text{even}_{j-1}; \text{odd}_{j-1}) = \text{Split}(s_j)$$

Remember that in the previous example a was an even sample while b was an odd sample.

Predict

The even and odd subsets are interspersed. If the signal has a local correlation structure, the even and odd subsets will be highly correlated. In other words, given one of the two sets, it should be possible to predict the other one with reasonable accuracy. We always use the even set to predict the odd one. In the Haar case the prediction is particularly simple. An odd sample $s_{j,2l+1}$ will use its left neighbouring even sample $s_{j,2l}$ as its predictor. We then let the detail $d_{j-1,l}$ be the difference between the odd sample and its prediction:

$$d_{j-1,l} = s_{j,2l+1} - s_{j,2l}$$

which defines an operator P such that

$$d_{j-1} = \text{odd}_{j-1} - P(\text{even}_{j-1})$$

As we already argued, it should be possible to represent the detail more efficiently. Note that if the original signal is a constant then all details are exactly zero.

Update

One of the key properties of the coarser signals is that they have the same average value as the original signal, i.e., the quantity

$$S = \frac{1}{2^j} \sum_{l=0}^{2^j-1} s_{j,l}$$

is independent of j . This results in the fact that the last coefficients $s_{0,0}$ is the DC component or overall average of the signal. The update stage ensures this by letting

$$s_{j-1,l} = s_{j,2l} + d_{j-1,l}/2$$

Substituting this definition, we easily verify that

$$\sum_{l=0}^{2^j-1} s_{j-1,l} = \sum_{l=0}^{2^j-1} (s_{j,2l} + d_{j-1,l}/2) = 1/2 \sum_{l=0}^{2^j-1} \sum_{l=0}^{2^j-1} (s_{j,2l} + s_{j,2l+1}) = \frac{1}{2} \sum_{l=0}^{2^j} s_{j,l}$$

which defines an operator U of the form:

$$s_{j-1} = \text{even}_{j-1} + U(d_{j-1})$$

This can be computed in-place: the even locations can be overwritten with the averages and the odd ones with the details. An abstract implementation is given by

$$(\text{odd}_{j-1}, \text{even}_{j-1}) = \text{Split}(s_j)$$

$$\text{odd}_{j-1} = \text{odd}_{j-1} - P(\text{even}_{j-1})$$

$$\text{even}_{j-1} = \text{even}_{j-1} + U(\text{odd}_{j-1})$$

These three stages are depicted in a wiring diagram in Figure 10.3. We can immediately build the inverse scheme as shown in the wiring diagram in Figure 10.4. Again, we have three stages.

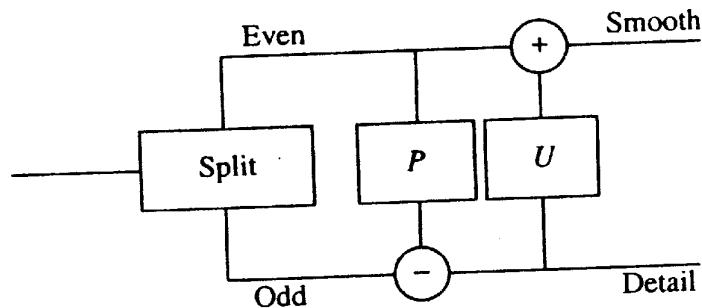


FIGURE 10.3 Prediction and updating steps of lifting scheme.

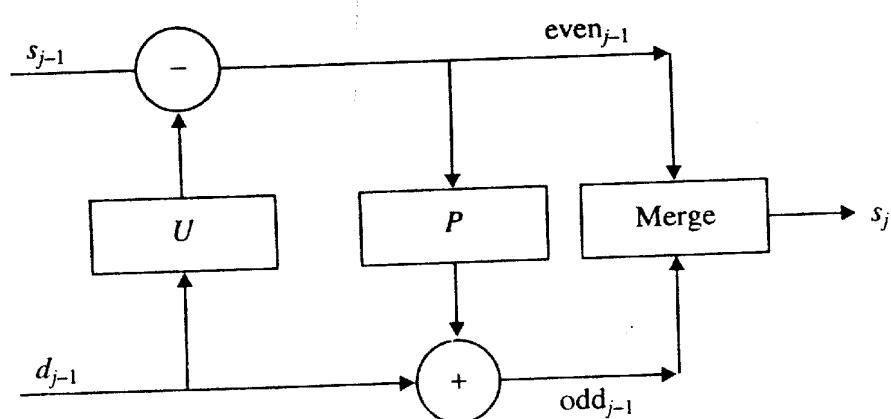


FIGURE 10.4 Wiring diagram for inverse of lifting scheme.

Undo update

Given even d_{j-1} and s_{j-1} we can recover the even samples by subtracting update information

$$\text{even}_{j-1} = s_{j-1} - U(d_{j-1})$$

In the case of Haar, we compute this by letting

$$s_{j,2l} = s_{j-1,l} - d_{j-1,l}/2$$

Undo predict

Given even d_{j-1} and s_{j-1} we can recover the odd samples by adding the prediction information

$$\text{odd}_{j-1} = d_{j-1} - P(\text{even}_{j-1})$$

In the case of Haar, we compute this by letting

$$s_{j,2l+1} = d_{j-l,l} + s_{j,2l}$$

Merge

Now, that we have the even and odd samples, we simply have to zipper them together to recover the original signal. This is the inverse lazy wavelet:

$$s_j = \text{Merge}(\text{even}_{j-1}, \text{odd}_{j-1})$$

Assuming that the even slots contain the averages and the odd ones contain the differences, the implementation of the inverse transform is:

$$\text{even}_{j-1} = \text{even}_{j-1} - U(\text{odd}_{j-1})$$

$$\text{odd}_{j-1} = \text{odd}_{j-1} + P(\text{even}_{j-1})$$

$$s_j = \text{Merge}(\text{odd}_{j-1}, \text{even}_{j-1})$$

The inverse transform is always found by reversing the order of the operations and flipping the signs.

The lifting scheme has a number of algorithmic advantages:

In-place

All the calculations can be performed in-place, which can be an important memory savings.

Efficiency

In many cases the number of floating point operations needed to compute both smooth and detail parts is reduced since sub-expressions are reused.

Parallelism

“Unrolling” a wavelet transform into a wiring diagram exhibits its inherent SIMD parallelism at all scales, with single write and multiple read semantics. But perhaps more importantly lifting has some structural advantages that are both theoretically and practically relevant.

Inverse transform

Writing the wavelet transform as a sequence of elementary predict and update (lifting) steps, it is immediately obvious what the inverse transform is: simply run the code backwards. In the classical setting, the inverse transform can typically only be found with the help of Fourier techniques.

Generality

This is the most important advantage. Since the design of the transform is performed without reference to Fourier techniques, it is very easy to extend it to settings in which, for example, samples are not placed evenly or constraints such as boundaries need to be incorporated. It also carries over directly to curves, surfaces and volumes. It is for these reasons that we built our exposition entirely around the lifting scheme.

10.2.3 The Linear Wavelet Transform Using Lifting

One way to build other wavelet transforms is through the use of different predict and/or update steps. What is the incentive behind improving predict and update? The Haar transform uses a predictor that is correct in case the original signal is a constant. It eliminates zeroth order correlation. We say that the order of the predictor is one. Similarly, the order of the update operator is one as it preserves the average or zeroth order moment. In many cases it is desirable to have predictors which can exploit coherence beyond zeroth order correlation and it is often desirable to preserve higher order moments beyond the zeroth in the successively coarser versions of the function. In this section we build a predictor and update that are of order two. This means that the predictor will be exact in case the original signal is a linear (between two consecutive even indexed signal values) and the update will preserve the average and the first moment. Refer Figure 10.5. This turns out to be fairly easy. For an odd sample, $(s_{j,2l+1})$ we let the predictor be the average of the neighbouring sample on the left ($s_{j,2l}$) and the neighbouring sample on the right ($s_{j,2l+2}$).

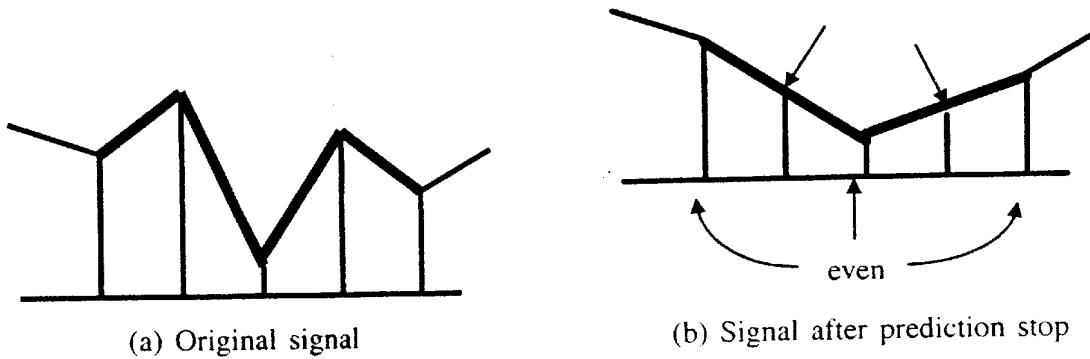


FIGURE 10.5 Linear prediction.

The detail coefficient is given by

$$d_{j,1} = s_{j,2l+1} - \frac{1}{2}(s_{j,2l} + s_{j,2l+2}) \quad (10.17)$$

Figure 10.6 illustrates this idea. Notice that if the original signal was a first degree polynomial (between two consecutive even indexed samples), i.e., if $s_l = \alpha l + \beta$ for some α and β , this prediction is always correct and all details are zero. In other words, the detail coefficients measure to which extent the original signal fails to be linear. The expected value of their magnitudes is small. In terms of frequency content, the detail coefficients capture high

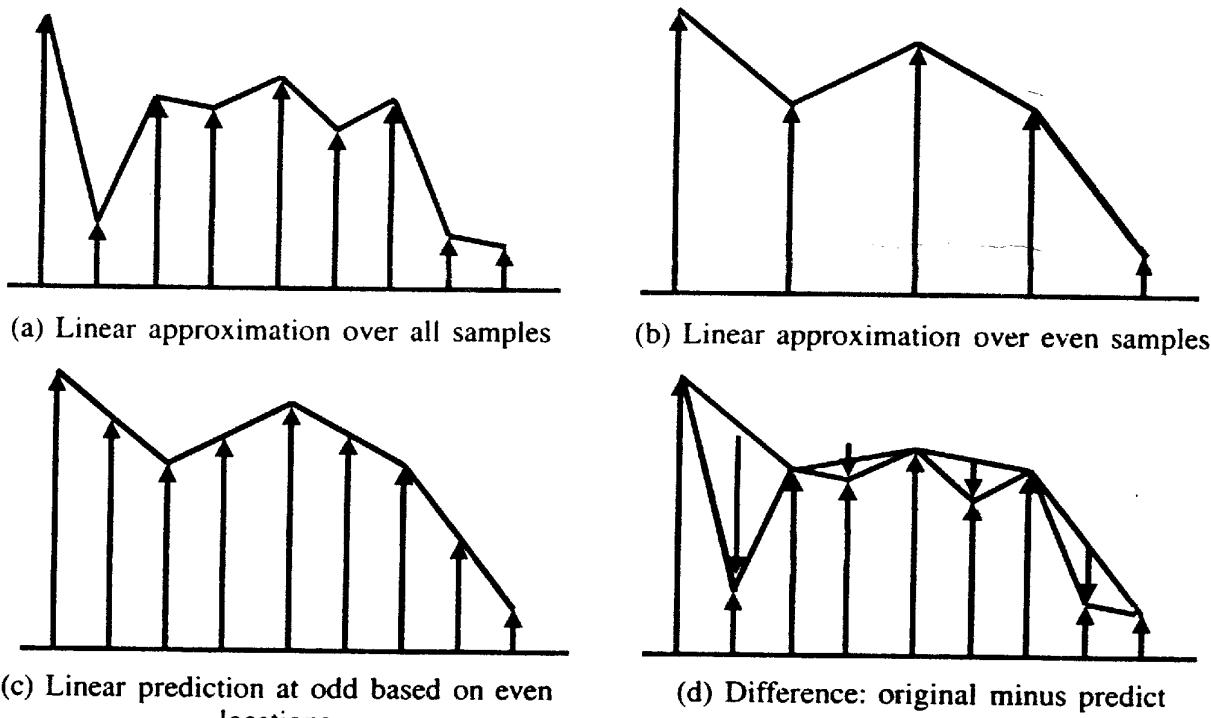


FIGURE 10.6 Results of linear prediction operation.

frequencies present in the original signal. In the update stage, we first assure that the average of the signal is preserved or

$$\sum_l s_{j-1,l} = \frac{1}{2} \sum_l s_{j,l} \quad (10.18)$$

We, therefore, update the even samples $s_{j,2l}$ using the previously computed detail signals $d_{j-1,l}$. Again we use the neighbouring wavelet coefficients and propose an update of the form:

$$s_{j-1,l} = s_{j,2l} + A(d_{j-1,l-1} + d_{j-1,l}) \quad (10.19)$$

To find A , we compute the average as follows:

$$\begin{aligned} \sum_l s_{j-1,l} &= \sum_l s_{j,2l} + 2A \sum_l d_{j-1,l-1} \\ &= (1 - 2A) \sum_l s_{j,2l} + 2A \sum_l s_{j,2l+1} \end{aligned}$$

From this we get $A = 1/4$ as the correct choice to maintain the average. Because of the symmetry of the update operator, we also preserve the first order moment. On substituting $A = 1/4$ in Eq. (10.19), we obtain the following equation:

$$s_{j-1,l} = s_{j,2l} + \frac{1}{4}(d_{j-1,l-1} + d_{j-1,l}) \quad (10.20)$$

Equation (10.20) is literally the lifting step. Even coefficients are given a lift to maintain the average of the whole signal. The amount of lift given is $1/4$ of the sum of difference values

(or $d_{j-1,l}$ coefficients) on either side of the even indexed signal. Figure 10.7 schematically shows the lifting process. The bold line segments represent the original signal. Signal between index pairs $(2k - 2, 2k)$ and $(2k + 2, 2k + 4)$ are already linear and hence prediction coefficient (or error) is zero. The shaded area represents the loss in area when we approximate the signal between the index $2k - 2$ and $2k + 4$ in a linear fashion. To compensate this loss, $1/4$ of prediction error d_k is added to even indexed coefficients standing on either side of it. The area under the dark curve between indexes $2k - 2$ and $2k + 4$ is same as the area under the light curve between the same indexes.

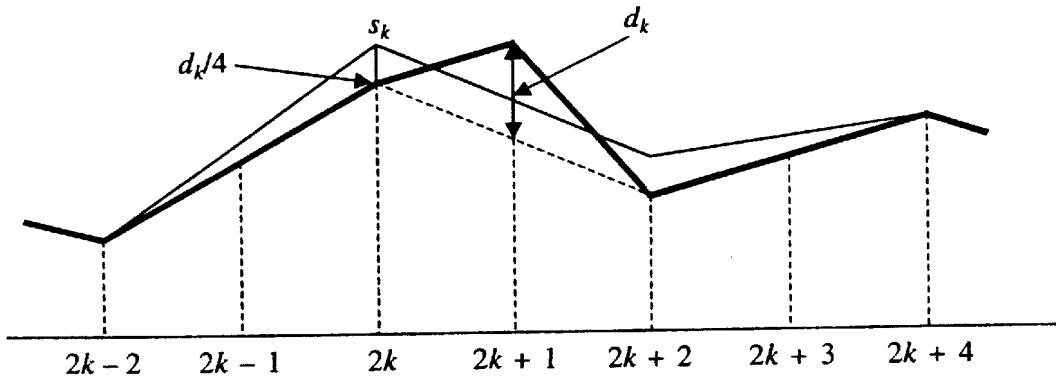


FIGURE 10.7 Illustration of lifting.

One step in the wavelet transform is shown in the scheme in Figure 10.8. By iterating this scheme, we get a complete wavelet transform. The inverse is as easy to compute, letting

$$s_{j,2l} = s_{j-1,l} - \frac{1}{4}(d_{j-1,l-1} + d_{j-1,l}) \quad (10.21)$$

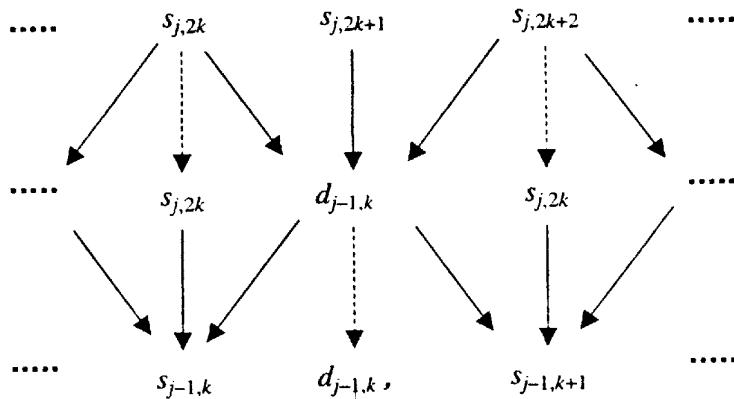


FIGURE 10.8 Linear prediction and updating steps.

to recover the even and

$$s_{j,2l+1} = d_{j,l} - \frac{1}{2}(s_{j,2l} + s_{j,2l+2}) \quad (10.22)$$

to recover the odd samples.

The wavelet transform presented earlier is the biorthogonal (2,2) of Cohen-Daubechies-Feauveau [7]. One might not immediately recognize this but by substituting the predict in the update, one can check that the coarser coefficients are given by

$$\begin{aligned}
 s_{j-1,l} &= s_{j,2l} + \frac{1}{4}(d_{j-1,l} + d_{j-1,l}) \\
 &= s_{j,2l} + \frac{s_{j,2l+1} - \frac{s_{j,2l} + s_{j,2l+2}}{2} + s_{j,2l-1} - \frac{s_{j,2l-2} + s_{j,2l}}{2}}{4} \\
 &= s_{j,2l} + \frac{s_{j,2l+1}}{4} - \frac{s_{j,2l} + s_{j,2l+2}}{8} + \frac{s_{j,2l-1}}{4} - \frac{s_{j,2l-2} + s_{j,2l}}{8} \\
 &= -\frac{1}{8}s_{j,2l-2} + \frac{1}{4}s_{j,2l-1} + \frac{3}{4}s_{j,2l} + \frac{1}{4}s_{j,2l+1} - \frac{1}{8}s_{j,2l+2} \\
 &= \tilde{h}(-k)*s \quad (\text{Convolution of analysis filter with signal coefficients})
 \end{aligned} \tag{10.23}$$

The set of coefficients for the biorthogonal (2,2) of Cohen-Daubechies-Feauveau wavelets are given in Table 10.2. Note that the filter coefficients are same up to a scale factor.

TABLE 10.2 CDF(2,2) Biorthogonal Filter Coefficients

k	z^2	z^1	z^0	z^{-1}	z^{-2}	z^{-3}
\tilde{h}^{new}	$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$	
\tilde{g}			$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{-1}{4}$	
h		$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{4}$		
g^{new}		$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$

Note that, Eq. (10.23), when written in this form (which is not lifting) the transform cannot be computed in-place. Also to find the inverse transform one would have to rely on Fourier techniques (convolution is usually implemented using FFT). This is much less intuitive and does not generalize to irregular settings.

10.2.4 Higher Order Wavelet Transform

For linear wavelet transform, we used two neighbouring even coefficients to predict the odd coefficients. We can use more than two coefficients to predict odd coefficients and accordingly we get higher order wavelet transform. For example, in Figure 10.9, we use four even coefficients to predict an odd coefficient.

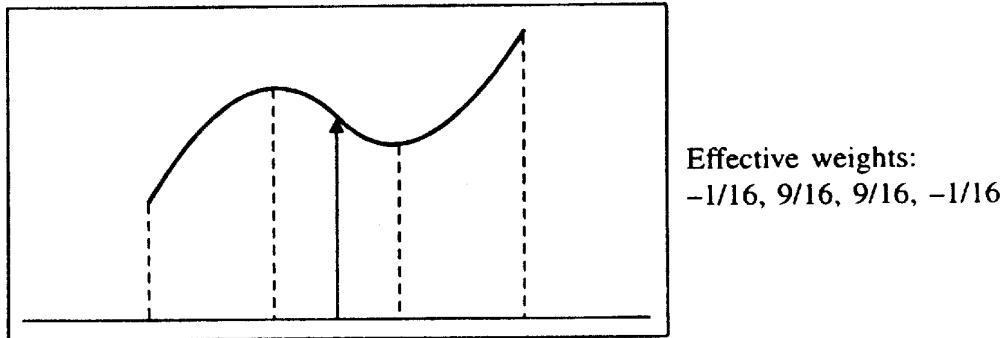


FIGURE 10.9 Devising higher order wavelet transform.

However, the method becomes cumbersome for higher order wavelets especially when we want to design wavelets with shapes that matches signals. Computing coefficients for updating and predicting in such cases using this approach is difficult. Therefore, we go for another approach to lifting that is due to Wim Sweldons [2, 6].

10.3 LIFTING SCHEME IN THE Z-DOMAIN

In this section, we describe, making of new biorthogonal filters based on existing biorthogonal filters using lifting scheme. The whole computation is done in Z-domain. At the same time, we try to relate the theory to time domain biorthogonal wavelet bases. Figure 10.10 shows general scheme of a wavelet transform. The forward transform uses two analysis filters $\tilde{h}(-n)$ (low pass) and $\tilde{g}(-n)$ (high pass) followed by downsampling. $\tilde{H}(z^{-1})$ and $\tilde{G}(z^{-1})$ are respective Z-transforms. In the inverse transform, at first we upsample the filtered signals and then apply two filters $h(n)$ (low pass) and $g(n)$ (high pass). We already know that the filters are perfectly matched to produce perfect reconstruction.

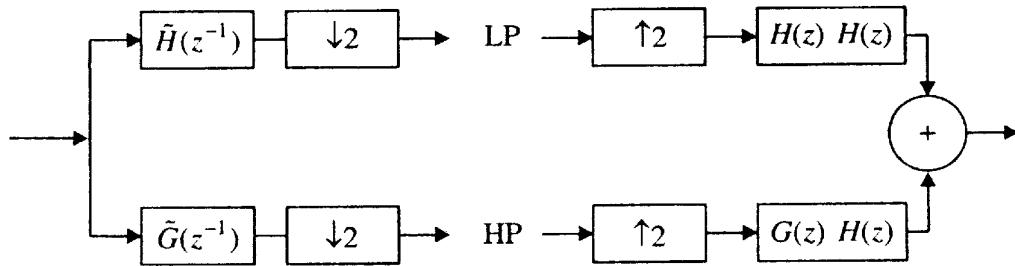


FIGURE 10.10 Analysis and synthesis steps which does perfect reconstruction.

Given the perfect reconstruction filter banks, filters can be easily modified to achieve desired properties without violating perfect reconstruction property. Figure 10.11 shows this basic idea. Filter coefficients $u(k)$ in the left part of the diagram modifies the low pass filtered signal by subtracting from it a weighted sum of high pass filtered signal coefficients. On the right part of the diagram $u(k)$ nullifies this change by adding the same quantity back to the low

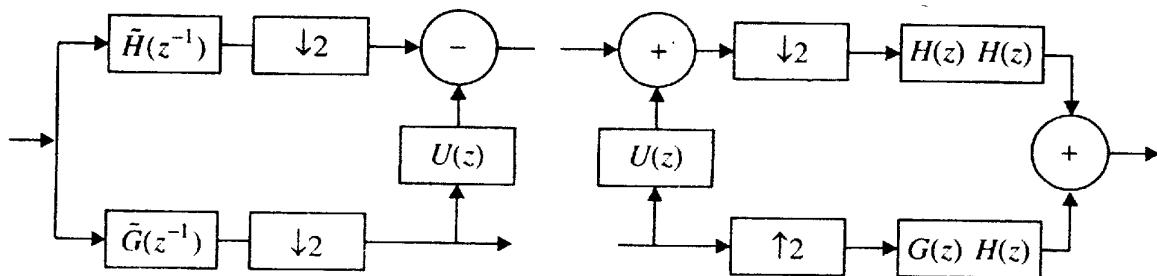


FIGURE 10.11 Additional lifting steps that does not violate reconstruction.

pass filtered signal. What really happens in the middle of the diagram of Figure 10.11 shown separately in Figure 10.12 can be visualized using the diagram shown in Figure 10.13. Let $\{u(k), k = -1, 0, 1\}$ and $\{a, b, c\}$ be the lifting filter coefficients.

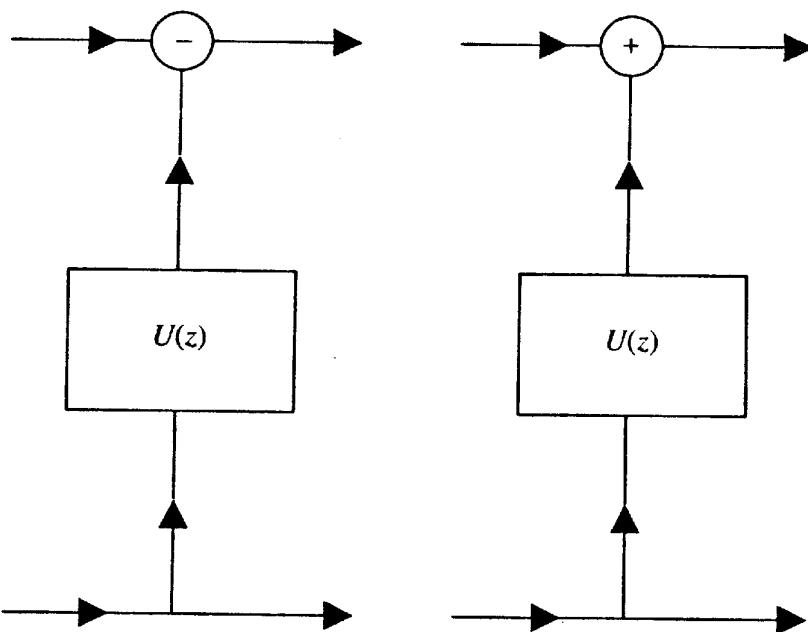


FIGURE 10.12 The lifting step that does not affect perfect reconstruction.

The high pass filtered signal is shown as unfilled circles at the bottom of the figure. Shaded circle on the top left portion of Figure 10.13 are low pass filtered signal. The filter $u(k)$ on the left of Figure 10.13 modifies $s_{j,k}$ coefficients to obtain modified low pass filtered coefficients. In the Figure 10.13 the $s_{j,k}$ and $d_{j,k}$ coefficients are assumed to move along two lines and the three (in this example) $d_{j,k}$ coefficients, namely $d_{j,m-1}, d_{j,m}, d_{j,m+1}$ passing through the box at any moment and $u(k)$ coefficients (a, b, c) are used to modify the $s_{j,m}$ th coefficient. The outputs are shown as shaded squares. The filter $u(k)$ on right of the Figure 10.13 does the ‘undo’ operation giving the original $s_{j,k}$ back. The whole operation does not affect the reconstruction of the signal.

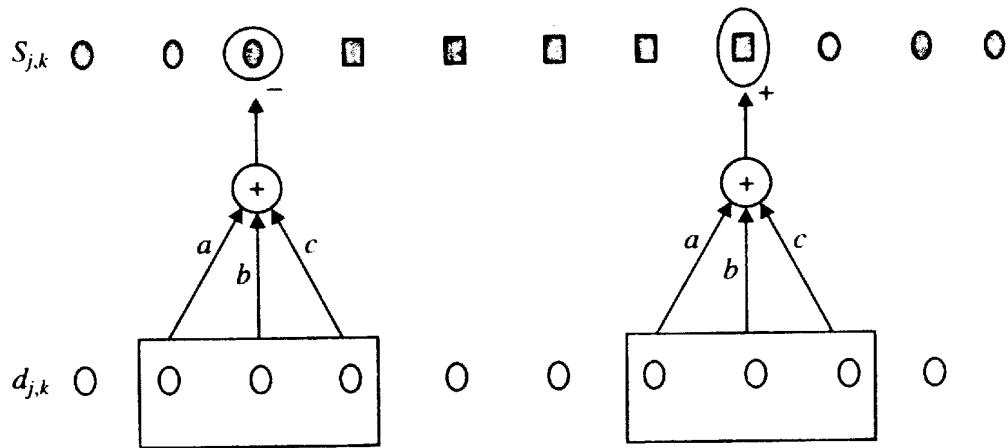


FIGURE 10.13 Lifting operations.

Consider synthesis bank of filters alongwith $u(k)$. The Z-transform version of the filter bank is shown in Figure 10.14. The update filter $U(z)$, if moved to the location beyond the

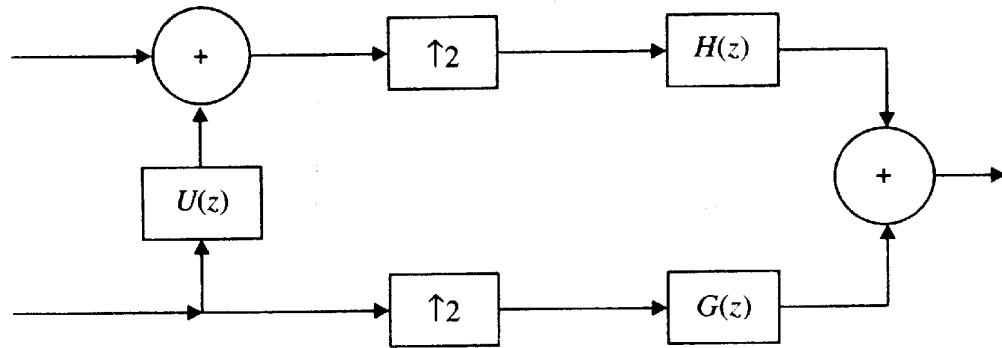


FIGURE 10.14 Signal synthesis stage.

upsampling operator, becomes $U(z^2)$ (see Figures 10.15a and 10.15b). It means that the effective lifting filter has become $\{a, 0, b, 0, c, 0\}$ —the upsampled version of original $\{a, b, c\}$. Finally, the equivalent low pass and high pass filters on the synthesis side is given by

$$H^{\text{new}}(z) = H(z) \quad (10.24)$$

and

$$G^{\text{new}}(z) = G(z) + U(z^2) H(z) \quad (10.25)$$

Similarly, on the analysis side we bring the $U(z)$ term to the left of downsampling operators according to the identity shown in Figures 10.16 and 10.17.

$$\tilde{H}^{\text{new}}(z^{-1}) = \tilde{H}(z^{-1}) - U(z^2) \tilde{G}(z^{-1}) \quad (10.26)$$

and

$$\tilde{G}^{\text{new}}(z^{-1}) = \tilde{G}(z^{-1}) \quad (10.27)$$

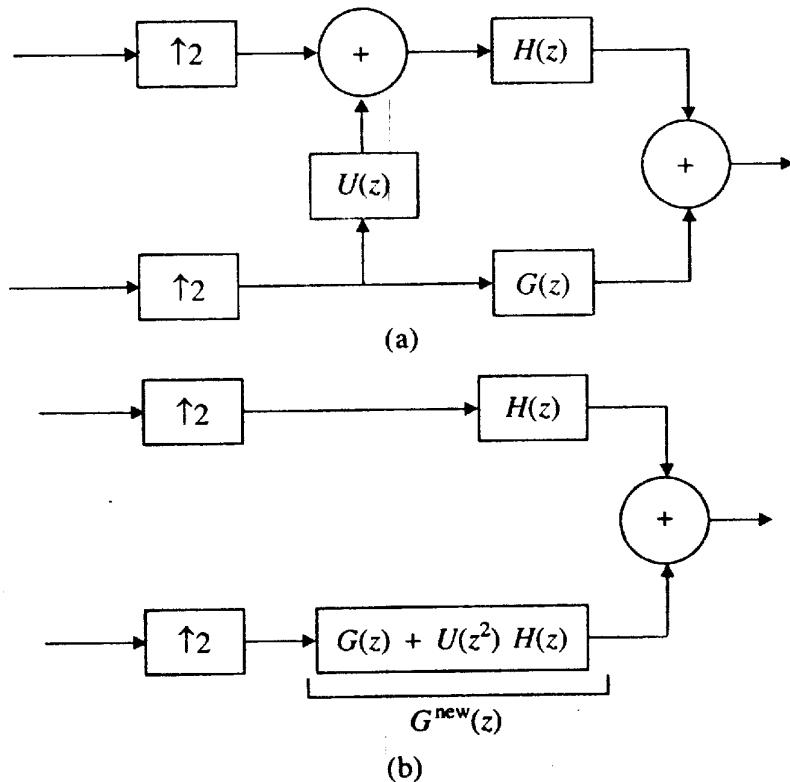


FIGURE 10.15 Modifying filters in the synthesis stage.

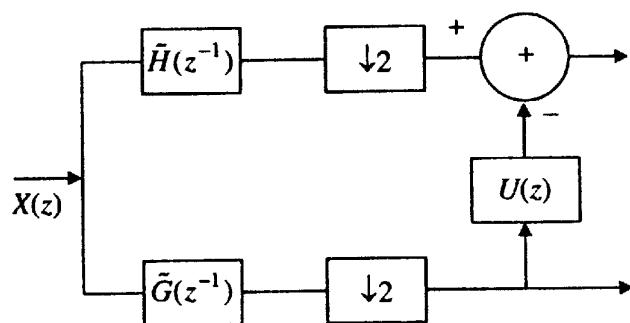


FIGURE 10.16 Analysis stage with update filter.

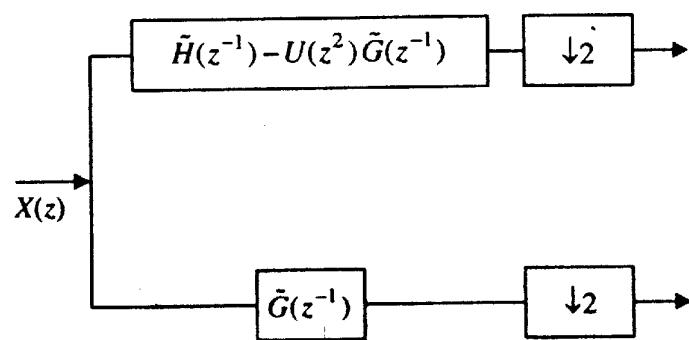


FIGURE 10.17 Modification of analysis filters.

Let us now find out the new high pass filter $g^{\text{new}}(n)$ on synthesis side and corresponding wavelet function $\psi(t)$. We have

$$G^{\text{new}}(z) = G(z) + U(z^2)H(z)g^{\text{new}}(n)$$

Then

$$g^{\text{new}}(n) = g(n) + \sum_k r(k) h(n-k) \quad (10.28)$$

where

$$r(k) = \begin{cases} u(k/2) & k \text{ is even} \\ 0 & k \text{ is odd} \end{cases}$$

Alternatively

$$r(2k) = u(k)$$

and

$$r(2k+1) = 0$$

So,

$$g^{\text{new}}(n) = g(n) + \sum_k u(k) h(n-2k) \quad (10.29)$$

The corresponding wavelet is:

$$\begin{aligned} \psi^{\text{new}}(t) &= \sum_n g^{\text{new}}(n) \phi(2t-n) \\ &= \sum_n g(n) \phi(2t-n) + \sum_k u(k) \sum_n h(n-2k) \phi(2t-n) \\ &= \psi(t) + \sum_k u(k) \sum_l h(l) \phi(2t-2k-l) \\ &= \psi(t) + \sum_k u(k) \phi(t-k) \quad \left[\text{As } \phi(t) = \sum_l h(l) \phi(2t-l) \right] \end{aligned} \quad (10.30)$$

In the analysis side, the filter which has undergone change is the low pass filter. Let us find out new low pass filter and the corresponding scaling function. We have

$$\tilde{H}^{\text{new}}(z^{-1}) = \tilde{H}(z^{-1}) - U(z^2)\tilde{G}(z^{-1})$$

$$\tilde{h}^{\text{new}}(-n) = \tilde{h}(-n) - \sum_k r(k) \tilde{g}(-n-k)$$

or

$$\tilde{h}^{\text{new}}(n) = \tilde{h}(n) - \sum_k r(k) \tilde{g}(n-k)$$

$$\tilde{h}^{\text{new}}(n) = \tilde{h}(n) - \sum_k u(k) \tilde{g}(n-2k) \quad (10.31)$$

The corresponding scaling function is:

$$\begin{aligned}
 \tilde{\phi}^{\text{new}}(t) &= \sum_n \tilde{h}^{\text{new}}(n) \tilde{\phi}^{\text{new}}(2t-n) \\
 &= \sum_n \tilde{h}(n) \tilde{\phi}^{\text{new}}(2t-n) - \sum_k u(k) \sum_n \tilde{g}(n-2k) \tilde{\phi}^{\text{new}}(2t-n) \\
 &= \sum_n \tilde{h}(n) \tilde{\phi}^{\text{new}}(2t-n) - \sum_k u(k) \sum_l \tilde{g}(l) \tilde{\phi}^{\text{new}}(2t-2k-l) \\
 &= \sum_n \tilde{h}(n) \tilde{\phi}^{\text{new}}(2t-n) - \sum_k u(k) \tilde{\psi}^{\text{new}}(t-k) \\
 &\quad \left[\text{As } \tilde{\psi}^{\text{new}}(t) = \sum_l \tilde{g}(l) \tilde{\phi}^{\text{new}}(2t-l) \right]
 \end{aligned} \tag{10.32}$$

Since $\tilde{g}(n)$ is unchanged the new wavelet function in the analysis side is given by

$$\psi^{\text{new}}(t) = \sum_l \tilde{g}(l) \tilde{\phi}^{\text{new}}(2t-l)$$

Putting all the result together, we obtain

$$\phi^{\text{new}}(t) = \phi(t) \tag{10.33}$$

$$\psi^{\text{new}}(t) = \psi(t) + \sum_k u(k) \phi(t-k) \tag{10.34}$$

$$\tilde{\phi}^{\text{new}}(t) = \sum_n \tilde{h}(n) \tilde{\phi}^{\text{new}}(2t-n) - \sum_k u(k) \tilde{\psi}^{\text{new}}(t-k) \tag{10.35}$$

$$\tilde{\psi}^{\text{new}}(t) = \sum_l \tilde{g}(l) \tilde{\phi}^{\text{new}}(2t-l) \tag{10.36}$$

The lifting theorem enunciated by Wim Sweldens [2, 6] is exactly the result we derived.

10.3.1 Design Example 1

Let

$$H(z) = \sqrt{2} \left\{ \frac{1}{4}z, \frac{1}{2}, \frac{1}{4}z^{-1} \right\}$$

$$G(z) = \sqrt{2}z^{-1}$$

$$\tilde{H}(z) = \sqrt{2}$$

$$\tilde{G}(z) = \sqrt{2} \left\{ \frac{-1}{4}, \frac{1}{2}z^{-1}, \frac{-1}{4}z^{-2} \right\}$$

Table 10.3 shows the filter coefficients from which it can be easily seen that the filters satisfy biorthogonal conditions. Filters \tilde{h} and \tilde{g} are orthogonal. Similarly \tilde{h} and g are orthogonal.

TABLE 10.3 Filter Coefficients

K	z^2	z^1	z^0	z^{-1}	z^{-2}
\tilde{h}			$\sqrt{2}$		
\tilde{g}			$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{-1}{4}$
h		$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{4}$	
g				$\sqrt{2}$	

Corresponding scaling function and wavelets are given in Figure. 10.18.

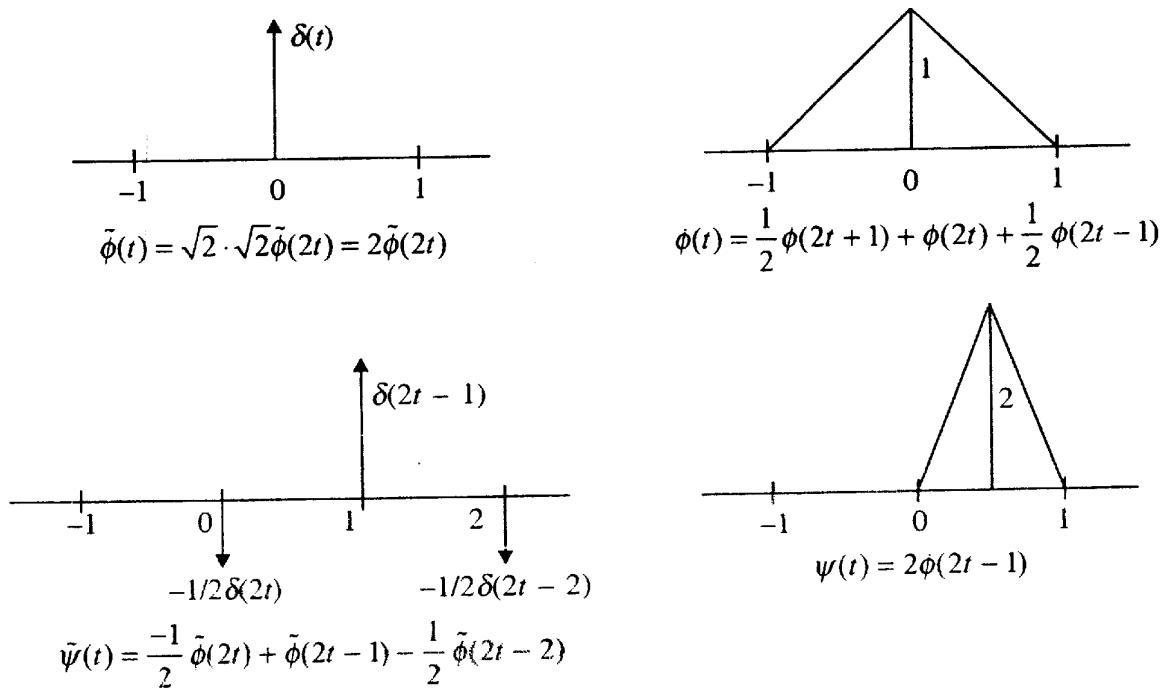


FIGURE 10.18 Scaling and wavelet functions.

From Figure 10.18 we observe that analysis scaling and wavelet functions are delta (impulse) functions. Also we note that the area under synthesis wavelet function is not zero and hence does not possess any vanishing moments.

Let us apply lifting step to add vanishing moment property to synthesis wavelet. Suppose that the new wavelet has the form:

$$\psi^{\text{new}}(t) = \psi(t) + \alpha\phi(t) + \alpha\phi(t-1)$$

Our goal is to make zeroth moment of $\psi^{\text{new}}(t)$ vanish, i.e.,

$$\begin{aligned} \int_{-\infty}^{\infty} \psi^{\text{new}}(t) dt &= \frac{1}{2} \cdot 1 \cdot 2 + \alpha \cdot 1 + \alpha \cdot 1 \\ &= 0 \text{ when } \alpha = \frac{-1}{2} \end{aligned}$$

So the new wavelet is $\psi^{\text{new}}(t) = \psi(t) - \frac{1}{2} \phi(t) - \frac{1}{2} \phi(t-1)$. Figure 10.19 shows the construction process.

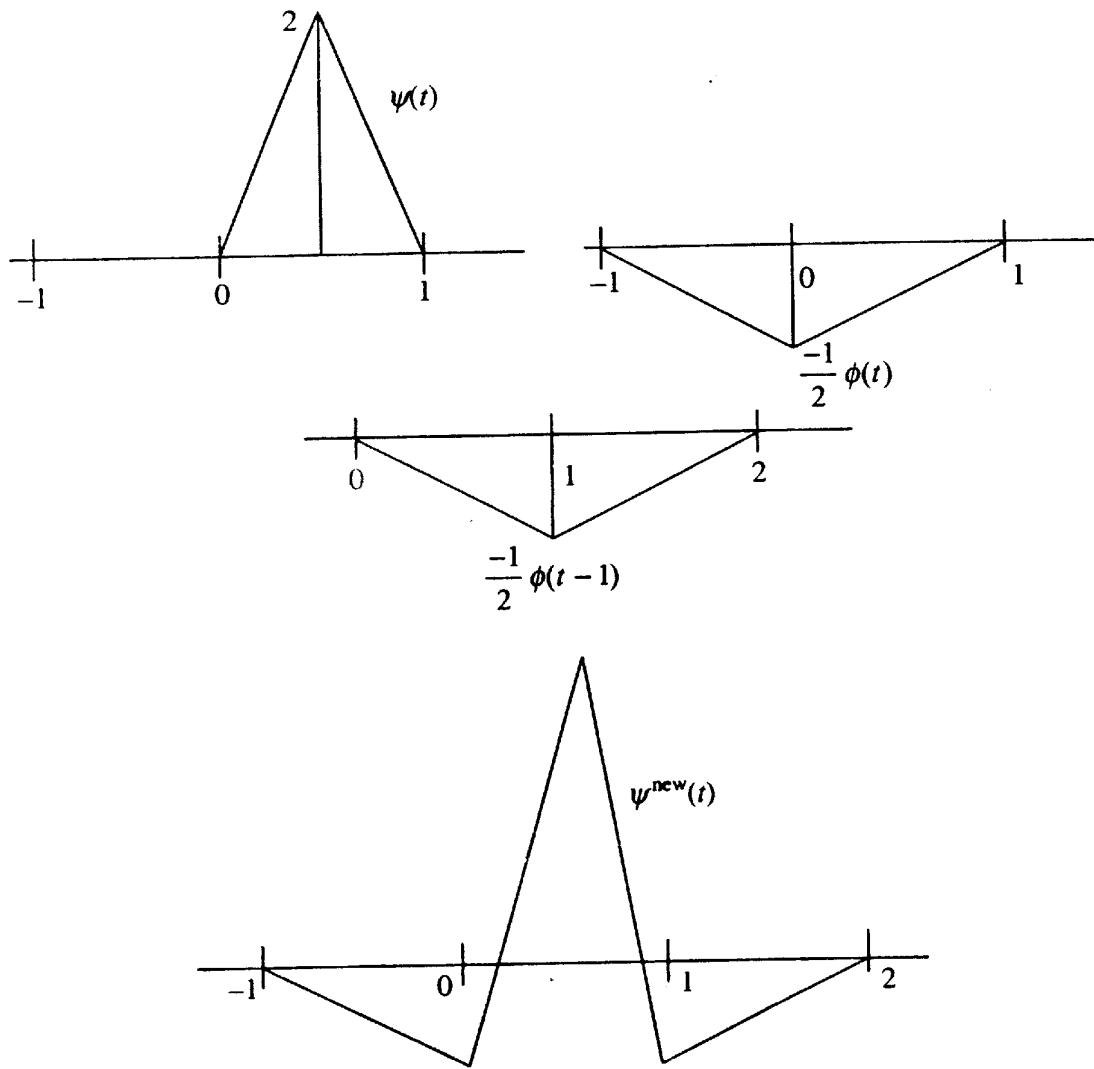


FIGURE 10.19 Construction of new wavelet from an old one.

Let us now find out the filter $G^{\text{new}}(z)$ corresponding to the new wavelet $\psi^{\text{new}}(t)$.

$$\begin{aligned}\psi^{\text{new}}(t) &= \psi(t) - \frac{1}{2} \phi(t) - \frac{1}{2} \phi(t-1) \\ &= 2\phi(2t-1) - \frac{1}{2} \left\{ \frac{1}{2} \phi(2t+1) + \phi(2t) + \frac{1}{2} \phi(2t-1) \right\} - \frac{1}{2} \left\{ \frac{1}{2} \phi(2t-1) + \phi(2t-2) + \frac{1}{2} \phi(2t-3) \right\} \\ &= \frac{-1}{4} \phi(2t+1) - \frac{1}{2} \phi(2t) + \frac{3}{2} \phi(2t-1) - \frac{1}{2} \phi(2t-2) - \frac{1}{4} \phi(2t-3)\end{aligned}$$

So,

$$G^{\text{new}}(z) = \sqrt{2} \left\{ -\frac{1}{8} z - \frac{1}{4} + \frac{3}{4} z^{-1} - \frac{1}{4} z^{-2} - \frac{1}{8} z^{-3} \right\}$$

This can be rewritten as:

$$\begin{aligned}G^{\text{new}}(z) &= \sqrt{2} \left\{ z^{-1} + \frac{-(1+z^{-2})}{2} \left(\frac{1}{4} z + \frac{1}{2} + \frac{1}{4} z^{-1} \right) \right\} \\ &= G(z) + U(z^2) H(z)\end{aligned}$$

Similarly,

$$\begin{aligned}\tilde{H}^{\text{new}}(z^{-1}) &= \tilde{H}(z^{-1}) - U(z^2) \tilde{G}(z^{-1}) \\ &= \sqrt{2} \left\{ 1 + \frac{(1+z^{-2})}{2} \left(\frac{-1}{4} + \frac{1}{2} z^1 + \frac{-1}{4} z^2 \right) \right\} \\ &= \sqrt{2} \left\{ -\frac{1}{8} z^{-2} + \frac{1}{4} z^{-1} + \frac{3}{4} + \frac{1}{4} z - \frac{1}{8} z^2 \right\}\end{aligned}$$

Therefore,

$$\tilde{H}^{\text{new}}(z) = \sqrt{2} \left\{ -\frac{1}{8} z^2 + \frac{1}{4} z^1 + \frac{3}{4} + \frac{1}{4} z^{-1} - \frac{1}{8} z^{-2} \right\}$$

The final filters are tabulated in Table 10.4.

TABLE 10.4 Final Filters

k	z^2	z^1	z^0	z^{-1}	z^{-2}	z^{-3}
\tilde{h}^{new}	$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$	
\tilde{g}			$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{-1}{4}$	
h		$\sqrt{2} \cdot \frac{1}{4}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{4}$		
g^{new}		$\sqrt{2} \cdot \frac{-1}{8}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{3}{4}$	$\sqrt{2} \cdot \frac{-1}{4}$	$\sqrt{2} \cdot \frac{-1}{8}$

Note again the orthogonality of filters. Figure 10.20 shows the plot of corresponding wavelet and scaling function.

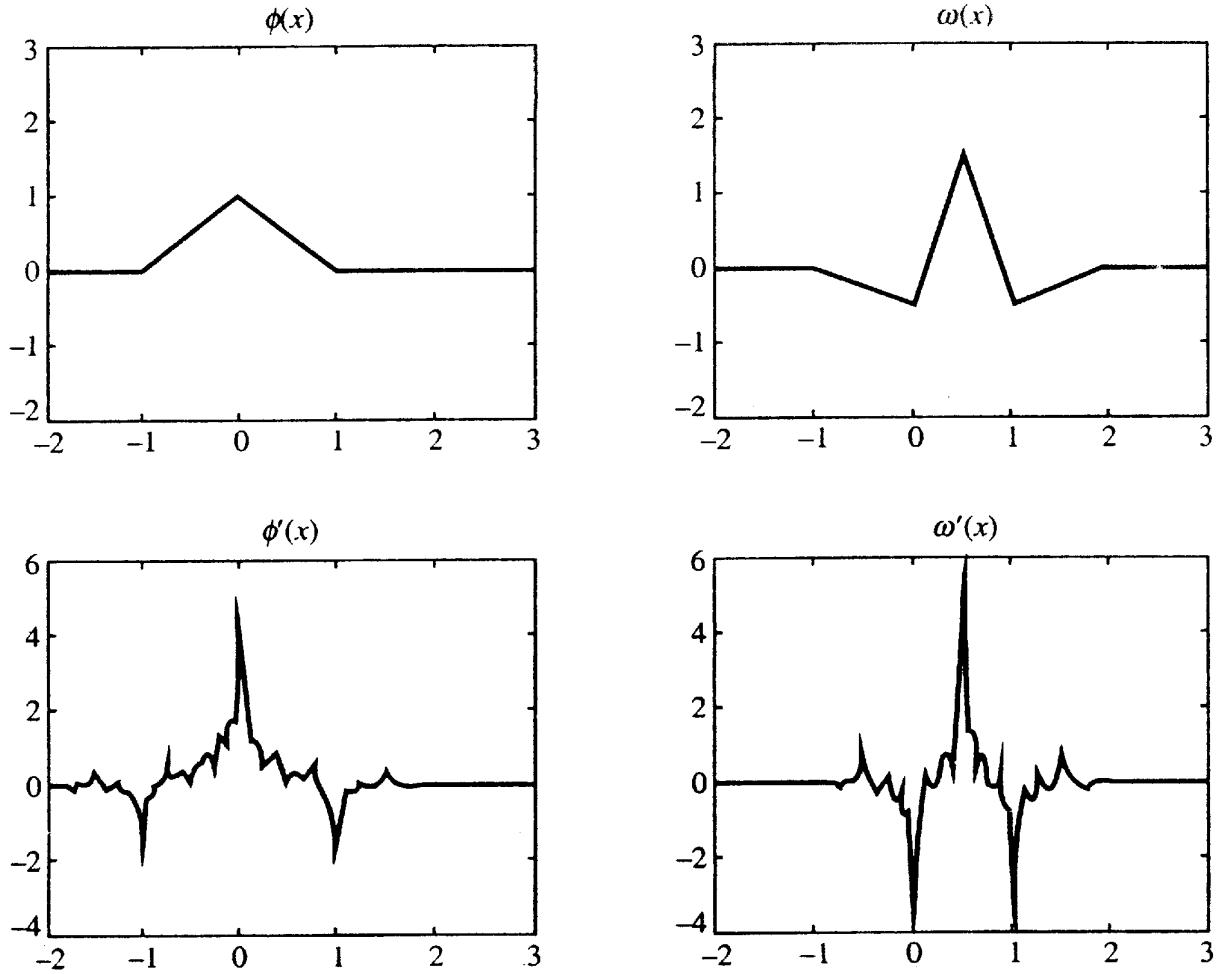


FIGURE 10.20 Biorthogonal wavelet and scaling functions corresponding to filters given in Table 10.4.

10.3.2 Example 2: Lifting Haar Wavelet

Consider the following Haar wavelet system:

$$H(z) = \sqrt{2} \left\{ \frac{1}{2}, \frac{1}{2}z^{-1} \right\}$$

$$G(z) = \sqrt{2} \left\{ \frac{1}{2}, -\frac{1}{2}z^{-1} \right\}$$

$$\tilde{H}(z) = \sqrt{2} \left\{ \frac{1}{2}, \frac{1}{2}z^{-1} \right\}$$

$$\tilde{G}(z) = \sqrt{2} \left\{ \frac{-1}{2}, \frac{1}{2} z^{-1} \right\}$$

Let

$$\psi^{\text{new}}(t) = \psi(t) + \alpha_1 \phi(t+1) + \alpha_2 \phi(t-1)$$

We want new wavelet to have two vanishing moments.

$$\begin{aligned} \int_{-\infty}^{\infty} \psi^{\text{new}}(t) dt &= \int_{-\infty}^{\infty} [\psi(t) + \alpha_1 \phi(t+1) + \alpha_2 \phi(t-1)] dt \\ &= 0 + \alpha_1 + \alpha_2 \end{aligned}$$

Thus,

$$\int_{-\infty}^{\infty} \psi^{\text{new}}(t) dt = 0 \Rightarrow \alpha_1 = -\alpha_2$$

Now,

$$\begin{aligned} \int_{-\infty}^{\infty} t \psi^{\text{new}}(t) dt &= \int_{-\infty}^{\infty} t [\psi(t) + \alpha_1 \phi(t+1) + \alpha_2 \phi(t-1)] dt \\ &= \int_0^{1/2} t dt - \int_{1/2}^1 t dt + \alpha_1 \int_{-1}^0 t dt + \alpha_2 \int_1^2 t dt \\ &= \frac{1}{8} - \frac{3}{8} - \frac{\alpha_1}{2} + \frac{3\alpha_2}{2} = 0 \end{aligned}$$

Solving above two equations, we obtain $\alpha_1 = -1/8$ and $\alpha_2 = 1/8$.

$$\begin{aligned} \psi^{\text{new}}(t) &= \psi(t) - \frac{1}{8} \phi(t+1) + \frac{1}{8} \phi(t-1) \\ &= \phi(2t) - \phi(2t-1) - \frac{1}{8} [\phi(2t+1) + \phi(2t+2)] + \frac{1}{8} [\phi(2t-2) + \phi(2t-3)] \end{aligned}$$

So,

$$G^{\text{new}}(z) = \sqrt{2} \left\{ \frac{-1}{16} z^2 + \frac{-1}{16} z^1 + \frac{1}{2} + \frac{1}{2} z^{-1} + \frac{1}{16} z^{-2} + \frac{1}{16} z^{-3} \right\}$$

Proceeding as in the first example, we obtain

$$\tilde{H}^{\text{new}}(z) = \sqrt{2} \left\{ \frac{-1}{16} + \frac{1}{16} z^{-1} + \frac{-1}{2} z^{-2} + \frac{1}{2} z^{-3} + \frac{1}{16} z^{-4} + \frac{-1}{16} z^{-5} \right\}$$

The four filters are given in Table 10.5. Observe that $g^{\text{new}}(z)$ is orthogonal to $\tilde{h}^{\text{new}}(z)$. It turns that this is precisely one of the biorthogonal filters of Cohen et al. [7].

TABLE 10.5 Final Filters

k	z^2	z^1	z^0	z^{-1}	z^{-2}	z^{-3}	z^{-4}	z^{-5}
\tilde{h}^{new}				$\sqrt{2} \cdot \frac{-1}{16}$	$\sqrt{2} \cdot \frac{1}{16}$	$\sqrt{2} \cdot \frac{-1}{2}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{16}$
\tilde{g}				$\sqrt{2} \cdot \frac{-1}{2}$	$\sqrt{2} \cdot \frac{1}{2}$			
h				$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{2}$			
g^{new}	$\sqrt{2} \cdot \frac{-1}{16}$	$\sqrt{2} \cdot \frac{-1}{16}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{2}$	$\sqrt{2} \cdot \frac{1}{16}$	$\sqrt{2} \cdot \frac{1}{16}$		

10.4 MATHEMATICAL PRELIMINARIES FOR POLYPHASE FACTORIZATION

We outline some of the background mathematical ideas relating to wavelets and the lifting scheme.

10.4.1 Laurent Polynomial

A *Laurent polynomial* is an expression of the form:

$$p(z) = \sum_{k=a}^b c_k z^{-k} \quad (10.37)$$

We define the *degree* of a non-zero Laurent polynomial as $|p(z)| = b - a$ and the *degree* of the zero polynomial is defined to be $-\infty$. While computing $|p(z)|$, make sure c_a and c_b are non-zero.

Example: What is the Laurent degree of the monomial

$$p(z) = 3z^{-7}$$

Solution: Here both b and a is 7. Therefore,

$$|p(z)| = 7 - 7 = 0$$

Laurent degree of monomial is zero. Ans.

Example: What is the Laurent degree of the monomial

$$p(z) = 1/3 - 3z^{-7}$$

Solution:

$$p(z) = -3z^{-7} + 1/3$$

Therefore,

$$|p(z)| = 7 - 0 = 7 \text{ Ans.}$$

A *finite filter* may be described as a set of coefficients $h = \{h(k): k \in Z\}$, which are zero outside the finite range $k_b \leq k \leq k_e$. The Z-transform of a finite filter is the Laurent polynomial whose coefficients are the filter coefficients, namely

$$H(z) = \sum_{k=k_b}^{k_e} h(k) z^{-k} \quad (10.38)$$

The limiting points k_b and k_e are arbitrary subject to the restriction that $(k_e - k_b)$ is the length of the filter. Equivalently, the Z-transform is defined only up to a monomial factor.

10.4.2 The Euclidean Algorithm

The *Euclidean algorithm* is a method for finding the greatest common divisor (gcd) of two integers. The division algorithm states that *for any integers a and b , where $\text{abs}(a) > \text{abs}(b)$ (here abs denotes absolute value) and $b \neq 0$, there exist integers q_1 and r_1 (the reason for the subscripts will become clear) such that $a = q_1 b + r_1$ with $0 \leq r_1 \leq b$.*

The Euclidean algorithm can be used to find the gcd as follows. Firstly note that if $r_1 = 0$, then $\text{gcd}(a, b) = b$. If $r_1 \neq 0$, we see that any common divisor of b and r_1 is a divisor of a and, thus is a common divisor of a and b . Rewriting the above expression as $r_1 = a - q_1 b$, shows that any common factor of a and b is a factor of r_1 , and is a common factor of r_1 and b . We have shown that the common divisors of a and b are the same as those of r_1 and b , and hence $\text{gcd}(a, b) = \text{gcd}(r_1, b)$. Thus, the problem has been reduced to finding the gcd of the smaller pair of numbers (r_1, b) .

We may now apply the division algorithm to r_1 and b to get $b = q_2 r_1 + r_2$ with $0 \leq r_2 \leq r_1$, and repeating the argument above, we see that $\text{gcd}(r_1, b) = \text{gcd}(r_1, r_2)$. We iterate this process, each time obtaining a smaller remainder until eventually $r_i = 0$ for some i . We then have $r_{i-2} = q_i r_{i-1}$ and, thus we have $\text{gcd}(a, b) = \text{gcd}(r_{i-2}, r_{i-1}) = r_{i-1}$. The algorithm has a simple computational implementation, in recursive form:

$$a_i \leftarrow b_{i-1} \quad (10.39)$$

and

$$b_i \leftarrow a_{i-1} - a_i q_i \quad (10.40)$$

The Euclidean algorithm can be generalized to the Laurent polynomials. The procedure is similar to that shown before, with the generalization being that we require $|r| < |b|$, where $|\cdot|$ is the Laurent degree.

An important point is that the quotients are not unique in the Euclidean algorithm. That is, although the gcd is unique (up to an invertible factor), the quotients can be chosen in several different ways, which differ in more than just a monomial factor. To illustrate this more clearly, we will consider two examples of the Euclidean algorithm, one for integers and one for Laurent polynomials.

Example of the Euclidean algorithm for integers

Suppose we want to find the gcd of 170 and 26. We carry out the algorithm

$$\begin{aligned} 170 &= 26(6) + 14 \\ 26 &= 14(1) + 12 \\ 14 &= 12(1) + 2 \\ 12 &= 2(6) \end{aligned}$$

Thus, the gcd is 2 and the quotients are 6, 1, 1, 6. Alternatively, we have

$$\begin{aligned} 170 &= 26(7) - 12 \\ 26 &= -12(-2) + 2 \\ -12 &= 2(-6) \end{aligned}$$

So we get the same gcd but the quotients are 7, -2, -6.

Example of the Euclidean algorithm for Laurent polynomials

Since gcd is unique up to an invertible (monomial) factor, the quotients can be chosen in several non-trivially distinct ways. We illustrate this with Laurent polynomials. Let $a = (1/z) + 3 - 2z$ and $b = [(-6/z^2) + (3/z)]$. We wish to find gcd (a, b). Note that $|a| = 1 - (-1) = 2$ while $|b| = -1 - (-2) = 1$. We set up the first step of the Euclidean algorithm:

$$\frac{1}{z} + 3 - 2z = q_1 \left(\frac{-6}{z^2} + \frac{3}{z} \right) + r_1$$

Since the remainder must have degree less than that of b , it must be monomial. Rearranging the equation for r_1 , we get

$$r_1 = \left(\frac{1}{z} + 3 - 2z \right) - q_1 \left(\frac{-6}{z^2} + \frac{3}{z} \right)$$

We must choose q_1 so that r_1 will be monomial. One way to do this is to choose q_1 so that the highest monomials in each term in this expression are equal and will thus cancel.

The highest term in $[(1/z) + 3 - 2z]$ is z , so we will try to cancel the z term from each. This means that q_1 will need to be of the form $z^2[c + (d/z)]$ in order for the terms to have the same highest exponents. The parameters c and d are then found by the method of undetermined coefficients. Substituting, we find that the first remainder has the form:

$$\begin{aligned} r_1 &= \left(\frac{1}{z} + 3 - 2z \right) - z^2 \left(c + \frac{d}{z} \right) \left(\frac{-6}{z^2} + \frac{3}{z} \right) = \left(\frac{1}{z} + 3 - 2z \right) + \left(6c - 3cz + \frac{6d}{z} - 3d \right) \\ &= z(-3c - 2) + 3 + 6c - 3d + \frac{6d + 1}{z} \end{aligned}$$

Setting the coefficients of the z and constant terms to zero, we get

$$c = -\frac{2}{3}, \quad d = -\frac{1}{3}$$

Thus, our first quotient is:

$$q_1 = z^2 \left(c + \frac{d}{z} \right) = \frac{-2z^2}{3} - \frac{z}{3}$$

and the first remainder is:

$$r_1 = \frac{6d + 1}{z} = \frac{-1}{z}$$

We now apply the Euclidean algorithm to this remainder and b to get

$$b = q_2 r_1 + r_2 \Rightarrow r_2 = b - q_2 r_1$$

Once again, the remainder r_2 must have degree less than that of r_1 . Since r_1 has degree 0, it follows that r_2 must be zero (recall that the degree of the zero polynomial is defined as $-\infty$). As before, we choose a quotient form so that both powers are matched and find that the second remainder has the form $(c + 3)/z + (d - 6)z^2$.

We can set both terms and, thus, the remainder to zero to get $c = -3$, $d = 6$, so the second quotient is $(6/z) - 3$. Alternatively, we could have decided to eliminate the lowest two powers from the original expression or even the lowest and the highest power. This gives three possible factorizations, each with different quotients. The gcd here is defined only up to a power of z : the gcd is ‘greatest’ in the sense that it has greatest possible Laurent degree. In this case, we found that the gcd was $-1/z$ (the last non-zero remainder). The other factorization branches lead to a monomial gcd also. In this case the original Laurent polynomial a can be retrieved via $a = q_1 b + r$ where r is the gcd and the reader can easily verify that the following three factorizations are all valid:

$$a = \left(\frac{-2z^2}{3} + \frac{-z}{3} \right) b - \frac{1}{z}, \quad a = \left(\frac{2z^2}{3} + \frac{-z}{6} \right) b - \frac{1}{2}, \quad a = \left(\frac{-7z^2}{12} + \frac{-z}{6} \right) b - \frac{z}{4}$$

The gcd is the same for each factorization up to a monomial factor but the first quotients differ non-trivially. The last quotients differ only by monomial factors because at this stage we have no freedom over which powers to eliminate.

Note that multiplying a by z and b by z^2 converts these Laurent polynomials into (ordinary) polynomials in z . Polynomial division then gives

$$az = -2z^2 + 3z + 1 = \left(-\frac{2z}{3} - \frac{1}{3} \right) (bz^2) - 1$$

which is equivalent to the first factorization above, providing a simple computational check. Similarly, the last factorization can be obtained by dividing a/z by bz (since these are ordinary polynomials in a/z), i.e.,

$$\frac{a}{z} = -2 + \frac{3}{z} + \frac{1}{z^2} = \left(-\frac{7}{12} - \frac{1}{6z} \right) (bz) - \frac{1}{4}$$

Furthermore, there are fast algorithms for polynomial division and these can be used for the computations. However, the second factorization cannot be obtained in this way.

In the next section, we will factorize certain 2×2 matrices whose entries are Laurent polynomials (specifically Z-transforms as defined above) by executing the Euclidean algorithm

on two of the entries. The factors are matrices whose entries are determined by the quotients resulting from the algorithm. Thus, using the different sets of quotients, we can obtain several different matrix factorizations.

10.4.3 Factoring Wavelet Transform into Lifting Steps—A Z-domain Approach

At the start of this chapter, we have shown how wavelet transform expressed in time-domain matrix form is converted into a polyphase matrix. Elements of polyphase matrix are Laurent polynomials. Equation (10.5)

$$\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} h_0 + \frac{h_2}{z} & h_1 + \frac{h_3}{z} \\ zg_0 + g_2 & zg_1 + g_3 \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} = P(z) \begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$$

can be written for a general wavelet decomposition as:

$$\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \\ G_{\text{even}}(z) & G_{\text{odd}}(z) \end{bmatrix} \begin{bmatrix} S(z) \\ D(z) \end{bmatrix} \quad (10.41)$$

$\begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$ represents even and odd indexed part of signal. This separation of signal into even and odd indexed parts is the first step in lifting scheme based wavelet transform. The factorization of $P(z)$ into elementary matrices leads to faster and simpler implementation of wavelet transform. The whole process can be schematically depicted in Figure 10.21.

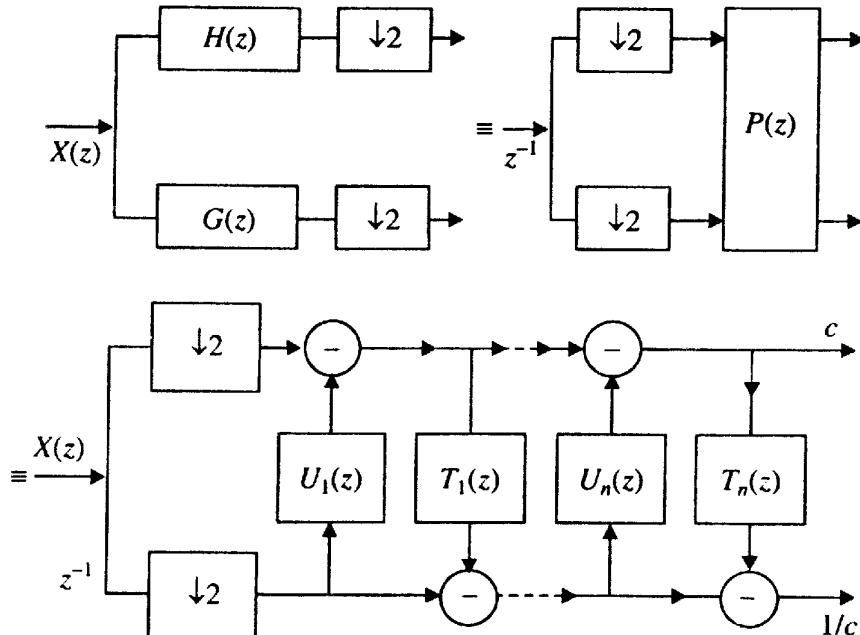


FIGURE 10.21 | Factorization of polyphase matrix.

Our goal is to perform the above change of representation of the form.

The approach is based on Euclidean algorithm for greatest common divisor. We start with

$$A_0(z) = H_{\text{even}}(z) \quad (10.42)$$

and

$$B_0(z) = H_{\text{odd}}(z) \quad (10.43)$$

Then we iterate using the following assignments and operations:

$$A_i(z) = B_{i-1}(z) \quad (10.44)$$

and

$$B_i(z) = A_{i-1}(z) \% B_{i-1}(z) = A_{i-1}(z) - Q_i(z) B_{i-1}(z) \quad (10.45)$$

where $\%$ is the remainder operator and $Q_i(z)$ is the quotient obtained when $A_{i-1}(z)$ is divided by $B_{i-1}(z)$. As shown earlier, the quotient is not unique.

Repeat until $i = n$ at which stage

$$A_n(z) = c; \quad B_n(z) = 0$$

where c is gcd [$H_{\text{even}}(z)$, $H_{\text{odd}}(z)$].

The two equations,

$$A_i(z) = B_{i-1}(z) \quad (10.46)$$

$$B_i(z) = A_{i-1}(z) - Q_i(z) B_{i-1}(z) \quad (10.47)$$

can be written in matrix form as:

$$\begin{bmatrix} A_i(z) & B_i(z) \end{bmatrix} = \begin{bmatrix} A_{i-1}(z) & B_{i-1}(z) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -Q_i(z) \end{bmatrix} \quad (10.48)$$

If we start writing from $A_0(z)$ and $B_0(z)$, we have

$$\begin{aligned} \begin{bmatrix} A_1(z) & B_1(z) \end{bmatrix} &= \begin{bmatrix} A_0(z) & B_0(z) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -Q_1(z) \end{bmatrix} \\ \begin{bmatrix} A_2(z) & B_2(z) \end{bmatrix} &= \begin{bmatrix} A_1(z) & B_1(z) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -Q_2(z) \end{bmatrix} \\ &= \begin{bmatrix} A_0(z) & B_0(z) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -Q_1(z) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -Q_2(z) \end{bmatrix} \end{aligned}$$

So after n iterations, we obtain

$$\begin{aligned} \begin{bmatrix} c & 0 \end{bmatrix} &= \begin{bmatrix} A_0(z) & B_0(z) \end{bmatrix} \prod_{i=1}^n \begin{bmatrix} 0 & 1 \\ 1 & -Q_i(z) \end{bmatrix} \\ &= \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \end{bmatrix} \prod_{i=1}^n \begin{bmatrix} 0 & 1 \\ 1 & -Q_i(z) \end{bmatrix} \end{aligned} \quad (10.49)$$

Inverting this result, we obtain

$$\begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \end{bmatrix} = \begin{bmatrix} c & 0 \end{bmatrix} \prod_{i=n}^1 \begin{bmatrix} Q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \quad (10.50)$$

Suppose that n is even ($n = 2m$), we can obtain a polyphase matrix of the form:

$$\hat{P}(z) = \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \prod_{i=2m}^1 \begin{bmatrix} Q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \quad (10.51)$$

In matrix $\begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix}$, in the second row $1/c$ is chosen because it ensures that determinant of $\hat{P}(z)$ is 1. A valid polyphase matrix should have 1 as its determinant. The resulting polyphase matrix is obviously not the one we require.

Let

$$\hat{P}(z) = \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \\ \hat{G}_{\text{even}}(z) & \hat{G}_{\text{odd}}(z) \end{bmatrix} \quad (10.52)$$

To recover $P(z)$ from $\hat{P}(z)$, we introduce one more lifting step. The idea is illustrated in Figure 10.22.

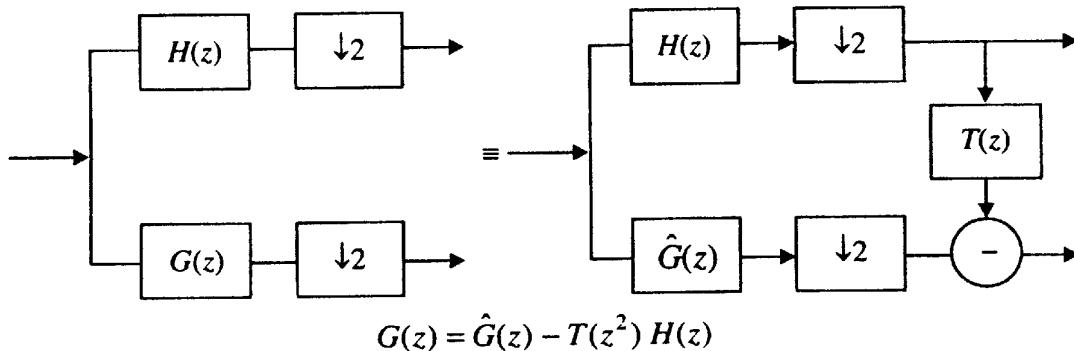


FIGURE 10.22 A lifting to retrieve $P(z)$ from $\hat{P}(z)$.

From the relation,

$$G(z) = \hat{G}(z) - T(z^2)H(z)$$

we obtain

$$G_{\text{even}}(z) = \hat{G}_{\text{even}}(z) - T(z)H_{\text{even}}(z) \quad (10.53)$$

and

$$G_{\text{odd}}(z) = \hat{G}_{\text{odd}}(z) - T(z)H_{\text{odd}}(z) \quad (10.54)$$

In matrix form,

$$\begin{bmatrix} G_{\text{even}}(z) & G_{\text{odd}}(z) \end{bmatrix} = \begin{bmatrix} -T(z) & 1 \end{bmatrix} \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \\ \hat{G}_{\text{even}}(z) & \hat{G}_{\text{odd}}(z) \end{bmatrix} = \begin{bmatrix} -T(z) & 1 \end{bmatrix} \hat{P}(z) \quad (10.55)$$

Therefore,

$$\begin{aligned} \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \\ G_{\text{even}}(z) & G_{\text{odd}}(z) \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ -T(z) & 1 \end{bmatrix} \hat{P}(z) \\ &= \begin{bmatrix} 1 & 0 \\ -T(z) & 1 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \prod_{i=2m}^1 \begin{bmatrix} Q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \end{aligned} \quad (10.56)$$

Further, it can be easily shown that

$$\begin{bmatrix} 1 & 0 \\ -T(z) & 1 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} = \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c^2 T(z) & 1 \end{bmatrix} \quad (10.57)$$

Using the above result, we obtain

$$\begin{aligned} \begin{bmatrix} H_{\text{even}}(z) & H_{\text{odd}}(z) \\ G_{\text{even}}(z) & G_{\text{odd}}(z) \end{bmatrix} &= \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c^2 T(z) & 1 \end{bmatrix} \prod_{i=2m}^1 \begin{bmatrix} Q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c^2 T(z) & 1 \end{bmatrix} \prod_{k=m}^1 \begin{bmatrix} Q_{2k}(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} Q_{2k-1}(z) & 1 \\ 1 & 0 \end{bmatrix} \end{aligned} \quad (10.58)$$

Rewriting each factor in the product as a permutation of columns or rows, we obtain

$$\begin{aligned} \begin{bmatrix} Q_{2k}(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} Q_{2k-1}(z) & 1 \\ 1 & 0 \end{bmatrix} &= \begin{bmatrix} 1 & Q_{2k}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ Q_{2k-1}(z) & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & Q_{2k}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ Q_{2k-1}(z) & 1 \end{bmatrix} \end{aligned} \quad (10.59)$$

So $P(z)$ can be written as:

$$P(z) = \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c^2 T(z) & 1 \end{bmatrix} \prod_{k=m}^1 \begin{bmatrix} 1 & Q_{2k}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ Q_{2k-1}(z) & 1 \end{bmatrix} \quad (10.60)$$

Now, the wavelet transform $\begin{bmatrix} S^{\text{new}}(z) \\ D^{\text{new}}(z) \end{bmatrix} = [P(z)] \begin{bmatrix} S(z) \\ D(z) \end{bmatrix}$ can be visualized as shown in Figure 10.23.

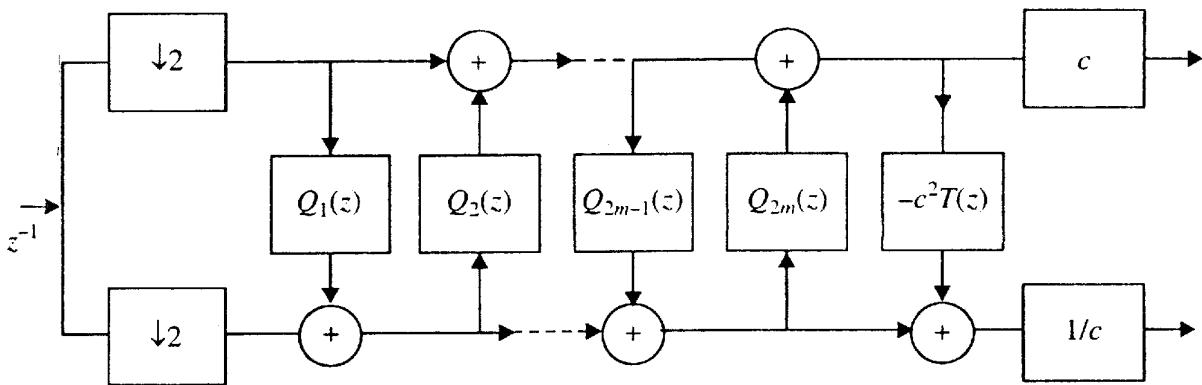


FIGURE 10.23 Lifting scheme.

Example: Factoring Haar wavelet polyphase matrix.

For Haar, we have

$$H(z) = \frac{1}{\sqrt{2}}(1 + z^{-1}), \quad H_{\text{even}}(z) = \frac{1}{\sqrt{2}}, \quad H_{\text{odd}}(z) = \frac{1}{\sqrt{2}}$$

and

$$G(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$$

Now,

$$A_0(z) = H_{\text{even}}(z) = \frac{1}{\sqrt{2}}$$

and

$$B_0(z) = H_{\text{odd}}(z) = \frac{1}{\sqrt{2}}$$

Then

$$A_1(z) = B_0(z) = \frac{1}{\sqrt{2}} = c \quad (c \text{ is the gcd})$$

and

$$B_1(z) = A_0(z) \% B_0(z) = 0$$

Therefore,

$$Q_1(z) = A_0(z)/B_0(z) = 1$$

Since the remainder is zero, iterations stop here.

$$\begin{aligned} \hat{P}(z) &= \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} Q_1(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \end{aligned}$$

From $\hat{P}(z)$, we observe that $\hat{G}(z) = \frac{2}{\sqrt{2}}$. We now extract $G(z)$ from $\hat{G}(z)$ using the following relation (Refer Figure 10.23).

$$G(z) = \hat{G}(z) - T(z^2) H(z)$$

$$\frac{1}{\sqrt{2}}(1 - z^{-1}) = \frac{2}{\sqrt{2}} - T(z^2) \frac{1}{\sqrt{2}}(1 + z^{-1})$$

This implies $T(z^2) = 1$ and hence $T(z) = 1$.

$$P(z) = \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c^2 T(z) & 1 \end{bmatrix} \begin{bmatrix} Q_1(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1/2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

10.5 DEALING WITH SIGNAL BOUNDARY

A digital filter is applied to a signal by convolution [see Section 2.3 and Eqs. (2.4) and (2.9)]. Convolution, however, is defined only *within* a signal. In order to result in a mathematically correct, reversible wavelet transform, *each* signal coefficient must enter into filter length/2 calculations of convolution (here, the subsampling process by factor of 2 is already incorporated). Consequently, every filter longer than 2 coefficients or *taps*, i.e., every filter except *Haar*, requires a solution for the boundary coefficients of the signal.

The boundary treatment becomes even more important the shorter the analyzed signal is. An audio piece, considered as a one-dimensional discrete signal over time and at a sampling rate of 44100 samples per second contains so many coefficients in its interior that—*independent* of its actual length—the boundary plays only a subordinate role. Still images, however, are signals of a relatively short length (in rows and columns), thus the boundary treatment is very important. Two common boundary policies are *circular convolution* and *padding*.

10.5.1 Circular Convolution

The idea of circular convolution is to ‘wrap’ the signal around at the boundary, i.e., wrap the end of a signal to the beginning (or vice versa). Figure 10.24 (a) illustrates this approach with a signal of length 8 and a filter of 6 taps. The convolution of the signal entries 1 to 6 with the filter results in the entry *a* in the time-scale domain. In the same manner, the convolution of the signal entries 3 to 8 with the filter results in the entry *b* in the time-scale domain. The process has not finished yet, but for the next convolution, the signal entries 5 to 8 are not enough and two more signal entries are required. Furthermore, the entries 1 and 2 need to be included in *filter_length/2*, i.e., in 3 calculations. Thus, they are being ‘wrapped’ around and enter the calculation of the time-scale coefficient *c*. The same is done with *d*.

Figure 10.24 demonstrates the convolution with a low-pass filter. The number of approximation coefficients is only half as many. A convolution with a high-pass filter would produce an equal number of detail coefficients.

In doing so, circular convolution is the only boundary treatment that maintains the number of coefficients for a wavelet transform, thus simplifying storage handling. However, the time information contained in the time-scale domain of the wavelet transformed coefficients ‘blurs’.

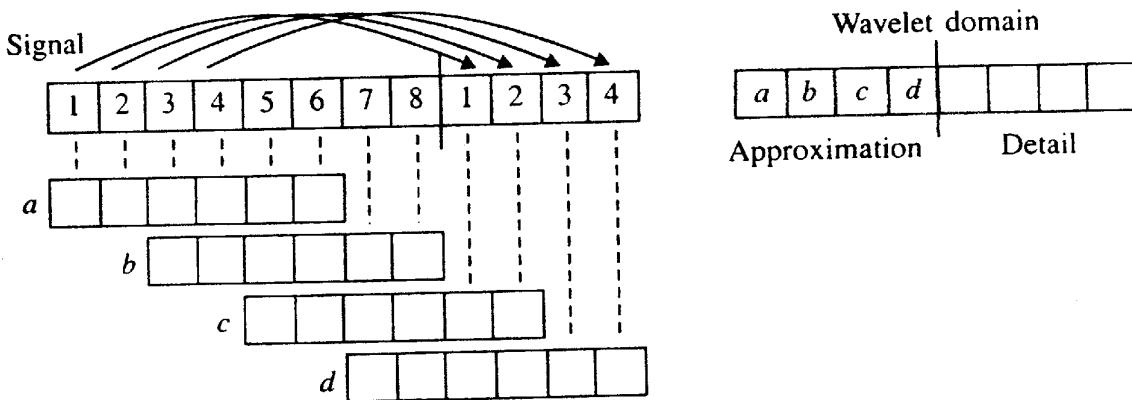


FIGURE 10.24 Circular convolution.

The coefficients in the time-scale domain that are next to the right border (respectively, left border) also affect signal coefficients that are located on the left (respectively, on the right). This means that in Figure 10.24, the information on pixels 1 and 2 of the left border of the original signal is contained not only in entries of the time-scale domain that are located on the left, but also on the right side, i.e., in the entries a , c and d of the time-scale domain. c and d are the coefficients that, due to circular convolution, contain information from the ‘other’ side of the signal.

10.5.2 Padding Policies

Padding policies add coefficients to the signal on either of its borders. The border pixels of the signal are padded with *filter_length*-2 coefficients. Consequently, each signal coefficient enters into *filter_length*/2 calculations of convolution and the transform is reversible. Many padding policies exist: *zero padding*, where ‘0’s are added, *constant padding*, where the signal’s boundary coefficient is padded, *mirror padding*, where the signal is mirrored at the boundary and *spline padding*, where the last n_0 border coefficients are extended by spline interpolation, etc.

In all the padding policies, the storage space in the wavelet domain is physically enlarged by each iteration step, see Figure 10.25. Though the amount of storage space required can be

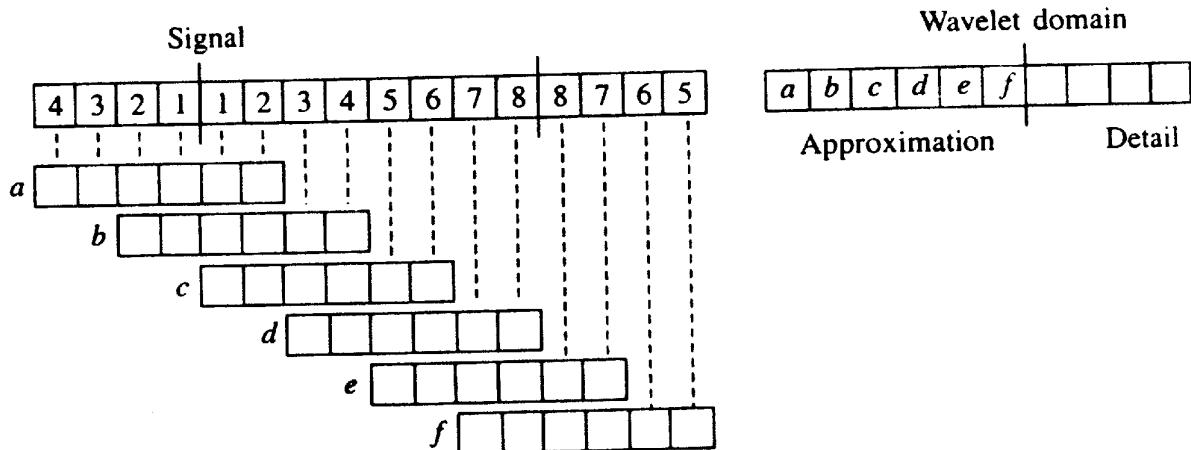


FIGURE 10.25 Mirror padding.

calculated in advance, it nevertheless remains sophisticated. Storage handling is strongly affected by the fact that only the *iterated* parts expand, thus expansion depends on the selected decomposition policy and on the iteration level. A strength of all padding approaches, however, is that the time information is preserved. This means that signal coefficients that are located on the left (respectively, on the right) are represented by time-scale coefficients at the same location.

10.5.3 Iteration Behaviour

Convolving the signal with a filter is reasonable only for a signal length greater than the filter length and each iteration step reduces the size of the approximating signal by a factor of 2. This does not affect the iteration behaviour of padding policies. In circular convolution, however, the decomposition depth varies with the filter length: the longer the filter, the fewer decomposition iterations are possible. For an image of 256×256 pixels, the Daubechies-2 filter bank with 4 taps allows a decomposition depth of seven levels while the Daubechies-20 filter bank with 40 taps reaches signal length after only three decomposition levels. Table 10.6 gives some more iteration levels for circular convolution.

TABLE 10.6 Iteration Levels for Circular Convolution

<i>Filter bank</i>	<i>Taps</i>	<i>Iterations</i>
Daubechies-2	4	7
Daubechies-3	6	6
Daubechies-4	8	6
Daubechies-5	10	5
Daubechies-10	20	4
Daubechies-15	30	4
Daubechies-20	40	3

Depending on the selected quantization policy for compression, the number of iterations can strongly affect the quality of a decoded signal.

SUMMARY

In this chapter we have shown how discrete wavelet transform or two-band subband filtering with finite filters can be decomposed into a finite sequence of simple filtering steps which we call as lifting steps. The decomposition correspond to factorization of the polyphase matrix of the wavelet filters into elementary matrices. Lifting scheme was then geometrically interpreted. Finally, we used lifting concept to derive new improved biorthogonal wavelets from existing biorthogonal wavelets.

EXERCISES

- 10.1** Find $H^{\text{new}}(z)$ for the following identity:

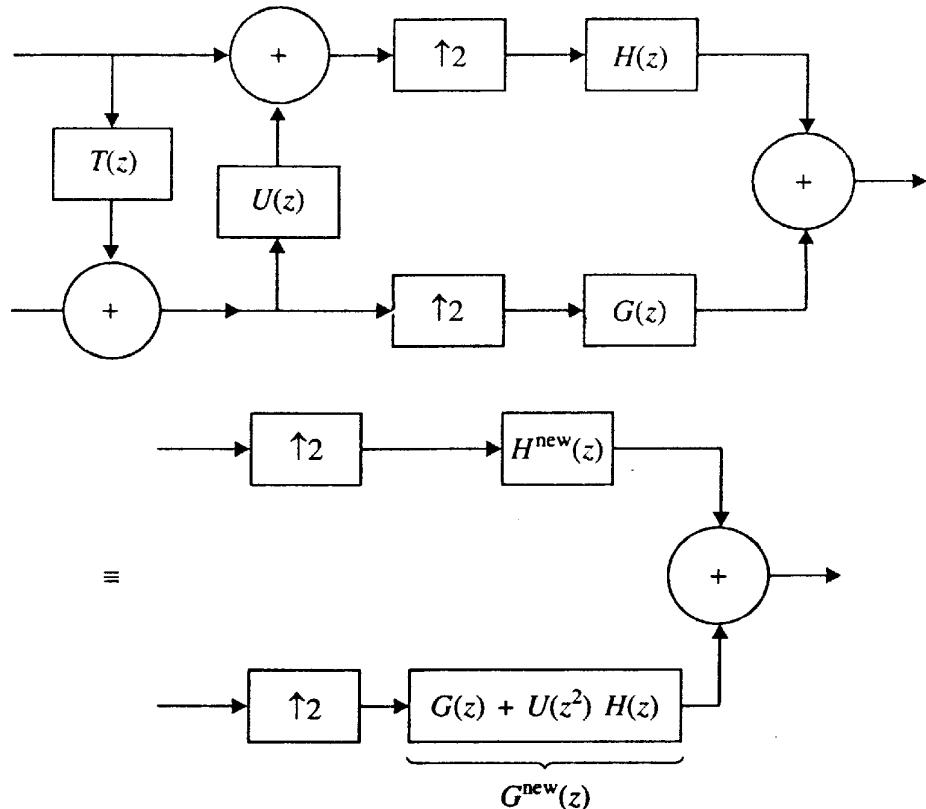
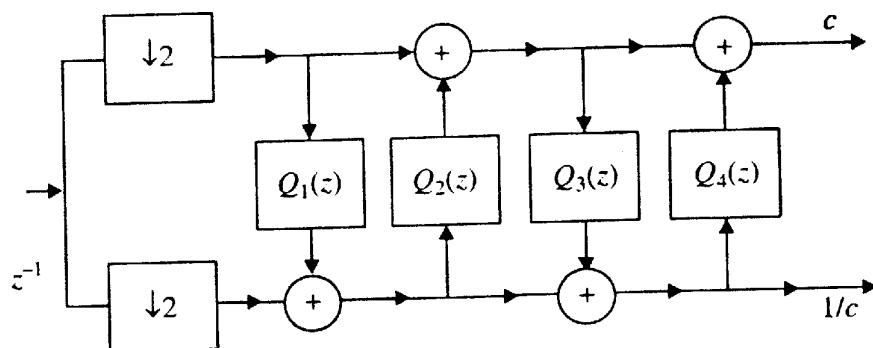


FIGURE 1 Finding equivalent filters.

- 10.2** Write a MATLAB code for finding wavelet coefficients using lifting scheme. Code for Haar and Daubechies 4-tap wavelets.
- 10.3** Find the inverse transform for the following forward wavelet transform in lifting steps:

$$\begin{aligned}
 s_l &\leftarrow x_{2l} \\
 d_l &\leftarrow x_{2l+1} \\
 s_l &\leftarrow s_l - 0.412287d_l \\
 d_l &\leftarrow d_l + 0.352388s_l - 1.56514s_{l+1} \\
 s_l &\leftarrow 0.492152d_{l-1} + 0.0284591d_l + s_l \\
 d_l &\leftarrow d_l - 0.38962s_l \\
 s_l &\leftarrow 1.9182s_{l+1} \\
 d_l &\leftarrow 0.521321d_{l-1}
 \end{aligned}$$

- 10.4 Find the scaling and wavelet filters corresponding to the lifting scheme shown below.



$$\begin{aligned}
 Q_1(z) &= \alpha(1 + z) & \alpha &= -1.586134342 \\
 Q_2(z) &= \beta(1 + z^{-1}) & \beta &= -0.05298011854 \\
 Q_3(z) &= \gamma(1 + z) & \gamma &= 0.8829110762 \\
 Q_4(z) &= \delta(1 + z^{-1}) & \delta &= 0.4435068522 \\
 c & & c &= 1.149604398
 \end{aligned}$$

FIGURE 2 Wavelet transform using lifting scheme.

REFERENCES

- [1] Daubechies and W. Sweldens, Factoring wavelet transforms into lifting steps, *Journal of Fourier Analysis and Applications*, **4**(3): 245–267, 1998.
- [2] Sweldens, W., The Lifting Scheme: A construction of second-generation wavelets, *SIAM Journal on Mathematical Analysis*, **29**(2):511–546, 1997.
- [3] Sweldens, W. and P. Schröder, Building your own wavelets at home, *ACM SIGGRAPH Course Notes*, pp. 15–87, 1996.
- [4] Zorin D., P. Schröder and W. Sweldens, Interpolating Subdivision for meshes with arbitrary topology, *Computer Graphics Proceedings (SIGGRAPH 96)*, pp. 189–192, 1996.
- [5] Zorin, D., et al., Subdivision for modeling and animation. *SIGGRAPH 2000 Course Notes*, 2000.
- [6] Sweldens, W., The Lifting Scheme: A custom-design of biorthogonal wavelets. *Journal of Applied and Computational Harmonic Analysis*, **3**:186–200, 1996.
- [7] Cohen, A., I. Daubechies, and J. Feauveau, Biorthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, **45**, 485–560, 1992.

CHAPTER

11

Image Compression

INTRODUCTION

Data compression is the process of converting data files into smaller files for efficiency of storage and transmission. As one of the enabling technologies of the multimedia revolution, data compression is a key to rapid progress being made in information technology. It would not be practical to put images, audio and video alone on websites without compression.

What is data compression? Why do we need it? You may have heard of JPEG (Joint Photographic Experts Group) and MPEG (Moving Pictures Experts Group), which are standards for representing images and video. Data compression algorithms are used in those standards to reduce the number of bits required to represent an image or a video sequence. Compression is the process of representing information in a compact form. Data compression treats information in digital form, that is, as binary numbers represented by bytes of data with very large data sets. For example, a single small "4 × 4" size colour picture, scanned at 300 dots per inch (dpi) with 24 bits/pixel of true colour, will produce a file containing more than 4 megabytes of data. At least three floppy disks are required to store such a picture. This picture requires more than one minute for transmission by a typical transmission line (64 k bit/second ISDN). That is why large image files remain a major bottleneck in a distributed environment. Although increasing the bandwidth is a possible solution, the relatively high cost makes this less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes. In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. There have been many types of compression algorithms developed. These algorithms fall into two broad types: *lossless algorithms* and *lossy algorithms*. A lossless algorithm reproduces the original exactly. A lossy algorithm, as its name implies, loses some data. Data loss may be unacceptable in many applications. For example, text compression must be lossless because a very small difference can result in statements with totally different meanings. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for

example, the exact reconstructed value of each sample of the image is not necessary. Depending on the quality required of the reconstructed image, varying amounts of loss of information can be accepted.

11.1 OVERVIEW OF IMAGE COMPRESSION TECHNIQUES

Currently used image compression techniques can be separated into two groups: *lossless* and *lossy* compression. Lossless compression allows the reconstruction of the original image data from the compressed image data.

The underlying idea of any compression scheme is to remove the correlation present in the data. Correlated data is characterized by the fact that one can, given one part of the data, fill in the missing part.

Several types of correlation exist. Some examples are:

Spatial correlation: One can often predict the value of a pixel in an image by looking at the neighbouring pixels.

Spectral correlation: The Fourier transform of a signal is often smooth. This means that one can predict one frequency component by looking at the neighbouring frequencies.

Temporal correlation: In a digital video, most pixels of two neighbouring frames change very little in the time direction (e.g. the background).

One of the standard procedures for lossy compression is through transform coding, as indicated in Figure 11.1, where *discrete cosine transform* is used for transformation of pixel

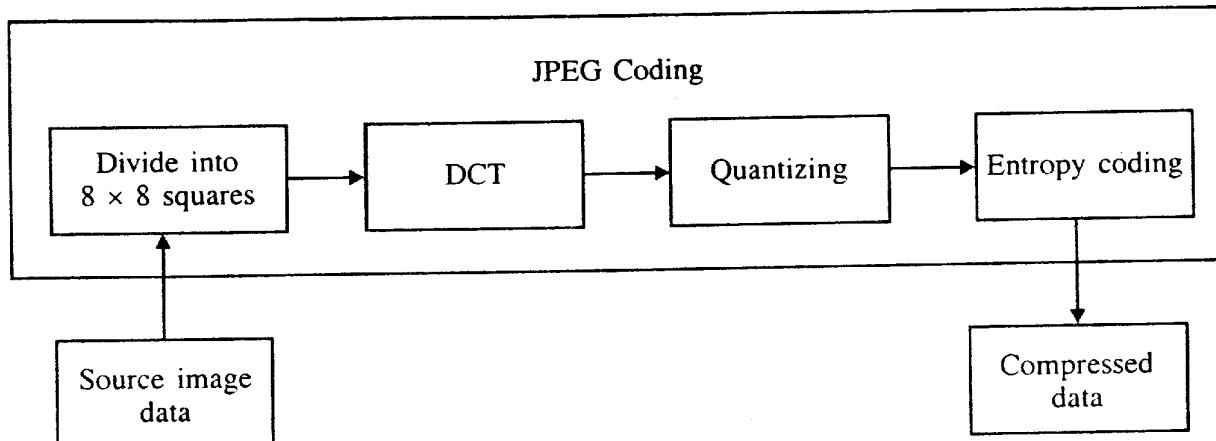


FIGURE 11.1 Simplified JPEG coding scheme for lossy compression.

data. The idea is to represent the data using a different mathematical basis in the hope that this new representation will reveal or unravel the correlation. By this we mean that in the new basis, the majority of the coefficients are so small that they can be set to zero. The information is thus packed into a small number of coefficients. Compression is achieved by calculating the transform associated with this basis, setting coefficients below a threshold to zero and lossless encoding of the non-zero coefficients.

In case one knows precisely the correlation present in a data set, it is possible to find the optimal transform. It is the so-called Karhunen-Loeve representation. The optimal basis, the one with the best information packing quality, is given by the eigenvectors of the correlation matrix. This theoretical optimal representation, however, has several practical disadvantages:

1. In most cases the correlation matrix is not known.
2. The algorithm to calculate the eigenvectors of a matrix has cubic complexity. Given the fact that the dimension of the problem in the case of image compression is 512×512 , we realize that it is impossible to compute the Karhunen-Loeve basis.
3. Suppose one knows the optimal basis, calculating the transform is a quadratic algorithm, which in most cases still is unacceptable.
4. The basis depends on the data set. It can only be used in case one knows precisely which set the data belong to.

This tells us that we need a transform with the following properties:

1. The transform is independent of the data set.
2. A fast (linear or linear-logarithmic) algorithm to calculate the transform exists.
3. The transform is capable of removing the correlation for a large, general set of data.

A possible candidate for a transform is the Fast Fourier Transform (FFT). It definitely has the first two properties. However, it does not always have the third property. The basis functions are perfectly local in frequency, but not local at all in time. Therefore, it is unable to reveal local temporal correlation. Most signals have both local frequency and spatial correlation. We need a transform that is adapted to this behaviour. More precisely, we need a basis which is local in time and frequency. There are two ways to construct such a basis:

1. One can divide the spatial domain into pieces and use a Fourier series on each piece separately. This way one gets a local trigonometric basis.
2. One can use a wavelet basis.

Both these methods result in a transform, which is data-independent, fast and which yields a compact representation for a large, general set of data. In both the cases, one can allow some limited but quite powerful data-dependency. This is done by considering a family of closely related bases, out of which one can select the best one. This process is called **best basis selection**.

With lossy compression a higher compression rate is possible by allowing small differences between original and reconstructed images. Compression rates for natural images with lossless compression are generally small, ranging from 1 : 1 for uncompressable data to 1 : 3. Typical values, using TIF-LZW encoding for example, are around 1 : 1.5. Lossy compression achieves compression rates of 1 : 5 to 1 : 30 using standard techniques (e.g. JPEG) and up to 1 : 300 using newer techniques (e.g. wavelet LWF, WV; fractal compression FIF). As mentioned earlier, most lossy compression techniques are based on two-dimensional transforms, followed by quantization and encoding stages. The loss of information is introduced by the quantization stage which intentionally rejects less relevant parts of the image information (disregarding rounding errors in the transformation step). Specially adapted techniques which

achieve high compression rates exist for artificial images. These techniques are based on Run Length Encoding (RLE), sometimes followed by an entropy coder. Examples of this techniques are: the fax-encoding standard and the image formats PCX and RLE. The confusingly large number of compression techniques prevents many potential users from directly applying image compression. A technique and an image format which covers a wide range of alternative compression requirements would be preferable. JPEG is one attempt at creating such a technique for natural images, assembling a number of compression techniques in a common standard. The following section describes its functionality. While JPEG integrates a wide range of compression techniques, these methods can only be realized by introducing several very different working modes. In contrast wavelet compression allows the integration of various compression techniques into one algorithm.

TABLE 11.1 Overview of Image Compression Techniques

<i>Techniques</i>			
Lossless		<ul style="list-style-type: none"> • Entropy coder (e.g. Huffman, LZW) • Run length coding 	
Lossy		<ul style="list-style-type: none"> • Transformation (DCT, DWT, Fractal transformation) • Quantizing “Simple” Qu (Linear, Uniform, Characteristic) Bit plane quantization, Successive approximation Qu • Encoding Entropy coder, Quadtree coding 	
<i>Achievable compression rates</i>			
	<i>Natural images (Photos, Scans)</i>	<i>Artificial images (Drawings, Cartoons)</i>	<i>Image file formats</i>
Lossless	1 : 1.5–1 : 2	1 : 1.5–1:20	<ul style="list-style-type: none"> • TIFF-LZW • Fax, BMP (RLE), PCX
Lossy	<ul style="list-style-type: none"> • 1 : 1.5–1 : 30 visual lossless • 1 : 10–1 : 300 lossy 	1 : 1.5–1 : 300	<ul style="list-style-type: none"> • JPG • LWF, WV • FIF

11.1.1 The JPEG-Standard (ITU T.81)

The popular current image compression techniques are implementations of the JPEG-standard (ITU T.81). The JPEG-standard covers a collection of compression techniques: it contains a lossless mode, two progressive modes and a simple lossy technique which encodes the image

sequentially from the upper left to the lower right corner. Every mode can be split into different versions, so for instance one can choose between Huffman and arithmetic coding, though some useful combinations are partially missed, for example, JPEG does not contain a progressive lossless mode of operation. The large number of different coding techniques makes a complete implementation of the JPEG-standard extensive while a minimal JPEG decoder has only to realize the sequential lossy mode of operation. If one has to choose the best suited JPEG compression technique, one has the problem of ensuring that the target system can decode the images. The lossy JPEG compression initially divides the source image into squares of 8×8 pixels. Discrete Cosine Transform (DCT) is applied on each block resulting in another 8×8 matrix of values. We then quantize this matrix giving more importance to low frequency coefficients. This process causes large number of zeroes in the transformed image matrix. These values are then entropy coded and output to a file. Since transform is done on 8×8 matrix (for reasons of speed of computation), the reconstructed image may contain discontinuities (blocking artifacts) at the boundaries of the squares, resulting in one of the major JPEG disadvantages. Figure 11.1 shows the coding scheme of JPEG lossy compression.

These blocking artifacts of the JPEG technique are visible only in enlarged images but cause problems in many image processing applications. Particular problems occur when analyzing algorithms are applied to JPEG compressed images or when the images are re-rastered for printing. When a compressed image is re-compressed (e.g., with a higher compression rate or after some processing), these additional edges further degrade the quality of the image. Wavelet based image compression techniques work on full image and is found to be superior to DCT. In Section 11.2 we will explore transform based on wavelet and how the transformed image is utilized in compression.

11.2 WAVELET TRANSFORM OF AN IMAGE

In the discrete wavelet transform, an image signal can be analyzed by passing it through an analysis filter bank followed by a decimation operation. This analysis filter bank, which consists of a low pass and a high pass filter at each decomposition stage, is commonly used in image compression.

When a signal passes through these filters, it is split into two bands. The low pass filter, which corresponds to an averaging operation, extracts the coarse information of the signal. The high pass filter, which corresponds to a differencing operation, extracts the detail information of the signal. The output of the filtering operations is then decimated by two.

A two-dimensional transform (see Figures 11.2, 11.3 and 11.4) can be accomplished by performing two separate one-dimensional transforms. First, the image is filtered along the x -dimension using low pass and high pass analysis filters and decimated by two. Low pass filtered coefficients are stored on the left part of the matrix and high pass filtered on the right. Because of decimation the total size of the transformed image is same as the original image (see Figure 11.2). Then, it is followed by filtering the sub-image along the y -dimension and decimated by two. Finally, we have split the image into four bands denoted by LL, HL, LH and HH after one-level decomposition (see Figure 11.3). Figure 11.4(a) shows second stage of filtering. A four-level decomposition is shown in 11.4(b).

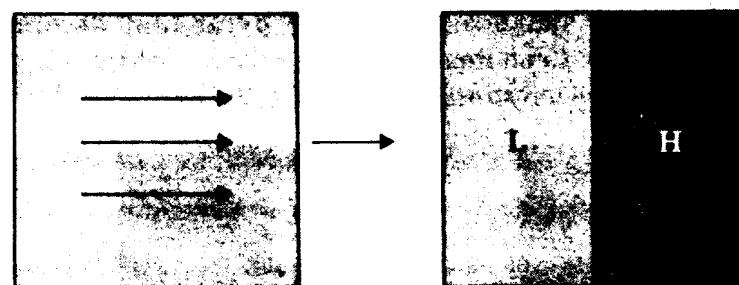


FIGURE 11.2 Horizontal transform-2 subbands.

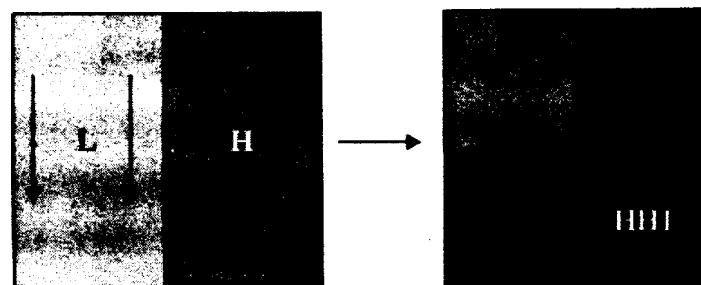
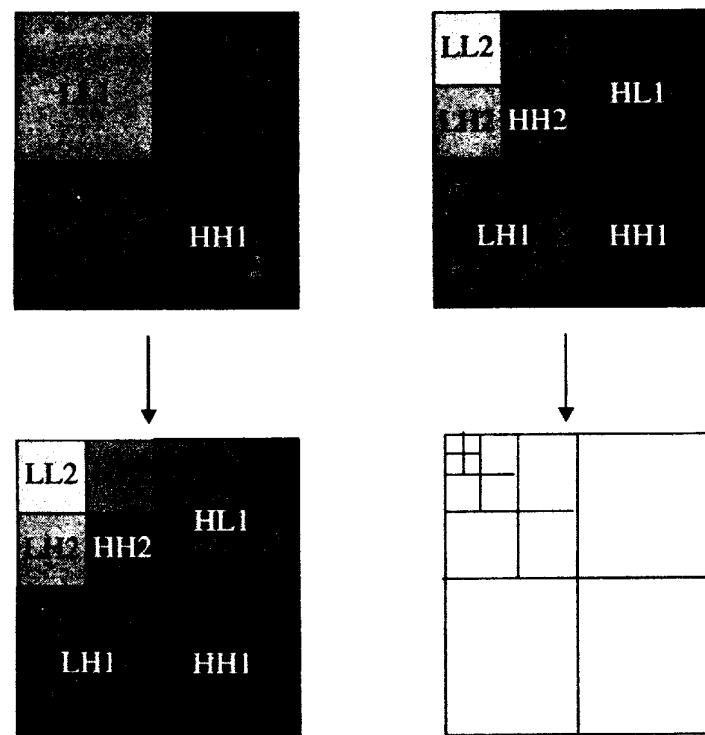


FIGURE 11.3 Vertical transform-4 subbands.



(a) Second level filtering (b) Third and forth level filtering

FIGURE 11.4 Steps in pyramidal decompositions of an image.

This process of filtering the image is called **pyramidal decomposition of image**. Figure 11.5 shows what happens to an image for each of the operation depicted in Figures 11.2 to 11.4. Figure 11.6 shows how transformed image finally look like.

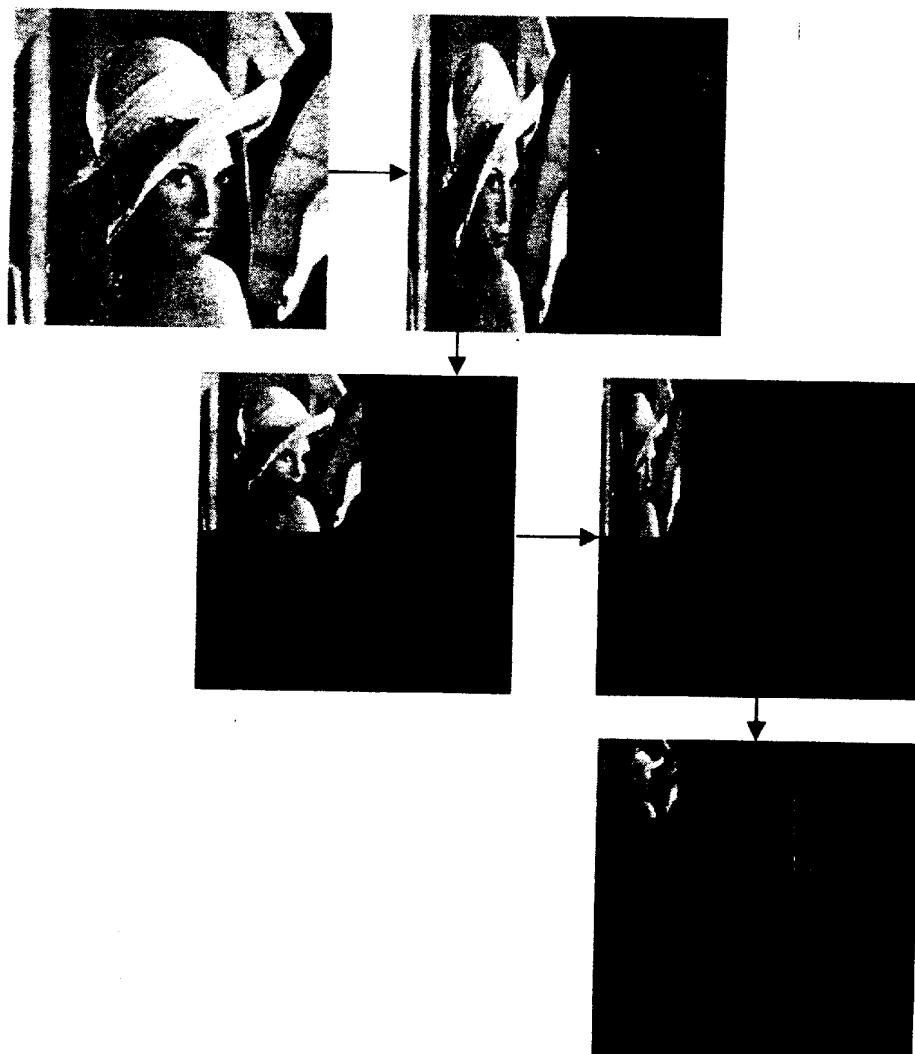


FIGURE 11.5 2-D wavelet decomposition of an image.



FIGURE 11.6 Lena's image (left) and a four level decomposition of Lena's image (right).
(Here the detail level coefficients are enhanced by one digit.)

The reconstruction of the image can be carried out by the following procedure. First, we will upsample the data by a factor of two on all the four subbands at the coarsest scale and filter the subbands in each dimension. Then we sum the four filtered subbands to reach the low-low subband at the next finer scale. We repeat this process until the image is fully reconstructed.

11.3 QUANTIZATION

Quantization refers to the process of approximating the continuous set of values in the image data with a finite (preferably small) set of values. The input to a quantizer is the original data and the output is always one among a finite number of levels. The quantizer is a function whose set of output values are discrete and usually finite. Obviously, this is a process of approximation and a good quantizer is one which represents the original signal with minimum loss or distortion.

There are two types of quantization: scalar quantization and vector quantization. In scalar quantization, each input symbol is treated separately in producing the output while in vector quantization the input symbols are clubbed together in groups called vectors, and processed to give the output. This clubbing of data and treating them as a single unit increases the optimality of the vector quantizer but at the cost of increased computational complexity. Here, we'll take a look at scalar quantization.

Scalar quantization allows individual wavelet coefficients to be converted to a quantized value with the conversion being independent from coefficient to coefficient. The following are some of the scalar quantizers that we will be discussing in the coming sections:

- Uniform quantizer
- Subband uniform quantizer
- Non-uniform quantizer

11.3.1 Uniform Quantization

In uniform quantization, the quantization level is an important parameter as it determines the value of the step size. A large quantization level will result in a small step size while a small quantization level will result in a large step size. For a given quantization level, the step size is constant and controls the quality of an image. A large step size gives a poorer approximation of the image while a small step size will give a better one.

The equation for computing the step size is:

$$\Delta = \frac{m_k - m_0}{L}$$

where

Δ denotes step size

m_k denotes maximum value

m_0 denotes minimum value

L denotes quantization levels

The range of the input data under quantization will be divided into $L + 1$ levels of equal intervals as shown in Figure 11.7.

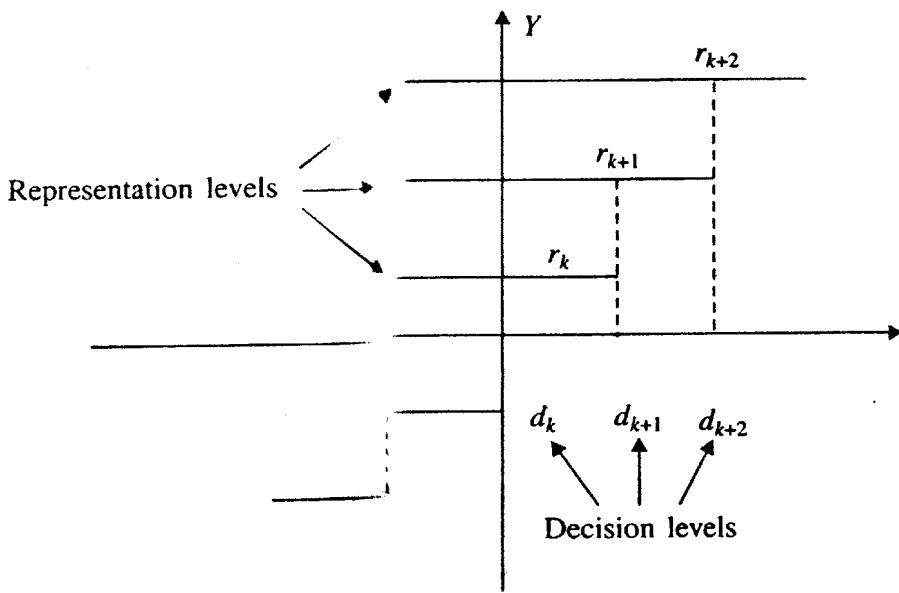


FIGURE 11.7 Uniform Quantizer.

In the quantization process, a coefficient x will be specified by k if it lies in the partition cell:

$$d_k \leq x \leq d_{k+1}$$

where $k = 1, 2, 3, \dots, L$.

The dequantizer uses this index k to compute the reconstructed value, r_k , using the following equation to minimize the distortion:

$$r_k = \frac{d_k + d_{k+1}}{2}$$

11.3.2 Subband Uniform Quantization

Subband uniform quantization is based on the fact that the wavelet coefficients in the higher subbands are more significant than in lower subbands. Therefore, the coefficients in the higher subbands are quantized finely while the coefficients in the lower subbands are quantized coarsely to achieve a better compression ratio. Subband uniform quantization applies the uniform quantization technique described earlier. The only difference is that the step size varies from subband to subband. A larger quantization level is used to compute the step size for the higher subbands and a smaller quantization level is used to compute the step size for the lower subbands.

11.3.3 Uniform Dead-Zone Quantization

The characteristic of wavelet coefficients is such that many of their values are very small in magnitude. To take advantage of that, we can specify a threshold value, t , that sets these coefficients to zero as shown in the following equation:

$$\text{If } -t \leq x \leq t \text{ then } y = 0$$

Following this, we apply uniform quantization. This technique is called **uniform dead-zone quantization** as shown in Figure 11.8, where it is assumed $t = d_k$.

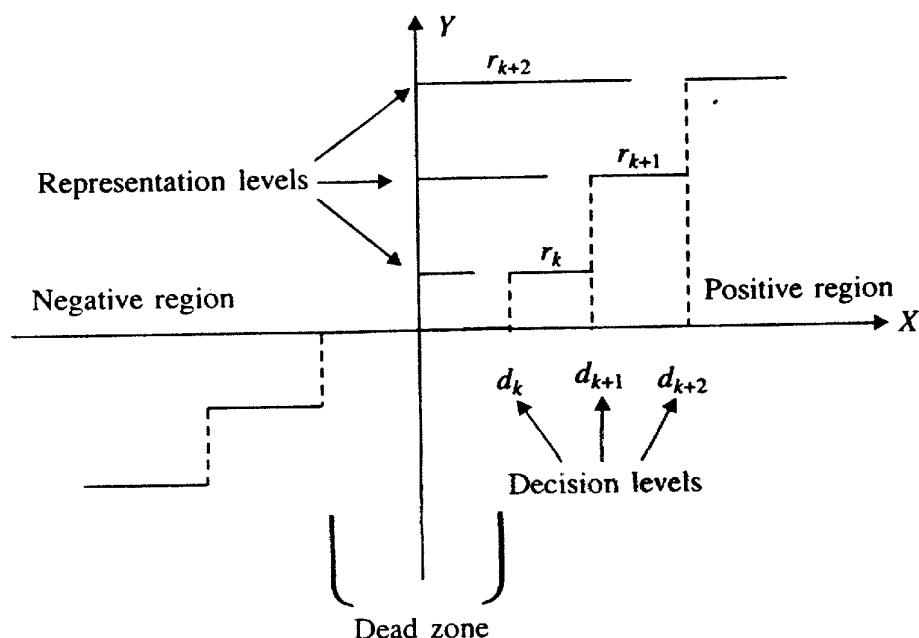


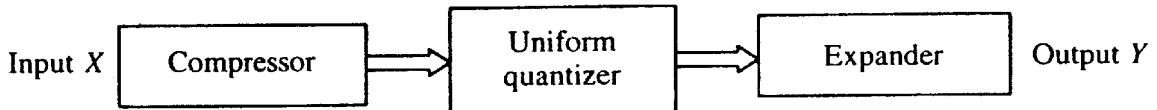
FIGURE 11.8 Dead-zone quantization.

The reason for setting these coefficients with small magnitude to zero is to make the image sparse so that it is applicable for the run-length encoding technique, which we will describe in the next few sections.

11.3.4 Non-Uniform Quantization

One of the non-uniform quantization techniques is the μ -law compander. This is the North American standard for speech compression. This technique compresses the data non-uniformly using a compressor and then feeds the compressed data to a uniform quantizer. To reconstruct the data, an expander is used.

The non-uniform quantization technique described here (see Figure 11.9) allows the dynamic range of the wavelet coefficients to be reduced by amplifying the lower amplitudes of the coefficients and compressing the higher ones. Thus, it gives a better signal to quantization noise ratio.

**FIGURE 11.9** Non-uniform quantizer.

The formula used for the μ -law compressor is:

$$y = \frac{V \log(1 + \mu|x|/V)}{\log(1 + \mu)} \operatorname{sgn}(x)$$

where

x denotes the input data

V denotes the maximum value of data

μ denotes the μ -law parameter of the compander

y denotes the output data

The formula for the expander is:

$$x = \frac{V}{\mu} (e^{|y|\log(1+\mu)/V} - 1) \operatorname{sgn}(y)$$

11.4 ENTROPY ENCODING

The quantized data contains redundant information. It is a waste of storage space if we were to save the redundancies of the quantized data. One way of overcoming this problem is to use entropy encoding. It is an example of a lossless data compression technique that provides a means of removing the redundancies in the quantized data without any loss of information. We will discuss two types of lossless data compression schemes in this section. They are Huffman encoding and Run-length encoding.

11.4.1 Huffman Encoding

Huffman encoding is a statistical compression technique developed by David Huffman. It uses the probability of occurrence of symbols to determine the codeword representing the symbols. The length of the codeword is variable. Symbols with higher probability of occurrence will have shorter codeword lengths while symbols with lower probability of occurrence will have longer codeword lengths. As a result, the average codeword length per symbol reduces and this leads to a smaller output data size. In order to use Huffman encoding (see Figure 11.10 for example) to encode the quantized data, we have to obtain the frequencies of occurrence of the symbols from the input, symbols can be number of alphabets. See Section 11.5 to know how symbols come into picture in the image compression process. Having done this, the following steps are employed:

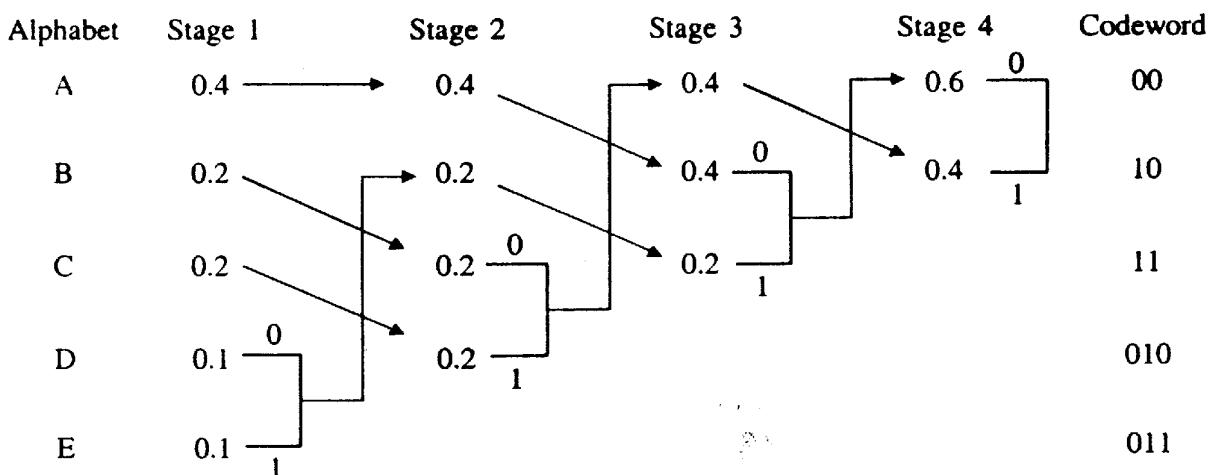


FIGURE 11.10 Huffman encoding.

1. Line up the symbols in the descending order of their probability of occurrences.
2. Combine the two source symbols with the least probabilities to form a new symbol. This new symbol will have a probability that is the sum of the probabilities of the two source symbols. We will assign a '0' to the left of the new symbol and a '1' to the right of the new symbol.
3. Repeat step 1. If there are many symbols with the same probability then the newly created symbol will be placed as high as possible in the list. This procedure is repeated until a binary tree containing all symbols has been created.

After that, we will trace the binary tree starting from its root to the leaf node that holds a symbol. The accumulation of these bit sequences will form the codeword that will be stored in the codeword table. This process is repeated for all the symbols in the list. To decode the encoded data, we will start at the root of the binary tree and traverse down the tree until we reach the leaf node base on the incoming bit sequence. Each time we reach the leaf node, we have decoded a symbol. This process is repeated until all the bit sequences in the incoming input have been decoded.

11.4.2 Run Length Encoding

Run length encoding is based on the idea of encoding a consecutive occurrence of the same symbol. This is achieved by replacing a series of repeated symbols with a count and the symbol. In the case of the wavelet coefficients where dead-zone quantization is used, there will be a lot of zero values that can be exploited with RLE. The following illustrates using RLE to encode input data with frequent runs of zero.

Illustration

Original data

12 0 0 0 13 14 15 0 0 0 0 0 9 0 0

Encoded data

12	0	3	13	14	15	0	6	9	0	0
----	---	---	----	----	----	---	---	---	---	---

From the illustration, we observe that the compression takes place when the number of consecutive zeroes is more than two. Such zeroes will be replaced with the RLE symbol shown in Figure 11.11. If the number of consecutive zeroes is less than two then the zeroes are not encoded.

Zero (1 byte)	Count (1 byte)
------------------	-------------------

FIGURE 11.11 RLE encoded-symbol(1).

If the data contains a large number of consecutive zeroes, the data size can be greatly reduced using RLE. On the other hand, if the content of the data is random then this encoding technique might increase the data size instead. To allow RLE to be used in encoding the consecutive runs of other characters, we can add a control character to the RLE encoded-symbol as shown in Figure 11.12.

CTRL (1 byte)	Count (1 byte)	Symbol (1 byte)
------------------	-------------------	--------------------

FIGURE 11.12 RLE encoded-symbol(2).

The following example illustrates the used of this encoded-symbol.

*Illustration**Original data*

12 13 13 13 13 14 14 14 14 0 0 0 0 0 9 9 0

Encoded data

12	!	4	13	!	4	14	!	5	0	9	9	0
----	---	---	----	---	---	----	---	---	---	---	---	---

For this type of RLE encoder, the compression takes place when the number of consecutive characters is more than three. Such characters will be replaced with the RLE symbol shown in Figure 11.12. If the number of consecutive characters is less than two then the characters will not be encoded. In the decoding process, the encoded data will be expanded according to the count of zeroes or other characters in the encoded-symbols.

11.5 EZW CODING (EMBEDDED ZERO-TREE WAVELET CODING)

An EZW encoder was specially designed by Shapiro [1] to use with wavelet transforms. In fact, EZW coding is more like a quantization method. It was originally designed to operate on

images (2D-signals) but it can also be used on other dimensional signals. The EZW encoder is based on progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to JPEG encoded images. Progressive encoding is also known as **embedded encoding** which explains the 'E' in EZW. Coding an image using the EZW scheme, together with some optimizations, results in a remarkably effective image compressor with the property that the compressed data stream can have *any* bit rate desired. Any bit rate is only possible if there is information loss somewhere, so that the compressor is lossy. However, lossless compression is also possible with an EZW encoder but of course with less spectacular results.

The EZW encoder is based on two important observations:

1. Natural images in general have a low pass spectrum. When an image is wavelet transformed, the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.
2. Large wavelet coefficients are more important than small wavelet coefficients.

These two observations are used by encoding the wavelet coefficients in decreasing order, in several passes. For every pass, a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold, it is encoded and removed from the image; if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited, the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance).

A wavelet transform, transforms a signal from the time domain to the joint time-scale domain. This means that the wavelet coefficients are two-dimensional. If we want to compress the transformed signal, we have to code not only the coefficient values but also their position in time/space. When the signal is an image then the position in time is better expressed as the position in space. After wavelet transforming an image, we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband (see Figure 11.13). One of each four descendants also has four descendants in the next higher subband and we see a *quad-tree* emerge: every parent has four children.

A zerotree is a quad-tree of which all nodes (value of coefficients at that location) are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeroes. To clutter this definition, we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured. The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad-tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case then the whole tree can be coded with a single zerotree symbol. Now, if the image is scanned in a predefined order, going from high scale to low, implicitly many positions are coded through the use of zerotree symbols.

images (2D-signals) but it can also be used on other dimensional signals. The EZW encoder is based on progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to JPEG encoded images. Progressive encoding is also known as **embedded encoding** which explains the 'E' in EZW. Coding an image using the EZW scheme, together with some optimizations, results in a remarkably effective image compressor with the property that the compressed data stream can have *any* bit rate desired. Any bit rate is only possible if there is information loss somewhere, so that the compressor is lossy. However, lossless compression is also possible with an EZW encoder but of course with less spectacular results.

The EZW encoder is based on two important observations:

1. Natural images in general have a low pass spectrum. When an image is wavelet transformed, the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.
2. Large wavelet coefficients are more important than small wavelet coefficients.

These two observations are used by encoding the wavelet coefficients in decreasing order, in several passes. For every pass, a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold, it is encoded and removed from the image; if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited, the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance).

A wavelet transform, transforms a signal from the time domain to the joint time-scale domain. This means that the wavelet coefficients are two-dimensional. If we want to compress the transformed signal, we have to code not only the coefficient values but also their position in time/space. When the signal is an image then the position in time is better expressed as the position in space. After wavelet transforming an image, we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband (see Figure 11.13). One of each four descendants also has four descendants in the next higher subband and we see a *quad-tree* emerge: every parent has four children.

A zerotree is a quad-tree of which all nodes (value of coefficients at that location) are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeroes. To clutter this definition, we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured. The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad-tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case then the whole tree can be coded with a single zerotree symbol. Now, if the image is scanned in a predefined order, going from high scale to low, implicitly many positions are coded through the use of zerotree symbols.

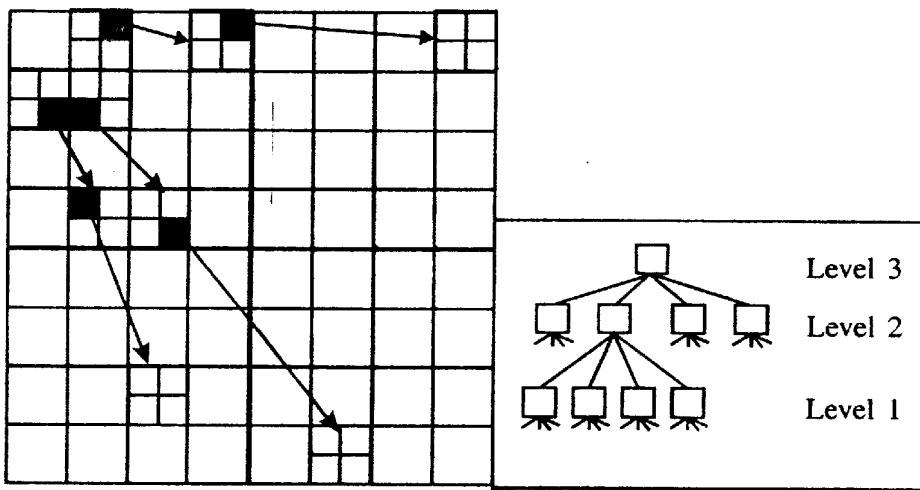


FIGURE 11.13 The relations between wavelet coefficients in different subbands as quad-trees.

EZW algorithm

The first step in the EZW coding algorithm is to determine the initial threshold. It can be calculated by the following equation:

$$T_0 = 2^{\lfloor \log_2 (\text{Max}(|C(x,y)|)) \rfloor}$$

where MAX(.) means the maximum coefficient value of the image and $C(x, y)$ denotes the coefficient. We start the coding loop with this initial threshold.

From the algorithm given in Figure 11.14, we see that two passes are used to code the image. In the first pass, the *dominant pass*, the image is scanned and a symbol is outputted for every coefficient. Notice two different scan orders (see Figure 11.15) can be applied to this algorithm, however, Morton scan is simpler to code. Only four symbols are used to code the coefficients.

```

MAIN PROCEDURE
threshold = initialThreshold;
DO
{
    dominantPass();
    subordinatePass();
    threshold = threshold/2;
}
WHILE (threshold > stopThreshold);
END MAIN PROCEDURE

```

FIGURE 11.14 Algorithm of the main loop.

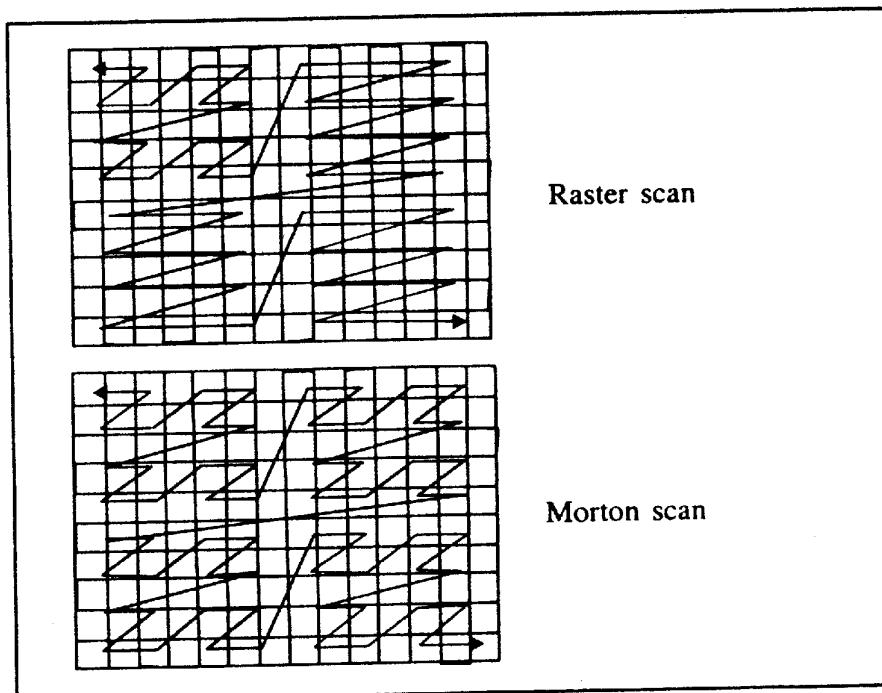


FIGURE 11.15 Two different scan orders.

1. If the coefficient is larger than the threshold a **P** (positive significant) is coded.
2. Else if the coefficient is smaller than minus the threshold an **N** (negative significant) is coded.
3. Else if the coefficient is the root of a zerotree then a **T** (zerotree) is coded.
4. Else if the coefficient is smaller than the threshold but it is not the root of a zerotree, then a **Z** (isolated zero) is coded. This happens when there is a coefficient larger than the threshold in the subtree. The effect of using the **N** and **P** codes is that when a coefficient is found to be larger than the threshold (in absolute value or magnitude), its two most significant bits are outputted. Notice that in order to determine, if a coefficient is the root of a zerotree or an isolated zero, we will have to scan the whole quad-tree. This process is time consuming. Also, to prevent outputting codes for coefficients in identified zerotrees, we will have to keep track of them. This means memory for book keeping.

Finally, all the coefficients that are in absolute value larger than the current threshold are extracted and placed without their sign on the subordinate list and their positions in the image are filled with zeroes. This will prevent them from being coded again. The second pass, the *subordinate pass*, is the refinement pass. We output the next most significant bit of all the coefficients on the subordinate list. The main loop ends when the threshold reaches the stop threshold value. Figures 11.16, 11.17 and 11.18 are the supporting routines of EZW encoder.

```

PROCEDURE Bool Zerotree(int threshold, Coefficient Current
Coefficient)
BEGIN
    Put the current efficient into Children coefficient list;
    WHILE (Children coefficient list is not empty)
    BEGIN
        Get a coefficient from the Children coefficient list;
        IF coefficient >= threshold, THEN
            RETURN FALSE
        ELSE
            Put the four children into the Children coefficient
            list;
        END IF
    END
    RETURN TR
END PROCEDURE

```

FIGURE 11.16 Algorithm to determine if a coefficient is a root of zerotree.

```

PROCEDURE DominantPass()
BEGIN
    initialize coded coefficient list;
    WHILE (coefficient list not empty)
        get one coded coefficient from the list;
        IF coefficient was coded as P, N or Z then
            code next scanned coefficient;
            put the coefficient into coded coefficient list;
        IF coefficient was coded as P or N then
            add abs(coefficient) to subordinate list;
            set coefficient position to zero;
        END IF
    END IF
    END WHILE
END PROCEDURE

```

FIGURE 11.17 Algorithm for the dominant pass.

Here we use a coefficient list to keep track of the identified zerotrees. To start this loop, we have to initialize the coefficient by manually adding the first quad-tree root coefficients to the coefficient list. The step of “code next scanned coefficient” checks the next uncoded coefficient of the image, indicated by the scanning order and outputs a P, N, T or Z. After coding the

```

PROCEDURE SubordinatePass
BEGIN
    subordinateThreshold = currentThreshold/2;
    FOR all elements on subordinate list do
        BEGIN
            IF (coefficient > subordinateThreshold)
                output a one;
            coefficient = coefficient - subordinateThreshold;
            ELSE output a zero;
        END FOR
    END PROCEDURE

```

FIGURE 11.18 Algorithm for the subordinate pass.

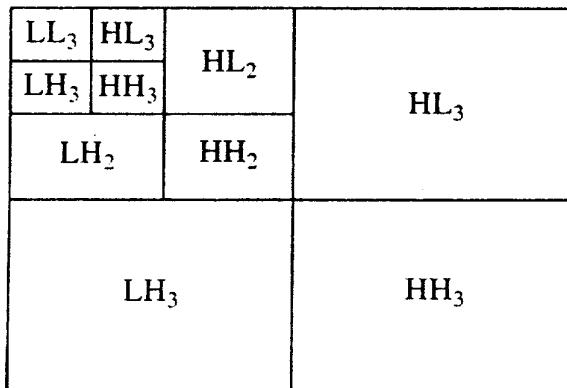
coefficient, it is put in the coded coefficient list. Thus, the coded coefficient list contains only coefficients which have already been coded, i.e., a P, N, T or Z has already been outputted for these coefficients. Finally, if a coefficient is coded as a P or N, we remove it from the image and place it on the subordinate list. This loop stops as long as the coefficients at the last level, the highest subbands, are coded as zerotrees. The subordinate pass follows the dominant pass:

Example: We will demonstrate the details of the EZW algorithm using a simple example. The coefficients to be coded are from a three-level wavelet transform of an 8×8 image and are shown in Figure 11.19.

53	-22	21	-9	-1	8	-7	6
14	-12	13	-11	-1	0	2	-3
15	-8	9	7	2	-3	1	-2
34	-2	-6	10	6	-4	4	-5
-6	5	-1	1	1	3	-1	5
6	1	3	0	-2	2	6	0
4	2	1	-4	-1	0	-1	4
0	-2	7	5	-3	2	-2	3

FIGURE 11.19 An example three-level wavelet decomposition used to demonstrate the EZW algorithm.

For clarity, the entropy coding is not shown, which means the coder output will be a sequence of symbols for the dominant pass. The largest coefficient in the transform is 53 which results in an initial threshold of $T_0 = 32$. The labelling scheme used in the example is shown in Figure 11.20.

**FIGURE 11.20** The subband labelling scheme for a three-level, 2-D wavelet transform.

The results for the first dominant pass are tabulated in Table 11.2, with corresponding comments given as follows:

TABLE 11.2 Results for the First Dominant Pass

<i>Subband</i>	<i>Coefficient value</i>	<i>Symbol</i>	<i>Reconstruction value</i>	<i>Comment</i>
LL ₃	53	P	48	(1)
HL ₃	-22	T	0	(2)
LH ₃	14	Z	0	(3)
HH ₃	-12	T	0	
LH ₂	15	T	0	
LH ₂	-8	T	0	
LH ₂	34	P	48	
LH ₂	-2	T	0	
LH ₁	4	Z	0	
LH ₁	2	Z	0	
LH ₁	0	Z	0	
LH ₁	-2	Z	0	

Comment-1: The coefficient has a magnitude greater than or equal to the threshold 32 and is positive. The resulting symbol is positive significant (P) and the decoder knows that this symbol lies in the interval [32, 64] and that its reconstruction value is 48.

Comment-2: This coefficient and all of its descendants (comprising all of subbands HL₂ and HL₁) are less than the threshold of 32, which causes this symbol to be coded as a zero-tree root (T). As a result, the remaining coefficients in subbands HL₂ and HL₁ are not coded in this dominant pass.

Comment-3: This coefficient is less than the threshold 32 but one of its descendants, coefficient 34 in subband LH₂, is significant relative to the threshold, preventing this symbol to be coded as a zero-tree root (T). As a result, this coefficient is coded as isolated zero (Z).

In the first subordinate pass the encoder sends a 0 or 1 to indicate if the significant coefficients are in the intervals [32, 48] or [48, 64] respectively. Thus, the encoder outputs are 1 and 0 corresponding to the reconstruction values of $(48 + 64)/2 = 56$ and $(32 + 48)/2 = 40$. The results of the first subordinate pass are summarized in Table 11.3.

TABLE 11.3 Output of the Subordinate Pass

<i>Coefficient magnitude</i>	<i>Symbol</i>	<i>Reconstruction magnitude</i>
53	1	56
34	0	40

For the second dominant pass, the coefficients 53 and 34 do not have to be coded again. The wavelet transform thus appears as in Figure 11.21, where the * indicates a previously significant coefficient.

*	-22	21	-9	-1	8	-7	6
14	-12	13	-11	-1	0	2	-3
15	-8	9	7	2	-3	1	-2
*	-2	-6	10	6	-4	4	-5
-6	5	-1	1	1	3	-1	5
6	1	3	0	-2	2	6	0
4	2	1	-4	-1	0	-1	4
0	-2	7	5	-3	2	-2	3

FIGURE 11.21 The example wavelet transform after the first dominant pass. The symbol * is used to represent symbols found to be significant on a previous pass.

The threshold for the second dominant pass is $T_1 = 16$ and the results of this pass are summarized in Table 11.4, with the corresponding comment.

TABLE 11.4 Output of Second Dominant Pass

<i>Subband</i>	<i>Coefficient value</i>	<i>Symbol</i>	<i>Reconstruction value</i>	<i>Comment</i>
HL ₃	-22	N	-24	
LH ₃	14	T	0	(4)
HH ₃	-12	T	0	
LH ₂	21	P	24	
LH ₂	-9	T	0	
LH ₂	13	T	0	
LH ₂	-11	T	0	
LH ₁	-1	Z	0	
LH ₁	8	Z	0	
LH ₁	-1	Z	0	
LH ₁	0	Z	0	

Comment-4: Since the coefficient 34 of subband LH₂ was found to be significant on a previous pass, its value can be considered zero for purposes of computing zerotrees. As a result, the coefficient 14 becomes a zerotree root on the second dominant pass.

The process continues alternating between dominant and subordinate passes until a desired fidelity or bit rate is achieved.

Example: Transformed image is given in the following diagram (Figure 11.22):

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

FIGURE 11.22 Data for the EZW coding example.

EZW output:

D1: PNZTPTTTZTTTTTTPTT

S1: 1010

D2: ZTNPTTTTTTTT

S2: 100110

D3: ZZZZPPNPPNNTNNPTPTNTTTTTTTPTTTPPTTTTTPTTTPPTTTTTTT

S3: 10011101111011011000

D4: ZZZZZZZT₂TZ₂NNNNPT₂PT₂PNPTNTTTTPT₂PN₂PPPPPTTTTTP₂PT₂TT₂PN₂PP

S4: 1101111011001000001110110100010010101100

D5: ZZZZZTZZZZZTPZZZTTPTTTNPTPPTPTTTNPPNTTPNNPTPTTTPPTT

S5: 10111100110100010111101011011001000000000110110110011000111

D6: ZZZTTZTTZTTTNNNTT

refer to the dominant and subordinate pass.

11.5.1 EZW Performance

Having presented the EZW algorithm, the main question to be asked is: How well does it perform? The answer is that the performance is quite good. When EZW was first introduced, it gave compression performance as good or better than the other algorithms that existed at that time. Figure 11.23 shows example of EZW performance comparing it with the results from the original JPEG-standard. It is notable that EZW is able to achieve its good performance with a relatively simple algorithm. EZW does not require complicated bit allocation procedures like subband coding does it doesn't require training or codebook storage like vector quantization does and it doesn't require prior knowledge of the image source like JPEG does (to optimize quantization tables).

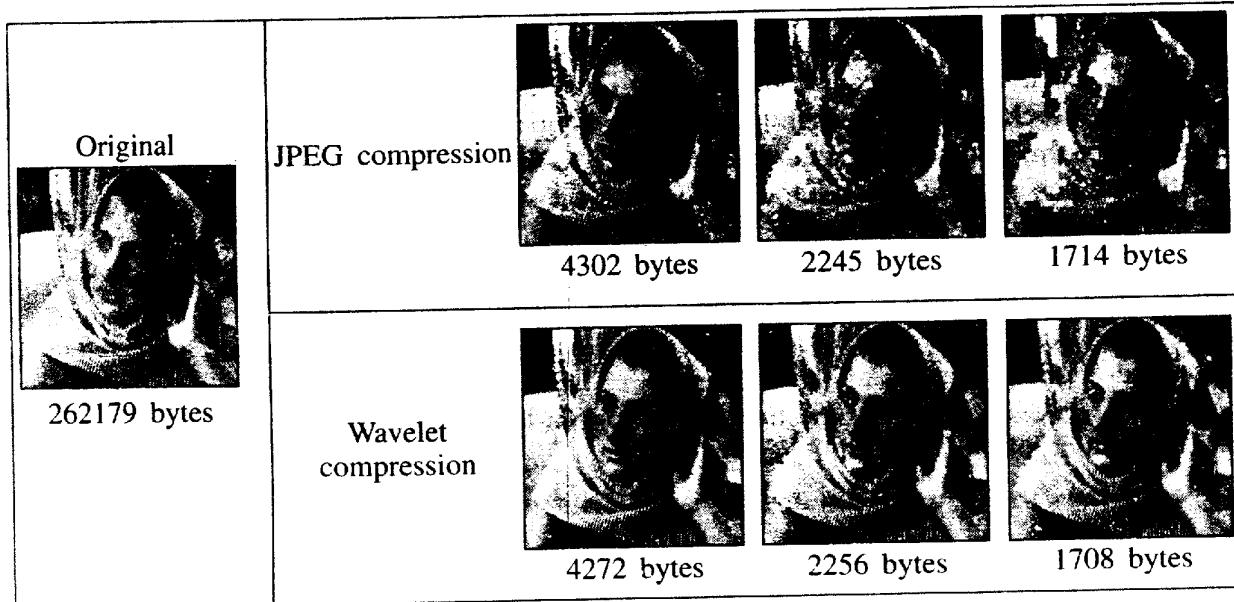


FIGURE 11.23 Performance comparison of JPEG with EZW.

EZW also has the desirable property, resulting from its successive approximation quantization, of generating an embedded code representation. This means that an image is coded at one rate and this code can be used to generate the code for the same image at higher or lower rates. To generate a higher rate or more detailed representation, simply continue the coding where the original representation left off and concatenate these bits to the original code. To generate a lower rate or less detailed code, just truncate bits off from the original code to get the desired lower code rate. The resulting codes, at either higher or lower rate, would be exactly the same as those generated from scratch using the EZW algorithm. One desirable consequence of an embedded bit stream is that it is very easy to generate coded outputs with the exact desired size. Truncation of the coded output stream does not produce visual artifacts since the truncation only eliminates the least significant refinement bits of coefficients rather than eliminating entire coefficients as is done in subband coding.

11.6 SPIHT (SET PARTITIONING IN HIERARCHICAL TREE)

From the inspiration of EZW, Said and Pearlman proposed SPIHT four years later. Under the same assumption of zerotrees, SPIHT scans wavelet coefficients along quad-tree instead of subband. Another modification is, SPIHT identifies significance only by the magnitude of wavelet coefficients and encodes the sign separately into a new bit stream. Combined with adaptive arithmetic coding, SPIHT improved the compression performance quite much and caused a number of new developments [7, 8, 9].

11.7 EBCOT (EMBEDDED BLOCK CODING WITH OPTIMIZED TRUNCATION)

In EBCOT, each subband is partitioned into relatively small blocks of wavelet coefficients called **code block**. EBCOT generates embedded bit stream separately for each code block. These bit streams can be truncated independently to different lengths. The actual block coding algorithm which generates a separate embedded bit stream for each code block is combined with bit plane coding and adaptive arithmetic coding. Each code block is again divided into 2-D sequences of subblocks, whose size is 16*16. (In JPEG2000, the size is 4*4). Then, for each bit plane, the significance map of subblocks is firstly encoded through quad-tree coding. Then, four different primitive coding operations are used to encode those significant subblocks. Those operations are zero coding, run-length coding, sign coding and magnitude refinement. And at last, EBCOT organizes the final big stream in layers with optimized truncation so as to make it both resolution and SNR scalable.

SUMMARY

Digital images are widely used in computer applications. Uncompressed digital images require considerable storage capacity and transmission bandwidth. Efficient image compression solutions are becoming more critical with the recent growth of data intensive, multimedia-based web applications.

Embedded Zerotree Wavelet (EZW) coding is among the best (if not the best) coding scheme for image compression. It provides efficient use of storage space and bandwidth. In addition reduces the complexity and cost to store and/or transmit images. Much work has been done (and still is) in order to improve this coding scheme. Many new algorithms like SPIHT and EBCOT is emerging which could be used to compress the images cheaply, fast and efficiently.

WEBLEMS (WEB BASED PROBLEMS)

- 11.1** Download MATLAB code for EZW from the site <http://perso.wanadoo.fr/polyvalens/clemens/ezw> and find EZW output for the following data:

$$X = [63 \ -34 \ 49 \ 10 \ 7 \ 13 \ -12 \ 7 \\ -31 \ 23 \ 14 \ -13 \ 3 \ 4 \ 6 \ -1 \\ 15 \ 14 \ 3 \ -12 \ 5 \ -7 \ 3 \ 9 \\ -9 \ -7 \ -14 \ 8 \ 4 \ -2 \ 3 \ 2 \\ -5 \ 9 \ -1 \ 47 \ 4 \ 6 \ -2 \ 2 \\ 3 \ 0 \ -3 \ 2 \ 3 \ -2 \ 0 \ 4 \\ 2 \ -3 \ 6 \ -4 \ 3 \ 6 \ 3 \ 6 \\ 5 \ 11 \ 5 \ 6 \ 0 \ 3 \ -4 \ 4]$$

- 11.2** Use the code mentioned in Weblem 1 to decompress the output from the Weblem 1 and plot the difference.
- 11.3** Use GOOGLE to locate the paper titled “Implementing the SPIHT Algorithm in MATLAB” written by Aldo Morales and Sedig Agili. Write additional MATLAB code necessary to make it a full fledged image compression code.
- 11.4** Using SPIHT algorithm find the compressor output for the following input data:

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

REFERENCES

- [1] Shapiro, J.M., Embedded image coding using zerotrees of wavelet coefficients, *IEEE Transactions on Signal Processing*, Vol. 41, No. 12 (1993), pp. 3445–3462.
- [2] Sayood, K., *Introduction to Data Compression*, Second edition, Academic Press, 2000.
- [3] Saha, S., Image Compression—from DCT to Wavelets, <http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html>.

- [4] Valens, C., EZW Coding, <http://perso.wanadoo.fr/polyvalens/clemens/ezw>.
- [5] Hilton, M.L., B.D. Jawerth, and A. Sengupta, Compressing still and moving images with wavelets, *Multimedia Systems*, Vol. 2 and No. 3, April 18, 1994.
- [6] Creusere, C.D., A new method of robust image compression based on the embedded zerotree wavelet algorithm, *IEEE Transactions on Image Processing*, Vol. 6, No. 10 (1997), pp. 1436–1442.
- [7] Said, A. and W. Pearlman, An image multiresolution representation for lossless and lossy compression, *IEEE Trans. on Image Processing*, September 1996.
- [8] Kassim, A. and W. Lee, Embedded colour image coding using SPHIT with partially linked spatial orientation trees, *IEEE Trans. Circuits and Syst. Video Tech.*, February 2003.
- [9] Mukherjee, D. and S. Mitra, Vector SPIHT for Embedded Wavelet Video and Image Coding, *IEEE Trans. Circuits and Syst. Video Tech.* March 2003.

CHAPTER

12

Denoising

INTRODUCTION

Applied scientists and engineers who work with data obtained from the real world know that signals do not exist without noise. Under ideal conditions, this noise may decrease to such negligible levels while the signal increases to such significant levels, that for all practical purposes denoising is not necessary. Unfortunately, the noise corrupting the signal, more often than not, must be removed in order to recover the signal and proceed with further data analysis. Should this noise removal take place in the original signal (time-space) domain or in a transform domain? If the latter, should it be the time-frequency domain via the Fourier transform or the time-scale domain via the wavelet transform? Research in this area for the last 15 years is finally coming to the conclusion that wavelet shrinkage method does the job more efficiently than most other methods of denoising. In this chapter we give a brief description of some of the simple methods of denoising using wavelet.

12.1 A SIMPLE EXPLANATION AND A 1-D EXAMPLE

But what is wavelet shrinkage denoising? First, it is not smoothing (despite the use by some authors of the term *smoothing* as a synonym for the term *denoising*). Whereas smoothing removes high frequencies and retains low frequencies, denoising attempts to remove whatever noise is present and retain whatever signal is present regardless of the frequency content of the signal. For example, when we denoise music corrupted by noise, we would like to preserve both the treble and the bass. Second, it is denoising by shrinking (i.e., non-linear soft thresholding) in the wavelet transform domain. Third, it consists of three steps:

1. a linear forward wavelet transform
2. a non-linear shrinkage denoising
3. a linear inverse wavelet transform

Because of the *non-linear* shrinking of coefficients in the transform domain, this procedure is distinct from those denoising methods that are entirely linear. Finally, wavelet shrinkage denoising is considered a non-parametric method. Thus, it is distinct from parametric methods in which parameters must be estimated for a particular model that must be assumed *a priori*. (For example, the most commonly cited parametric method is that of using least squares to estimate the parameters a and b in the model $y = ax + b$.) Figure 12.1 displays a practical 1-D example demonstrating the three steps of wavelet shrinkage denoising with plots of a known test signal with added noise, the wavelet transform (from step 1), the denoised wavelet transform (from step 2) and the denoised signal estimate (from step 3).

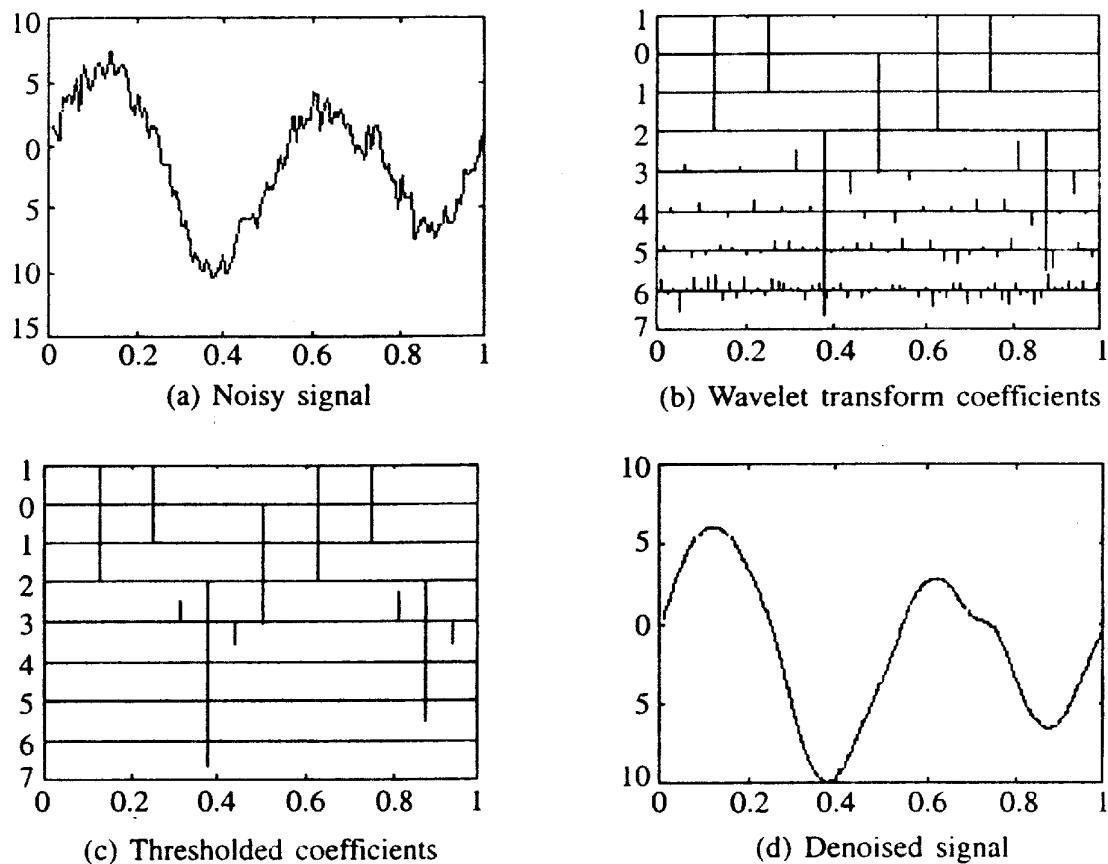


FIGURE 12.1 Steps in denoising a signal.

12.2 DENOISING USING WAVELET SHRINKAGE—STATISTICAL MODELLING AND ESTIMATION

Consider the standard univariate non-parametric regression setting

$$X_i(t) = S_i(t) + \sigma \varepsilon_i(t), \quad i = 1, 2, \dots, n \quad (12.1a)$$

where $X_i(t)$ s are assumed to come from zero-mean normal distribution, ε_i are independent standard normal $-N(0, 1)$ —random variables and noise level σ may be known or unknown. The

goal is to recover the underlying function S from the noisy data, $X = (X_1, X_2, \dots, X_n)'$, without assuming any particular parametric structure for S .

For images, the model is:

$$X_{i,j}(t) = S_{i,j}(t) + \sigma \varepsilon_{i,j}(t), \quad i = 1, 2, \dots, I, \quad j = 1, 2, \dots, J, \quad \varepsilon_{i,j} \sim N(0,1) \quad (12.1b)$$

The three main steps of denoising using wavelet coefficient shrinkage technique are as follows:

Step 1: Calculate the wavelet coefficient matrix w by applying a wavelet transform W to the data:

$$w = W(X) = W(S) + W(\sigma \varepsilon) \quad (12.2)$$

Step 2: Modify the detail coefficients (wavelet coefficients) of w to obtain the estimate \hat{w} of the wavelet coefficients of S :

$$w \rightarrow \hat{w} \quad (12.3)$$

Step 3: Inverse transform the modified detail coefficients to obtain the denoised coefficients:

$$\hat{S} = W^{-1}(\hat{w}) \quad (12.4)$$

The number n of the wavelet coefficients w in Eq. (12.2) varies depending on the type of transform (decimated or undecimated) used. w consists of both scaling coefficients and wavelet coefficients. In decimated wavelet transform, the number of coefficients in w is same as number of data points.

The first step in denoising is to select a wavelet for the forward and inverse transformation W and W^{-1} in Eq. (12.2) and Eq. (12.4), respectively. There are variety wavelet that can be used which differ in their support, symmetry and number of vanishing moments. In addition to a wavelet, we also need to select number of multiresolution levels and the option for handling values near the edge of the image. There are several boundary treatment rules including periodic, symmetric, reflective, constant and zero-padding.

12.3 NOISE ESTIMATION

Thresholding methods can be grouped into two categories: global thresholds and level-dependent thresholds. The former means that we choose a single value for threshold λ to be applied globally to all empirical wavelet coefficients while the latter means that a possibly different threshold value λ_j is chosen for each resolution level j . In what follows, we consider both global and level-dependent thresholds. These thresholds require an estimate of the noise level σ . The usual standard deviation of the data values is clearly not a good estimator, unless the underlying function S is reasonably flat. Donoho and Johnstone (1994) considered estimating σ in the wavelet domain and suggested a robust estimate that is based only on the empirical wavelet coefficients at the finest resolution level. The reason for considering only the finest level is that the corresponding empirical wavelet coefficients tend to consist mostly of noise. Since there is some signal present even at this level, Donoho and Johnstone (1994)

proposed a robust estimate of the noise level σ (based on the median absolute deviation) given by

$$\hat{\sigma} = \frac{\text{median} \{ |w_k| : k = 1, 2, \dots, n/2 \}}{0.6745} \quad (12.5)$$

Here w_k s are detail coefficients at the finest level. If we have started with n sampled data, after first level of filtering, we will have $n/2$ detail coefficients. The estimator (12.5) has become very popular in practice and it is used in subsequent sections unless stated otherwise.

The remainder of this section explains the details in the shrinkage (sometimes called thresholding step) in Eq. (12.3). Let w denote a single detail coefficient and \hat{w} denote its shrink version. Let λ be the threshold and $D^\lambda(\cdot)$ denote the shrinkage function which determines how threshold is applied to the data and $\hat{\sigma}$ be the estimate of the standard deviation σ of the noise in Eq. (12.1). Then

$$\hat{w} = \hat{\sigma} \cdot D^\lambda(w/\hat{\sigma}) \quad (12.6)$$

By dividing w with $\hat{\sigma}$ we standardize the w coefficients to get w_s and to this standardized w_s , we apply the threshold operator. After thresholding, the resultant coefficients are multiplied with $\hat{\sigma}$ to obtain \hat{w} .

If $\hat{\sigma}$ is built into the thresholding model or if the data is normalized with respect to noise standard deviation, equation for estimated value of w is:

$$\hat{w} = D^\lambda(w) \quad (12.7)$$

The normalization of the data with respect to noise-standard deviation is done as follows in MATLAB. The MATLAB code here uses formula given in Eq. (12.5). Once data is normalized with respect to noise-standard deviation, the regression model given in Eq. (12.1a) changes to

$$X_i(t) = S_i(t) + \varepsilon_i(t), \quad i = 1, 2, \dots, n$$

where $X_i(t)$ s are assumed to come from zero-mean normal distribution, and ε_i s are independent standard normal $-N(0, 1)$ variates.

```
function [y,coef] = NormNoise(x,qmf)
% NormNoise - Estimates noise level, Normalize signal to
% noise level 1
% Usage
% [y,coef] = NormNoise(x,qmf)
% Inputs
% x 1-d signal
% qmf quadrature mirror filter
% Outputs
% y 1-d signal, scaled so wavelet coefficients
% at finest level have median absolute deviation 1.
% coef estimation of 1/sigma
%
% Description
```

```

% This is required pre-processing to use any of the
% DeNoising
% tools on naturally-occurring data.

u = DownDyadHi(x,qmf);
% DownDyadHi() is the matlab function for 1-D wavelet
% analysis which return
% high pass filtered coefficients
s = median(abs(u));
if s ~= 0
    y = 0 .6745 .* x ./s;
    coef = .6745/s;
else
    y = x;
    coef = 1;
end

```

Note that the noise estimate, the threshold and the shrinkage function could depend on either multiresolution level or the subband (in case of image processing), though we have suppressed this dependence in our notation.

12.4 SHRINKAGE FUNCTIONS

The denoising methods we consider differ in the choices for $D^\lambda(\cdot)$, λ and $\hat{\sigma}$. We can obtain different denoisers by considering different

- Shrinkage functions that determine how the threshold is applied
- Noise estimates
- Shrinkage rules to determine the threshold λ

Since some shrinkage rules depend on the shrinkage functions and the noise estimates, we need to first select $D(\cdot)$ and $\hat{\sigma}$ before we determine λ .

For one-dimensional data, we can calculate the thresholds either globally, with one threshold for all the coefficients or on a level dependent basis, with K different thresholds for the K different dyadic levels. In two dimensions, in addition to these possibilities, we can also calculate thresholds in a subband-dependent manner and obtain $3K$ thresholds for the $3K$ detail coefficient subbands.

The shrinkage function determines how the thresholds are applied to the data. Figures 12.2 and 12.3 display four commonly used thresholding functions, scaled to the interval $[-1, 1]$. The horizontal axis represents detail coefficients and y axis represents corresponding thresholding function. The dotted vertical lines indicate the values of the single threshold $\pm\lambda$ for the hard, soft and garrote functions. The semisoft function requires two thresholds $\pm\lambda_1$ and $\pm\lambda_2$, represented by the four vertical lines in its graph. The mathematical expression for each of the function, given threshold λ for data w (in any arbitrary domain-signal transform or otherwise) are:

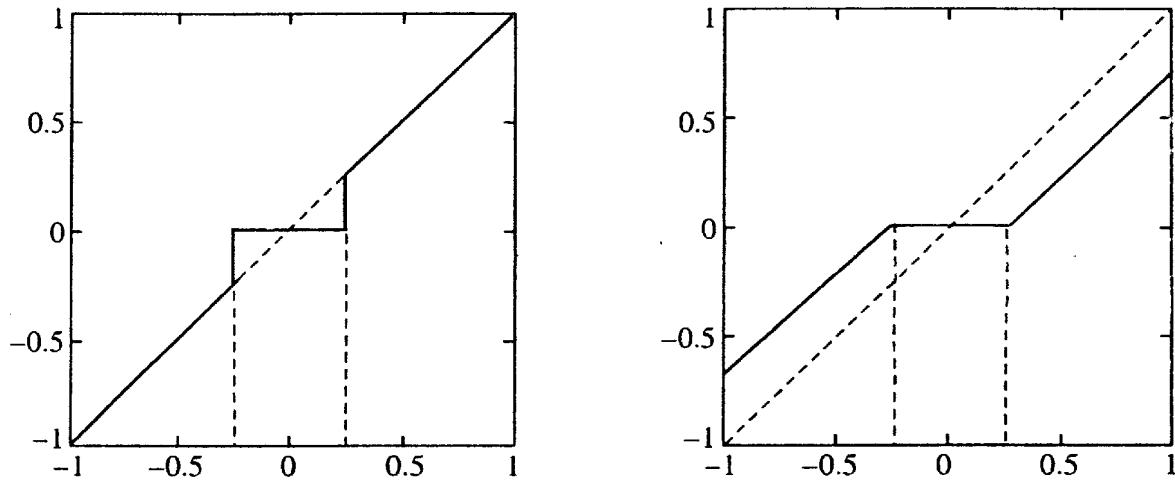


FIGURE 12.2 Shrinkage functions for hard (left figure) and soft (right figure) thresholding.

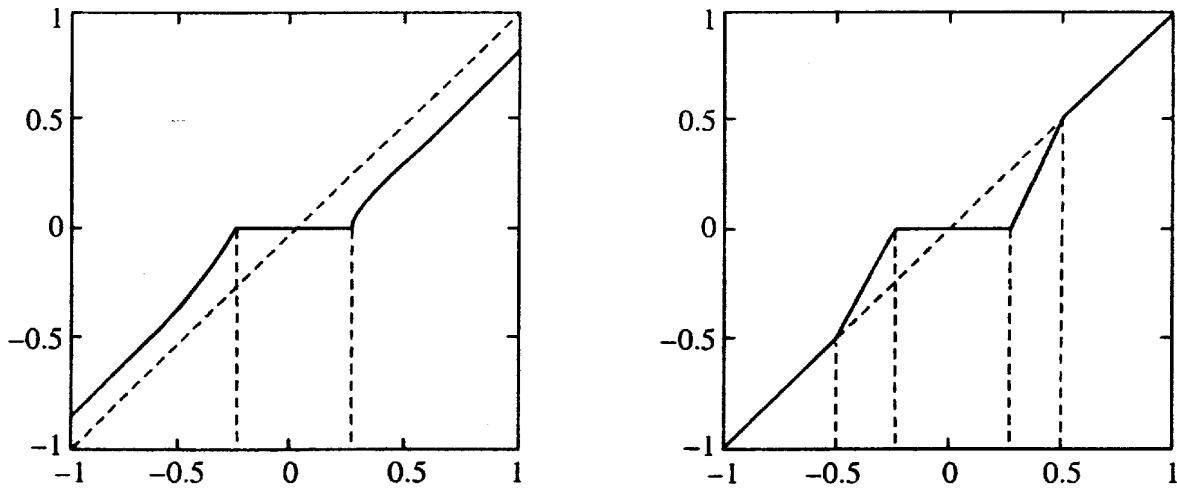


FIGURE 12.3 Shrinkage functions for garrote and semisoft (right figure) thresholding.

1. **For hard threshold:**

$$D_H^\lambda(w) \equiv \begin{cases} w & \text{for all } |w| > \lambda \\ 0 & \text{otherwise} \end{cases} \quad (12.8)$$

2. **For soft threshold:**

$$D_S^\lambda(w) \equiv \operatorname{sgn}(w) \max(0; |w| - \lambda) \quad (12.9)$$

3. **For garrote:**

$$D_G^\lambda(w) \equiv \begin{cases} \left(w - \frac{\lambda^2}{w} \right) & \text{for all } |w| > \lambda \\ 0 & \text{otherwise} \end{cases} \quad (12.10)$$

4. *Semisoft:*

$$D_{SS}^{\lambda_1, \lambda_2}(w) \equiv \begin{cases} 0 & |w| \leq \lambda \\ \text{sgn}(w) \frac{\lambda_2(|w| - \lambda_1)}{\lambda_2 - \lambda_1} & \lambda_1 < |w| \leq \lambda_2 \\ w & |w| > \lambda_2 \end{cases} \quad (12.11)$$

12.5 SHRINKAGE RULES

The shrinkage rules determine how the thresholds are calculated. Let λ denotes the threshold. For convenience, the possible dependence of λ on the multiresolution level or on the subband is suppressed in the notation. Certain rules calculate the threshold independent of the shrinkage function while others obtain different thresholds for different threshold functions. In addition certain rules assume a unit noise scale, $\sigma = 1$; others do not. We indicate the assumptions of each method as we describe them in more detail.

12.5.1 Universal

The universal rule was proposed by Donoho and Johnstone as a global rule for one-dimensional signals. Regardless of the shrinkage function, for a signal of size n , with noise from a standard normal distribution $N(0,1)$, the threshold is:

$$\lambda^U = \sqrt{2 \log n}$$

If data is not normalized with respect to noise-standard deviation, we estimate $\hat{\sigma}$, according to Eq. (12.5) and use $\lambda^U = \hat{\sigma} \sqrt{2 \log n}$ as threshold.

12.5.2 Minimizing the False Discovery Rate

Introduced by B. Vidakovic [3] for one-dimensional data, the minFDR rule determines the same global threshold for all shrinkage functions by keeping the expected value of the fraction of coefficients erroneously included in the reconstruction below a given fraction q . Given the N wavelet coefficients $\{w_k, k = 1, 2, \dots, N\}$, first it compute p -values,

$$p_k = 2[1 - \Phi(|w_k|/\hat{\sigma})]$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution and $\hat{\sigma}$ is an estimate of the noise-standard deviation. Then, it orders the p_k values as:

$$p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(N)}$$

Starting with $k = 1$, let m be the largest index such that

$$p_{(m)} \leq \frac{m}{N} q$$

The threshold is then obtained as:

$$\lambda = \hat{\sigma} \Phi^{-1} \left(1 - \frac{P(m)}{2} \right)$$

As $\hat{\sigma}$ is already in, the threshold is applied according to Eq. (12.7). Thus,

$$\lambda^{\min FDR} = \hat{\sigma} \Phi^{-1} \left(1 - \frac{P(m)}{2} \right)$$

12.5.3 Top

The *Top* rule for one-dimensional signal is a global method, independent of the shrinkage function. Given p as the fraction of the largest coefficients to keep, the threshold λ is set to be the $(1 - p)$ th quantile of the empirical distribution of the absolute values of the wavelet coefficients. It is applied to the coefficients using Eq. (12.7).

12.5.4 Sure

For one-dimensional data, thresholds derived by minimizing Stein's Unbiased Risk Estimate (SURE) depend on the shrinkage function and on the multiresolution level. The generalization to images can be achieved in either level- or subband-dependent manner. In the latter case, the threshold on subband s is:

$$\lambda_s = \arg \min_{\lambda \geq 0} [SURE(\lambda, w_s)]$$

where w_s denotes the detail coefficients from subband s and $SURE(\lambda, w_s)$ denotes the corresponding Stein's unbiased estimate of the risk corresponding to a specific shrinkage function. For example, the threshold on subband s to be used with the soft shrinkage function,

$$\lambda_s^S = \arg \min_{\lambda \geq 0} [SURE^S(\lambda, w_s)] \quad (12.12)$$

where

$$SURE^S(\lambda, w_s) = N_s + \sum_{k=1}^{N_s} [\min(|w_k|, \lambda)]^2 - 2[\# \text{ of } w_k : |w_k| \leq \lambda]$$

and N_s is the number of coefficients w_k in $\{w_s\}$.

The threshold above assumes $\sigma = 1$ (noise-standard deviation). For data with non-unit noise variance, the coefficients are standardized by an appropriate $\hat{\sigma}$ estimate before calculating the threshold in Eq. (12.12). The level dependent implementation is similar, except that instead of using the coefficients on subband, one uses the coefficients on a level.

It was shown by Donoho and Johnstone that, in the case where the wavelet coefficient decomposition is sparse, a hybrid method combining the universal and the SURE thresholds is preferable over SURE. This hybrid method, when combined with the soft shrinkage function is referred to as *SureShrink* in the literature. If

$$\frac{1}{N_s} \sum_{n=1}^{N_s} \left(\left(\frac{w_n}{\hat{\sigma}} \right)^2 - 1 \right) \leq \frac{(\log_2 N_s)^{3/2}}{\sqrt{N_s}}$$

then *SureShrink* uses universal threshold, otherwise the SURE threshold is used for the coefficients on subband s .

12.5.5 Translation Invariant Thresholding

It has been noted by practitioners that the wavelet thresholding with the universal threshold suffers from artifacts of various kinds. In other words, in the vicinity of discontinuities, the wavelet thresholding estimators can exhibit pseudo-Gibbs phenomenon, alternating undershoot and overshoot of a specific target level. Coifman and Donoho proposed the use of the translation invariant wavelet thresholding scheme which helps to suppress these artifacts. The idea is to correct unfortunate misalignment between features in the function of interest and features in the basis. When the function of interest contains several discontinuities, the approach of Coifman and Donoho is to apply a range of shifts in the function and average over several results so obtained.

12.5.6 BayesShrink

The BayesShrink rule uses a Bayesian mathematical framework for images to derive subband dependent thresholds that are nearly optimal for soft thresholding. The formula for the threshold on a given subband s for the model given by the equation with zero mean variable X

$$X_{i,j}(t) = S_{i,j}(t) + \sigma \varepsilon_{i,j}(t), \quad i = 1, 2, \dots, I, \quad j = 1, 2, \dots, J, \quad \varepsilon_{i,j} \sim N(0, 1)$$

is:

$$\lambda_s = \frac{\hat{\sigma}^2}{\hat{\sigma}_s^2}$$

where $\hat{\sigma}^2$ is the estimated noise variance and $\hat{\sigma}_s^2$ is the estimated signal variance on the subband considered (see Figure 12.4). Refer Chapter 11 on image compression to learn more about subbands.

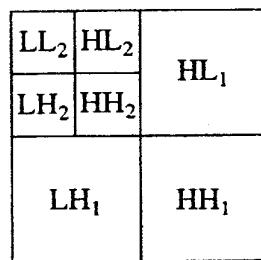


FIGURE 12.4 Subbands of the filtered image.

The noise variance is estimated as the median of the absolute deviation of the diagonal detail coefficients on the finest level (i.e., subband HH₁). The estimate of the signal standard deviation in subband s is:

$$\hat{\sigma}_s = \sqrt{\max(\hat{\sigma}_X^2 - \hat{\sigma}_s^2, 0)}$$

where $\hat{\sigma}_X^2 = \frac{1}{N_s} \sum_{k=1}^{N_s} w_k^2$ is an estimate of the variance of the observations, with N_s being the number of the wavelet coefficients w_k on the subband under consideration. In case $\hat{\sigma}^2 \geq \hat{\sigma}_s^2$, the threshold is set to $\lambda_s = \max(|w_k|)$ and all the coefficients from the subband is set to zero. These thresholds are applied according to Eq. (12.6).

12.6 DENOISING IMAGES WITH MATLAB

A quick way to denoise an image with it is to use the wavelet toolbox of MATLAB 6 which allow us to decompose an image and denoise it. First of all, we have to apply the DWT to the image to separate the horizontal, vertical and diagonal details at different level (up to 5). More details about 2D wavelet analysis is given in Chapter 11.

Figure 12.5 shows the MATLAB window. Here we have the original image on the upper left corner. We apply the DWT based on Haar wavelet at 2 levels on the image. The resulting

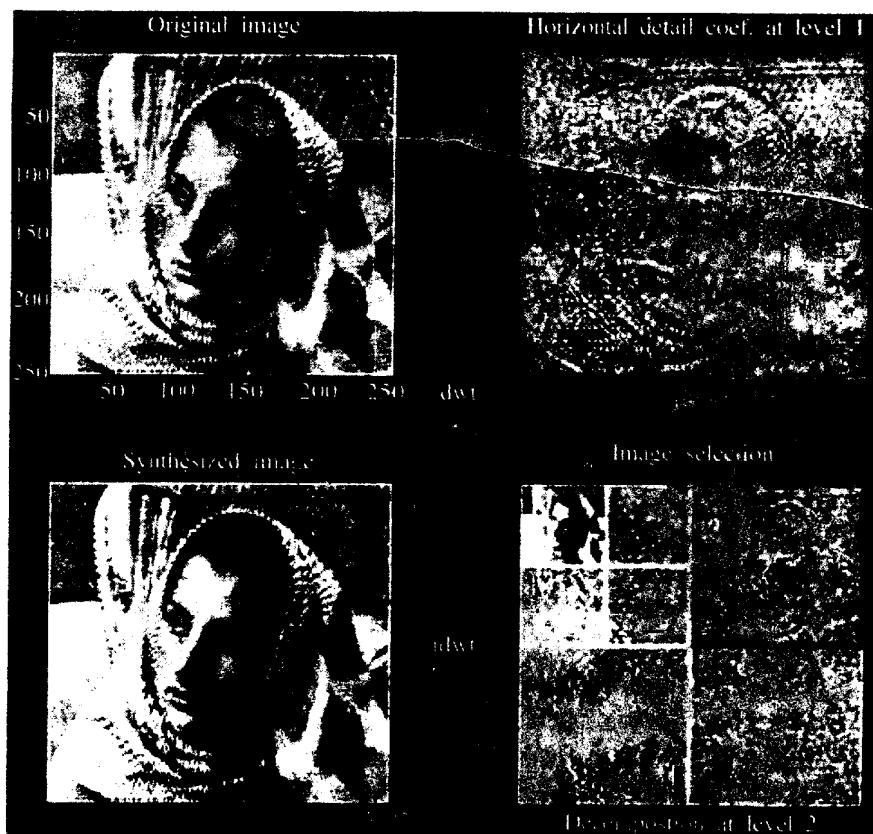


FIGURE 12.5 Denoising an image.

image is on the bottom right corner. Using the inverse DWT, we can of course recover our image. If we don't make any change, we have, like here, the same noisy image on the bottom left corner. We can see the horizontal detail coefficient at level 1 on the upper right corner.

The denoising method consists of choosing a threshold value for coefficients in each subband level. We then play with these coefficients to minimize the speckles without losing information about the original image.

The denoising window of Matlab Toolbox is shown in Figure 12.6. Using the scroll bars, we can choose threshold for the coefficients in different levels. This is one of the easiest way

Level	Select	Threshold
2		72.36
1		62.19

FIGURE 12.6 MATLAB window to choose threshold level.

of doing the operation of denoising. Figure 12.7 shows the histogram of wavelet coefficients which appear in the same denoising window of Matlab Toolbox. Looking at these histograms also one can decide the threshold level. Usually, a level 5 decomposition of the image allows us to have complete control over denoising by choosing different threshold for different levels. But often, a level 2 decomposition is good enough for simple noises.

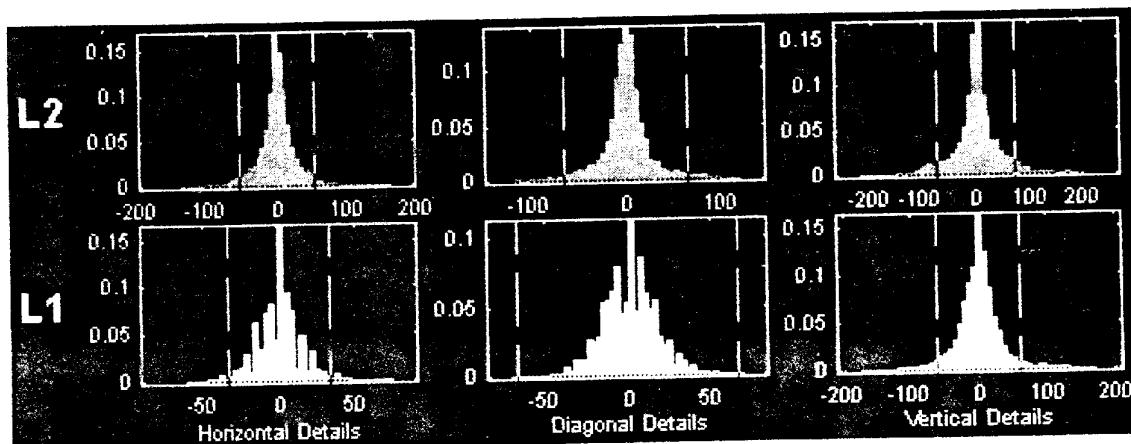


FIGURE 12.7 Distribution (Histogram) of wavelet coefficients at various subbands.

The way to find good thresholds for all the coefficient is not easy and automation of finding these thresholds is very hard to compute. It often leads to statistical method to recover the best thresholds and it is not sure that the algorithms find the best one. According to specialists, manual thresholding stays the best way to find the best denoising result even if it takes a lot of time to achieve.

12.7 MATLAB PROGRAMS FOR DENOISING

(This code uses functions in MATLAB wavelet toolbox.)

1. *VisuThresh – Visually calibrated Adaptive Smoothing*
function [x] = VisuThresh(y, type)
if nargin < 2,
 type = 'Soft';
end
thr = sqrt(2*log(length(y))) ;
if strcmp(type, 'Hard'),
 x = HardThresh(y, thr);
else
 x = SoftThresh(y, thr);
end
% end of function VisuThresh
function x = HardThresh(y, t)
x = y .* (abs(y) > t);
% end of function HardThresh

function x = SoftThresh(y, t)
res = (abs(y) - t);
res = (res + abs(res))/2;
x = sign(y).*res;
% end of function SoftThresh
2. *SureShrink: Adaptive Threshold Selection Using Principle of SURE*
function [x, thresh] = SUREThresh(y)
% SURE refers to Stein's Unbiased Risk Estimate.

thresh = ValSUREThresh(y);
x = HardThresh(y, thresh);
% end of function SureThresh

function thresh = ValSUREThresh(x)
a = sort(abs(x)).^2 ;
b = cumsum(a);
n = length(x);
c = linspace(n-1, 0, n);
s = b+c.*a;

```

risk = (n - (2 .* (1:n)) + s)/n;
[guess, ibest] = min(risk);
thresh = sqrt(a(ibest));
% end of function ValSUREThresh

```

12.8 SIMULATION FOR FINDING EFFECTIVENESS OF THRESHOLDING METHOD

We have a variety of wavelets and thresholding method to choose for denoising a particular signal. It has been found that for certain kind of signals certain wavelet and thresholding method maximally remove noise. The only way of finding this combination is simulation. In this section we describe a simulation method where we find the performance of different thresholding methods on different kinds of signals that is corrupted with different levels of noise.

In all cases, the data (x_i, y_i) are generated from a model of the form:

$$y_i = f(x_i) + \sigma \varepsilon_i, \quad i = 1, 2, \dots, N$$

where $\{x_i\}$ are equispaced in $[0, 1]$ with $x_0 = 0, x_N = 1$

The factors to vary during simulation are:

1. The sample sizes N
2. The test functions $f(x)$
3. The values of σ

We may choose three levels for factors one and three.

For each combination of these factor levels, a simulation run is to be repeated 100 times (a good enough number of repetition) holding all factor levels constant, except the ε_i which are regenerated for each run. In order to compare the behaviour of the various estimation methods, we may use the following six different criteria: MSE, L1, RMSE, RMSB, MXDV and CPU. These criteria are computed as follows:

MSE: This is the average over 100 runs of

$$\frac{1}{n} \sum_{i=1}^n [f(x_i) - \hat{f}(x_i)]^2$$

L1: This is the average over the 100 runs of

$$\frac{1}{n} \sum_{i=1}^n |f(x_i) - \hat{f}(x_i)|$$

RMSE: The mean squared error can be computed for each run and average over the 100 runs. Then its square root is taken.

RMSB: Let $\bar{f}(x_i)$ be the average of $\hat{f}(x_i)$ over the 100 runs. The RMSB is the square root of

$$\frac{1}{n} \sum_{i=1}^n [f(x_i) - \bar{f}(x_i)]^2$$

MXDV: This is the average over 100 runs of

$$\max_{1 \leq i \leq n} |f(x_i) - \hat{f}(x_i)|$$

CPU: This is the average over 100 runs of the CPU time.

Symmlet 8 wavelet basis [as described on page 198 of Daubechies (1992)] and the Coiflet 3 basis [as described on page 258 of Daubechies (1992)] are the commonly used wavelet system for simulation experiments in denoising. The set of test functions we may use for comparison are following:

Step: This function is very hard to estimate with linear methods because of its jumps, but relatively easy for non-linear wavelet estimators.

Wave: This is a sum of two periodic sinusoids. Since this signal is smooth, linear methods compare favourably with non-linear ones.

Blip: This is essentially the sum of a linear function with a Gaussian density and has been often used as a target function in non-parametric regression. To make the jump induced by the assumed periodicity visually clear, the function has been periodically rotated so the jump is at $x = 0.8$.

Blocks: This step function has many more jumps than the Step above and has been used in several Donoho and Johnstone papers, for example, Donoho and Johnstone (1994).

Bumps: This also comes from Donoho and Johnstone and is very challenging for any smoother.

HeaviSine: Another Donoho and Johnstone example. This looks promising for linear smoothers except for the two jumps.

Doppler: The final Donoho and Johnstone example. The time varying frequency makes this very hard for linear methods, with power spread all across the spectrum. It is more suitable for the wavelets, with their space time localization.

Angles: This function is piecewise linear and continuous but has big jumps in its first derivatives.

Parabolas: This function is piecewise parabolic. The function and its first derivative are continuous but there are big jumps in its second derivative. It is ideal for the Symmlet 8.

Figure 12.8 shows the twelve signals that are commonly used in simulation studies. These are available in wavelet toolbox.

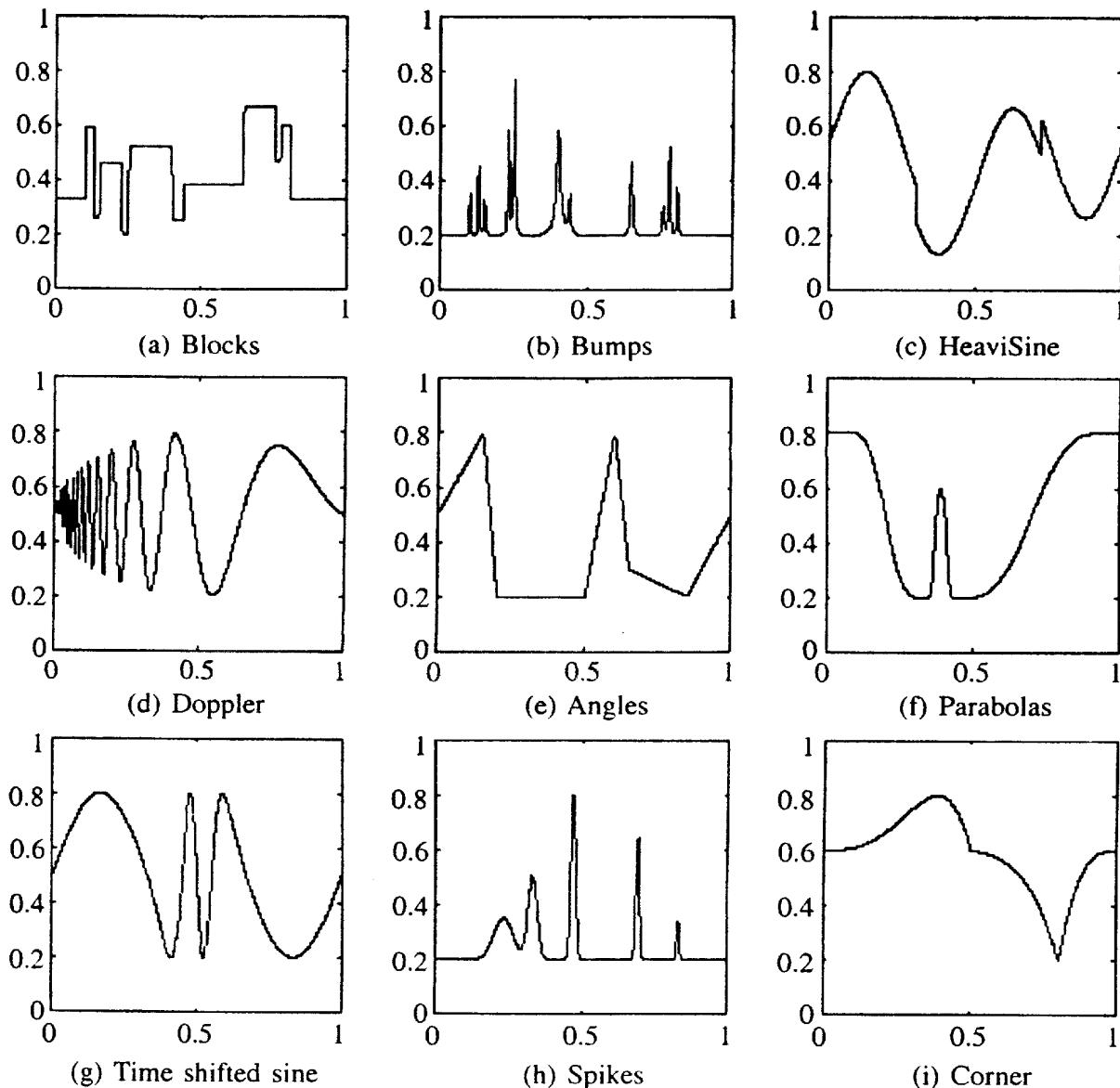


FIGURE 12.8 Test signals used in simulation studies.

SUMMARY

Fourier smoothing has long been the method of choice to suppress noise but recently methods based on the wavelet transformation have become increasingly popular. In principle they offer much greater flexibility to analyze and process data. Whereas Fourier-filtering affects all data points in the same manner, wavelets allow different parts of spectra to be filtered individually, in principle promising a considerably refined and improved treatment.

EXERCISES

- 12.1** Generate a test signal of the type shown in Figure 1(a) and then corrupt with white noise as shown in Figure 1(b). Find wavelet transform coefficients for the Haar wavelet with filter coefficients as $(1, 1)$ and the wavelet transform coefficients with the coefficients $(1, 0, 0, 1)$. Then denoise by setting all the wavelet coefficients on the 4 lowest scales to zero when they are in absolute value less than 1. Plot the resulting denoised signal. Explain the differences observed.

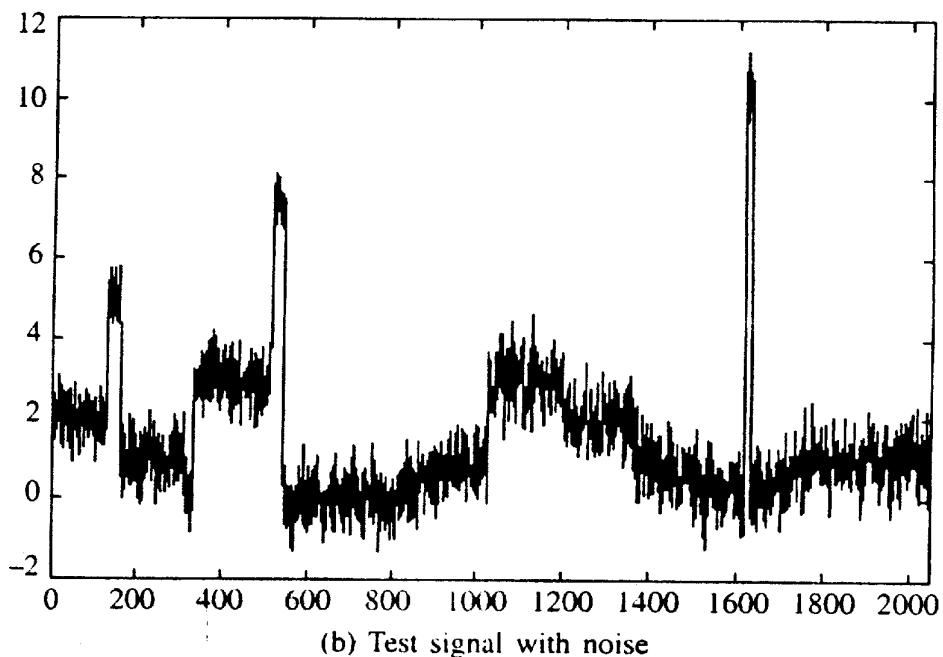
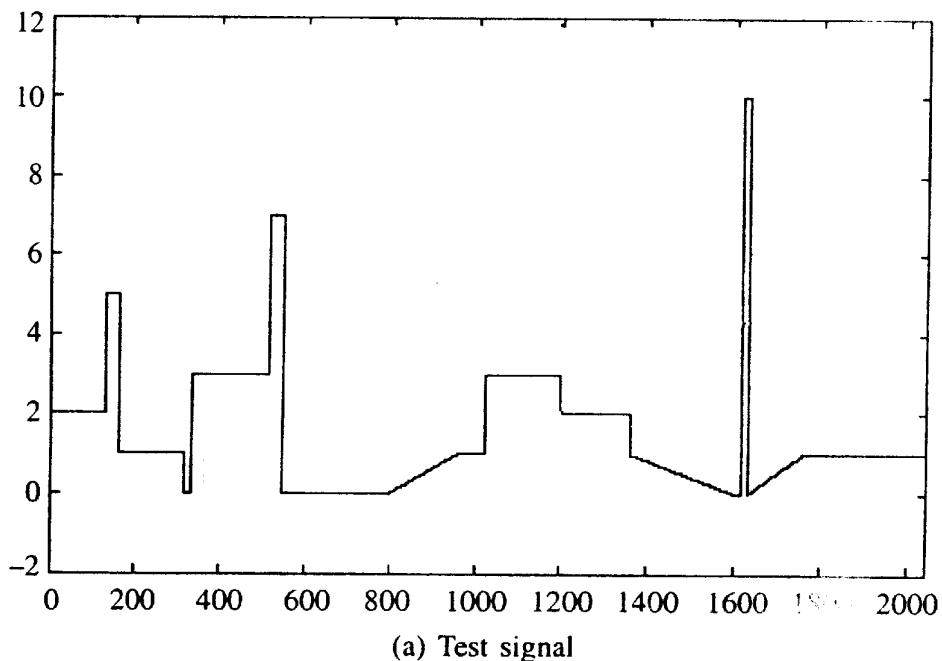


FIGURE 1 Signals for testing denoising ability of wavelet.

- 12.2 Prove that if the variance of the signal is a constant then the variance of the wavelet transform is a constant in each resolution level. Is this true for any wavelet transform (orthogonal, biorthogonal, ...)?
- 12.3 Consider a signal s with correlation matrix C and suppose S is the 1D wavelet transform of s and that S has correlation matrix D . Prove that D is a 2D wavelet transform of C . Is it a square or a rectangular transform?

REFERENCES

- [1] Donoho, D. and I. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association*, **90**: 1200–1224, 1995.
- [2] Donoho, D. and I. Johnstone, Minimax estimation via wavelet shrinkage, *Annals of Statistics*, **26**: 879–921, 1998.
- [3] Vidakovic, B., *Statistical Modeling by Wavelets*, John Wiley & Sons, NY, 1999.

CHAPTER

13

Spline Wavelets: Introduction and Applications to Computer Graphics

13.1 INTRODUCTION TO CURVES AND SURFACES

There are three forms for representing curves and surfaces. They are parametric, implicit and explicit forms. Implicit and explicit forms are referred as non-parametric forms.

A plane curve can be represented in the parametric form as $x = x(t)$ and $y = y(t)$ where coordinates of the points (x, y) are expressed as a function of parameter t within a closed interval $t_1 \leq t \leq t_2$. The functions $x(t)$ and $y(t)$ are assumed to be continuous with sufficient number of continuous derivatives. The parametric curve is said to be of class r , if functions have continuous derivatives up to the order r , inclusively. In vector notation the parametric curve can be specified by a vector valued function $r = r(t)$. The implicit equation for the plane curve is of the form:

$$f(x, y) = 0$$

If it is linear, it is straight-line curve and if $f(x, y)$ is of the second order, i.e. $ax^2 + 2bxy + cy^2 + 2dx + 2ey + h = 0$ then it represents a variety of plane curves called **conic sections**. The implicit equation of a plane curve can be written as an intersection curve between a parametric surface and a plane. The explicit form can be considered as a special case of parametric and implicit form. If t can be considered as a function of x or y , we can easily eliminate t and generate the explicit form:

$$y = F(x) \quad \text{or} \quad x = G(y)$$

The parametric representation of a space curve is:

$$x = x(t), y = y(t), z = z(t), t_1 \leq t \leq t_2$$

The implicit representation for a space curve can be written as intersection between two implicit surfaces $f(x, y, z) = 0 \cap g(x, y, z) = 0$ or parametric and implicit surfaces $r = p(\sigma, t) \cap r = q(u, v)$. If t can be expressed as a function of x, y or z , t can be easily eliminated to generate the explicit form:

$$y = Y(x) \quad \text{and} \quad z = Z(x)$$

Similar to curves, there are three ways to represent surfaces, namely parametric, implicit and explicit methods. In parametric representation, the coordinates (x, y, z) of the surface patch are expressed as functions of parameters u and v in a closed rectangle:

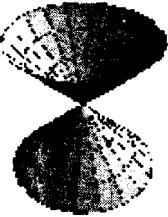
$$x = x(u, v), \quad y = y(u, v), \quad z = z(u, v), \quad u_1 \leq u \leq u_2, \quad v_1 \leq v \leq v_2 \quad (13.1)$$

The functions $x(u, v)$, $y(u, v)$ and $z(u, v)$ are continuous and possess a sufficient number of continuous partial derivatives. The parametric surface is said to be of class r , if functions have continuous partial derivatives up to the order r , inclusively. In vector notation the parametric curve can be specified by a vector valued function $r = r(u, v)$. An implicit surface is defined as locus of points whose coordinates (x, y, z) satisfy the equation of the form $f(x, y, z) = 0$. When f is linear in variables x, y, z , it represents a plane. When f is of second degree in x, y, z , it represents quadrics:

$$ax^2 + by^2 + cz^2 + dxy + eyz + hxz + kx + ly + mz + n = 0 \quad (13.2)$$

Some of the quadric surfaces such as elliptic paraboloid, hyperbolic paraboloid and parabolic cylinder have explicit forms. Paraboloid of revolution is a special type of elliptic paraboloid where major and minor axes are the same. The rest of the quadrics have implicit form including ellipsoid, elliptic cone, elliptic cylinder, hyperbolic cylinder, hyperboloid of one sheet or two sheets where hyperboloid of revolution is a special form. The natural quadrics, sphere, circular cone and circular cylinder, are widely used in mechanical design and CAD/CAM systems and they result from standard manufacturing operations like rolling, turning, filleting, drilling and milling. Some standard quadric surfaces are tabulated in Table 13.1.

TABLE 13.1 Standard Quadric Surfaces

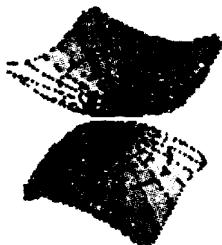
	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0$	Forms a cone. A signal of this quadric surface is the constant equalling zero.
	$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$	Forms an ellipsoid. The sign of each variable is positive.

(cont.)

TABLE 13.1 Standard Quadric Surfaces (Cont.)

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

Forms a hyperboloid of one sheet. A signal of this shape is the presence of **one** minus sign. Can you see the difference between this equation and the equation for the cone given above?



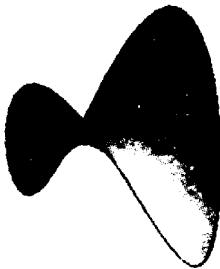
$$\frac{z^2}{c^2} - \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

Forms a hyperboloid of two sheets. A signal of this shape is the presence of **two** minus signs.



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z}{c}$$

Forms an elliptic paraboloid. In this equation, x and y are quadratic and z is linear. The quadratic terms have the **same** sign.



$$\frac{y^2}{b^2} - \frac{x^2}{a^2} = \frac{z}{c}$$

Forms a hyperbolic paraboloid. This equation also contains one linear variable and two quadratic terms. The quadratic terms in this equation **differ** in sign.

If the implicit equation can be solved for one of the variables as a function of the other two, say, z is solved in terms of x and y , we obtain an explicit surface $z = F(x, y)$. A summary of representation of curves and surfaces is given in Table 13.2.

TABLE 13.2 Curves and Surfaces

<i>Geometry</i>	<i>Parametric</i>	<i>Implicit</i>	<i>Explicit</i>
Plane curves	$x = x(t)$ and $y = y(t)$, $t_1 \leq t \leq t_2$	$f(x, y) = 0$ or $r = r(u, v) \cap \text{plane}$	$y = F(x)$
Space curves	$x = x(t), y = y(t), z = z(t)$, $t_1 \leq t \leq t_2$	$f(x, y, z) = 0 \cap g(x, y, z) = 0$ $r(u, v) = 0 \cap f(x, y, z) = 0$ $r = p(\sigma, t) \cap r = q(u, v)$	$y = y(x) \cap$ $z = z(x)$
Surfaces	$x = x(u, v), y = y(u, v)$, $z = z(u, v)$, $u_1 \leq u \leq u_2, v_1 \leq v \leq v_2$	$f(x, y, z) = 0$	$z = f(x, y)$

Comparisons of different methods of curves and surface representation are shown in Table 13.3.

TABLE 13.3 Comparison of Different Methods of Curves and Surface Representation

	<i>Disadvantages</i>	<i>Advantages</i>
Explicit	<ul style="list-style-type: none"> Infinite slopes are impossible if $f(x)$ is a polynomial. Axis dependent (difficult to transform). Closed and multi-valued curves are difficult to represent. 	<ul style="list-style-type: none"> Easy to trace.
Implicit	<ul style="list-style-type: none"> Difficulty in fitting and manipulating free-form shapes. Axis dependent. Complex to trace. 	<ul style="list-style-type: none"> Closed and multi-valued curves and infinite slopes can be represented. Point classification (solid modelling, interface check) is easy. Intersections/offsets can be represented.
Parametric	<ul style="list-style-type: none"> High flexibility complicates intersections and point classifications. 	<ul style="list-style-type: none"> Closed and multi-valued curves and infinite slopes can be represented. Axis dependent (easy to transform). Easy to trace. Easy in fitting and manipulating free-form shapes.

13.1.1 Spline Curves and Surfaces

The idea of spline is based on the use of the “drafting spline”, a thin flexible strip used to draw a smooth curve passing through a given series of points. The physical spline is shaped by attaching appropriate lead weights to the flexible strip. If the weights act as simple supports, the curve becomes piecewise cubic polynomial, continuous up to second derivatives at each support, i.e., continuous with respect to position, tangent and the curvature. Spline curve is a curve with piecewise cubic polynomial function whose first and second derivatives are continuous across various curve sections. It refers to a composite curve formed with polynomial

sections satisfying specific continuity conditions at the boundary of the pieces. Two sets of orthogonal spline curves define a spline surface. Splines are used in graphical applications to design curves and surface shapes, to digitize drawings for computer storage and to specify animation paths for objects or camera in a scene. Typical CAD applications of splines include the design of automobile bodies, aircraft and spacecraft surfaces and ship hulls.

A curve is specified by a set of coordinate position called **control points**. These control points are then fitted with piecewise continuous parametric polynomial functions in two ways. If the polynomial sections are fitted so that the curve passes through each control point, the resulting curve said to interpolate the set of control points. When the polynomials are fitted to the general control-point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points. Interpolation curves are used to digitize drawings or to specify animation paths (refer Figure 13.1). Approximation curves are used as design tools to structure object surfaces. Approximation spline surfaces are created for design applications.



(a) Interpolation—the curve goes precisely through the points



(b) Approximation—the curve doesn't necessarily go through the points only near to

FIGURE 13.1 Interpolation and approximation curves.

A spline curve is defined, modified and manipulated with operations on control points. By interactively selecting the spatial positions for control points, a designer can set up the initial curve and later reposition the control points to restructure the shape of the curve. The curve can be rotated, scaled or translated by applying transformations to the control points. Extra control points can be inserted to aid a designer. The convex polygon, which encloses the set of control points, is called the **convex hull** and it provides a measure for the deviation of a curve or a surface in the region bounding the control points. The set of line segments connecting the control points is called the **control graph**.

To ensure smooth transition from one section of piecewise parametric curve to the next, various continuity conditions have to be satisfied. Each section of a spline is described with a set of parametric coordinate functions of the form:

$$x = x(u), \quad y = y(u), \quad z = z(u), \quad u_1 \leq u \leq u_2 \quad (13.3)$$

We set parametric continuity by matching the parametric derivatives of the adjoining curve sections at their common boundary. Zero-order parametric continuity described a C^0 continuity, means simply that the curves meet. That is, the value of x , y and z evaluated at u_2 for the first

curve section are equal, to the values of x, y, z , respectively evaluated at u_1 for the next curve section. First parametric continuity, known as C^1 continuity means that the first parametric derivatives (tangent lines) of the coordinate functions for the two successive curve sections are equal at the intersection. Second-order parametric continuity or C^2 continuity, means that both first and second parametric derivatives of the two curve sections are same at the intersection. Higher order parametric continuities are defined similarly. Figure 13.2 shows examples of C^0 , C^1 and C^2 continuity. With second order continuity, the rates of change of the tangent vectors

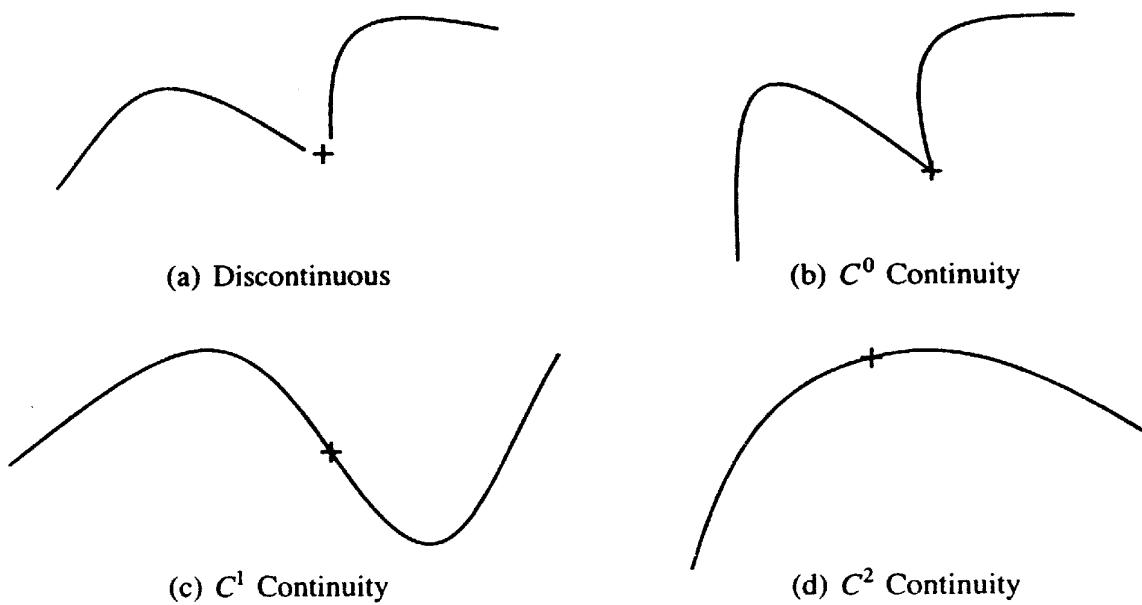


FIGURE 13.2 Piecewise construction of curve by joining two curve segments using different orders of continuity.

for the two sections are equal at the intersection. Thus, the tangent lines transitions form smoothly from one section of the curve to the next. But with first order continuity, the rates of change of the tangent vectors for the two sections can be quite different so that the general shapes of the two adjacent sections can change abruptly. First order continuity is often sufficient for digitizing drawings and some design applications while the second order continuity is useful for setting up animation paths for camera motion and many precision CAD requirements which requires smooth transition across the boundary.

Geometric continuity is an alternate method for joining two curve sections in which, we only require the parametric derivatives of the two sections to be proportional to each other at the common boundary instead of equal to each other. Zero-order geometric continuity described as G^0 continuity, is the same as zero-order parametric continuity. That is, the value of x, y and z evaluated at u_2 for the first curve section are equal, to the values of x, y, z respectively evaluated at u_1 for the next curve section. First-order geometric continuity, known as G^1 continuity means that the first parametric derivatives (tangent lines) of the coordinate functions for the two successive curve sections are proportional at the intersection. The direction of tangent vectors should be the same but the magnitudes may not be necessarily the same. Second-order geometric continuity or G^2 continuity, means that both first and second parametric

derivatives of the two curve sections are proportional at their boundary. Under G^2 continuity, curvatures of the two curve sections will match at the joining position. A curve generated with geometric continuity is similar to the one generated with parametric continuity but with the slight differences in the curve shape.

There are three equivalent methods for specifying a particular spline representation:

1. We can state the set of boundary conditions that are imposed on the spline.
2. We can state the matrix that characterizes the spline.
3. We can state the set of blending functions (or basis functions) that determine how specified geometric constraints on the curve are combined to calculate positions along the curved path.

Consider the parametric cubic polynomial representation for the x coordinate along the path of a spline section:

$$a_x u^3 + b_x u^2 + c_x u + d_x, \quad 0 \leq u \leq 1 \quad (13.4)$$

Boundary conditions for this curve might be set on the end point coordinates $x(0)$ and $x(1)$ and on the parametric first derivatives $x'(0)$ and $x'(1)$. These four boundary conditions are sufficient to determine the four coefficients a_x , b_x , c_x and d_x . From the boundary conditions, we can obtain the matrix that characterizes this spline curve as follows:

$$x(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{Bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{Bmatrix} = U.C \quad (13.5)$$

where U is the row matrix of powers of parameters u , C is the coefficient column matrix. We can write the boundary conditions in the matrix form and solve for the coefficient matrix C as:

$$C = M_{\text{spline}} * M_{\text{geom}} \quad (13.6)$$

where M_{geom} is the four element column matrix containing the geometric constraint values (boundary conditions) on the spline and M_{spline} is 4 by 4 matrix that transforms the geometric constraint values to the polynomial coefficients and provides a characterization for the spline curve. Matrix M_{geom} contains control points coordinates values and other geometric constraints that have been specified.

$$x(u) = U.M_{\text{spline}} * M_{\text{geom}} \quad (13.7)$$

The matrix M_{spline} characterizing a spline representation, sometimes called **basis matrix**, is particularly useful for transforming from one spline representation to another. Finally, we can expand the equations to obtain a polynomial representation for coordinate x in terms of the geometric constraint parameters as follows:

$$x(u) = g_k \text{BF}_k(u) \quad (13.8)$$

where g_k constraint parameters, such as control point coordinates and slope of the curve at control points and $\text{BF}_k(u)$ are polynomial blending functions. Splines are mostly specified by characterizing matrix and blending functions.

13.1.2 Cubic Spline Interpolation Methods

Interpolation polynomials can also be used to design object shapes. Cubic splines offer a compromise between flexibility and speed of computation. Compared to lower order polynomials, it is more flexible for modelling arbitrary curve shapes and compared to higher order polynomials, cubic splines require less calculation and memory and they are more stable. The parametric cubic polynomial that is to be fitted between each pair of control points can be represented by the following equations:

$$\begin{aligned}x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z\end{aligned}\quad (13.9)$$

For each of these three equations, we have to determine the values of the coefficients a , b , c and d in the polynomial representation for each of n curve sections between the $n + 1$ control points. This is done by setting the boundary conditions at the joint between the curve sections so that we can obtain numerical values for all the coefficients.

13.1.3 Hermite Spline Interpolation

A Hermite spline is an interpolating piecewise cubic polynomial with a specified tangent at each control point. Unlike natural cubic splines, Hermite splines can be adjusted locally because each curve section is only dependent on its endpoint constraints. If $P(u)$ represents a parametric cubic point function for the curve section between control points P_k and P_{k+1} then the boundary conditions that define this Hermite curve are:

$$P(0) = P_k, \quad P(1) = P_{k+1}, \quad P'(0) = DP_k, \quad P'(1) = DP_{k+1} \quad (13.10)$$

With DP_k and DP_{k+1} specifying the values for the parametric derivatives and control points P_k and P_{k+1} , respectively, we can write

$$P(u) = au^3 + bu^2 + cu + d, \quad 0 \leq u \leq 1$$

or

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} \quad (13.11)$$

The derivatives of the point function can be expressed as:

$$P'(u) = [3u^2 \quad 2u \quad 1 \quad 0] \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} \quad (13.12)$$

Substituting the end point values 0 and 1 into the previous equations, we get

$$\begin{Bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 1 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} \quad (13.13)$$

$$\begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 1 \end{bmatrix}^{-1} \begin{Bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{Bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{Bmatrix} \quad (13.14)$$

$$p(u) = [u^3 \quad u^2 \quad u \quad 1] M_H \begin{Bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{Bmatrix} \quad (13.15)$$

where M_H , the Hermite matrix is the inverse of the boundary constraint matrix. Finally, we can determine expressions for the Hermite blending functions by carrying the matrix multiplications and collecting the coefficients for the boundary constraints to obtain the polynomial form:

$$\begin{aligned} p(u) &= p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2) \\ &= p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + Dp_{k+1} H_3(u) \end{aligned} \quad (13.16)$$

The values of $P(0)$, $P(1)$, $P'(0)$, $P'(1)$ are called **geometric coefficients** and represent the known vector quantities in the equation. The polynomial coefficients of these vector quantities $H_k(u)$ for $k = 0, 1, 2, 3$ are referred to as blending functions because they blend the boundary constraint values (endpoint coordinates and slopes) to obtain each coordinate positions along the curve. By varying the parameter t in these blending functions from 0 to 1, several points on the curve segment can be obtained. Hermite polynomials can be useful for some digitizing applications where it may not be too difficult to specify or approximate curve slopes.

Cardinal splines and Kochek-Bartel splines are variations of Hermite polynomials. Cardinal splines are interpolating piecewise cubics with specified endpoint tangents at the boundary of each curve section. The difference is that we do not have to give the values for endpoint tangents. For a cardinal spline, the value for the slope at control points is calculated from the coordinates of the two adjacent control points. A parameter t called the **tension parameter** is introduced in the equations for parametric derivatives, which control how loosely or tightly the cardinal spline fits the input control points. Kochek-Bartel splines are extension of cardinal splines. Two additional parameters are introduced into the constraint equations to provide for further flexibility in adjusting the shape of the curve sections. The additional parameters apart from the tension parameter t are the bias parameter b and the continuity parameter c . The bias parameter b is used to adjust the amount that the curve bends at each end of a section, so that the curve sections can be skewed towards one end or the other. Parameter c controls continuity of the tangent vector across the boundaries of the sections.

13.1.4 Cubic Splines

Cubic splines are represented by piecewise cubic polynomials with second derivative continuity at the joint between the segments. With parameter normalized between zero and one, the cubic spline is just a special case of Hermite interpolation where the first derivative values at the ends of each segment are so chosen as to ensure second derivative continuity. The commonly used constraints are:

- (a) known end tangent vectors P'_0 and P'_{m-1}
- (b) known derivatives at the endpoints P_0 and P_{m-1} both made equal to zero that is referred to as natural cubic spline. Taking the i th and $(i-1)$ th segments and applying the boundary conditions yield results in the equation:

$$P'_{i-1} + 4P'_i + P'_{i+1} = 3(P_{i+1} - P_{i-1})$$

The iterative application of this equation to all cubic spline segments with the first case of known end tangent vectors results in the following matrix form of the cubic spline:

$$\begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix} \begin{bmatrix} P'_0 \\ P'_1 \\ . \\ . \\ . \\ P'_{m-1} \end{bmatrix} = \begin{bmatrix} P' \\ 3(P_2 - P_0) \\ . \\ . \\ 3(P_{m-1} - P_{m-3}) \\ P'_{m-1} \end{bmatrix} \quad (13.17)$$

Solution of this matrix equation yields the values of all tangent vectors.

$$\begin{bmatrix} P'_0 \\ P'_1 \\ . \\ . \\ . \\ P'_{m-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P'_0 \\ 3(P_2 - P_0) \\ . \\ . \\ 3(P_{m-1} - P_{m-3}) \\ P'_{m-1} \end{bmatrix}$$

or

$$[P'] = [M]_{Cs}^{-1} [G]_{Cs} \quad (13.18)$$

The matrix M_{Cs} is tri-diagonal and inversion of the matrix is computationally less intensive. Solution of this system of m equations in m unknowns can be obtained by Gaussian elimination.

For the second case of natural cubic splines the first and second derivatives at the first and last points are set to zero. The equation is:

$$\begin{aligned}
 P(t) &= a_3t^3 + a_2t^2 + a_1t + a_0 & P'_1 &= 3a_3t^2 + 2a_2t + a_1 \\
 P(0) &= a_0 & P(1) &= 3a_3t^2 + 2a_2t + a_1 \\
 P'(0) &= a_1 & P'(1) &= 3a_3 + 2a_2 + a_1 \\
 \therefore a_0 &= P(0) & a_1 &= P'(0) \\
 a_2 &= -3P(0) + 3P(1) - 2P'(0) - P'(1) & a_2 &= -2P(0) - 2P(1) + P'(0) + P'(1)
 \end{aligned} \tag{13.19}$$

Putting first and second derivatives at endpoints as zero yields the following equations:

$$2P'_0 + P'_1 = 3(P_1 - P_0)$$

and

$$P'_{m-2} + 2P'_{m-1} = 3(P_{m-1} - P_{m-2})$$

The m equations in m unknowns can be expressed in terms of matrix as follows:

$$\left[\begin{array}{cccccc|c} 2 & 1 & . & . & . & . & . & P'_0 \\ 1 & 4 & 1 & . & . & . & . & . \\ . & 1 & 4 & 1 & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & 1 & 4 & 1 & . \\ . & . & . & . & . & 1 & 2 & P'_{m-1} \end{array} \right] = \left[\begin{array}{c} 3(P_1 - P_0) \\ 3(P_2 - P_0) \\ 3(P_3 - P_1) \\ . \\ 3(P_{m-1} - P_{m-3}) \\ 3(P_{m-1} - P_{m-2}) \end{array} \right] \tag{13.20}$$

Natural cubic splines are interpolation curves where the two adjacent curve sections have the same first and second parametric derivatives. They have C^2 continuity. If there are $n + 1$ control points then we have n curve sections with a total of $4n$ polynomial coefficients to be determined. At each $n - 1$ interior points, we have four boundary conditions; the two curve sections on either side of the control point must have the same first and second parametric derivatives at the control points and each curve must pass through that control point. Thus we have $4n - 4$ equations to satisfy $4n$ polynomial coefficients. An additional equation is obtained from the first control point P_0 , the position of the beginning of the curve, and another condition from the last control point P_n . The remaining two conditions can be obtained either by setting second derivatives at P_0 and P_n to zero or by adding two extra dummy points at each end of the original control point sequence P_{-1} and P_{n+1} . Though natural cubic splines are a mathematical model for drafting splines, they have a major disadvantage that they do not allow local control and if the position of anyone one control point is altered, the entire curve is affected.

The blending function matrix for normalized cubic spline is the same one used in Hermite interpolation:

$$[t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{13.21}$$

Example: Consider four two-dimensional point vectors $P_0[0 \ 0]$, $P_1[2 \ 1]$, $P_2[4 \ 4]$, $P_3[6 \ 0]$. For the case of known end tangent vectors, the tangent vectors are found as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P'_0 \\ P'_1 \\ P'_2 \\ P'_3 \end{bmatrix} = \begin{bmatrix} (1 \ 1) \\ 3[(4-0) \ (4-0)] \\ 3[(6-2) \ (0-2)] \\ (-1 \ 1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 12 & 12 \\ 12 & -3 \\ 1 & 1 \end{bmatrix} \quad (13.22)$$

P'_0 and P'_3 are known values and hence P'_1 and P'_2 can be determined.

For case of natural cubic splines where the first and second derivatives at endpoints are put as zero, the tangent vectors are calculated as follows:

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} P'_0 \\ P'_1 \\ P'_2 \\ P'_3 \end{bmatrix} = \begin{bmatrix} 6 & 3 \\ 12 & 12 \\ 12 & -3 \\ 6 & -12 \end{bmatrix} \quad (13.23)$$

or

$$\begin{bmatrix} P'_0 \\ P'_1 \\ P'_2 \\ P'_3 \end{bmatrix} = \frac{1}{45} \begin{bmatrix} 26 & -7 & 2 & -1 \\ -7 & 14 & -4 & 2 \\ 2 & -4 & 14 & -7 \\ -1 & 2 & -7 & 26 \end{bmatrix} \begin{bmatrix} 6 & 3 \\ 12 & 12 \\ 12 & -3 \\ 6 & -12 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 2 & 3 \\ 2 & 0 \\ 2 & 6 \end{bmatrix} \quad (13.24)$$

13.1.5 Splines in Signal and Image Processing

Splines constitute an elegant framework for dealing with interpolation and discretization problems. They find wide applications in computer graphics and computer aided design. Spline representations are ideally suited for signal and image processing applications. The multiresolution properties of splines make them ideal candidate for constructing wavelet bases which has got lot of applications in image processing like image reconstruction from projected data, image compression, image registration, edge detection, etc. Their main advantages can be summarized as follows:

- One can always obtain a continuous representation of a discrete signal by fitting it with a spline in one or more dimensions. The fit may be exact (interpolation) or approximate (least-squares or smoothing splines). Spline fits are usually preferable to other forms of representations (e.g. Lagrange polynomial interpolation) because they have a lesser tendency to oscillate (minimum curvature property).
- Polynomial splines can be expressed as linear combinations of B-spline basis functions. For equally spaced knots, the spline parameters (B-spline coefficients) may be determined by simple digital filtering. There is no need for matrix manipulations!

- The primary reason for working with the B-spline representation is that the B-splines are compactly supported. They are the shortest functions with an order of approximation $n + 1$. This short support property is a key consideration for computational efficiency. Their simple analytical form also greatly facilitates manipulations.
- Splines are smooth and well-behaved functions (piecewise polynomials). Splines of degree n are $(n - 1)$ continuously differentiable. As a result, splines have excellent approximation properties. Precise convergence rates and error estimates are available.
- Splines have multiresolution properties that make them very suitable for constructing wavelet functions and for performing multi-scale processing.
- B-splines and their wavelet counterparts have excellent localization properties; they are good templates for time-frequency signal analysis.
- The family of polynomial splines provides design flexibility. By increasing the degree n , we can progressively switch from the simplest piecewise constant ($n = 0$) and piecewise linear ($n = 1$) representations to the other extreme which corresponds to a band limited signal model ($n \rightarrow +\infty$).
- The conventional sampling procedure can be easily modified to obtain a spline representation of an analog signal. This essentially amounts to replacing Shannon's ideal low pass filter by another optimal pre-filter specified by the representation. In principle, there is no compelling reason other than historical for preferring the band limited model and its corresponding sinc interpolator to other ones.
- Spline techniques are also available for non-uniformly spaced data. The price to pay, however, is that one loses the convenient shift-invariant structure (filters) that was emphasized in this paper.

13.1.6 Bezier Curves and Surfaces

Bezier spline approximation method was developed by a French engineer, Pierre Bezier. It has got number of properties which are highly useful and convenient for the design of curves and surfaces. They are also easy to implement. For these reasons, Bezier splines are widely available in various CAD systems.

A Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative positions determine the degree of the Bezier polynomial. As with interpolation polynomials, a Bezier curve can be specified with boundary conditions, with a characterizing matrix or with blending functions. For general Bezier curves, the blending function specification is the most convenient. Suppose we are given $n + 1$ control point positions: $P_k(x_k, y_k, z_k)$ with k varying from 0 to n . The coordinate points can be blended to produce the following position vector $P(u)$ which describes the path of an approximating Bezier polynomial function between P_0 and P_n .

$$\sum_{k=0}^n p_k BEZ_{k,n}(u) \quad 1 \leq u \leq 0 \quad (13.25)$$

The Bezier blending functions $BEZ_{k,n}(u)$ are the *Bernstein polynomials*:

$$BEZ_{k,n}(u) = C(n, k)u^k(1-u)^{n-k} \quad \text{where } C(n, k) = \frac{n!}{k!(n-1)!} \quad (13.26)$$

Equivalently, we can define Bezier blending functions with recursive calculation:

$$BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n}(u) \quad n > k \geq 1 \quad (13.27)$$

With $BEZ_{k,k}(u) = u^k$ and $BEZ_{0,k}(u) = (1-u)^k$, we can represent the parametric equations for the individual curve coordinates:

$$x(u) = \sum_{k=0}^n x(k) BEZ_{k,n}(u), \quad y(u) = \sum_{k=0}^n y(k) BEZ_{k,n}(u), \quad z(u) = \sum_{k=0}^n z(k) BEZ_{k,n}(u) \quad (13.28)$$

As a rule, a Bezier curve is a polynomial of degree one less than the number of control points used; three points generate a parabola, four points a cubic curve and so forth. Bezier curves are commonly used in painting and drawing packages, as well as CAD systems, as they are easy to implement and powerful in curve design. The appearance of some of the Bezier curves for various selections of control points is shown in Figure 13.3.

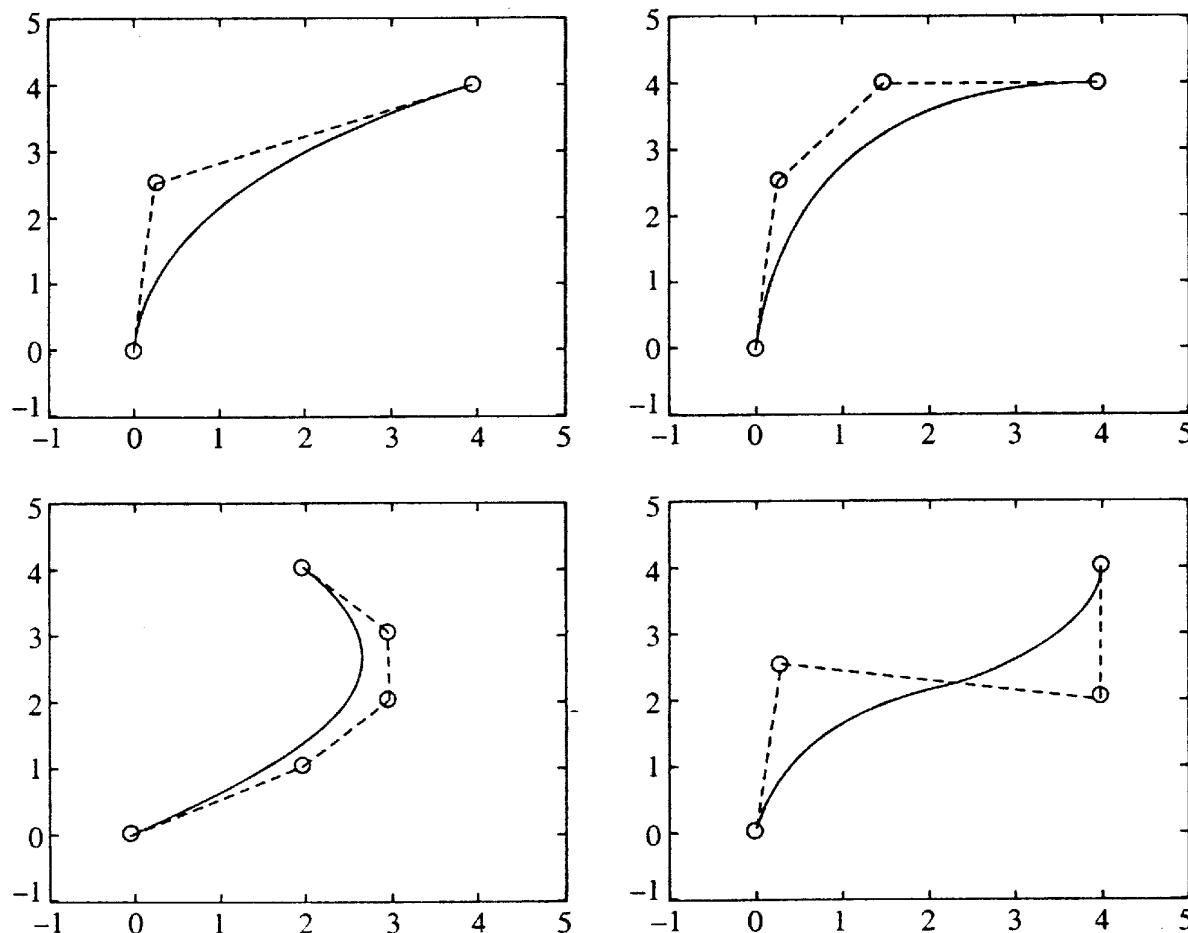


FIGURE 13.3 Examples of 2-D Bezier curves generated from 3, 4 and 5 control points.

Efficient methods for calculating coordinate positions along a Bezier curve can be set up using recursive calculations. For example, successive binomial coefficients can be calculated as:

$$C(n, k) = \frac{n-k+1}{k} C(n, k-1) \quad \text{for } n \geq k \quad (13.29)$$

13.1.7 Properties of Bezier Curves

The Bezier curve always passes through the first and last control points. That is:

$$P(0) = P_0 \quad \text{and} \quad P(1) = P_n \quad (13.30)$$

Values of the first derivatives of a Bezier curve at the endpoints can be calculated from the control point coordinates as:

$$P'(0) = -nP_0 + nP_1 \quad \text{and} \quad P'(1) = -nP_{n-1} + nP_n \quad (13.31)$$

Thus, the slope at the beginning of the curve is along the line joining the first two control points and the slope at the end of the curve is along the line joining the last two endpoints. The parametric second derivatives at the endpoints are calculated as:

$$P''(0) = n(n-1)[(P_2 - P_1)(P_1 - P_0)] \quad \text{and} \quad P''(1) = n(n-1)[P_{n-2} - P_{n-1}] \quad (13.32)$$

Another important property of any Bezier curve is that it lies within the convex hull (convex polygon boundary of the control points). This follows from the property of Bezier blending function that they are all positive and their sum is always 1.

$$\sum_{k=0}^n BEZ_{k,n}(u) = 1 \quad (13.33)$$

When complicated curves are to be generated, they can be formed by piecing several Bezier sections of lower degrees together. Piecing them together gives better control over the shape of the curve in small regions. Since Bezier curves passes through the endpoints, it is easy to match the curve sections (zero-order continuity). Also the Bezier curves have the important property that the tangent to the curve at an endpoint is along the line joining that endpoint to the adjacent control point. Therefore, to obtain first order continuity between curve sections, we can pick control points P'_0 and P'_1 of the new section to be along the same straight line as control points P_{n-1} and P_n (Figure 13.4). When the two curve sections have the same number of control points we have C^1 continuity by choosing the first control point of the new section as the last control point of the previous section and by positioning the second control point of the new section at position $P_n + (P_n - P_{n-1})$. Thus, the three control points are collinear and equally spaced. The C^2 continuity between two Bezier sections can be obtained by calculating the position of third control point of the new section in terms of the last three control points of the previous section as $P_{n-2} + 4(P_n - P_{n-1})$. Requiring second order continuity can be unnecessarily restrictive. This is especially true with cubic curves which has only four control points. The second order continuity fixes three control points leaving only one point that we can use to adjust the shape of the curve.

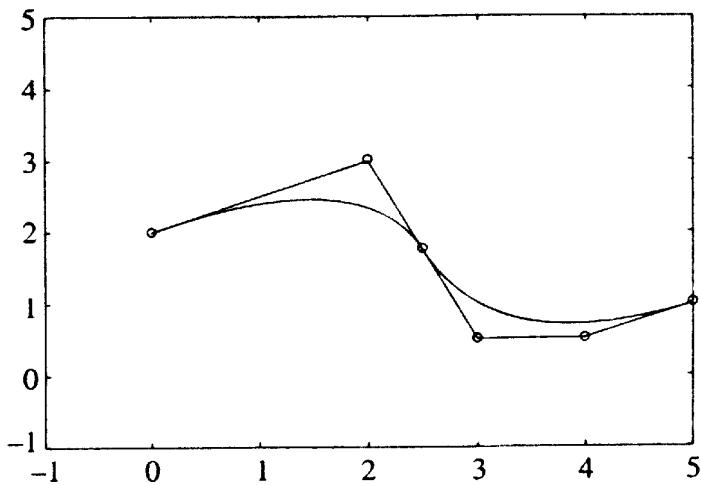


FIGURE 13.4 Piecewise approximation curve formed with two Bezier sections. C^0 and C^1 continuity is attained by setting end point of first curve and starting point of second curve equal and by making joining point and the two points on either side collinear.

13.1.8 Quadratic and Cubic Bezier Curves

Quadratic Bezier curves are generated with three control points. The three blending functions for the quadratic Bezier curve are obtained by substituting $n = 2$ in the equation for Bezier curves.

$$BEZ_{0,2}(u) = (1-u)^2; \quad BEZ_{1,2}(u) = 2u(1-u); \quad BEZ_{2,2}(u) = u^2 \quad (13.34)$$

The plots of the three Bezier blending functions are shown in Figure 13.5. The form of the blending function determines how control points influence the shape of the curves for values of parameter over the range from 0 to 1. At $u = 0$, the only non-zero blending function is $BEZ_{0,2}$, which has the value 1. At $u = 1$, the only non-zero function is $BEZ_{2,2}$ which has value 1 at that point. Thus, cubic Bezier curve will always pass through the control points P_0 , and P_2 . The other function, $BEZ_{1,2}$ influences the shape function of the curve at intermediate values of the parameter u . Blending function $BEZ_{1,2}$ is maximum at $u = 1/2$. The expressions for parametric derivatives can be used to construct piecewise curves:

$$P(u) = [u^2 \quad u \quad 1] M_{BEZ} \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \end{Bmatrix}$$

where

$$M_{BEZ} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad (13.35)$$

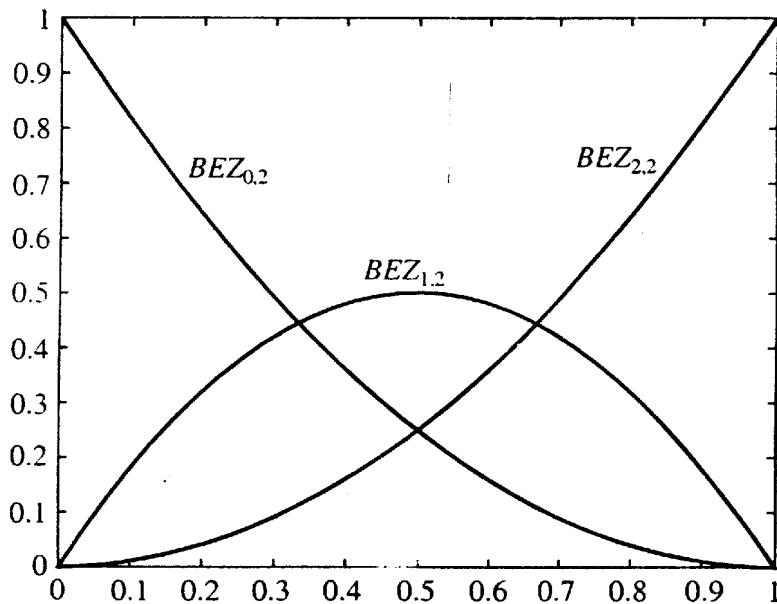


FIGURE 13.5 The three Bezier blending functions for a quadratic curve ($n = 2$).

Cubic Bezier curves are generated with four control points. The four blending functions for the cubic Bezier curves are obtained by substituting $n = 3$ in the equation for Bezier curves.

$$BEZ_{0,3}(u) = (1-u)^3; \quad BEZ_{1,3}(u) = 3u(1-u^2); \quad BEZ_{2,3}(u) = 3u^2(1-u); \quad BEZ_{3,3}(u) = u^3 \quad (13.36)$$

The plots of four cubic Bezier blending functions are given in Figure 13.6. The form of the blending function determines how control points influence the shape of the curves for values

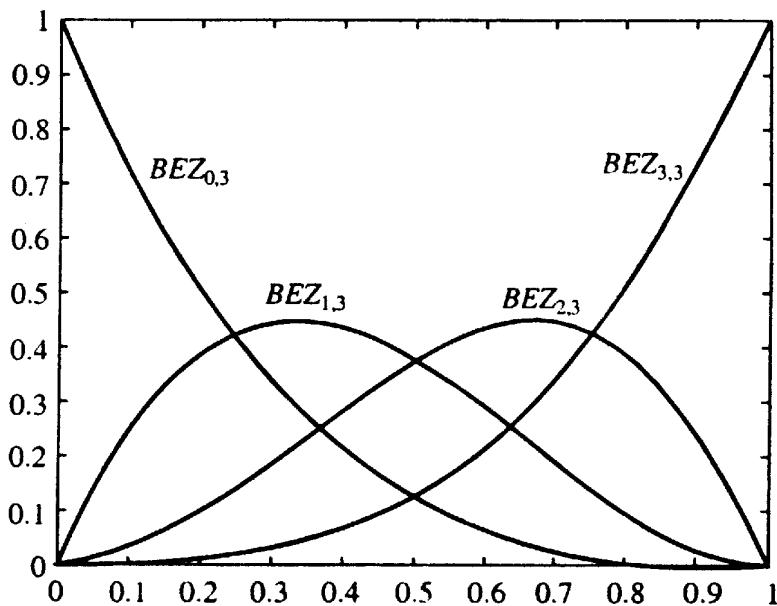


FIGURE 13.6 The four Bezier blending functions for cubic curves ($n = 3$).

of parameter over the range from 0 to 1. At $u = 0$, the only non-zero blending function is $BEZ_{0,3}$ which has the value 1. At $u = 1$, the only non-zero function is $BEZ_{3,3}$ which has value 1 at that point. Thus, cubic Bezier curve will always pass through the control points P_0 , and P_3 . The other functions, $BEZ_{1,3}$ and $BEZ_{2,3}$ influence the shape function of the curve at intermediate values of the parameter u so that the resulting curve tends towards P_1 and P_2 . Blending function $BEZ_{1,3}$ is maximum at $u = 1/3$ and $BEZ_{2,3}$ is maximum at $u = 2/3$. Each of the four blending functions is non-zero over the entire range of parameter u . Thus, Bezier curves do not allow for local control of the curve shape. If we try to reposition any control point, the entire curve shape will be affected.

These expressions for parametric derivatives can be used to construct piecewise curves. For four points degree 3,

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] M_{BEZ} \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix}$$

where

$$M_{BEZ} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (13.37)$$

with C^1 and C^2 continuity between sections. By expanding the polynomial expressions for the blending functions, we can write the cubic Bezier point function in the matrix form.

13.1.9 Parametric Cubic Surfaces

Parametric cubic surfaces are formed by parametric cubics or Hermite as boundary curves and the interior is defined by blending functions. Figure 13.7 shows one such patch. Each boundary curve is represented by its endpoints and end tangent vectors and is given by the equation:

$$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{Bmatrix} = [t][M]_H [G]_H \quad (13.38)$$

A similar equation is derived for the parameter s , to define the approximate boundary curves in that direction. The matrix M is used in both parametric directions, s and t , and the values of P vary according to the curve's position. For the same boundary curves, various surfaces can be obtained by varying the interior shape. To totally identify the surface, the interior shape in the vicinity of each corner is controlled by the cross derivative at the corner, also known as **twist vector**.

Since basis functions of two variables, s and t , are needed for parametric cubic surface representation, the surface basis is defined by the product of single variable basis. The surface or tensor product is known as **Coons bicubic surface** after the inventor and is represented by

$$P(s, t) = [s][M]_H [G]_H [M]^T_H [t]^T$$

where the matrix $[G]_H$ is given by

$$[G]_H = \begin{bmatrix} P(0,0) & P(0,1) & P_t(0,0) & P_t(0,1) \\ P(1,0) & P(1,1) & P_t(1,0) & P_t(1,1) \\ P_s(0,0) & P_s(0,1) & P_{st}(0,0) & P_{st}(0,1) \\ P_s(1,0) & P_s(1,1) & P_{st}(1,0) & P_{st}(1,1) \end{bmatrix}$$

$$= \left[\begin{array}{c|c} \text{Position of corner points} & \text{Derivatives with respect to } t \text{ at corner points} \\ \hline \text{Derivatives with respect to } s \text{ at corner points} & \text{Cross derivative at corner points} \end{array} \right] \quad (13.39)$$

Variations on the surface are obtained by changing the corner points, tangent derivatives or twist vectors. Parametric cubic surface can be simplified by setting all the twist vectors to zero. This is called **Ferguson** or **F-patch**. They are not commonly used in practice because they force the surface to flatten at the corners. For most engineering applications, non-zero twist vectors must be used. Parametric cubic surface patches can be pieced together with edges having C^1 continuity.

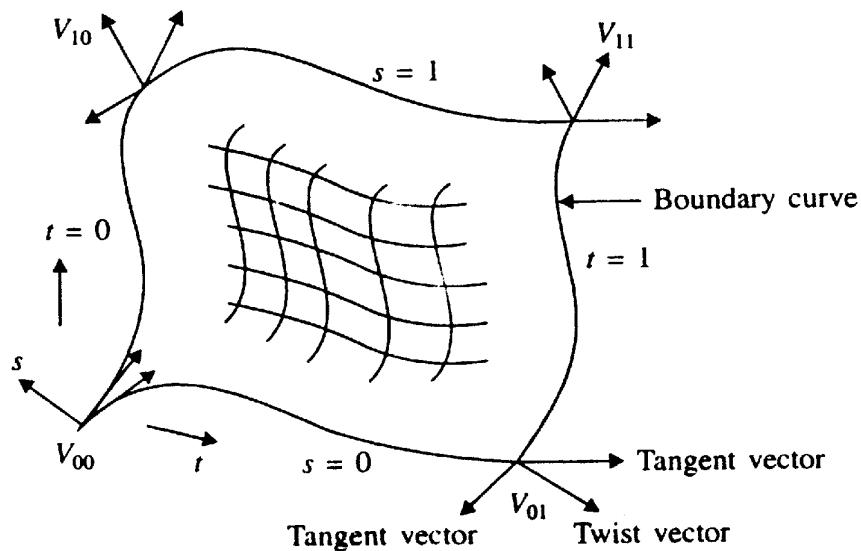


FIGURE 13.7 Parametric cubic surface and its defining parameters.

Normal to a surface is important in geometric modelling applications which is found by the cross product of the parametric derivatives at the point in question so that $n = P'_s \times P'_t$.

13.1.10 Bezier Surfaces

Two sets of orthogonal Bezier curves can be used to design an object surface by specifying by an input mesh of control points. The parametric vector functions for the Bezier surface is formed as the Cartesian product of Bezier blending functions:

$$\sum_{j=0}^n \sum_{k=0}^m P_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u) \quad (13.40)$$

with $P_{j,k}$ specifying the location of $(m + 1)$ by $(n + 1)$ control points. $BEZ_{j,m}$ and $BEZ_{k,n}$ are Bernstein blending functions in s and t directions. The degree of the blending functions need not be the same in the two parametric directions. Figure 13.8 shows the Bezier surface plots. Varying v from 0 to 1 with u fixed at one of the values in the unit interval plots each curve of constant u are obtained. Curves of constant v are plotted similarly.

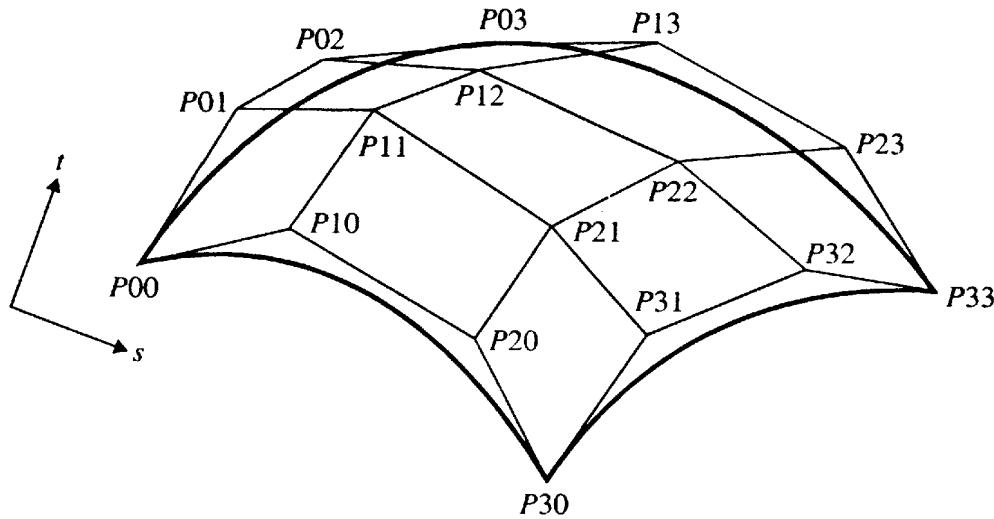


FIGURE 13.8 Bicubic Bezier surface patch.

Bezier surfaces have same properties as Bezier curves, as they provide a convenient method for interactive design applications. The properties of the Bezier surface are controlled by the blending functions. The surface takes the general shape of the control points and is contained within the convex hull of the control points. The corners of the surface and the corner control vertices are coincident. The interior control points $P11, P12, P21, P22$ define the interior shape of the surface. These points can be moved to modify the internal shape of the surface. The other control points form tangents with the four corner points to the four boundary curves $s = 0, s = 1, t = 0$ and $t = 1$. Two Bezier surface patches can be pieced together using boundary constraints. Smooth transition from one section to other is established by establishing both zero-order and first-order continuity at the boundary line. Zero order continuity is obtained

by matching control points at the boundary and first order continuity is obtained by choosing control points along a straight line across the boundary and maintaining a constant ratio of collinear line segments for each set of specified control points across section boundaries.

A Bezier surface can be represented in matrix form as:

$$Q(s, t) = [s][M]_B[V][M]_B^T[t]^T$$

For a bicubic surface this reduces to

$$Q(s, t) = [s^3 \ s^2 \ s \ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P(0,0) & P(0,1) & P(0,2) & P(0,3) \\ P(1,0) & P(1,1) & P(1,2) & P(1,3) \\ P(2,0) & P(2,1) & P(2,2) & P(2,3) \\ P(3,0) & P(3,1) & P(3,2) & P(3,3) \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}^T \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \quad (13.41)$$

To represent bicubic Bezier surface, 16 control points are needed and increase in the number of control points automatically increases the degree of the surface.

13.1.11 B-Spline Curves

B-Splines are the most widely used approximating splines. They have two advantages over Bezier splines:

1. the degree of B-spline polynomial can be set independently of the number of control points.
2. B-splines allow local control over the shape of the spline curve or surface.

But B-splines are more complex than Bezier splines. To calculate the coordinate positions along the B-spline curve, B-spline curves are expressed in a blending function formulation as follows:

$$P(u) = \sum_{k=0}^n P_k B_{k,d}(u) \quad u_{\min} \leq u \leq u_{\max} \quad 2 \leq d \leq n+1 \quad (13.42)$$

where the P_k are input set of $n + 1$ control points. There are several differences between this B-spline formulation and Bezier splines. The range of parameter u depends on how we choose the B-spline parameters and the B-spline blending functions $B_{k,d}$ are polynomials of degree $d - 1$ where the parameter d can be chosen to be any integer value in the range 2 up to the number of control points $n + 1$. Local control of B-spline is achieved by defining the blending functions over subintervals of the total range of u . Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas:

$$\begin{aligned} B_{k+1}(u) &= 1 \quad \text{if } u_k \leq u \leq u_{k+1} \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{13.43}$$

and

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u)$$

Each blending function is defined over d subintervals of the total range of u . The selected set of subinterval points u_j is referred to as *knot vector*. We can choose any value for the subinterval endpoints satisfying the relation $u_j \leq u_{j+1}$. Values for u_{\min} and u_{\max} then depend on the number of control points we select, the value we choose for parameter d and how we set up the subintervals (knot vector). Since it is possible to choose the elements of the knot vector so that the denominators in the previous calculations can have a value of 0. This formulation assumes any terms evaluated as % are to be assigned the value zero.

The properties of B-spline curves are:

- The polynomials curve has degree $d - 1$ and C^{d-2} continuity over the range of u .
- For $n + 1$ control points, the curve is described with $n + 1$ blending functions.
- Each blending function $B_{k,d}$ is defined over d subintervals of the total range of u , starting at knot value u_k .
- The range of parameter u is divided into $n + d$ subintervals by $n + d + 1$ values specified in the knot vector.
- With the knot values labelled $[u_0, u_1, u_2, \dots, u_{n+d}]$, the resulting B-spline curve is defined only in the interval from knot value u_{d-1} up to knot value u_{n+1} .
- Each section of the spline curve (between two successive knot values) is influenced by d control points.
- Any control point can affect the shape of at most d curve section. In addition B-spline curve lies within the convex hull of at most $d + 1$ control points, so that B-splines are tightly bound to the input position. For any value of u in the interval from knot value u_{d-1} to u_{n+1} , sum of all basis functions is 1.

When the spacing between knot values is constant, the resulting curve is called a **uniform B-spline**. For example, we can set uniform knot vectors $\{-1.5 \ -1.0 \ -0.5 \ 0 \ 0.5 \ 1.0 \ 1.5 \ 2.0\}$ or knot vector is normalized between 0 and 1 as $\{0.0 \ 0.2 \ 0.4 \ 0.6 \ 0.8 \ 1.0\}$ or sometimes, it is convenient to have knot vector starting from zero with a separation of 1 as: $\{0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7\}$. Uniform B-splines are uniform periodic functions. For given values of n and d , all blending functions have same slope. Each successive blending function is a shifted version of previous function.

$$B_{k,d}(u) = B_{k+1,d}(u + \Delta u) = B_{k+2,d}(u + 2\Delta u) \tag{13.44}$$

where Δu is the interval between adjacent knot values.

13.1.12 Cubic Periodic B-Splines

Cubic periodic B-splines are commonly used in graphics packages. Periodic splines are particularly useful for generating certain closed curves. For cubics, $d = 4$ and each blending function spans four intervals of the total range of u . If we are to fit a cubic curve we could use the integer knot vector $[0, 1, 2, 3, 4, 5, 6, 7]$ and the recurrence relations to obtain the blending functions normalized to the interval $0 \leq u \leq 1$. The boundary conditions for periodic cubic B-splines with four consecutive control points P_0, P_1, P_2 and P_3 are:

$$\begin{aligned} P(0) &= \frac{1}{6}(P_0 + 4P_1 + P_2) & P(1) &= \frac{1}{6}(P_1 + 4P_2 + P_3) \\ P'(0) &= \frac{1}{2}(P_2 - P_0) & P'(1) &= \frac{1}{2}(P_3 - P_1) \end{aligned} \quad (13.45)$$

The boundary conditions are defined with four control points and parametric derivatives (slopes) defined at the beginning and end of each curve section are parallel to the chords adjoining the adjacent control points. The B-spline curve starts at the position near P_1 and ends at position near P_2 . A matrix formulation for a cubic periodic B-spline with four control points can be written as:

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] M_B \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{Bmatrix}$$

where

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (13.46)$$

We can modify B-spline equation, by introducing the tension parameter as in cardinal splines, as:

$$M_{Bt} = \begin{bmatrix} -t & 12-9t & 9t-12 & t \\ 3t & 12t-18 & 18-15t & 0 \\ -3t & 0 & 3t & 0 \\ t & 6-2t & t & 0 \end{bmatrix} \quad (13.47)$$

which reduces to M_B when $t = 1$.

The matrix representation can be expanded to polynomial form. For example, for tension value $t = 1$, we have

$$\begin{aligned} B_{0,3} &= \frac{1}{6}(1-u)^3 \quad 0 \leq u \leq 1 & B_{2,3} &= \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1) \\ B_{1,3} &= \frac{1}{6}(3u^3 - 6u^2 + 4) & B_{3,3} &= \frac{1}{6}u^3 \end{aligned} \quad (13.48)$$

Figure 13.9 shows the plots of the four basis functions of the cubic periodic B-splines.

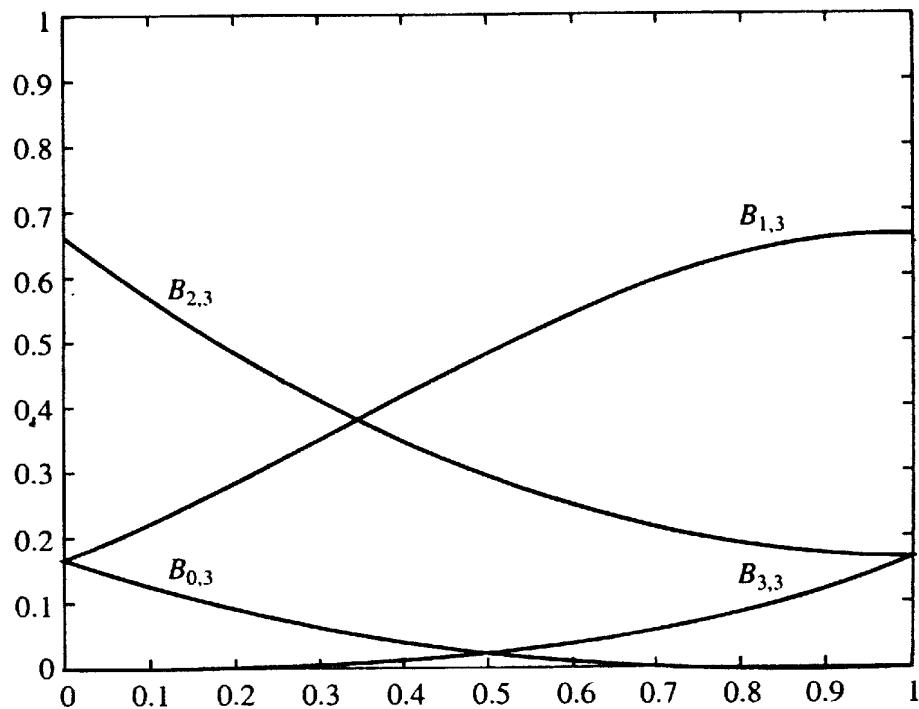


FIGURE 13.9 Basis functions for cubic periodic B-splines.

The uniform quadratic B-splines can be formulated like a cubic. Its representation in matrix form is:

$$P(u) = [u^2 \quad u \quad 1] M_B \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \end{Bmatrix}$$

where

$$M_B = \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \quad (13.49)$$

13.1.13 Conversion between Hermite, Bezier and B-Spline Representations

A free-form cubic curve is represented by the equation

$$x = [t^3 \quad t^2 \quad t \quad 1][M][V]$$

where $[V]$ is a matrix of control points and $[M]$ is the basis matrix. To convert one type of cubic curve to another the following expressions are used:

$$x = [t^3 \ t^2 \ t \ 1][M]_{\text{from}}[V]_{\text{from}} = [t^3 \ t^2 \ t \ 1][M]_{\text{to}}[V]_{\text{to}} \quad (13.50)$$

which yields

$$[M]_{\text{from}}[V]_{\text{from}} = [M]_{\text{to}}[V]_{\text{to}} \quad \text{and} \quad [V]_{\text{to}} = [M]_{\text{to}}^{-1}[M]_{\text{from}}[V]_{\text{from}} \quad (13.51)$$

Table 13.4 shows the conversion matrices for various free-form cubic curves.

TABLE 13.4 Conversion Matrices for Various Free-form Cubic Curves

<i>From →</i>	<i>Hermite</i>	<i>Bezier</i>	<i>B-spline</i>
<i>To ↓</i>			
Hermite	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}$	$\frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ -3 & 0 & 3 & 0 \\ 0 & -3 & 0 & 3 \end{bmatrix}$
Bezier	$\frac{1}{6} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 0 & 3 & 0 & -1 \\ 0 & 3 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}$
B-spline	$\frac{1}{6} \begin{bmatrix} -3 & 6 & -7 & -2 \\ 6 & -3 & 2 & 1 \\ -3 & 6 & -1 & -2 \\ 6 & -3 & 2 & 7 \end{bmatrix}$	$\begin{bmatrix} 6 & -7 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -7 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

13.1.14 Non-Uniform B-Splines

With non-uniform B-splines, we can choose multiple internal knot values and unequal spacing between the knot values. Some examples are: {0 1 2 3 3 4}, {0 2 2 3 3 3 6}, {0 0 0 1 1 3 3 3}, {0 0.2 0.6 0.9 1.0}. Non-uniform B-splines provide increased flexibility in controlling a curved shape. With unequally spaced intervals in the knot vector, we obtain different shapes for the blending functions in different intervals, which can be used to adjust the spline shapes. By increasing node multiplicity, we produce subtle variations in curve shape and even introduce discontinuities. Multiple knot values also reduce the continuity by 1 for each repeat of a particular value. Given a set of $n + 1$ control points, we set the degree of the polynomial and select the knot values. Then using the recurrence relations, we could either obtain the set of blending functions or evaluate the curve positions directly for the display of the curve. Graphical packages restrict the knot intervals to be either 0 or 1 to reduce

computation. A set of characteristic matrices can be stored and used to compute values along the spline curve without evaluating the recurrence relations for each curve points to be plotted. Figures 13.10 to 13.12 show the uniform and non-uniform B-spline plots.

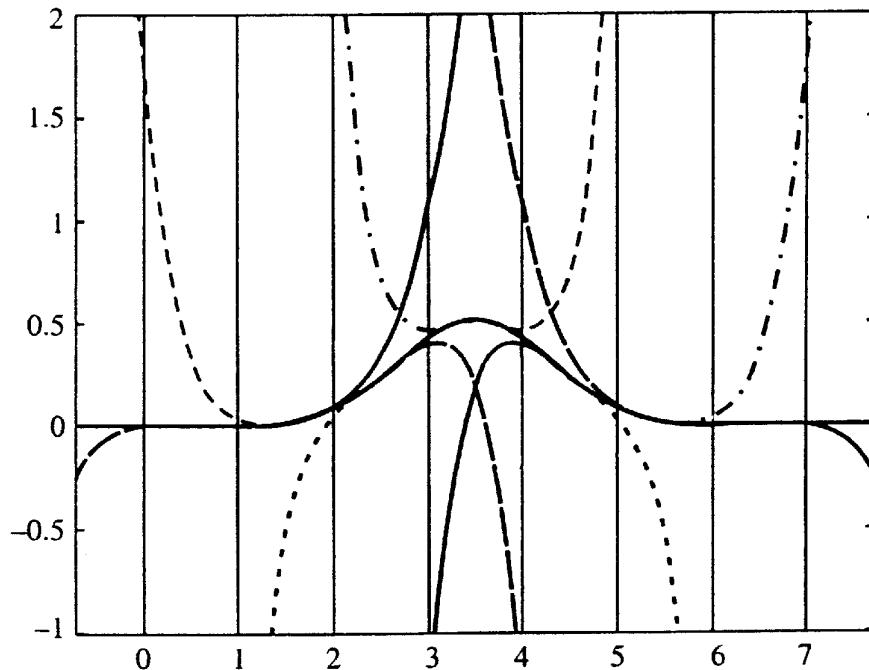


FIGURE 13.10 Uniform B-spline with knot $\{0, 1, 2, 3, 4, 5, 6, 7\}$.

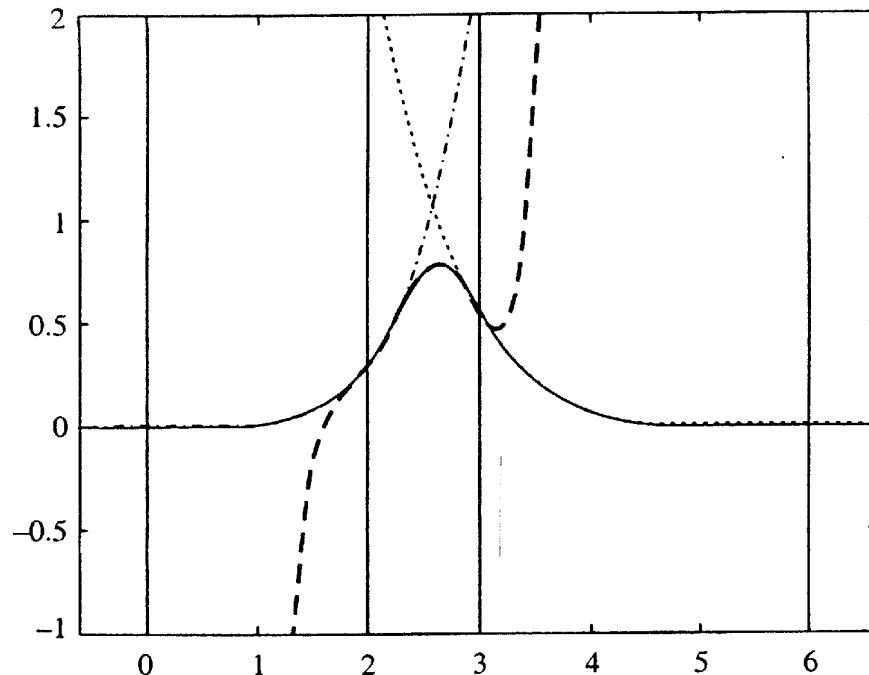


FIGURE 13.11 Non-uniform B-spline with multiple knots $\{0, 2, 2, 3, 3, 3, 6\}$.

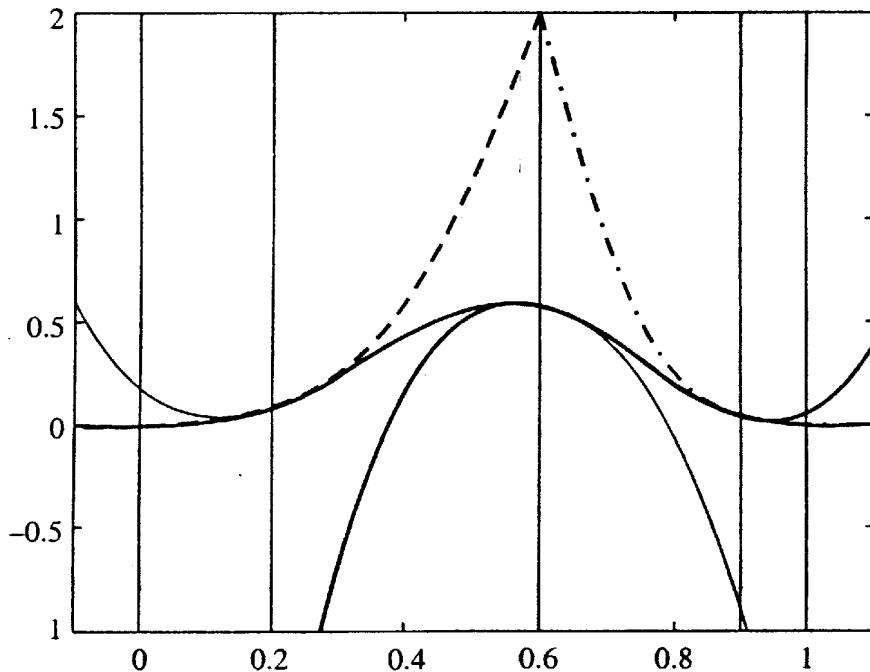


FIGURE 13.12 Non-uniform B-spline with knots $\{0, 0.2, 0.6, 0.9, 1.0\}$.

13.1.15 Relation between Spline, Bezier and B-Spline Curves

Bezier curves are special type of B-spline curves. Any system that contains B-splines in their full generality, allowing for multiple knots, is capable of representing a Bezier curve or for that matter, is a piecewise Bezier curve. A curve may be stored as an array holding B-spline control vertices and for evaluation purposes, may be converted to piecewise Bezier form.

B-splines form basis for all splines so that any spline curve can be written as a B-spline curve. In the design of curves, we can pass an interpolating curve through all data points or we can design a curve by interactively manipulating a B-spline polygon. A flexible system can have both interpolation and interactive manipulation. The curve can be generated using interpolation and subsequent modification can be done through interactive manipulation of control polygon. Every C^2 B-spline curve may be generated as an interpolating spline curve by entering the junction points, end tangent vectors and knot sequence into a C^2 cubic spline interpolator.

Cubic B-spline curves are more stable than curves in the piecewise Hermite form. This is because some of the Hermite basis functions are negative giving rise to numerically unstable non-convex combinations. One advantage of using Hermite form is that it stores interpolation points explicitly; they must be computed in the B-spline form. Another advantage is that the end conditions for C^2 spline interpolation are more easily formulated in the Hermite form than in the B-spline form. A significant argument against Hermite form is its lack of invariance under affine parameter transformation. An advantage of B-spline form is the storage. For B-spline curves, one needs a control point array of $L + 2$ plus a knot vector array of length $L + 1$ for a curve with L data points, resulting in overall storage requirement of $4L + 7$ reals. For a piecewise Hermite form, one needs a data array of length $2L$ (position and tangent at each data

point) plus a knot vector of $L + 1$ resulting in a storage requirement of $7L + 1$. For surfaces, the discrepancy is even larger: $3(L + 2)^2 + 2(L + 1)$ vs $12L^2 + 2(L + 1)$ reals (for Hermite, we need both u and v tangents and twist for each data point). When both forms are used for tensor product interpolation, Hermite form must solve three sets of linear equations while B-spline form must solve only two sets.

13.1.16 B-Spline Surfaces

B-spline surfaces are obtained as the cartesian product of B-spline blending functions in the form:

$$P(u, v) = \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} P_{k_1, k_2} B_{k_1, d_1}(u) B_{k_2, d_2}(v) \quad (13.52)$$

where the vector values for P_{k_1, k_2} specify positions of $n_1 + 1$ by $n_2 + 1$ control points. $B_{k_1, d_1}(u)$ and $B_{k_2, d_2}(v)$ are B-spline blending functions. The knot vectors in the two directions of parameterization can be classified as periodic/uniform, non-periodic, and non-uniform like B-spline curves. The surface lies within the defining polyhedron formed by the control points.

13.1.17 Beta-Splines and Rational Splines

Beta splines are generalization of B-splines, also referred as β splines, that are formulated by imposing geometric boundary continuity conditions on the first and second parametric derivatives.

The continuity parameters for beta splines are called **β parameters**. A rational function is simply the ratio of two polynomials. A rational B-spline is the ratio of two spline functions. A rational B-spline can be described with the position vector:

$$P(u) = \frac{\sum_{k=0}^n \omega_k P_k B_{k,d}(u)}{\sum_{k=0}^n \omega_k B_{k,d}(u)} \quad (13.53)$$

where P_k is a set of $n + 1$ control point positions and parameters, ω_k are weight factors for the control point. The greater the value of a particular ω_k , the closer the curve is pulled toward the control point P_k weighted by the parameter. When all the weight factors are set to the value 1 we have the standard B-spline curve since the denominator is 1. Rational B-splines are exact representation for quadric curves (conics) such as circles and ellipses. Non-rational splines, which are polynomials, can only approximate conics. All curve shapes can be modelled with rational splines without a separate library of curve functions to handle different design shapes. Another advantage of rational spline is that they are invariant with respect to the respective viewing transformation. This means that we can apply a perspective viewing transformation to the control points of the rational curve and we will obtain the correct view of the curve. Non-rational B-splines are not invariant with respect to a perspective viewing transformation.

Graphic design packages use non-uniform knot vector representations for constructing rational B-splines. These splines are referred to as NURBs (non-uniform rational B-splines). To plot a conic section with NURBs, we use a quadratic spline function ($d = 3$) and three control points.

$$\omega_0 = \omega_1 = 1, \omega_2 = \frac{r}{1-r} \quad \text{and} \quad P(u) = \frac{P_0 B_{0,3} + \left[\frac{r}{1-r} \right] P_1 B_{1,3} + P_2 B_{2,3}}{B_{0,3} + \left[\frac{r}{1-r} \right] B_{1,3} + B_{2,3}} \quad (13.54)$$

We can do this with the B-spline function with open knot vector $\{0\ 0\ 0\ 1\ 1\ 1\}$, which is a quadratic Bezier spline. We can set the weighing parameters as:

$$\begin{array}{ll} r > 1/2 \text{ and } \omega > 1 & \text{(hyperbola)} \\ r = 1/2 \text{ and } \omega = 1 & \text{(parabola section)} \\ r < 1/2 \text{ and } \omega < 1 & \text{(ellipse section)} \\ r = 0 \text{ and } \omega = 0 & \text{(straight line segment)} \end{array}$$

13.2 MULTIRESOLUTION METHODS AND WAVELET ANALYSIS

The simplest way to represent any information is to represent the data as time series (t_i, y_i) sequence of points. Each point provides complete information about the behaviour of the series at time t_i but absolutely no information about the behaviour of the series elsewhere. Many applications require analysis of the series at a broader scale. Fourier analysis is one of the classical tools for addressing the problems, which can be used to convert data into a form that is useful for analyzing frequencies. Fourier coefficients contain complete information about the behaviour of the series at one frequency but no information about its behaviour at other frequencies. Fourier techniques are also difficult to adapt to many situations of practical importance. For instance, most of the time series encountered in practice are finite and aperiodic. But the discrete Fourier transform can be applied only to periodic functions.

Multiresolution method is used to represent function with a collection of coefficients, each of which provides a limited information about both position and frequency of the function. The theory of wavelets provides an extremely useful mathematical toolkit for hierarchically decomposing function efficiently and correctly. A wavelet representation of functions consists of overall approximation coefficients and detail coefficients that influence the function at various scales. Unlike Fourier functions, wavelets can be adapted to represent a wide variety of functions, including functions with discontinuities, functions defined on bounded domains and functions defined on arbitrary topological type. Consequently, wavelets are equally well suited to problems involving images, open or closed curves and surfaces of any variety. For functions which are typically encountered, many of coefficients in a wavelet representation are either zero or negligibly small. This property offers the opportunity to compress data and to accelerate the convergence of iterative solution technique. Again transforming to and from a wavelet representation can generally be accomplished in linear time, allowing for very fast algorithms.

The starting point for multiresolution analysis is a nested set of linear function spaces $v^0 \subset v^1 \subset v^2, \dots$, with the resolution of functions in v^j increasing with j . The basis function for v^j are called **scaling functions**. The next step in multiresolution analysis is to define the wavelet

spaces, denoted by w^j . Each wavelet space w^j is required to be the complement of v^j in v^{j+1} . Thus, any function in v^{j+1} can be written as sum of unique function in v^j and a unique function in w^j . The functions we choose as basis for w^j are called **wavelets**. The wavelets in w^j are not required to be orthogonal to the scaling function in v^j .

The different scaling functions and wavelets for a given level j can be represented as row matrices. The dimension v^j is denoted by $v(j)$ and the dimension of w^j is denoted by $w(j)$ because w^j is the complement of v^j in v^{j+1} . The dimensions of these spaces should satisfy $v(j+1) = w(j) + v(j)$.

$$\phi^j(x) = [\phi_0^j(x) \dots \phi_{v(j)-1}^j(x)] \quad \Psi^j(x) = [\Psi_{v(j)}^j(x) \dots \Psi_{v(j)-1}^j(x)] \quad (13.55)$$

Having nested subspaces v^j is equivalent to having scaling functions that are refinable. That is, for all $j = 1, 2, \dots$, there must exist a matrix of constraints ϕ such that

$$\phi^{j-1}(x) = \phi^j(x)P^j \quad (13.56)$$

v^j and v^{j-1} have dimensions $v(j)$ $v(j-1)$, respectively. P^j is a $v(j)$ by $v(j-1)$ matrix.

Since the wavelet space w^{j-1} is by definition also a subspace of v^j , we can write the wavelets $\Psi^{j-1}(x)$ as linear combinations of the scaling functions $\phi^j(x)$. Therefore, there is a $v(j)$ by $v(j-1)$ matrix of constants Q^j satisfying

$$\Psi^{j-1}(x) = \phi^j(x)Q^j \quad (13.57)$$

In the Haar basis, at a particular level j , there are $v(j) = 2^j$ scaling functions and $w(j) = 2^j$ wavelets. Thus, there must be refinement matrices describing how the two scaling functions in v^1 and the two wavelets in w^1 can be made from the four scaling functions in v^2 .

$$P^2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad Q^2 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \quad (13.58)$$

In the case of the wavelets constructed on unbounded real line, all the wavelets have the same shape; in fact, they are just shifted and scaled versions of a single wavelet called the **mother wavelet**. In general,

$$\Psi_i^j(x) = \Psi(2^j x - i) \quad (13.59)$$

The Eqs. (13.56) and (13.57) can be written as:

$$\phi(x) = \sum p_i \phi(2x - i) \quad \Psi(x) = \sum q_i \phi(2x - i) \quad (13.60)$$

Using the block matrix notation,

$$[\phi^{j-1} | \Psi^{j-1}] = \phi^j [P^j | Q^j] \quad (13.61)$$

Substituting the matrices from the previous example into the equation along with the appropriate basis function yields

$$[\phi_0^1 \quad \phi_1^1 \quad \phi_0^1 \quad \phi_1^1] = [\phi_0^2 \quad \phi_1^2 \quad \phi_2^2 \quad \phi_3^2] \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad (13.62)$$

13.3 THE FILTER BANK

Consider a function in some approximation space v^j . The coefficients in terms of the scaling function basis can be written as a column matrix of values:

$$c^{j-1} = [c_0^j, \dots, c_{v(j)-1}^j]^T \quad (13.63)$$

Suppose we wish to create a lower resolution c^{j-1} of c^j with a smaller number of coefficients $v(j-1)$. This can be done by linear filtering or down sampling on the $v(j)$ entries of c^j . This can be expressed as a matrix equation:

$$c^{j-1} = A^j c^j \quad (13.64)$$

where A^j is a constant $v(j-1)$ by $v(j)$ matrix.

Since c^{j-1} contains fewer entries than c^j , it is clear that some details are lost in this process. The lost details d^{j-1} can be computed by $d^{j-1} = B^j c^j$ where B^j is a constant $w(j-1)$ by $w(j)$ matrix related to A^j . The pair of matrices A^j and B^j is called **analysis filters**. The process of splitting c^j into c^{j-1} and d^{j-1} is called **analysis or decomposition**. The original coefficients c^j can be recovered from c^{j-1} and d^{j-1} by using the matrices, P^j and Q^j . Recovering c^j from c^{j-1} and d^{j-1} is called **synthesis or reconstruction**. P^j and Q^j are called **synthesis filters**.

$$c^j = P^j c^{j-1} + Q^j d^{j-1} \quad (13.65)$$

The analysis or decomposition equations and the reconstruction equations appear as:

$$c_k^{j-1} = \sum_l a_{l-2k} c_l^j \quad d_k^{j-1} = \sum_l b_{l-2k} c_l^j \quad c_k^j = \sum_l (p_{k-2l} c_l^{j-1} + q_{k-2l} d_l^{j-1}) \quad (13.66)$$

The procedure for splitting c^j into a low-resolution part c^{j-1} and a detail part d^{j-1} can be recursively applied to the lower resolution part. The original coefficients can be expressed as a hierarchy of lower resolution parts c^0, \dots, c^j and details d^0, \dots, d^j and this recursive process is called **filter bank** (Figure 13.13).

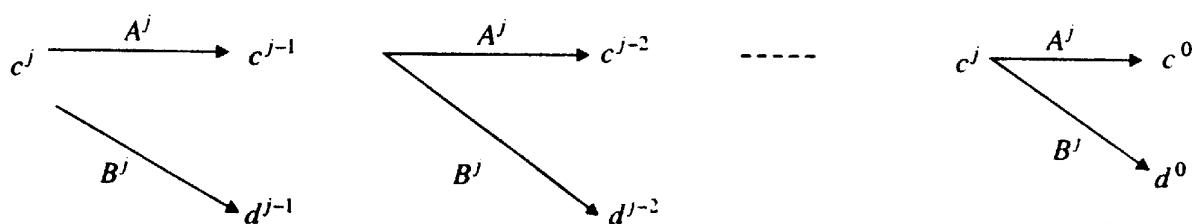


FIGURE 13.13 Filter bank.

A^j and B^j are formed by the matrices satisfying the relation:

$$[\phi^{j-1} | \Psi^{j-1}] \begin{bmatrix} A^j \\ B^j \end{bmatrix} = \phi^j \quad (13.67)$$

$[P^j | Q^j]$ and $[A^j/B^j]$ are square matrices. From the above equations, we get

$$\begin{bmatrix} A^j \\ B^j \end{bmatrix} = [P^j | Q^j]^{-1} \quad (13.68)$$

The matrices $[A_j/B_j]$ and $[P^j | Q^j]^{-1}$ must at least be invertible.

13.4 ORTHOGONAL AND SEMI-ORTHOGONAL WAVELETS

An orthogonal basis is one in which each basis function is orthogonal to every other basis function. An orthogonal multiresolution basis is one in which the scaling functions are orthogonal to one another, the wavelets are orthogonal to one another and each of the wavelets is orthogonal to every coarser scaling function. Wavelets that satisfy these conditions are called **orthogonal wavelets**. In mathematical notation, the conditions defining an orthogonal wavelet basis can be written as:

$$\left. \begin{array}{l} \langle \phi_k^j | \Psi_l^j \rangle = \delta_{k,l} \\ \langle \Psi_k^j | \Psi_l^j \rangle = \delta_{k,l} \\ \langle \phi_k^j | \Psi_l^j \rangle = 0 \end{array} \right\} \text{for all } j, k \text{ and } l \quad (13.69)$$

These conditions define an orthonormal wavelet basis, since the inner product of a basis function with itself is constrained to be 1. An orthogonal basis need not have this normalization but it will be assumed that orthogonal wavelet basis is also orthonormal. If I denotes the identity matrix and 0 denotes the zero matrix, the orthogonality conditions are:

$$\begin{aligned} \langle \phi^j | \phi^j \rangle &= I & \left[\left[\langle \phi^{j-1} | \Psi^{j-1} \rangle \middle| \langle \phi^{j-1} | \Psi^{j-1} \rangle \right] \right] &= I \\ \langle \Psi^j | \Psi^j \rangle &= I & \left[\left[\langle \phi^j | P^j | Q^j \rangle \middle| \phi^j | P^j | Q^j \rangle \right] \right] &= I \\ \langle \phi^j | \Psi^j \rangle &= 0 & \left[P^j | Q^j \right]^T \left[\phi^j | \phi^j \right] \left[P^j | Q^j \right] &= I \end{aligned} \quad (13.70)$$

Finally, we arrive at the result that

$$[P^j | Q^j]^T = [P^j | Q^j]^{-1} \quad (13.71)$$

That is, the inverse of the combined scaling function and wavelet synthesis matrix is the same as its transpose; a matrix with this property is called an **orthogonal matrix**.

Other than orthogonality, other desirable properties of wavelet basis for a particular application are:

- *Compact support.* The supports of scaling functions and wavelets are related to the spread of non-zero entries in P^j and Q^j . A more compact support improves efficiency of decomposition and reconstruction.
- *Smoothness of functions.* Best represented by smooth bases.
- *Vanishing moments.* A wavelet $\Psi(x)$ is said to have n vanishing moments if $\int \Psi(x)x^k dx$ is identically zero for $k = 0, \dots, n - 1$ but not for $k = n$. Wavelet with more vanishing points are desirable for applications that require numerical approximations of smooth operators.

If we desire smooth symmetric wavelets with compact supports, we must be willing to sacrifice orthogonality. We can construct a multiresolution basis in which the wavelets of a given resolution are at least orthogonal to all coarser scaling functions, though not necessarily orthogonal to each other. Wavelets designed to meet this criterion are referred to as semiorthogonal wavelets. In mathematical form, a semiorthogonal wavelet basis satisfies the condition:

$$\langle \phi_k^j | \Psi_l^j \rangle = 0 \quad \text{for all } j, k, l \quad (13.72)$$

Orthogonal wavelets are a special case of semiorthogonal wavelets, in which both the scaling functions are orthogonal to one another and the wavelets are orthogonal to one another.

13.5 SPLINE WAVELETS

Spline wavelets are class of wavelets with semiorthogonal basis, which is particularly useful for multiresolution curves. These wavelets are built from B-splines and have been developed by Chui and his colleagues. We choose here the endpoint interpolating B-splines, defined on the unit interval. If m is the degree of the basis function, we can define $v^j(m)$ to be a set of $2^j + m$ non-uniform B-splines that are constructed from the knot sequence given below

$$(x_0, \dots, x_{2^j+m}) = \frac{1}{2^j} (0, \underbrace{\dots, 0}_{m+1 \text{ times}}, 1, 2, \dots, 2^j - 1, \underbrace{2^j, \dots, 2^j}_{m+1 \text{ times}}) \quad (13.73)$$

Examples of the spline scaling functions at level $j = 1$, shown in Figure 13.14, for various degrees m , $2^j + m$, non-uniform B-splines that are constructed from the knot sequence.

It is not difficult to show that the spaces $v_0(m), v_1(m), \dots$ are nested as required by multiresolution analysis. The theory of knot insertion for B-splines can be used to develop expression for the entries of the refinement matrix P^j . The columns of P^j are sparse reflecting that B-spline basis functions are locally supported. The first and last m columns of P^j are relatively complicated but remaining columns are shifted versions of column $m + 1$. The entries of the interior columns are up to a common factor of $20m$, given by binomial coefficients. In the case of cubic splines ($m = 3$), the matrix P^j for $j \leq 3$ is:

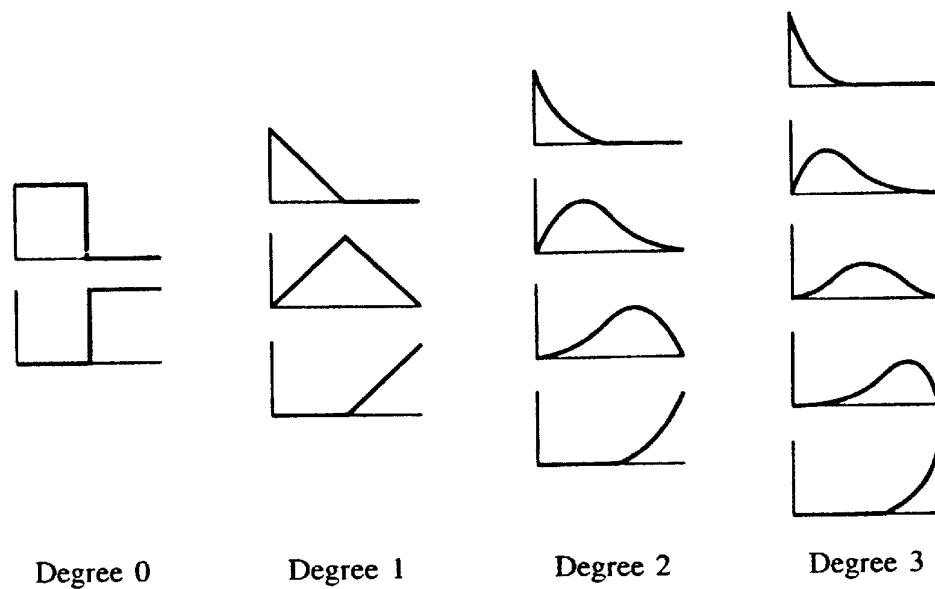


FIGURE 13.14 Endpoint interpolating B-spline scaling functions for $v'(m)$ with various degrees ($m = 0, 1, 2$).

$$P^j = \frac{1}{8} \begin{bmatrix} 8 \\ 4 & 4 \\ 6 & 2 \\ \frac{3}{2} & \frac{11}{2} & 1 \\ 4 & 4 \\ 1 & 6 & 1 \\ 4 & 4 \\ 1 & 6 \\ 4 \\ 1 \\ & & & & & 1 \\ & & & & & 4 \\ & & & & & 6 & 1 \\ & & & & & 4 & 4 \\ & & & & & 1 & \frac{11}{2} & \frac{3}{2} \\ & & & & & 2 & 6 \\ & & & & & 4 & 4 \\ & & & & & 8 \end{bmatrix}$$

The blank entries are taken to be zero and the dots indicate that the previous column is repeated, shifted down by two rows each time. Compared to other well known scaling functions like Daubechies, the error term in the case of splines is very small. It gives better approximation at coarser scale. Another major advantage of splines is that splines do not require equally spaced points.

To complete our development of a semiorthogonal B-spline multiresolution analysis, we need to find basis function for the spaces v^j . Once we have chosen scaling functions and their refinement matrices π_i , the wavelet matrices c^i are somewhat constrained. If the wavelets are to be orthogonal to the scaling functions, the columns of c^i must form a basis for the null space of $M^j = (P^j)^T [\langle \phi^j | \phi^j \rangle]$. Because the dimension of $v^j(m)$ is $2^j + m$, M^j has $2^j + m$ columns and only $2^{j-1} + m$ rows. The dimensions of the null space of M^j is $2^j + m$. Thus, there are $2^j + m$ wavelets in $w^j(m)$, each defined by a column of Q^j . To determine Q_j matrices, we need to decide what properties the wavelets should have. The structure Q_j matrix is similar to π matrices: the first and last m columns are unusual but the remaining interior columns are all shifted copies of single sequence. Figure 13.15 shows some typical endpoint interpolating B-spline wavelets that result from these constructions.

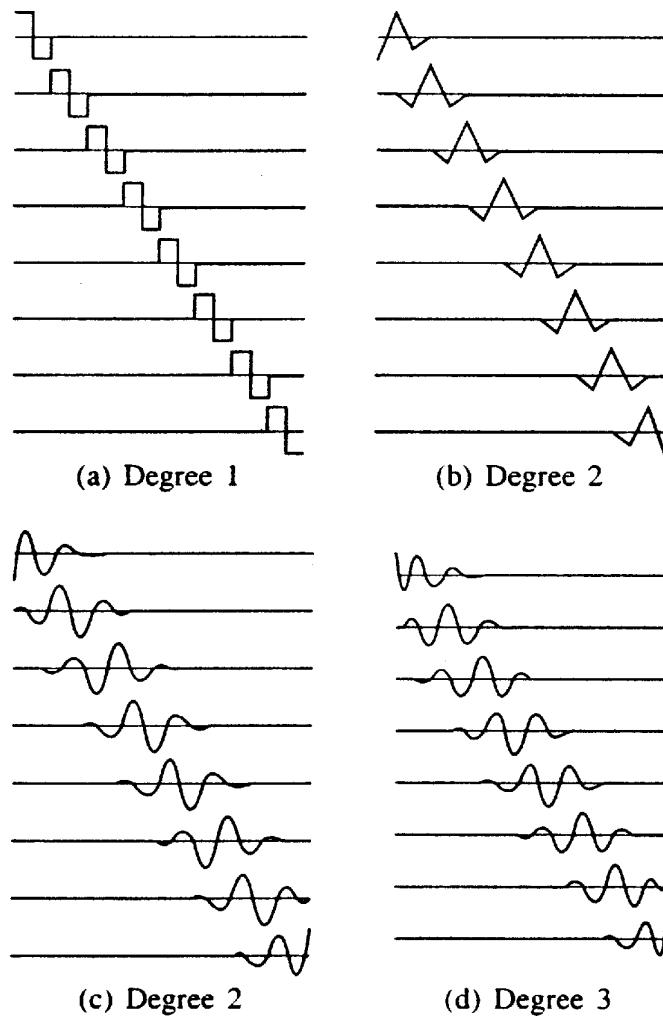


FIGURE 13.15 Endpoint interpolating B-spline wavelets for $w^3(m)$ with $m = 0, 1, 2$ and 3 .

The steps involved in constructing a semiorthogonal wavelet basis and its associated analysis and synthesis matrices are as follows:

- Select the scaling functions $\phi^j(x)$, either explicitly or via subdivision scheme. This choice determines the nested approximation spaces v^j and the synthesis filters P^j .
- Select an inner product defined on the functions in v^0, v^1, \dots . This choice determines the L^2 norm and defines orthogonality. Although the standard inner product is the common choice, a weighted inner product can be used to capture a measure of error that is meaningful in the context of the application at hand.
- Select wavelet synthesis matrices Q^j that satisfy $(P^j)^T[\langle \phi^j | \phi^j \rangle]Q^j = 0$. This choice determines the wavelets $\Psi^j(x)$ that span the spaces w^j . Together, the synthesis filters P^j and Q^j determine the analysis filters A^j and B^j by the equation.

13.6 PROPERTIES OF SPLINE WAVELETS

Ten good properties that make spline wavelets very popular are:

Closed form solution

The B-splines, which have been defined as the $(n + 1)$ -fold convolution of a unit box function, have simple explicit forms in both the time and frequency domain.

Simple manipulation

Splines are piecewise polynomial, which greatly simplifies their manipulation. In particular, it is straightforward to obtain spline derivatives and integrals. Differentiation corresponds to a reduction of the spline degree by one and integration will result in a corresponding increase in the degree. This type of relation may be useful if one uses spline wavelets for solving differential equations.

Symmetry

The B-splines are symmetric. It is therefore easy to construct spline wavelets that are either symmetric or antisymmetric by selecting the sequence with the appropriate symmetry. The advantage is that the corresponding wavelet transform can be implemented using mirror boundary conditions, which reduces boundary artifacts. This is usually not possible for non-linear phase wavelets such as the celebrated Daubechies wavelets.

Shortest and most regular scaling function of order L

B-spline of degree $n = L - 1$ is the shortest possible scaling function of order L . Having shortest functions reduces amount of computation.

Maximum regularity for a given order L

Regularity is the degree of differentiability. B-splines are not the smoothest scaling functions of order L —they are only optimal if we take into account the filter length. For instance, we can consider the refinement filter. B-splines are the smoothest scaling for a refinement filter of a given length.

m-scale relation

The B-splines satisfy a two-scale relation for any integer m . Unlike most other scaling functions, m is not restricted to a power of two. This is advantageous for designing fast non-dyadic wavelet algorithms.

Variational properties

Splines provide a “natural” signal interpolant that is optimal in the sense that it has the least oscillating energy. The spline is the interpolating function that oscillates the least. Cubic spline interpolants exhibit a *minimum curvature property* which justifies the analogy with the draftsman’s spline or French curve. This latter device is a thin elastic beam that is constrained to pass through a given set of points.

Best approximation properties

For a given order L , splines tend to give smaller truncated approximation errors than other standard wavelet families. Splines are shown to be π times better than Daubechies wavelets in some piecewise approximation theoretic sense. Splines are the best for approximating smooth functions. Another interesting consequence is that the asymptotic approximation error will be the same irrespective of the orthogonality properties of the transform.

Optimal time-frequency localization

B-spline wavelets have been shown to converge to modulated Gaussians. Therefore, spline wavelet bases that can be optimally localized in time and frequency. We can get as close as we wish to the time-frequency localization limit specified by the uncertainty principle. This makes them best choice for time-frequency analysis.

Convergence to the ideal lowpass filter

Splines provide a convenient framework that allows for a progressive transition between the two extreme cases of a multiresolution: the piecewise constant model (with $n = 0$) and the band limited model corresponding to a spline of degree infinite. The Battle-Lemarié scaling functions converge to $\text{sinc}(x)$ —the impulse response of the ideal lowpass filters—as the order of the spline goes to infinity. Their corresponding orthogonal wavelets converge to the ideal bandpass filter (modulated sinc). In practice, their approximation of an ideal filter starts to be good for $n \geq 3$.

(relative error < 5%). This type of convergence properties may be relevant for coding applications. For instance, it has been shown that the ideal bandpass decomposition is optimal in the sense that it maximizes the coding gain for all stationary processes with non-increasing spectral power density.

13.7 ADVANTAGES OF B-SPLINE BASED WAVELETS IN SIGNAL PROCESSING APPLICATIONS

Efficiency

B-spline wavelets are more computationally efficient. It can capture and process meaningful information contained in the signal as fast as possible. In contrast with wavelets based on Gaussian kernel, the B-spline representation of a signal is determined directly on the initial discrete set of points, avoiding problems caused by discretization in continuous scale-space.

Parallelism

Visual perception treats images on several levels of resolution simultaneously. B-spline techniques provide a good interpretation of “the human visual system, which can process hierarchical information simultaneously. B-splines provide a flexible way to process the multiscale information using coarse to fine strategy or in parallel.

Completeness and invertibility

Zero crossings or local extrema is used as a meaningful description of the signal. For a Gaussian based scale-space, the completeness is guaranteed by the finger print theorem. The map of zero crossing across scales determines the signal uniquely for almost all signals in the absence of noise. The finger print theorem is also true for B-spline kernels. Efficient frame algorithm can be designed to express an image as combinations of multiscale local partial derivatives.

Compactness

For compression application, we require a representation to be as compact as possible so that the image can be represented by the corresponding primitives using less space. All compact multiresolution is related to B-splines. The wavelet filters can be factored into B-spline filters and can be implemented more efficiently.

Causality

A multiscale feature detection method that does not introduce new features as the scale increases is said to possess the property of causality. The continuous causality property of Gaussian kernel is not shared by the B-spline. But causality still holds for discrete B-spline filtering in the discrete sense. The number of local extrema or zero crossings of the derivative of the signal does not increase after running average sum.

Orientation

Orientation analysis is an important task in early vision and image processing, for example in texture analysis. Efficient B-spline pyramid-like algorithm can be designed using the B-spline technique to analyze and synthesize an image from the multiorientational information at any number of angles in the dyadic scale-space. Note that, the wavelet transform can decompose the image only in three orientations.

Other advantages are:

- (a) In time frequency analysis, Gaussian kernel is the optimal function that minimizes the uncertainty principle. As the order increases, B-splines converge to Gaussian in both time and frequency domain.
- (b) B-splines resemble the response of receptive field and is also suitable for modelling the human visual system.
- (c) From the point of regularization theory, cubic spline is proved optimal. Cubic B-spline rather than Gaussian kernel is proved optimal for edge detection.
- (d) B-splines are the shortest basis function that provide a stable multiresolution analysis of the signal. B-spline plays an important role to bridge the traditional scale-space theory, dyadic scale-space frame and compact multiresolution representation.

13.8 B-SPLINE FILTER BANK

In order to use spline wavelets, it is necessary to implement the filter bank procedure incorporating the analysis filters A^j and B^j . These matrices allow us to determine c^{j-1} and d^{j-1} from c^j using matrix multiplication discussed in the previous sections. The analysis filters are uniquely determined by the inverse relation:

$$\begin{bmatrix} A^j \\ B^j \end{bmatrix} = [P^j | Q^j]^{-1} \quad (13.74)$$

However, when implementing the filter bank procedure for spline wavelets, it is generally not a good idea to form the filters A^j and B^j explicitly. Although P^j and Q^j are sparse, having only $O(m)$ entries per column, A^j and B^j are generally dense, so that matrix-vector multiplication would require quadratic instead of linear time. A better approach is to compute c^{j-1} and d^{j-1} from c^j by solving the sparse linear system

$$[P^j | Q^j] \begin{bmatrix} c^{j-1} \\ d^{j-1} \end{bmatrix} = c^j \quad (13.75)$$

In order to solve for $[c^{j-1}/d^{j-1}]$ the matrix $[P^j | Q^j]$ is made into banded matrix and solved in linear time by LU decomposition.

13.9 MULTIREOLUTION CURVES AND FACES

Curves play an important role in graphics applications including computer-aided design. Cross-sectional curves are frequently used in specification surfaces; key-frame animation, in which

curves are used to control parameter interpolation; three dimensional modelling and animation, in which the backbone curves are manipulated to specify object deformations; graphic design, in which curves are used to describe regions of constant colour or texture and font design, in which curves represent the outline of characters. In all these applications, the construction and manipulation of curves and surfaces is a core area of computer graphics. Basic operations such as interactive editing, variational modelling and compact representation of geometry provide many opportunities to take advantage of the unequal features of wavelets. Wavelets also have important contribution in the area of animation where manipulation and optimization of trajectories are considered. A common feature of any CAD modelling package for design of curves and surfaces is the interactive editing environment. For example, B-spline drawing primitives are available where a set of control vertices can be moved with the help of a mouse to achieve the desired shape. Multiple levels of resolution are necessary for building complicated shapes and this can be best achieved by hierarchical B-spline modelling. The same curve can be presented at multiple levels of resolution exposing a set of control vertices appropriate for each level. This representation does not correspond to a basis and a given curve does not possess a unique representation. If instead we encode the difference between successive levels of resolution a representation with respect to a B-spline wavelet basis results. This representation is unique and results in number of computational advantages, such as preconditioning.

13.10 WAVELET BASED CURVE AND SURFACE EDITING

A common paradigm for curve and surface editing is direct manipulation in an interactive environment. Examples include popular drawing programs, which often provide B-spline drawing primitives for curves. Similar tools are available in CAD modelling packages for design of surfaces. To achieve a desired shape, the user moves a set of control vertices with the help of a mouse. For building complicated shapes, hierarchical B-spline modelling can be used. They represent curve at multiple levels of resolution exposing a set of control vertices appropriate for each level. This representation does not correspond to a basis and a given curve does not possess a unique representation. If instead we encode the difference between successive levels of resolution a representation with respect to B-spline wavelet basis results. Consider a system, which uses semiorthogonal cubic B-spline wavelets. A curve, $\gamma(t) = (x(t), \gamma(t))$ is a given sequence of B-spline control knots at some finest resolution L . Performing a wavelet transform on these coefficients results in a wavelet representation of the underlying curve. While all internal computations are performed in the wavelet domain, the user is not presented with the wavelet coefficients for direct manipulation. The results of directly manipulating wavelet coefficients for editing purposes is non-intuitive. This is due to the shape of wavelet function. Pulling one of their control vertices results in a “wiggly” shape change, when one typically expects a smoother shape change. This is easily remedied by performing an inverse wavelet transform to the desired level of resolution and displaying the resulting B-spline control vertices. This way it is possible to intuitively alter the overall sweep of a curve by moving control knots on the coarse levels rather than moving many knots on the finer level. Conversely small details can be added at finer levels without disturbing the overall sweep of the curve. Figure 13.16 shows the examples of the editing modes. On the left, editing the overall sweep of a curve at a coarse level and on the right maintaining the overall sweep while changing the details.

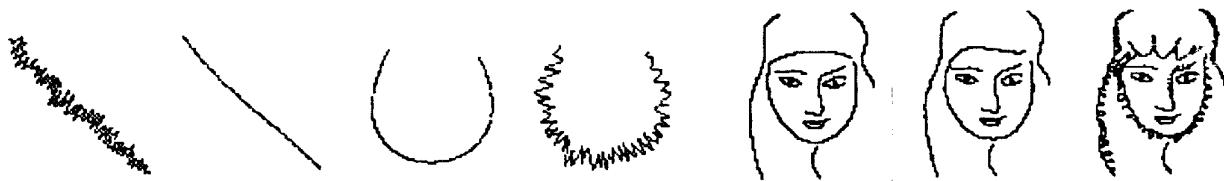


FIGURE 13.16 Wavelet based curve editing at finest and coarse resolutions and an example of keeping the overall shape while manipulating the details.

13.11 VARIATIONAL MODELLING AND THE FINITE ELEMENT METHOD

In the design of curves and surfaces, it is useful for the user to exercise control directly by specifying just the goal function and a few constraints and then letting the computer solve for the best geometric primitive that meets those constraints. This design process is called **variational modelling**, as the goal function or objective function is specified as the minimum of some integral and minimizing the integral is the domain of variational calculus. The first step is to define the objective function more precisely. We will begin by writing an expression for the curve in three-dimensional space as:

$$\gamma(t) = [\gamma_1(t) \quad \gamma_2(t) \quad \gamma_3(t)]^T \quad \text{for } t \in [0 \quad 1] \quad (13.76)$$

Next, we must specify the objective function. One measure of smoothness is the total curvature: the integral of the curvature over the length of the curve. The integral of the second derivative squared can be used as a good measure of curvature. The variational modelling problem then becomes

$$\text{Minimize } \int_0^1 |\gamma_i''(t)|^2 dt \text{ subject to the constraints, for } i = 1, 2, 3$$

For designing a smooth surface subject to certain constraints, we can write a parametric surface as:

$$\sigma(s, t) = [\sigma_1(s, t) \quad \sigma_2(s, t) \quad \sigma_3(s, t)]^T \quad \text{for } s, t \in [0 \quad 1] \quad (13.77)$$

If we approximate the total curvature of this surface by thin plate energy model, the variational modelling problem for the surface can be written as:

$$\int_0^1 \int_0^1 \left[\left| \frac{\partial^2 \sigma_1}{\partial s^2} \right|^2 + \left| \frac{\partial^2 \sigma_1}{\partial s \partial t} \right|^2 + \left| \frac{\partial^2 \sigma_1}{\partial t^2} \right|^2 \right] ds dt \text{ subject to constraints, for } i = 1, 2, 3 \quad (13.78)$$

The surface can be constrained to pass through a point by specifying σ at a particular s and t . The normal of the surface can be constrained by specifying $(\partial \sigma / \partial s, \partial \sigma / \partial t)$ at for some

s and t . Finding the curve or surface with minimum total curvature that satisfies a given set of constraints is a problem of variational calculus because the unknowns are functions. To make the problem computationally tractable, we need to reduce the problem to one in which the unknowns are a finite set of numbers.

Finite element method is a technique for finding an approximate solution to a problem by replacing the unknown function with an unknown linear combination of known basis functions. The basic steps are:

Step 1: Choose n set of basis functions $u(t) = [u_1(t), u_2(t), \dots, u_m(t)]$ called the **finite elements**.

Step 2: Write the unknown functions as linear combination of these finite elements. For example, a function $\gamma(t)$ would be written as:

$$\gamma(t) = \sum_{j=1}^m x u_j(t) = u(t)x \quad (13.79)$$

where $x = [x_1, x_2, \dots, x_m]^T$ is a set of unknown coefficients.

Step 3: Substitute the finite element approximation $u(t)x$ into the original problem involving $\gamma(t)$. Now the problem is easier to solve in terms of the finite set of unknown x .

Let us take the example of finding a smooth curve $\gamma(t)$ that interpolates some specified points. The variational problem is:

$$\text{Minimize } \int_0^1 |\gamma''(t)|^2 dt \text{ subject to } \begin{cases} \gamma(0) = 3 \\ \gamma\left(\frac{1}{2}\right) = 18 \\ \gamma(1) = 12 \end{cases} \quad (13.80)$$

We restrict $\gamma(t)$ to be quadratic function and we can express it as follows:

$$\gamma(t) = x_1 + x_2(t) + x_3t^2 + x_4t^3 + x_5t^4 = u(t)x$$

where

$$u(t) = [1 \quad t \quad t^2 \quad t^3 \quad t^4] \quad \text{and} \quad x = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] \quad (13.81)$$

The finite element representation of $\gamma(t)$ is given by

$$\begin{bmatrix} \gamma(1) \\ \gamma\left(\frac{1}{2}\right) \\ \gamma(1) \end{bmatrix} = \begin{bmatrix} u(1) \\ u\left(\frac{1}{2}\right) \\ u(1) \end{bmatrix} x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16} \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 3 \\ 18 \\ 12 \end{bmatrix} \quad (13.82)$$

The result is a set of linear equations represented as:

$$Ax = b$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16} \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 3 \\ 18 \\ 12 \end{bmatrix} \quad (13.83)$$

The finite element representation of $\gamma(t)$ can be substituted into the objective function as:

$$\begin{aligned} \int_0^1 |\gamma''|^2 dt &= \int_0^1 (u''(t)x)(u''(t)x) dt = \int_0^1 (x^T u''^T)(u''x) dt \\ &= \int_0^1 (x^T (u''^T u'') x) dt = x^T \left[\int_0^1 (u''^T u'') dt \right] x = \frac{1}{2} x^T H x \end{aligned} \quad (13.84)$$

where Hessian matrix H is given by

$$H = 2 \int_0^1 u''(t)^T u''(t) dt = \frac{4}{5} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 15 & 20 \\ 0 & 0 & 15 & 30 & 45 \\ 0 & 0 & 20 & 45 & 72 \end{bmatrix} \quad (13.85)$$

The variational modelling is now of the form:

$$\text{Minimize } \frac{1}{2} x^T H x \text{ subject to } Ax = b \quad (13.86)$$

To solve this problem, we introduce three Lagrange multipliers $\lambda = [\lambda_1 \ \lambda_2 \ \lambda_3]$. We can now include the constraints into part of a single unconstrained minimization problem:

$$\text{Minimize } \frac{1}{2} x^T H x + (Ax - b)^T \lambda \quad (13.87)$$

By taking derivatives of the function with respect to the unknowns x_i, λ_i and setting the derivatives equal to zero. The result is a set of linear equations:

$$\begin{cases} Hx + A^T \lambda = 0 \\ Ax - b = 0 \end{cases} \quad \text{or} \quad \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \quad (13.88)$$

Substituting the values of H, A and b and solving for x and λ , we get

$$x = [3 \ 41 \ 8 \ -80 \ 40] \quad \text{and} \quad \lambda = [256 \ -512 \ 256]$$

In this case we will not use the Lagrange multipliers. The quadric curve that goes through that constraint points while minimizing total curvature is:

$$\gamma(t) = 3 + 41t + 8t^2 - 80t^3 + 40t^4 \quad (13.89)$$

13.12 ADAPTIVE VARIATIONAL MODELLING USING WAVELETS

In variational curve and surface modelling, the curve or surface is modelled by minimizing some energy functional satisfying the set of constraints specified. The curve is decomposed into number of segments and is solved as a discretized version of the continuum problem. The resulting system can be very ill conditioned and although sparse leading to long solution time. The ill conditioning can be particularly addressed through the use of wavelets. If constrained modelling subjected to a quadratic energy functional because of second parametric derivatives in the functional, ill conditioning occurs when discretized in a nodal basis at some finest resolution. The ill conditioning gets worse with increasing subdivision: If the same problem is instead solved in the wavelet basis, diagonal preconditioners can be applied which will lead to systems whose condition number is uniformly bound independent of the size of mesh. This preconditioning strategy is easily absorbed right into the wavelet transform. Figure 13.17 shows a simple example of preconditioning.

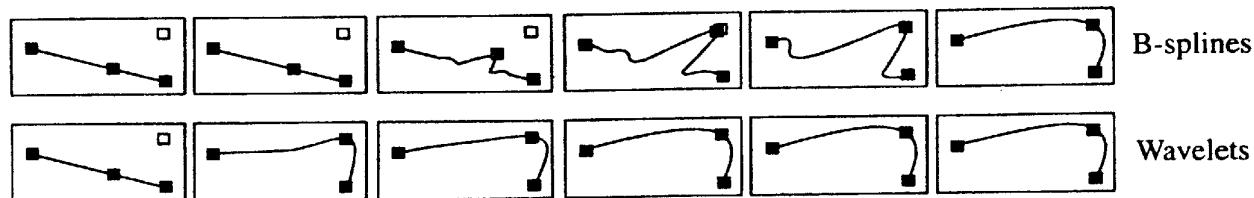


FIGURE 13.17 An example of preconditioning.

A simple point interpolation problem subject to a smoothness constraint. Going left to right the solution is presented after some number of iterations (0–1024). The top row shows the convergence in some finest level B-spline basis while the bottom row shows the convergence in the associated wavelet basis with preconditioning.

Wavelets have another advantage for these kinds of problems. They naturally lead to an error estimator driven adaptive meshing strategy. Wavelets have the ability to characterize local smoothness. Some parts of the curve need to be meshed finely while other parts require only coarse mesh. Initially optimization is attempted over a very coarse resolution curve. Next a refinement step based on already used wavelet coefficients is performed. Wherever wavelet coefficients are larger in magnitude finer level wavelets entered as new parameters. This adaptive wavelet basis with preconditioning leads to vastly faster solution algorithms. In this scenario the user specifies the geometry and degrees of freedom of some object. The degrees of freedom typically describe quantities such as joint angles, orientations and locations. All degrees of freedom are subject to Newtonian dynamics and a set of constraints given as equalities or inequalities. The user will then ask for a feasible solution satisfying some goal, such as moving from one location to another while consuming a minimal amount of fuel. This results in large search space over which to optimize the user specified requirements. Wavelets can be helpful because of their ability to lead better-conditioned system and because they enable adaptive refinement.

$$\begin{bmatrix} w^T H w^{-1} & w^{-T} A^T \\ A w^{-1} & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \quad (13.90)$$

where w represents wavelet transform and $\hat{x} = wx$ is a set of wavelet coefficients for the solution.

Wavelet like decomposition of surfaces would be useful for compression of original database. Making the compression ratio adaptive would allow algorithm to choose the most appropriate solution, e.g., the objects viewed from a distance could be compressed considerably more than the objects near by. Lifting scheme is a very general construction scheme for the second-generation wavelets. The idea behind the second-generation wavelets is that the scaling and dilation are not really fundamental to reap all the benefits of wavelets, such as space frequency localization and fast transform. The lifting scheme provides versatile tool to construct wavelets which still have all the desirables of traditional wavelets but allow for the accommodation of such custom constraints as boundary conditions weighted measures, irregular sampling and adaptive subdivision. This technique was employed to construct wavelets on the sphere where both irregular sampling and adaptive subdivisions were required. Application of spherical wavelets include modelling of reflection of surfaces, compression and processing of large spherical data sets such as topography data for earth and spherical image processing. Spherical wavelets are used to selectively sharpen and blur environmental maps, i.e., images that are defined over the set of directions.

SUMMARY

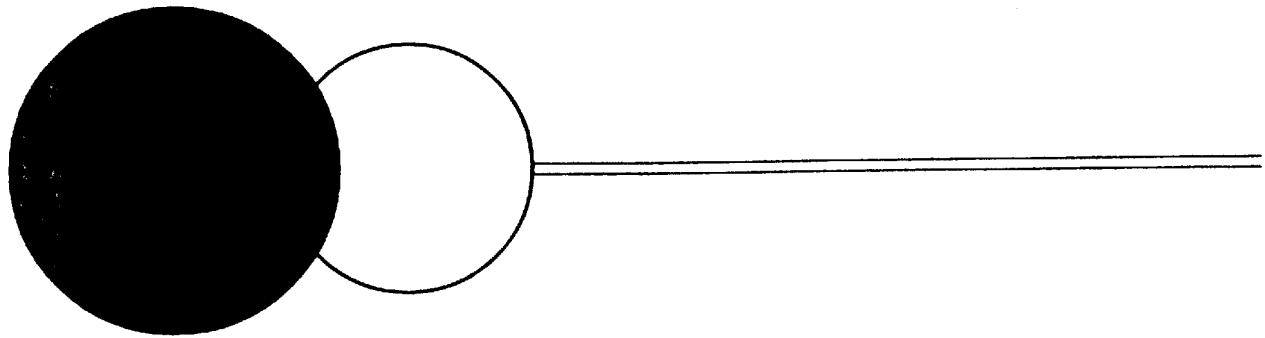
The special properties of splines are attracting lot of attention in signal processing. They have maximum regularity (and symmetry) with minimum support and complexity. Splines are outstanding in synthesis. They give approximation of higher order p and low constant C in the error term $C(\Delta t)^p$. B-splines based scale-space possess almost the same properties of Gaussian kernel. B-spline kernels outperforms the Gaussian in many aspects, notably, computational efficiency.

REFERENCES

- [1] Burrus, C. Sydney, Ramesh A. Gopinath, and Haitao Guo, *Introduction to Wavelets and Wavelet Transform—A Primer*, Prentice Hall International, 1998.
- [2] Rogers and Adams, *Mathematical Elements for Computer Graphics*, 1986.
- [3] Stollnitz, Eric J., Tony D. DeRose, and David H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

- [4] Les Piegel and Wayne Tiller, *The NURBS Book*, Springer-Verlag 1997.
- [5] Gilbert Strang and Truong Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley MA (USA), 1996.
- [6] Peter Schroder, Wavelets in computer graphics, *Proceedings of the IEEE*, Vol. 84, No. 4, pp. 615–625, 1996.
- [7] Prenter, P.M., *Splines and Variational Methods*, John Wiley & Sons, 1972.
- [8] Yu-Ping Wang and S.L. Lee, Scale-space derived from B-splines, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 1040–1055, 1998.





Appendix

VECTOR SPACES

Further reading on this section can be found in any standard book on linear algebra. To avoid having to keep track of what type of numbers we are using at the moment, we will do everything with complex numbers. Think of real numbers as a special case. Complex numbers are also called *scalars* in this setting.

If $z = x + iy$ is a complex number, its complex conjugate is $\bar{z} = x - iy$.

Z is the set of integers, R is the set of real numbers and C is the set of complex numbers.

Definition: A (complex) *vector space* V is a set of elements v, w, \dots , called **vectors**, with two arithmetic operations:

- *Addition*

$$v + w \in V \quad \text{if} \quad v, w \in V$$

- *Scalar multiplication*

$$\alpha v \in V \quad \text{if} \quad \alpha \in C, v \in V$$

Properties

- *Commutativity*

$$v + w = w + v$$

- *Associativity*

$$(v + w) + z = v + (w + z)$$

$$(\alpha\beta)v = \alpha(\beta v)$$

- *Distributivity*

$$\alpha(v + w) = \alpha v + \alpha w$$

$$(\alpha + \beta)v = \alpha v + \beta v$$

- *Zero vector.* There exists a vector $\mathbf{0} \in V$ such that

$$v + \mathbf{0} = \mathbf{0} + v = v \text{ for all } v \in V$$

- *Inverse Vector.* For all $v \in V$, there exists a vector $(-v) \in V$ such that

$$v + (-v) = (-v) + v = \mathbf{0}$$

Standard examples

- (a) $V = C^n$, $v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ or $\{v_1, v_2, \dots, v_n\}$ with componentwise addition and scalar multiplication.
- (b) $V =$ The space of infinite sequences of complex numbers, $v = \{v_j\}_{j=-\infty}^{\infty}$, again with componentwise addition and scalar multiplication.
- (c) $V =$ The space of complex-valued functions on $[a, b]$, with pointwise addition and scalar multiplication.

$$(v + w)(x) = v(x) + w(x)$$

$$(\alpha v)x = \alpha v(x)$$

Definition: A *linear combination* of vectors v_1, v_2, \dots, v_n is:

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n = \sum_j \alpha_j v_j$$

Definition: A collection $\{v_1, v_2, \dots, v_n\}$ of vectors is *linearly dependent* if one of them can be written as a linear combination of the others:

$$v_k = \sum_{j \neq k} \alpha_j v_j \text{ for some } k$$

Otherwise they are linearly independent.

Definition: A *basis* of V is a collection of linearly independent vectors such that any $v \in V$ can be written as a linear combination of basis vectors.

Note: Every basis contains the same number of vectors; this number is called the **dimension** of V .

Definition: A *norm* on a vector space V is a mapping which assigns to each $v \in V$ a norm (or length) $\|v\|$ with the properties:

- (1) $\|v\|$ is real, $\|v\| \geq 0$
- (2) $\|v\| = 0 \Leftrightarrow v = \mathbf{0}$
- (3) $\|\alpha v\| = \|\alpha\| \|v\|$, $\alpha \in C$
- (4) $\|v + w\| \leq \|v\| + \|w\|$ (the triangle inequality)

Standard examples

- (a) *The l_p norms.* Assume that $v = \{v_j\}$ (a finite vector or infinite sequence). For $1 \leq p \leq \infty$, we define the p -norm by

$$\|v\|_p = \left\{ \sum_j |v_j|^p \right\}^{1/p}$$

The only special case we need are:

$$\|v\|_1 = \sum_j |v_j|$$

$$\|v\|_2 = \sqrt{\sum_j |v_j|^2}$$

$$\|v\|_\infty = \sup_j |v_j|$$

The infinity norm $\|v\|_\infty$ is not covered by the initial definition. It is defined as the limit of the p -norms as p goes to ∞ , which turns out to be formula given. For finite vectors, all these norms are well defined but for infinite sequences they may not exist. Thus, we define

$$l_p = \text{The set of all infinite sequences } \{v_j\} \text{ for which } \|v\|_p < \infty$$

For each $1 \leq p \leq \infty$, this is a normed vector space and a subspace of the space of all infinite sequences.

The sequences in l_1 are called **summable**, the sequences in l_2 are called **square summable** and the sequences in l_∞ are **bounded**.

- (b) *The L^p norms.* Assume that v is a function on some interval $[a, b]$ (which could be finite or infinite). For $1 \leq p \leq \infty$, we define the p -norm by

$$\|v\|_p = \left\{ \int_a^b |v(x)|^p dx \right\}^{1/p}$$

The only special case we need are:

$$\|v\|_1 = \int_a^b |v(x)| dx$$

$$\|v\|_2 = \sqrt{\int_a^b |v(x)|^2 dx}$$

$$\|v\|_\infty = \text{Ess sup}|v(x)|$$

Again, the infinity norm is the limit of the p -norms. *Ess sup* stands for *essential supremum* which means the supremum except for sets of measure zero. For any given function, a p -norm may or may not exist. We introduce the spaces,

$L^p[a, b] =$ The set of all functions v on $[a, b]$ for which $\|v\|_p < \infty$

For $1 \leq p \leq \infty$, these are vector spaces and subspaces of the set of all functions on $[a, b]$. If the interval is not given, it is R :

$$L^2 = L^2(R)$$

The functions in L^1 are called **integrable**, the functions in L^2 are called **square integrable** and the functions in L^∞ are (essentially) **bounded**.

Definition: An *inner product* on a vector space is a mapping which assigns to each pair $v, w \in V$ a complex number $\langle v, w \rangle$ such that

$$\langle \alpha v + \beta w, z \rangle = \alpha \langle v, z \rangle + \beta \langle w, z \rangle$$

$$\langle v, \alpha w + \beta z \rangle = \bar{\alpha} \langle v, w \rangle + \bar{\beta} \langle v, z \rangle$$

$$\langle v, w \rangle = \overline{\langle w, v \rangle}$$

$$\langle v, v \rangle \geq 0 \text{ for all } v$$

$$\langle v, v \rangle = 0 \Leftrightarrow v = 0$$

Standard examples

(a) C^n and l_2 have the inner product

$$\langle v, w \rangle = \sum_j v_j \bar{w}_j$$

(b) $L^2[a, b]$ has the inner product

$$\langle v, w \rangle = \int_a^b v(x) \overline{w(x)} dx$$

Note: Every inner product defines a norm by

$$\|v\| = \sqrt{\langle v, v \rangle}$$

The 2-norms in l_2 and L^2 come from the standard inner products defined earlier.

ORTHONORMAL BASES

Two vectors v, w are *orthonormal* if they are orthogonal, that is, $\langle v, w \rangle = 0$, and they both have length 1:

$$\|v\|_2 = \|w\|_2 = 1$$

Likewise, a collection $\{v_1, v_2, \dots, v_n\}$ of vectors is orthonormal if they are pairwise orthogonal, and all have length 1:

$$\langle v_i, v_j \rangle = \delta_{ij}$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

(δ_{ij} is called the **Kronecker delta**) Among all possible bases of a vector space V , orthonormal bases are especially popular because of their many properties and their numerical stability. Assume that V is a vector space with an inner product and with a basis $\{v_j\}$. Any basis has the property that any $f \in V$ can be written in a unique way as a linear combination of basis vectors,

$$f = \sum_j \alpha_j v_j$$

However, the α_j may be hard to calculate and may not be bounded. For an orthonormal basis, both of these problems disappear:

$$\alpha_j = \langle f, v_j \rangle$$

and

$$\sum_j |\alpha_j|^2 \leq \|f\|_2^2 \quad (\text{Bessel's inequality})$$

Remark: Because of convergence problems, it is in general required that any vector can be written as a *finite* linear combination of basis vectors. For orthonormal bases, Bessel's inequality provides convergence, so infinite linear combinations are also allowed.

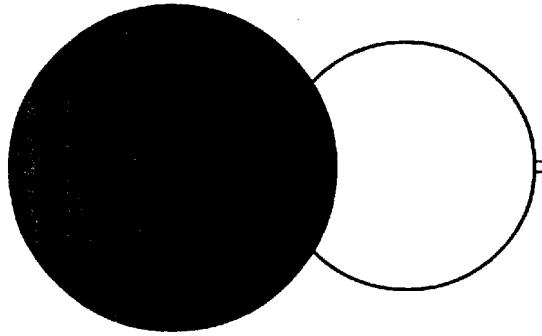
Thus, in an orthonormal basis, any $f \in V$ can be written as:

$$f = \sum_j \langle f, v_j \rangle v_j$$

More generally, if $\{v_1, v_2, \dots, v_n\}$ is a subset of $\{v_j\}$ then

$$Pf = \sum_{j=1}^n \langle f, v_j \rangle v_j$$

is the orthogonal projection of f onto the subspace spanned by $\{v_1, v_2, \dots, v_n\}$.



Index

Analysis filter, 268

Approximation
condition, 77
using spline, 242

B-spline, 253, 259, 261, 262, 264, 273, 274, 275, 276
surface, 265

Bases, 15, 16

Basis matrix, 244

Bayes shrink method, 229

Bernstein blending function, 257

Best basic selection, 198

Beta

parameter, 265
splines, 268

Bezir

curves, 250, 251, 252, 257, 258
spline, 250, 258, 261

Biorthogonal filters, 181

Biorthogonality, 125, 137

Box function, 133

Cardinal splines, 246

Cascade algorithm, 110

Circular convolution, 191, 193

Coiflet

basis, 234
wavelet, 81

Complex Fourier series, 24

Continuous

Fourier transform, 35
time frequency representation of signals, 37
wavelets transform, 34, 42, 43

Control

graph, 242
points, 242
Convex hull, 242
Convolution, 36, 89, 158
Cox-deBoor recursion formula, 258
Critical sampling, 44
Cubic
spline, 245, 247
surfaces, 255

Daubechies wavelet filters, 9, 68, 75, 79, 80, 104, 147,
193

Denoising, 221, 230

Dilation equation, 54, 67

Discrete wavelets transform, 34

Dominant pass algorithm, 212

Double shift orthogonality, 75

Down-sampling, 158

Dual lifting step, 153

Dyadic translates, 9

EBCOT, 218

Embedded encoding, 209

Entropy encoding, 206

- Euclidean algorithm, 183, 184, 185, 187
EZW
 algorithm, 210
 coding, 208
 performance, 217
- F**
- Filter
 bank, 89, 200, 268, 276
 coefficient, 74, 141, 158
 Finite element methods, 278, 279
 FIR filter, 94
 Forward wavelet transform, 159
 Fourier
 analysis, 4, 153
 techniques, 170
 transform, 4, 73
 Frequency response, 94
 Function space, 17, 51
- Gabor wavelets, 32
 Gaussian density, 234
 Generating
 $\phi(t)$, 120
 $\psi(t)$, 122
 Geometric
 coefficient, 246
 continuity, 243
 Gram-Schmidt orthogonalization, 20
- Haar
 wavelet transform, 161, 162, 163, 189
 wavelets, 5, 48, 55, 56
 Heavisine, 234
 Heisenberg's uncertainty principle, 45
 Hermite
 polynomial, 246
 spline, 245, 248, 261, 264
 Hessian matrix, 280
 Huffman encoding, 206
- Interpolation splines, 242, 245
 Inverse
 lifting, 158
 transform, 160
- JPEG standard, 199
- Karhunen-Loeve basis, 198
 Kochek-Bartel splines, 246
- Laurent polynomial, 182, 183, 184, 185
 Lazy wavelet transform, 164
 Lifting scheme, 153, 161, 163, 167, 171, 180, 190
 Local trigonometric bases, 9
 Localized frequency, 3
 LZW, 198
- Maple code, 106
 Morlet wavelets, 6, 32
 Morton scan images, 211
 Mother wavelet, 6, 267
 Multiresolution analysis (MRA), 100
 Multiwavelets, 10
- Noise estimation, 223
 Non-uniform quantizer, 203
 Normality, 137
 Normalization of
 data, 224
 Haar, 63
 Normalized coefficients, 22
 NURBS, 12, 266
- Orthogonal
 basis functions, 20
 functions, 16
 wavelet, 73, 269
 Orthogonality, 7
 Orthogonality of translates of $\phi(t)$, 50
 Orthonormal, 17, 51, 142
 spaces, 15
 vectors, 15
 Orthonormality, 15, 20, 75
- Padding, 191, 192
 Parametric wavelets, 102
 Parameterization, 102

- Perfect matching filters, 98
- Plotting Daubechies wavelets, 69
- Pollen-type parameterization, 102, 109
- Polyphase
 - factorization, 182
 - matrix, 153, 155, 158, 188, 189
 - recurrence relation, 104
- Prediction steps, 158
- Primal lifting step, 153
- Pyramidal algorithm, 5

- Quad tree, 209
- Quadrature mirror filters, 99
- Quadric surfaces, 239, 240
- Quantizer, 203
- Quintic spline wavelets, 32

- Raster scan images, 211
- Refinement relation, 54, 67
- Ridgelets, 12
- Riesz theorem, 150
- RLE, 199
- Runlength encoding, 207

- Scaled Haar wavelets, 57
- Scaling, 33, 49
 - function, 54, 74, 175, 266
- Second generation wavelets, 10
- Semiorthogonal wavelets, 269
- Shan wavelets, 32
- Shrinkage
 - functions, 225
 - rules, 227
- Signal decomposition, 86, 162
- Soft threshold, 226
- Spectral factorization, 150
- Spectrum, 34
- SPIHT, 218
- Spline
 - curves and surfaces, 241
 - wavelets, 238, 270, 273
- Subband coding, 5
 - uniform quantizer, 203
- Subdivision
 - scheme, 120
 - surfaces, 12

- Subordinate pass algorithm, 212
- Successive approximation, 114
- Support of wavelets system, 66
- Sure, 228
 - shrink, 228
- Symlet, 82
 - wavelets, 234
- Synthesis filter, 268

- Tension parameter, 246
- Threshold method, 233
- Thresholding, 229
- Time-domain relationship, 156
- Top rule, 228
- Translation, 33, 36, 49
 - invariance, 229
- Triangle scaling function, 67
- Trigonometric basis, 198
- Two-band analysis, 93

- Uncertainty principle, 40
- Uniform dead-zone quantizer, 205
- Uniform quantizer, 203
- Unitary, 144
- Universal rule, 227
- Update steps, 158
- Upsampling and filtering, 97

- Vanishing moment, 137, 151, 177, 181, 270
- Variational modelling, 278, 281
- Vector space, 15, 125

- Wave, 234
- Wavelet
 - function, 175
 - packets, 9
 - shrinkage, 222
 - transform, 31, 170, 200
- Windowed Fourier transform, 38, 41, 42

- Z-domain, 171
- Zero tree, 209
- Zeroth moment, 178
- Z-transform, 142, 171, 183

INSIGHT INTO WAVELETS

From Theory to Practice

K.P. Soman • K.I. Ramachandran

The rapid growth of the theory of wavelets and their application in diverse areas—ranging from oil exploration to bioinformatics, and astrophysics—has made it imperative that engineers and scientists pursuing these areas have a working knowledge of wavelets. In the past few years, the study of wavelets and the exploration of the principles governing their behaviour have brought about sweeping changes in the disciplines of pure and applied mathematics and sciences. Keeping this in mind, the authors have specifically written this comprehensive text to fulfill the curriculum requirements of a course on wavelets. It offers an introduction to wavelet analysis, in an easy-to-understand style, without mathematical rigour.

Beginning with a description of the origin of wavelets and a discussion on the recent developments, the book moves on to explain the basic concepts in Fourier series, continuous wavelet transform, discrete wavelet transform, lifting scheme and applications of wavelets in image compression, signal denoising and computer graphics.

Intended primarily as a textbook for the postgraduate students of computer science, electrical/electronics and communication engineering, this book would also be useful for the practising engineers and researchers.

KEY FEATURES

- Describes wavelet concepts from both the signal expansion and filter theory point of view.
- Gives clear concise explanation of biorthogonality and biorthonormal wavelet analysis.
- Contains a separate chapter on the applications of wavelets in computer graphics.
- Analyzes wavelet design from a time domain as well as a frequency domain point of view.
- Includes a very lucid introduction to parametric wavelets.
- Explains lifting-scheme based wavelet analysis and design in detail.

Rs. 250.00

Prentice-Hall of India

New Delhi

www.phindia.com

ABOUT THE AUTHORS

K.P. SOMAN, (Ph.D., IIT Kharagpur), is Head, Centre for Excellence in Computational Engineering and Networking, Amrita Vishwa Vidyapeetham, Coimbatore. He has presented/published over 50 papers in national and international conferences and journals. His areas of interest include high performance computing, design patterns, wavelet transform, fractal analysis, data mining and cryptography.

K.I. RAMACHANDRAN, Ph.D., is Professor, Centre for Excellence in Computational Engineering and Networking, Amrita Vishwa Vidyapeetham, Coimbatore. Professor Ramachandran has several papers in journals to his credit. His research interests include computational fluid mechanics, nonlinear dynamics, virtual instrumentation, wavelets and fractals.



ISBN 81-203-2650-4



9 788120 326507